ON-POLICY PARAMETER UPDATE DIRECTION UNDERLIES GENERALIZATION IN LLM POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

The strong generalization performance of on-policy post-training paradigms has motivated studies of their parameter update behaviors. However, these studies treat the observed behaviors only as byproducts in on-policy training, overlooking their potential to serve as optimization principles for improving the generalization of other paradigms such as supervised fine-tuning (SFT). To address this limitation, we investigate whether there exists a specific on-policy update behavior that can achieve such improvements. First, our theoretical and experimental analyses reveal that SFT updates parameters along consistent directions, while the on-policy paradigm continuously adjusts the direction during training. This difference inspires us to focus on the cumulative update direction of each parameter as a promising behavior. Then, we evaluate its effectiveness for improving generalization by proposing On-Policy direction-constrained Supervised Fine-Tuning (OPSFT), which constrains SFT updates to the direction identified by on-policy paradigms. The strong performance of OPSFT indicates that the generalization advantage of on-policy paradigms can be transferred to SFT through the parameter update direction. Once such a direction is identified, even SFT can generalize with its updates constrained to this direction. This finding offers two practical benefits by combining the strong generalization of on-policy paradigms with the advantages of SFT, including the high training efficiency and ability to leverage high-quality trajectories. For efficiency, we identify update directions that support strong generalization using a few on-policy training steps, and subsequently apply OPSFT to achieve high training efficiency. For leveraging high-quality trajectories, OPSFT can utilize these trajectories to continue improving a post-trained model along its update direction without disrupting the ability learned from on-policy training. Codes are provided in the supplementary material.

1 INTRODUCTION

The on-policy post-training has emerged as an important paradigm for enhancing the reasoning capabilities of large language models (LLMs) (Shao et al., 2024; Lu & Lab, 2025; Agarwal et al., 2024). Compared to the off-policy paradigm such as supervised fine-tuning (SFT) that optimizes the model using fixed ground-truth responses (Hinton et al., 2015; Gu et al., 2024), the on-policy paradigm optimizes the model using its self-generated responses and achieves strong generalization performance across diverse tasks (Zhang et al., 2025; Guo et al., 2025; Xiaomi et al., 2025).

The strong generalization of on-policy paradigms has motivated studies of their internal mechanisms to understand the reasons behind their success (Nguyen et al., 2025; Mukherjee et al., 2026). Prior studies investigate the effects of individual components that distinguish on-policy paradigms from SFT, such as the reverse KL divergence in OPD (Nguyen et al., 2025) and negative samples in GRPO (Abdolmaleki et al., 2025). Despite these efforts, on-policy paradigms and SFT differ in numerous components (Zhao et al., 2026), making component-wise analysis costly and fragmented. To obtain more unified conclusions, recent studies bypass individual components and focus directly on their resulting parameter updates. By comparing the updates of on-policy paradigms and SFT from both the spectrum (Wu et al., 2025; Zhu et al., 2025; Shen et al., 2026c) and weight matrix (Mukherjee et al., 2026; Zhang et al., 2026) perspectives, these studies provide insights into the distinctive parameter update behaviors of on-policy paradigms, such as the sparse update locations (Mukherjee et al., 2026) and the small spectrum shift (Shen et al., 2026c).

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

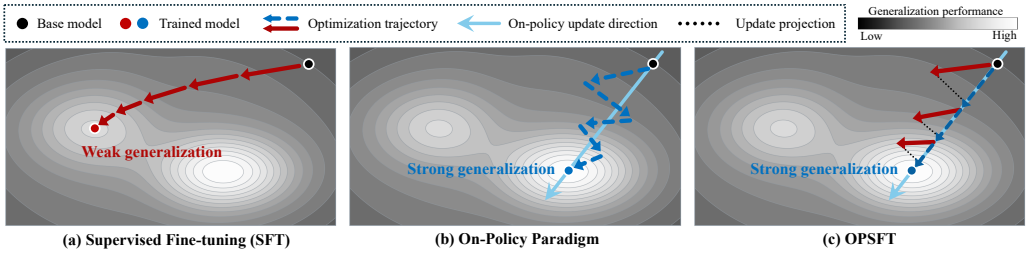

Figure 1: (a) SFT update parameters along consistent directions throughout training. (b) In contrast, on-policy paradigms continuously adjust the update direction. (c) To investigate whether the update direction identified by on-policy paradigms can improve the generalization of SFT, we introduce OPSFT that constrains the SFT update of each parameter to the corresponding on-policy direction.

However, current studies treat these optimization behaviors only as byproducts of on-policy training (Shen et al., 2026c; Wu et al., 2025), rather than exploring their contributions to generalization. This perspective overlooks the potential of these behaviors to serve as effective optimization principles for improving the generalization of SFT. Therefore, there remains a substantial gap between understanding the parameter update behaviors of on-policy paradigms and translating these insights into practical improvements in generalization performance.

In this paper, to bridge this gap, we investigate whether there exists an on-policy parameter update behavior that can improve the generalization of SFT. First, inspired by studies of update locations (Mukherjee et al., 2026), we extend the investigation target to update directions, which include both the location and sign of each parameter update. Our theoretical and experimental analyses demonstrate that, unlike SFT that updates along consistent directions (Figure 1a), the on-policy paradigm continuously adjusts the update direction throughout training (Figure 1b). This continual effort for direction adjustment motivates us to investigate the resulting cumulative update direction of on-policy training as a promising behavior. To evaluate its effectiveness, we propose On-Policy direction-constrained SFT (OPSFT), which retains SFT while constraining the update direction of each parameter to that identified by the on-policy paradigm (Figure 1c). By comparing the generalization performance of OPSFT with vanilla SFT and on-policy paradigms, we evaluate the ability of the on-policy update direction to improve SFT. Experiments across different models (Yang et al., 2025; Guo et al., 2025) and benchmarks (Li et al., 2023; Zhou et al., 2023; Zhang & Math-AI, 2024; 2025; Cui et al., 2025; He et al., 2026; Balunovic et al., 2026) demonstrate that OPSFT substantially outperforms vanilla SFT while achieving performance comparable to the on-policy paradigm. The results suggest that the parameter update direction identified by on-policy paradigms can effectively transfer their generalization advantage to SFT. Once this direction is identified, even SFT can achieve strong generalization performance when its updates are restricted to this direction.

This finding benefits post-training by integrating the strong generalization of on-policy paradigms with the advantages of SFT, including high training efficiency and the ability to leverage high-quality trajectories. For efficiency, we first conduct a few steps of on-policy training to identify the update direction and then apply OPSFT. By optimizing along the on-policy update direction without requiring continuous rollouts, we can achieve strong generalization and high training efficiency simultaneously. For leveraging high-quality trajectories, given a model post-trained with an on-policy paradigm, OPSFT can continue training this model along its on-policy update direction, thereby avoiding disruption to the capabilities learned during on-policy training. This strategy allows the model to leverage newly acquired trajectories for continuous performance improvement without repeating the entire SFT-then-RL process. Our contributions are illustrated as follows:

- We analyze the parameter update direction of on-policy paradigms and reveal their preference for continuously adjusting the update direction during training.
- We propose OPSFT that constrains updates of SFT to the on-policy direction. Its strong generalization performance indicates that the update direction identified by on-policy paradigms can transfer their generalization advantage to SFT.
- We leverage the investigation results to achieve strong generalization with high training efficiency, and apply OPSFT with newly acquired high-quality trajectories to continue improving models post-trained by on-policy paradigms.

2 Related Work

Optimization Behavior of On-Policy Post-Training. To understand the mechanisms behind the effectiveness of on-policy post-training, prior studies analyze the principal components of its optimization geometry (Wu et al., 2025; Zhu et al., 2025) and the impact of its training objectives on optimization behaviors (Nguyen et al., 2025; Abdolmaleki et al., 2025). More fundamentally, recent studies directly investigate parameter updates and discover that on-policy updates are localized within small sub-networks (Mukherjee et al., 2026). Unlike existing studies that focus only on update locations and treat them as byproducts of on-policy paradigms (Shen et al., 2026c; Yu et al., 2026a), we extend the investigation to the update direction and discover its ability to transfer the generalization advantage of on-policy paradigms to SFT.

Task-Relevant Parameter Localization. Sparsely updating parameters is widely adopted to avoid overfitting in fine-tuning tasks or improve interpretability (Zhang et al., 2024; Ansell et al., 2024; Gao et al., 2025; Shen et al., 2025; 2026a). Current methods typically obtain the relevance between parameters and tasks and only adjust the task-relevant parameters throughout training (Han et al., 2024; He et al., 2025). Unlike prior approaches that rely on heuristic strategies under off-policy paradigms to estimate task relevance (He et al., 2023; Shen et al., 2024; 2026b), we demonstrate that the on-policy post-training paradigm naturally serves as an effective mechanism for identifying not only task-relevant parameter update locations but also directions.

Generalization Ability of Supervised Fine-tuning. As a standard post-training approach, supervised fine-tuning (SFT) is widely adopted for its simplicity and efficiency in imitating expert demonstrations (Wu et al., 2026; Ren et al., 2026). However, its generalization performance falls significantly short of on-policy post-training paradigms, such as GRPO (Shao et al., 2024). To bridge this gap, existing methods aim to enhance SFT by incorporating importance-sampling data selection strategies (Qin & Springenberg, 2025), logit-weighted cross-entropy losses (Wu et al., 2026), or negative samples (Abdolmaleki et al., 2025). Unlike these methods that imitate on-policy paradigms with training objectives and dataset constructions, we leverage the parameter update direction identified by on-policy paradigms to improve the generalization of SFT.

3 On-Policy Update Direction Underlies Generalization

In this section, we explore whether there exists an on-policy parameter update behavior that can transfer the generalization advantage of on-policy paradigms to SFT. Inspired by previous studies that focus on parameter update locations (Shen et al., 2026c; Mukherjee et al., 2026), we extend the investigation to the direction (*i.e.*, sign) and its evolution throughout training.

3.1 Analysis of Parameter Update Directions

Theoretical Analysis. We compare gradient formulations of the on-policy paradigms and SFT to illustrate their different behaviors in parameter update direction at each training step. Given an input prompt x, a response trajectory τ, and the current policy $\pi_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^d$, the gradient of the trajectory log-probability with respect to parameters $\boldsymbol{\theta}$ is computed as follows:

$$s_{\boldsymbol{\theta}}(x, \tau) = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau \mid x). \tag{1}$$

For every input prompt x, $s_{\boldsymbol{\theta}}(x, \tau)$ has zero conditional expectation:

$$\mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}(\cdot|x)} \left[s_{\boldsymbol{\theta}}(x, \tau) \right] = \sum_{\tau} \pi_{\theta}(\tau \mid x) \nabla_{\theta} \log \pi_{\theta}(\tau \mid x) = \nabla_{\theta} \sum_{\tau} \pi_{\theta}(\tau \mid x) = \mathbf{0}_d, \tag{2}$$

where $\mathbf{0}_d \in \mathbb{R}^d$ denotes the zero vector. According to the derivations of previous methods (Shao et al., 2024), the policy gradient of on-policy paradigms can be represented as follows,

$$g_{\text{on}}(\boldsymbol{\theta}) = \mathbb{E}_{x,\ \tau \sim \pi_{\theta}(\cdot|x)} \left[A_{\boldsymbol{\theta}}(x, \tau) s_{\boldsymbol{\theta}}(x, \tau) \right], \tag{3}$$

where $A_{\boldsymbol{\theta}}(x, \tau) \in \mathbb{R}$ is the advantage of trajectory τ for prompt x under the policy parameterized by $\boldsymbol{\theta}$. The projection of $g_{\text{on}}(\boldsymbol{\theta})$ onto an arbitrary sign vector $\boldsymbol{v} \in \{-1, 0, 1\}^d$ can be formulated as:

$$\boldsymbol{v}^{\top} g_{\text{on}}(\boldsymbol{\theta}) = \mathbb{E}_{x,\ \tau \sim \pi_{\boldsymbol{\theta}}(\cdot|x)} \left[\boldsymbol{v}^{\top} A_{\boldsymbol{\theta}}(x, \tau) s_{\boldsymbol{\theta}}(x, \tau) \right]. \tag{4}$$

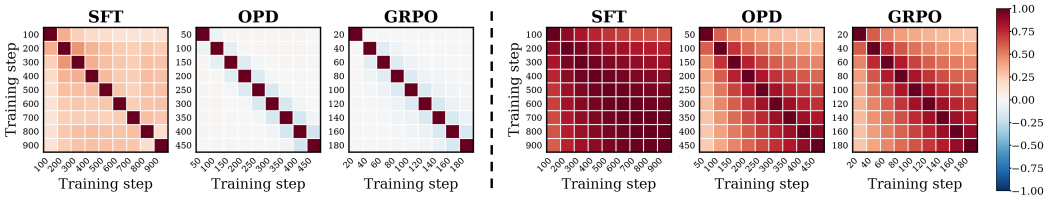

Figure 2: Comparisons of update directions between SFT and on-policy (OPD, GRPO) paradigms using Qwen3-1.7B on DeepMath. We present the cosine similarity of interval *(left)* and cumulative *(right)* parameter updates among different training stages.

Given the zero conditional expectation of $s_{\boldsymbol{\theta}}(x,\tau)$ in Equation 2, Equation 4 can be reformulated as the projection onto $\boldsymbol{v}$ of the covariance between $A_{\boldsymbol{\theta}}(x,\tau)$ and $s_{\boldsymbol{\theta}}(x,\tau)$:

$$\boldsymbol{v}^\top g_{\text{on}}(\boldsymbol{\theta}) = \mathbb{E}_x\left[\boldsymbol{v}^\top \operatorname{Cov}_{\tau\sim\pi_{\boldsymbol{\theta}}(\cdot|x)}\left[A_{\boldsymbol{\theta}}(x,\tau), s_{\boldsymbol{\theta}}(x,\tau)\right]\right], \tag{5}$$

For SFT, the projection of their gradients along $\boldsymbol{v}$ is:

$$\boldsymbol{v}^\top g_{\text{sft}}(\boldsymbol{\theta}) = -\mathbb{E}_x\left[\boldsymbol{v}^\top \mathbb{E}_{\tau\sim\pi_{\text{teacher}}(\cdot|x)}\left[s_{\boldsymbol{\theta}}(x,\tau)\right]\right]. \tag{6}$$

By comparing Equation 5 with Equation 6, we find that SFT updates parameters along the direction of $s_{\boldsymbol{\theta}}(x,\tau)$, where the trajectories τ are typically sampled from a fixed distribution π_{teacher}. In contrast, on-policy paradigms update parameters toward the direction of the covariance matrix between the advantages $A_{\boldsymbol{\theta}}(x,\tau)$ and the trajectory gradients $s_{\boldsymbol{\theta}}(x,\tau)$. Since the trajectories τ are sampled from the current policy $\pi_{\boldsymbol{\theta}}$ that evolves during training, the distribution of advantages $A_{\boldsymbol{\theta}}(x,\tau)$ changes with $\pi_{\boldsymbol{\theta}}$ rather than serving as a fixed target. Consequently, the parameter update direction of on-policy paradigms changes with the evolving advantage distribution throughout training.

Experimental Analysis. To verify the above theoretical analysis, we measure the cosine similarity among update directions at different training steps from both interval and cumulative perspectives, as shown in Figure 2. For interval updates, SFT optimizes towards positively correlated directions, while on-policy paradigms explore nearly orthogonal directions at different training stages. This observation is consistent with our analysis in Equation 5 and Equation 6. The accumulation of these interval updates consequently leads to substantially different cumulative update directions between SFT and on-policy paradigms. For cumulative updates, SFT exhibits highly similar directions throughout training (with cosine similarity close to 1.0), whereas on-policy paradigms exhibit positive yet substantially lower correlation (with cosine similarity around 0.5). These analyses suggest that, unlike SFT that updates parameters along nearly consistent directions, on-policy paradigms tend to continuously adjust their cumulative parameter update directions throughout training.

3.2 Applying On-Policy Update Directions to SFT

Inspired by the continuous efforts to adjust update directions in on-policy paradigms, we investigate whether the resulting cumulative on-policy update direction can serve as an effective optimization principle for transferring their generalization advantage to SFT.

Specifically, we constrain parameter updates in SFT to the directions identified by on-policy paradigms. For consistency with the theoretical analysis above, we represent the model parameters as a d-dimensional vector. Given the parameters before and after on-policy training $(\boldsymbol{\theta}_{\text{base}}, \boldsymbol{\theta}_{\text{on}})$, we obtain the sign vector $\boldsymbol{v} = \mathit{sign}(\boldsymbol{\theta}_{\text{on}} - \boldsymbol{\theta}_{\text{base}}) \in \{-1, 0, 1\}^d$ that determines the direction of the cumulative parameter update and constrains the gradient $\boldsymbol{g} \in \mathbb{R}^d$ of SFT according to $\boldsymbol{v}$ as follows:

$$\boldsymbol{g}_s = \mathbb{I}\left(\mathit{sign}(-\boldsymbol{g}) = \boldsymbol{v}\right) \odot \boldsymbol{g}, \tag{7}$$

where $\odot$ denotes the Hadamard product, $\mathit{sign}(\cdot)$ is the element-wise sign operator, and $\mathbb{I}(\cdot)$ represents the indicator function that retains gradient elements whose signs match $\boldsymbol{v}$ while discarding others. The constrained gradient $\boldsymbol{g}_s$ is then passed to the optimizer for gradient descent. By constraining the gradient at each training step, the parameter updates consistently follow the directions identified by the on-policy paradigm. In practice, considering that the regularization terms inherent to the optimizer may affect the imposed constraint, we further constrain the update direction after each optimizer step in Appendix A. For brevity in the subsequent discussion, we refer to this On-Policy direction-constrained SFT as OPSFT. By comparing OPSFT with vanilla SFT and the on-policy paradigm that provides the update direction, we can evaluate the ability of the on-policy update direction to transfer the generalization advantage of on-policy paradigms to SFT.

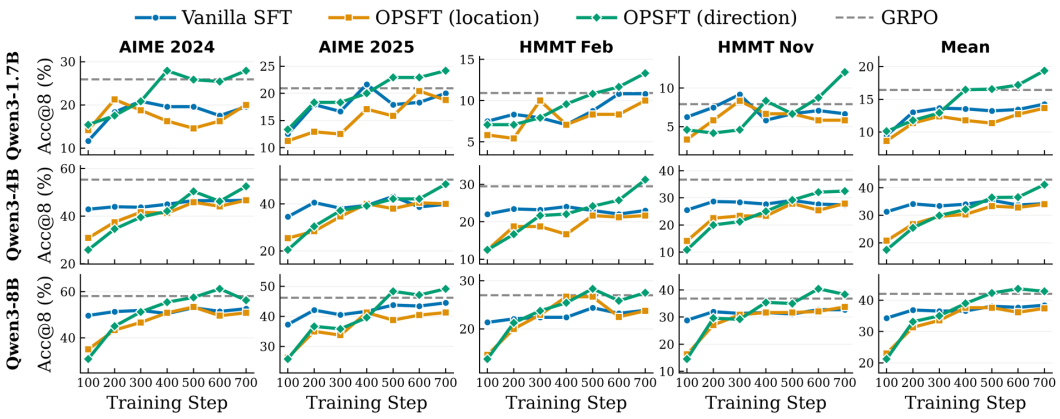

Figure 3: Performance comparison across different models and benchmarks during training. We compare vanilla SFT, OPSFT with location constraints, OPSFT with direction constraints, and the on-policy paradigm that provides update locations and directions (GRPO).

Table 1: Performance on out-of-domain benchmarks. We compare the base model, the final checkpoints of vanilla SFT, GRPO, and OPSFT trained on the DeepMath dataset.

Model	Method	IFEval	ARC	HaluEval	Hellaswag	Winogrande	PIQA	Mean
Qwen3-4B	Base	87.65	87.71	72.20	52.25	65.82	74.97	73.43
	SFT	86.45	87.88	73.40	**52.41**	66.30	**75.46**	73.65
	GRPO	88.61	87.88	73.70	52.25	66.22	75.03	73.95
	OPSFT	**88.97**	**87.97**	**73.80**	52.22	**66.46**	75.24	**74.11**
Qwen3-8B	Base	87.05	90.53	75.60	57.20	67.96	76.77	75.85
	SFT	87.53	90.61	72.70	**57.21**	68.19	**76.99**	75.54
	GRPO	**89.09**	90.53	76.00	57.14	67.80	76.66	76.20
	OPSFT	88.13	**90.70**	**77.10**	57.15	**68.35**	76.88	**76.39**

3.3 Experimental Results

Experimental Setups. We train Qwen3-1.7B/4B/8B (Yang et al., 2025) and DeepSeek-R1-Distill-Llama-3-8B (Guo et al., 2025) on the DeepMath (He et al., 2026), DAPO (Yu et al., 2026b), and Eurus-RL-Code (Cui et al., 2025) datasets with trajectories generated by Qwen3-30B-A3B. For baseline methods, we compare OPSFT with vanilla SFT (Gu et al., 2024), OPD (Lu & Lab, 2025), GRPO (He et al., 2026), and OPSFT only sharing the same update locations as on-policy paradigms. For evaluation, we consider both the in-domain (Zhang & Math-AI, 2024; 2025; Balunovic et al., 2026) and out-of-domain datasets (Zhou et al., 2023; Clark et al., 2018; Li et al., 2023; Zellers et al., 2019; Sakaguchi et al., 2021; Bisk et al., 2020). Due to space limitations, the experiments on OPD are presented in Appendix B. More details on training and evaluation are provided in Appendix A.

On-Policy Update Direction Underlies Generalization. In Figure 3, we provide the generalization performance of different methods across various benchmarks and model scales. OPSFT consistently achieves substantial performance gains over vanilla SFT, reaching performance comparable to GRPO that provides the on-policy update directions to OPSFT. The results indicate that the generalization advantage of on-policy paradigms can be transferred to SFT by sharing the same parameter update direction. This conclusion challenges the conventional finding that "SFT memorizes, while RL generalizes" (Chu et al., 2025). Once such an update direction is identified, even SFT can achieve strong generalization performance with its updates constrained to this direction. Furthermore, OPSFT constrained only by the parameter update locations performs even worse than vanilla SFT. This suggests that although prior studies (Mukherjee et al., 2026; Shen et al., 2026c; Yu et al., 2026a) have revealed the distinctive update location of on-policy paradigms, the location alone is insufficient to transfer their generalization advantages. In contrast, extending the constraint from update location $\{0, 1\}^d$ to direction $\{-1, 0, 1\}^d$ achieves substantial gains in generalization performance. These results further highlight the importance of parameter update directions in LLM post-training. Raw data of Figure 3 are provided in Appendix B.

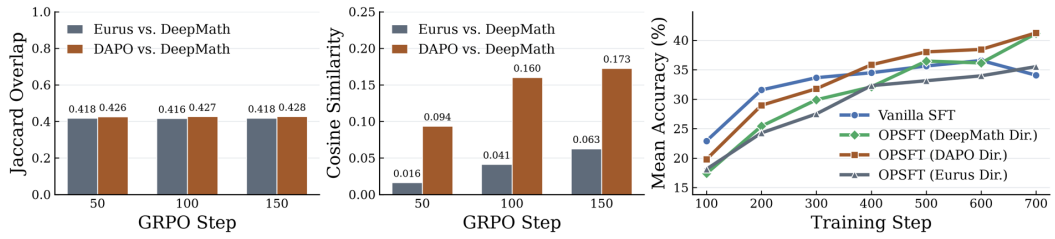

Figure 4: We compare parameter update locations *(left)* and directions *(middle)* between datasets from the same (DAPO vs. DeepMath) and different domains (Eurus vs. DeepMath). Then, we train OPSFT on DeepMath using on-policy update directions identified by different datasets *(right)*. Mean accuracy is computed across the AIME24/25 and HMMT-Feb/Nov benchmarks.

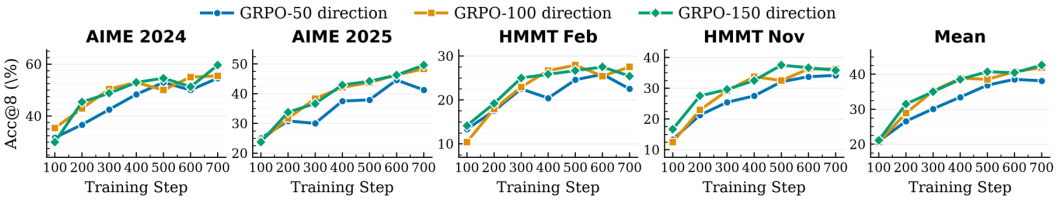

Figure 5: Accuracy evolution of OPSFT using the update direction identified at different GRPO training steps. Experiments are conducted using Qwen3-8B on the DeepMath dataset.

OOD Capability. We train the model on the DeepMath dataset and evaluate its performance on datasets from other domains. As shown in Table 1, the results across different benchmarks and model scales demonstrate that OPSFT consistently outperforms vanilla SFT and even slightly surpasses GRPO, which provides the on-policy update direction for OPSFT. This further highlights the effectiveness of on-policy update directions in improving the generalization performance of SFT, consistent with our findings in Figure 3. The on-policy update direction enables SFT to achieve strong performance on in-domain tasks while facilitating the transfer of newly acquired reasoning capabilities from the math domain to other domains.

Reusability Across Datasets and Domains. In Figure 4, we compare the on-policy update locations and directions across datasets within the same and different domains using Qwen3-4B. First, the update location overlaps are similar between datasets from the same and different domains (around 0.4), whereas the update directions exhibit substantially higher similarity within the same domain than across different domains. This suggests that update directions play an important role in capturing domain-specific capabilities. Second, even within the same domain, the update directions identified by different datasets can differ substantially, with the directions identified by DAPO and DeepMath exhibiting a cosine similarity of only around 0.2. This suggests the existence of multiple update directions that can support effective generalization within the same domain. Nevertheless, OPSFT trained on DeepMath using the direction identified by DAPO achieves strong generalization, even slightly outperforming OPSFT using the direction identified by DeepMath itself. This result indicates that on-policy update directions can be reused across datasets within the same domain. In contrast, such reusability does not extend across domains, as evidenced by the weak performance of OPSFT on the math domain when using the direction identified by Eurus, a code-domain dataset.

According to the above experiments, the on-policy update direction enables SFT to achieve strong generalization on in-domain and out-of-domain tasks with reusability across datasets within the same domain. These findings indicate that the on-policy update direction underlies strong generalization in LLM post-training and reveal its potential to improve the post-training pipelines.

4 Applications of On-Policy Update Direction

Motivated by the ability of the on-policy update direction to support strong generalization, we further leverage this direction to combine the generalization advantage of on-policy paradigms with the advantages of SFT. First, we perform a small number of GRPO steps to identify an update direction that supports strong generalization, and then apply OPSFT to achieve efficient training (Section 4.1). Second, given a model post-trained by an on-policy paradigm, we conduct OPSFT along its original

Table 2: Accuracy and training time of different methods across multiple models and benchmarks. All models are trained on the DeepMath dataset. The time of OPSFT includes both identifying the update direction with the on-policy paradigm and the subsequent OPSFT training.

Model	Method	Time	AIME24	AIME25	HMMT25-Feb	HMMT25-Nov	Mean
Qwen3-1.7B	GRPO	4.9h	16.67	17.08	7.92	6.67	12.09
	SFT	3.5h	19.48	17.92	8.96	6.25	13.15
	DFT	3.9h	18.25	17.72	**10.00**	6.67	13.16
	OPSFT	**2.3h**	**20.00**	**21.67**	9.58	**9.17**	**15.11**
Qwen3-4B	GRPO	16.5h	50.83	45.31	**26.35**	**33.33**	38.96
	SFT	8.1h	46.67	39.79	23.02	27.40	34.22
	DFT	8.6h	46.44	39.19	25.08	28.04	34.69
	OPSFT	**6.4h**	**52.92**	**50.42**	25.83	31.25	**40.11**
Qwen3-8B	GRPO	19.3h	54.17	**44.58**	**28.33**	34.17	40.31
	SFT	13.2h	52.60	44.48	23.85	32.71	38.41
	DFT	13.9h	54.23	43.92	24.50	33.23	38.97
	OPSFT	**8.9h**	**57.92**	42.92	26.67	**39.17**	**41.67**
DeepSeek-R1-LLaMA-8B	GRPO	18.2h	36.25	29.17	17.50	17.92	25.21
	SFT	13.6h	24.16	23.75	12.50	11.67	18.02
	DFT	14.1h	31.67	24.58	14.58	8.75	19.90
	OPSFT	**8.1h**	**37.91**	**30.41**	**17.92**	**20.00**	**26.56**

Table 3: Performance comparisons between SFT and OPSFT across multiple benchmarks using different models post-trained by GRPO. All models are trained on the DeepMath dataset.

Model	Method	AIME24	AIME25	HMMT25-Feb	HMMT25-Nov	Mean
Qwen3-1.7B	Base (post-trained)	16.67	17.08	7.92	6.67	12.09
	SFT	17.92	19.17	**10.83**	6.25	13.54
	OPSFT	**26.25**	**22.08**	10.42	**8.33**	**16.77**
Qwen3-4B	Base (post-trained)	50.83	**45.31**	26.35	33.33	38.96
	SFT	50.00	42.08	24.17	28.75	36.25
	OPSFT	**56.67**	44.17	**27.92**	**36.67**	**41.36**
Qwen3-8B	Base (post-trained)	54.17	44.58	**28.33**	34.17	40.31
	SFT	52.92	39.58	25.00	32.50	37.50
	OPSFT	**57.08**	**48.75**	27.91	**35.83**	**42.39**
DeepSeek-R1-LLaMA-8B	Base (post-trained)	36.25	29.17	17.50	**17.92**	25.21
	SFT	21.25	22.08	11.67	10.42	16.36
	OPSFT	**37.10**	**31.67**	**17.92**	**17.92**	**26.15**

update direction, thereby leveraging newly acquired high-quality trajectories to further improve the model without disrupting the capabilities learned during on-policy training (Section 4.2). Training and evaluation details are provided in Appendix A.

4.1 Training Efficiency Improvement

We train the model on the DeepMath dataset using GRPO (100 steps), SFT (700 steps), DFT (Wu et al., 2026) (700 steps), and OPSFT (50 GRPO steps for direction identification followed by 100 OPSFT steps). The results across different model scales, architectures, and benchmarks are reported in Table 2. First, compared with SFT and its improved version DFT, OPSFT achieves substantially better performance while requiring less training time by leveraging the on-policy update direction. For example, on DeepSeek-R1-Distill-LLaMA-8B, OPSFT achieves a mean accuracy of 26.56 in only 8.1h, substantially outperforming DFT that achieves 19.90 with 14.1h of training. Second, compared with GRPO, OPSFT reduces training time by more than 50% (*e.g.*, 8.9h vs. 19.3h on Qwen3-8B) while achieving better generalization performance (*e.g.*, 41.67 vs. 40.31 mean accuracy on Qwen3-8B). These results demonstrate that update directions identified during the early stages of on-policy training are already sufficient to substantially improve the generalization of SFT. More importantly, OPSFT is not limited by the on-policy paradigm used to identify the direction, as it can even outperform 100-step GRPO when using an update direction identified after only 50 GRPO

Table 4: Accuracy and training time of different methods on code tasks. Experiments are conducted with Qwen3-4B on the Eurus dataset.

Method	Time	HumanEval+	MBPP+	LCBv6	Mean
GRPO	20.3h	80.79	**69.04**	21.75	57.19
SFT	9.8h	80.50	63.90	18.51	54.30
OPSFT	**9.5h**	**84.00**	67.30	**22.90**	**58.07**

Table 5: Performance comparisons between SFT and OPSFT using Qwen3-4B post-trained by GRPO on the Eurus dataset.

Method	HumanEval+	MBPP+	LCBv6	Mean
Base	80.79	**69.04**	21.75	57.19
SFT	70.60	50.30	16.22	48.71
OPSFT	**85.80**	68.20	**26.15**	**60.05**

Table 6: Ablation studies of the parameter precisions. We report the accuracy and the proportion of updated parameters across different methods. Experiments are conducted with Qwen3-8B.

Precision	Method	Updated Param.	AIME24	AIME25	HMMT25-Feb	HMMT25-Nov	Mean
FP32	Base	0.000%	27.08	20.83	9.17	8.33	16.35
	GRPO	9.499%	58.13	46.15	26.98	36.77	42.01
BF16	SFT	2.702%	54.16	42.50	27.08	29.58	38.33
	OPSFT	0.408%	56.77	45.73	26.15	35.10	**40.94**
FP32	SFT	89.286%	52.60	44.48	23.85	32.71	38.41
	OPSFT	9.454%	56.25	49.17	27.50	38.33	**42.81**

steps. Moreover, we provide results on code tasks in Table 4. The results are consistent with those on math tasks, demonstrating the generality of our findings across different domains.

4.2 Improve Models After On-Policy Post-Training

In Table 3, we investigate whether OPSFT can further improve post-trained models by leveraging newly acquired high-quality trajectories. We first conduct 100 steps of GRPO and then continue training the resulting model on reasoning trajectories from the DeepMath dataset for 700 steps. Directly applying SFT to the post-trained model leads to performance degradation (*e.g.*, the mean accuracy of Qwen3-4B drops from 38.96 to 36.25). This suggests that even high-quality trajectories cannot be directly used to improve a post-trained model, as SFT may disrupt the capabilities acquired during on-policy post-training. In contrast, by constraining parameter updates to the on-policy update direction, OPSFT further improves the post-trained model (*e.g.*, increasing the mean accuracy of Qwen3-4B from 38.96 to 41.36). Furthermore, we provide results on code tasks in Table 5, which are consistent with those observed on math tasks. As high-quality trajectories can continuously accumulate in practice, OPSFT provides a promising way for incrementally improving post-trained models without restarting the entire post-training process from the base model.

4.3 Ablation Studies and Analysis

Effects of the Direction from Different Training Steps. We train OPSFT using update directions identified by the on-policy paradigm at different training stages. The performance across different benchmarks is reported in Figure 5. First, we observe that update directions identified at later stages of GRPO consistently lead to better generalization performance in OPSFT. Moreover, the marginal benefit of using later-stage update directions gradually diminishes as GRPO training progresses. Specifically, the improvement from the direction identified at 100 steps over that identified at 50 steps is larger than the improvement from the 150-step direction over the 100-step direction.

Effects of Parameter Precisions. In Table 6, we report the performance across different models and benchmarks under BF16 and FP32 parameter precision. Under BF16 precision, many small gradients result in parameter updates below the numerical precision of BF16 and are therefore rounded to zero, resulting in relatively sparse updates for vanilla SFT (only 2.702% of parameters are updated). Nevertheless, OPSFT induces substantially sparser updates than vanilla SFT by constraining parameter updates to the on-policy update direction. Despite its extremely sparse updates, OPSFT still significantly outperforms vanilla SFT under BF16 precision (40.94% vs. 38.33%), achieving a 24.59% improvement over the base model while updating only 0.408% of the parameters. Under FP32, the updates of both SFT variants become substantially denser. Vanilla SFT updates approximately 90% of the parameters, whereas OPSFT updates only 9.45%, comparable to GRPO. Moreover, FP32 pre-

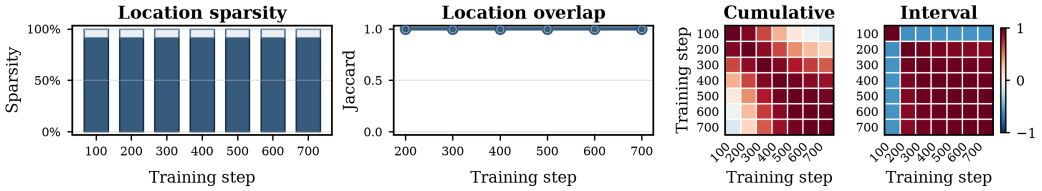

Figure 6: Update behaviors of OPSFT when training Qwen3-8B on the DeepMath dataset. We report the location sparsity, Jaccard overlap of update locations, and matrix-level cosine similarity of cumulative and interval updates across different training steps.

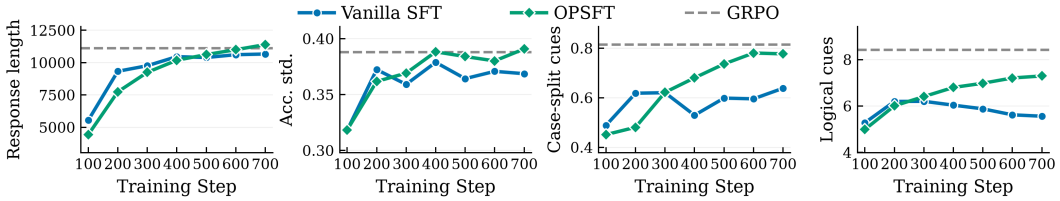

Figure 7: Reasoning behavior comparisons among different methods. We report response length, accuracy standard deviation, the number of case-split cues per 1k words, and the number of logical cues per 1k words throughout training with Qwen3-8B on the DeepMath dataset.

cision leads to better performance than BF16 precision, with OPSFT even outperforming GRPO, which is used to identify its update direction (42.81% vs. 42.01%).

Optimization Behavior of OPSFT. In Figure 6, we analyze the evolution of update locations and matrix-level directions in OPSFT throughout training. For update locations, OPSFT updates are constrained to a subset of parameters by the on-policy update direction, resulting in dense updates within this subset. For update directions, we find that OPSFT exhibits substantial differences between the early and later stages of training. This behavior suggests that the parameter-level direction (*i.e.*, sign) constraint of OPSFT redirects the SFT optimization trajectory toward a matrix-level direction conducive to generalization, after which OPSFT can continue optimizing along this direction.

Reasoning Behavior. In Figure 7, we compare vanilla SFT and OPSFT with GRPO in terms of reasoning behaviors. First, we find that vanilla SFT, OPSFT, and GRPO exhibit similar response lengths. Moreover, compared with vanilla SFT, OPSFT exhibits a higher standard deviation in accuracy across multiple responses, approaching that of GRPO. This suggests that incorporating the on-policy update direction promotes greater uncertainty in the model's reasoning outcomes. To investigate these behaviors at a finer granularity, we analyze the frequencies of case-splitting cues (*e.g.*, `if`, `otherwise`, `consider a case`) and logical cues (*e.g.*, `therefore`, `because`) throughout training. The results indicate that the direction constraint makes the reasoning behaviors of OPSFT more closely resemble those of on-policy paradigms, rather than simply inducing longer reasoning.

5 Conclusion

In this paper, we explore whether there exists a parameter update behavior in on-policy paradigms that can be leveraged to transfer their generalization advantage to SFT. First, our theoretical and experimental analyses reveal that on-policy paradigms tend to continuously adjust their parameter update directions during training. Inspired by this phenomenon, we propose OPSFT, which constrains parameter updates of SFT to the direction identified by the on-policy paradigm. Extensive experiments show that OPSFT achieves performance comparable to that of the corresponding on-policy paradigm, demonstrating that the on-policy update direction can serve as an effective optimization principle for improving the generalization of SFT. Based on this finding, OPSFT enables integrating the strong generalization of on-policy paradigms with the advantages of SFT, including efficient training and the ability to leverage high-quality trajectories. For future work, we will investigate how different components of on-policy paradigms contribute to identifying generalization-friendly directions, and how the property of reasoning trajectories affects model generalization through OPSFT. For limitation discussions, please refer to Appendix C.2.

AI USE STATEMENT

In this work, we used generative AI tools for assist with translation. We have not used generative AI tools for interpret results, propose or refine hypotheses, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for identify relevant literature. We have reviewed all AI-assisted work. We have ensured the authenticity of the relevant literature provided by AI and the correctness of the citations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We assert that the methodology and applications of this study do not involve human subjects, generate harmful insights, or engage in applications that could lead to discrimination, bias, or social harm. All datasets used in this research are publicly available and widely used in academic studies. We commit to adhering to all applicable data privacy and intellectual property laws and regulations. We confirm that we have read and understood the ICLR Code of Ethics and pledge to comply with its guidelines in all aspects of our research, paper submission, and subsequent interactions

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we provide comprehensive details on datasets, pre-trained models in Section 3.3. Details of training and evaluation strategies computing can be found in Appendix A.1. Source code has been submitted as supplementary materials.

REFERENCES

Abbas Abdolmaleki, Bilal Piot, Bobak Shahriari, Jost Springenberg, Tim Hertweck, Michael Bloesch, Rishabh Joshi, Thomas Lampe, Junhyuk Oh, Nicolas Heess, et al. Learning from negative feedback, or positive feedback or both. In *International Conference on Learning Representations*, volume 2025, pp. 7414–7436, 2025.

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

Alan Ansell, Ivan Vulić, Hannah Sterz, Anna Korhonen, and Edoardo M Ponti. Scaling sparse fine-tuning to large language models. *arXiv preprint arXiv:2401.16405*, 2024.

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

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Han Chen, Garvesh Raskutti, and Ming Yuan. Non-convex projected gradient descent for generalized low-rank tensor regression. *Journal of Machine Learning Research*, 20(5):1–37, 2019.

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.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018. URL https://api.semanticscholar.org/CorpusID:3922816.

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

Leo Gao, Achyuta Rajaram, Jacob Coxon, Soham V Govande, Bowen Baker, and Dan Mossing. Weight-sparse transformers have interpretable circuits. *arXiv preprint arXiv:2511.13653*, 2025.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

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.

Andi Han, Jiaxiang Li, Wei Huang, Mingyi Hong, Akiko Takeda, Pratik Kumar Jawanpuria, and Bamdev Mishra. Sltrain: a sparse plus low rank approach for parameter and memory efficient pretraining. *Advances in Neural Information Processing Systems*, 37:118267–118295, 2024.

Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Sensitivity-aware visual parameter-efficient fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11825–11835, 2023.

Haoze He, Juncheng B Li, Xuan Jiang, and Heather Miller. Smt: Fine-tuning large language models with sparse matrices. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 138306–138322, 2026.

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

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

Feng Ju, Zeyu Qin, Rui Min, Zhitao He, Lingpeng Kong, and Yi R Fung. Reasoning path divergence: A new metric and curation strategy to unlock llm diverse thinking. *arXiv preprint arXiv:2510.26122*, 2025.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, J. Nie, and Ji rong Wen. Halueval: A large-scale hallucination evaluation benchmark for large language models. *ArXiv*, abs/2305.11747, 2023.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models. *Advances in Neural Information Processing Systems*, 38:132119–132138, 2026.

Phuc Minh Nguyen, Chinh D La, Duy MH Nguyen, Nitesh V Chawla, Binh T Nguyen, and Khoa D Doan. The reasoning boundary paradox: How reinforcement learning constrains language models. *arXiv preprint arXiv:2510.02230*, 2025.

Chongli Qin and Jost Tobias Springenberg. Supervised fine tuning on curated data is reinforcement learning (and can be improved). *arXiv preprint arXiv:2507.12856*, 2025.

Qihan Ren, Peng Wang, Ruikun Cai, Shuai Shao, Dadi Guo, Yuejin Xie, Yafu Li, Quanshi Zhang, Xia Hu, Jing Shao, et al. Rethinking generalization in reasoning sft: A conditional analysis on optimization, data, and model capability. *arXiv preprint arXiv:2604.06628*, 2026.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

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.

Shufan Shen, Junshu Sun, Xiangyang Ji, Qingming Huang, and Shuhui Wang. Expanding sparse tuning for low memory usage. *Advances in Neural Information Processing Systems*, 37:76616–76642, 2024.

Shufan Shen, Zhaobo Qi, Junshu Sun, Qingming Huang, Qi Tian, and Shuhui Wang. Enhancing pre-trained representation classifiability can boost its interpretability. In *International Conference on Learning Representations*, volume 2025, pp. 88422–88446, 2025.

Shufan Shen, Junshu Sun, Qingming Huang, and Shuhui Wang. Vl-sae: Interpreting and enhancing vision-language alignment with a unified concept set. *Advances in Neural Information Processing Systems*, 38:45235–45265, 2026a.

Shufan Shen, Junshu Sun, Shuhui Wang, and Qingming Huang. Kernelized sparse fine-tuning with bi-level parameter competition for vision models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026b.

Zhennan Shen, Yanshu Li, Qingyu Yin, Chak Tou Leong, Zhilin Wang, Yanxu Chen, Rongduo Han, Sunbowen Lee, and Yi R Fung. On the geometry of on-policy distillation. *arXiv preprint arXiv:2606.07082*, 2026c.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why rlvr may or may not escape its origin. *arXiv preprint arXiv:2507.14843*, 2025.

Yongliang Wu, Yizhou Zhou, Ziheng Zhou, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, et al. On the generalization of sft: A reinforcement learning perspective with reward rectification. In *International Conference on Learning Representations*, volume 2026, pp. 27550–27571, 2026.

LLM Xiaomi, Bingquan Xia, Bowen Shen, Dawei Zhu, Di Zhang, Gang Wang, Hailin Zhang, Huaqiu Liu, Jiebao Xiao, Jinhao Dong, et al. Mimo: Unlocking the reasoning potential of language model–from pretraining to posttraining. *arXiv preprint arXiv:2505.07608*, 2025.

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

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

Yuming Yang, Mingyoung Lai, Wanxu Zhao, Xiaoran Fan, Zhiheng Xi, Mingqi Wu, Chiyue Huang, Jun Zhao, Haijun Lv, Jian Tong, et al. Which reasoning trajectories teach students to reason better? a simple metric of informative alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 42123–42150, 2026b.

Guo Yu, Wenlin Liu, Yulan Hu, Hao-Xuan Ma, Jun-Peng Jiang, and Han-Jia Ye. Dense supervision, sparse updates: On the sparsity and geometry of on-policy distillation. *arXiv preprint arXiv:2606.13657*, 2026a.

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. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026b.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Chong Zhang, Yue Deng, Xiang Lin, Bin Wang, Dianwen Ng, Hai Ye, Xingxuan Li, Yao Xiao, Zhanfeng Mo, Qi Zhang, et al. 100 days after deepseek-r1: A survey on replication studies and more directions for reasoning language models. *arXiv preprint arXiv:2505.00551*, 2025.

Jiaying Zhang, Lei Shi, Jiguo Li, Jun Xu, Jiuchong Gao, Jinghua Hao, and Renqing He. Geora: Geometry-aware low-rank adaptation for rlvr. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24207–24221, 2026.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Zhi Zhang, Qizhe Zhang, Zijun Gao, Renrui Zhang, Ekaterina Shutova, Shiji Zhou, and Shanghang Zhang. Gradient-based parameter selection for efficient fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28566–28577, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

Shiwan Zhao, Zhihu Wang, Xuyang Zhao, Jiaming Zhou, Caiyue Xu, Chenfei Liu, Liting Zhang, Yuhang Jia, Yanzhe Zhang, Hualong Yu, et al. Large language model post-training: A unified view of off-policy and on-policy learning. *arXiv preprint arXiv:2604.07941*, 2026.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *ArXiv*, abs/2311.07911, 2023.

Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Zhizhou Sha, Jinwon Lee, et al. The path not taken: Rlvr provably learns off the principals. *arXiv preprint arXiv:2511.08567*, 2025.

Table A7: Training hyperparameters of GRPO in math tasks.

Hyper-parameter	Value
Train Batch Size	128
Micro Batch Size	128
Rollout n	8
Maximum Prompt Length	2048
Maximum Response Length	16,384
Temperature	1.0
Top-p	1.0
LR	1×10^{-6}
Optimization Steps	150
KL Coefficient	0.0

Table A8: Training hyperparameters of GRPO in code tasks.

Hyper-parameter	Value
Train Batch Size	128
Micro Batch Size	128
Rollout n	8
Maximum Prompt Length	2048
Maximum Response Length	8192
Temperature	1.0
Top-p	1.0
LR	1×10^{-6}
Optimization Steps	150
KL Coefficient	0.0

A Implementation Details

A.1 Training Datasets

On-policy paradigms. We filter the DeepMath (He et al., 2026) dataset to select 57K samples with a difficulty level greater than or equal to 6 to form the math RL data, and use Eurus-RL-Code (Cui et al., 2025) as the code RL data, which consists of 25K samples.

SFT. For math, we construct SFT trajectories from the DeepMath (He et al., 2026) training prompts using `Qwen3-30B-A3B-Instruct-2507`. We decode one completion per prompt with temperature 0.6, top-$p = 0.95$, and a maximum of $16{,}384$ newly generated tokens (with an $18{,}432$-token model context limit). A trajectory is retained only when the teacher response is judged correct against the instance ground truth by the mathematical answer verifier (`math_verify`); malformed, empty, or incorrect generations are discarded. For code, we use the same model and protocol following the math data-generation setting with Eurus (Cui et al., 2025). We sample one completion per prompt and retain it only if the evaluator successfully executes the generated program against the reference tests. We require every response to contain a Python code block and store execution metadata together with the prompt, response, and ground truth. After verification, the math dataset contains $44{,}810$ training and 914 examples. The final Eurus code dataset contains $9{,}585$ training examples.

A.2 Training Settings

On-policy paradigms. For on-policy paradigms in the main paper, we apply Group Relative Policy Optimization (GRPO) (Shao et al., 2024). A reward of 1.0 is given when the final answer is correct in math reasoning or when all unit tests pass in code generation; otherwise, the reward is 0.0. The training hyperparameters in math and code training are put in Table A7 and Table A8, respectively.

SFT. For both math and code experiments, we perform SFT and OPSFT using the cross-entropy objective. All experiments use FP32 training with FSDP2 sharding over 8 NVIDIA H20 GPUs. We use a global batch size of 64, a maximum sequence length of 16,384 tokens, and right truncation for overlength examples. Gradient checkpointing is enabled. Optimization is performed with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$), weight decay of 0.01, and gradient-norm clipping at 1.0. We use a constant learning-rate schedule with no warm-up steps. Unless otherwise specified, the learning rate is 1×10^{-7}, and the random seed is fixed to 42. Models are trained for 700 optimizer steps, and checkpoints are saved every 100 steps. For the experiment in Table 4.1 and Table 4, we retain all settings above and use a learning rate of 1×10^{-6}. Unless otherwise specified, all experiments use FP32 parameter precision by default. Moreover, OPSFT enforces the direction constraint at both the gradient and parameter levels. At each optimization step, gradients that induce updates opposite to the cumulative GRPO update direction are masked out. After the optimizer step, we further restore any parameter whose cumulative displacement from the initial weights has moved in the opposite direction. This two-level constraint ensures that neither AdamW's weight decay nor its momentum-based updates can violate the prescribed on-policy update direction.

A.3 Evaluation Details

For the evaluation of math reasoning, we select four competition-level benchmarks: AIME24 (Zhang & Math-AI, 2024), AIME25 (Zhang & Math-AI, 2025), HMMT25 (February) (Balunovic et al., 2026), and HMMT25 (November) (Balunovic et al., 2026). For the evaluation of code generation, we select three test sets: HumanEval+, MBPP+ (Liu et al., 2023), and LiveCodeBench (v6 only, February 2025∼May 2025) (Jain et al., 2025). In all evaluations, we set the temperature to 1.0, top-p to 1.0, and the maximum generation length to 16,384. On each math reasoning benchmark, we sample 8 solutions for each problem, whereas on each code generation benchmark, we sample 4 solutions per problem. We then report the average accuracy of each model on each benchmark. We adopt Math-Verify as a verifier to validate answer correctness for math reasoning benchmarks.

A.4 Pseudo Code of OPSFT

```
# Inputs:
#   theta_0[name]: parameter value before SFT
#   M[name]: exact-support boolean mask
#   D[name]: cumulative OPD/GRPO parameter update
#   batch: one SFT minibatch
#   optimizer: e.g., AdamW
optimizer.zero_grad()
# Compute the standard SFT gradient on one minibatch.
loss = compute_sft_loss(model, batch)
loss.backward()
# Stage 1: constrain the instantaneous SFT gradient.
for name, parameter in model.named_parameters():
    if parameter.grad is None:
        continue
    gradient = parameter.grad
    # Location: only OPD exact-support coordinates receive gradients.
    gradient[~M[name]] = 0
    # Direction: -gradient is the first-order parameter movement.
    # Block gradients whose induced movement opposes D[name].
    opposite_direction = (
        M[name]
        & (gradient != 0)
        & (D[name] != 0)
        & (gradient * D[name] > 0)
    )
    gradient[opposite_direction] = 0
clip_gradient_norm(model.parameters(), max_norm)
optimizer.step()
# Stage 2: hard projection after the optimizer update.
with no_gradient():
    for name, parameter in model.named_parameters():
        # Enforce exact support.
        parameter[~M[name]] = theta_0[name][~M[name]]
        # Enforce agreement between cumulative SFT displacement and
    D[name].
        displacement = parameter - theta_0[name]
        wrong_direction = (
            M[name]
            & (D[name] != 0)
            & (displacement * D[name] < 0)
        )
        parameter[wrong_direction] = theta_0[name][wrong_direction]
```

Listing 1: Pseudo code of OPSFT. M is the parameter update location extracted from on-policy paradigms, and D is the element-wise parameter update direction.

Table A9: Accuracy of different methods on Qwen3-1.7B using the DeepMath dataset.

Method	AIME24	AIME25	HMMT25-Feb	HMMT25-Nov	Mean
Base	13.54	11.35	4.79	4.68	8.59
OPD	24.17	20.93	11.67	10.21	16.75
Vanilla SFT	18.90	18.60	10.40	7.40	13.83
OPSFT (location)	18.90	18.90	9.40	6.70	13.48
OPSFT (direction)	23.02	23.95	10.93	8.95	16.71

Table A10: Training trajectories for Qwen3-1.7B. Entries report Acc@8 (%).

Method	Step	AIME24	AIME25	HMMT Feb	HMMT Nov	Mean
Vanilla SFT	100	11.67	12.50	7.50	6.25	9.48
	200	18.33	17.92	8.33	7.50	13.02
	300	20.83	16.67	7.92	9.17	13.65
	400	19.58	21.67	7.08	5.83	13.54
	500	19.58	17.92	8.75	6.67	13.23
	600	17.50	18.33	10.83	7.08	13.44
	700	19.58	20.00	10.83	6.67	14.27
OPSFT (location)	100	14.17	11.25	5.83	3.33	8.65
	200	21.25	12.92	5.42	5.83	11.35
	300	18.75	12.50	10.00	8.33	12.40
	400	16.25	17.08	7.08	6.67	11.77
	500	14.58	15.83	8.33	6.67	11.35
	600	16.25	20.42	8.33	5.83	12.71
	700	20.00	18.75	10.00	5.83	13.65
OPSFT (direction)	100	15.42	13.33	7.08	4.58	10.10
	200	17.50	18.33	7.08	4.17	11.77
	300	20.83	18.33	7.92	4.58	12.92
	400	27.92	20.00	9.58	8.33	16.46
	500	25.83	22.92	10.83	6.67	16.56
	600	25.42	22.92	11.67	8.75	17.19
	700	27.92	24.17	13.33	12.08	19.38

B Additional Experiments

Experiments on other on-policy training paradigms. In Table A9, we conduct experiments on the OPD update direction following the settings in Figure 3. For OPD, we use Qwen3-30B-A3B as the teacher model and Qwen3-1.7B as the student model, and train the student for 150 steps. All other training settings follow G-OPD (Yang et al., 2026a). We observe consistent results with our previous experiments. Compared with vanilla SFT, OPSFT with both location and direction constraints on the OPD update direction achieves substantial performance improvements (16.71% vs. 13.83% of mean accuracy), which are similar to the OPD performance (16.71% vs. 16.75% of mean accuracy). These results demonstrate the generality of our findings across different on-policy paradigms.

Raw data of Figure 3. We provide the raw data of Qwen3-1.7B, Qwen3-4B, and Qwen3-8B in Table A10, Table A11, and Table A12, respectively.

Ablations of Update Location and Direction. In Table A13, we provide ablation studies of the update location and direction identified by the on-policy paradigm. First, compared with randomly constraining the parameter update locations, selecting the trainable parameters following the on-policy paradigm achieves better performance. This result highlights the ability of the on-policy paradigm to identify task-relevant parameters. Building on this observation, imposing random direction constraints leads to a performance drop, whereas using the update directions identified by the on-policy paradigm achieves the best performance. This further demonstrates that the performance gains of OPSFT arise from updating parameters along directions that support generalization, rather than merely from the regularization effect of gradient masking.

Table A11: Training trajectories for Qwen3-4B. Entries report Acc@8 (%).

Method	Step	AIME24	AIME25	HMMT Feb	HMMT Nov	Mean
Vanilla SFT	100	42.92	34.48	22.08	25.52	31.25
	200	43.96	40.52	23.44	28.65	34.14
	300	43.75	38.12	23.23	28.44	33.38
	400	45.00	39.27	24.06	27.71	34.01
	500	46.56	43.12	23.02	29.06	35.44
	600	46.46	38.65	22.08	27.71	33.73
	700	46.67	39.79	23.02	27.40	34.22
OPSFT (location)	100	30.83	25.42	12.50	14.17	20.73
	200	37.50	28.33	18.75	22.50	26.77
	300	41.67	34.58	18.75	23.33	29.58
	400	41.25	40.00	16.67	23.33	30.31
	500	45.83	37.92	21.67	27.92	33.34
	600	44.17	40.42	21.25	25.42	32.81
	700	46.67	40.00	21.67	27.92	34.06
OPSFT (location)	100	25.83	20.42	12.50	10.83	17.39
	200	34.58	30.42	16.67	20.00	25.42
	300	39.58	37.08	21.67	21.25	29.89
	400	42.08	39.17	22.08	25.00	32.08
	500	50.42	42.08	24.17	29.17	36.46
	600	46.25	42.08	25.80	32.08	36.55
	700	52.50	48.33	31.25	32.50	41.14

Table A12: Training trajectories for Qwen3-8B. Entries report Acc@8 (%).

Method	Step	AIME24	AIME25	HMMT Feb	HMMT Nov	Mean
Vanilla SFT	100	49.69	37.29	21.35	28.85	34.30
	200	51.35	42.08	22.08	31.98	36.87
	300	51.88	40.52	22.40	31.35	36.54
	400	50.52	41.67	22.40	31.67	36.56
	500	53.02	43.75	24.38	31.15	38.08
	600	51.46	43.44	23.23	32.71	37.71
	700	52.60	44.48	23.85	32.71	38.41
OPSFT (location)	100	35.00	26.25	14.58	16.25	23.02
	200	43.33	35.00	20.00	27.08	31.35
	300	46.67	33.75	22.92	30.83	33.54
	400	50.83	41.25	26.67	31.67	37.61
	500	53.33	38.75	26.67	31.67	37.61
	600	49.58	40.42	22.50	32.08	36.14
	700	50.83	41.25	23.75	33.75	37.39
OPSFT (direction)	100	30.83	25.83	13.75	14.58	21.25
	200	45.00	36.67	21.25	29.58	33.12
	300	51.25	35.83	23.75	29.17	35.00
	400	55.42	39.58	25.42	35.42	38.96
	500	57.50	48.33	28.33	35.00	42.29
	600	61.25	47.08	25.83	40.42	43.64
	700	56.25	49.17	27.50	38.33	42.81

Principle components of weight matrices. In Figure A8, we analyze the weight updates of different methods from a spectral perspective. By comparing the updates across different layers, we find that constraining SFT updates to the on-policy update direction at the weight level also makes its spectral properties more similar to those of the on-policy paradigm. First, the singular values of the update matrices in OPSFT are more evenly distributed, resembling those of GRPO, whereas the updates of vanilla SFT are concentrated in the top few singular values. This observation is further reflected by the substantially higher stable ranks of GRPO and OPSFT. Moreover, compared with SFT, which induces relatively large rotations of the principal singular vectors of the weights, OPSFT and GRPO

Table A13: We report the performance of SFT with (i) random update location constraints, (ii) on-policy update location constraints, (iii) on-policy update location constraints with random update sign constraints, and (iv) the complete on-policy update direction constraints used in OPSFT. Experiments are conducted with Qwen3-4B on the DeepMath dataset. GRPO is selected as the on-policy paradigm that provides update location and direction constraints.

Location constraint	Direction constraint	AIME24	AIME25	HMMT25-Feb	HMMT25-Nov	Mean
Random	-	41.67	34.17	24.17	23.33	30.84
On-policy	-	46.67	40.00	21.67	27.92	34.07
On-policy	Random	37.50	31.67	18.33	23.33	27.71
On-policy	On-policy	52.50	48.33	31.25	32.50	41.15

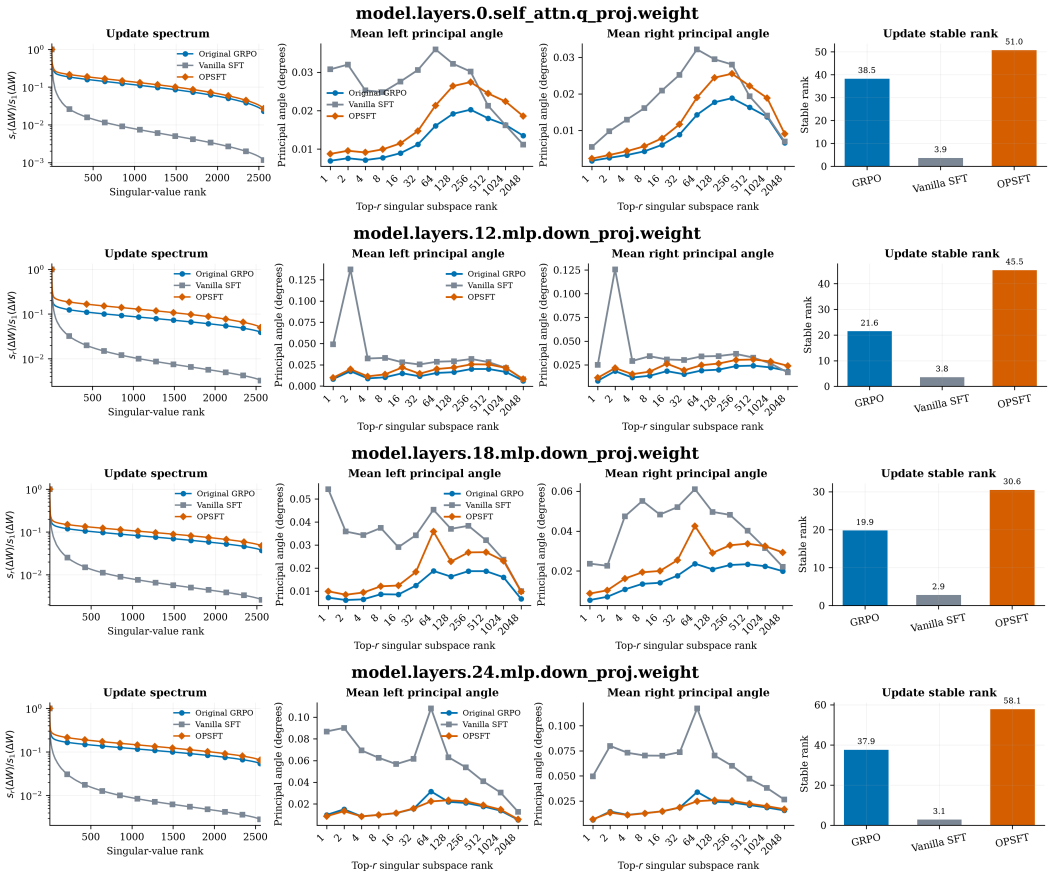

Figure A8: We provide the principal values of parameter updates, the rotation angle of parameter matrices' principal singular vectors, and the stable rank of parameter updates across different layers and methods. Experiments are conducted using Qwen3-4B trained on the DeepMath dataset.

exhibit substantially smaller rotations. The spectral similarities between OPSFT and GRPO suggest that these properties may arise as a consequence of the on-policy update direction, rather than being intrinsic to the on-policy training paradigm itself. Once the update direction is identified, even SFT can induce similar spectral behaviors.

C Additional Discussions

C.1 Discussions with Gradient Projection Methods

OPSFT is closely related to existing projected gradient descent methods, which are typically proposed to reduce training costs (Chen et al., 2019; Zhao et al., 2024). From an implementation perspective, both OPSFT and these methods constrain model updates. The key difference is that we characterize the update direction through on-policy paradigms, while other methods determine the projection subspace by decomposing the SFT gradient matrix. More importantly, through gradient projection, we reveal that this direction is a key factor in transferring the strong generalization of on-policy paradigms to SFT. This finding extends the application of gradient projection beyond efficiency, demonstrating its potential for improving the generalization performance on complex reasoning tasks.

C.2 Limitations

Although the proposed OPSFT demonstrates the critical role of the on-policy update direction in the generalization performance of LLM post-training, our experiments are conducted with high-quality reasoning trajectories, *i.e.*, trajectories generated by a teacher model with substantially stronger performance than the model being post-trained on the corresponding tasks. In practical scenarios, since on-policy paradigms continuously train on trajectories generated by the current policy, their trajectories may be better than those collected by SFT in terms of diversity, correctness, and their suitability for the target model (Ju et al., 2025). However, existing studies on measuring the quality and suitability of reasoning trajectories remain limited (Yang et al., 2026b), preventing us from explicitly accounting for this factor in this paper. In future work, we will investigate how the properties of reasoning trajectories affect model generalization and further examine their impact on the resulting update direction and the performance of OPSFT.