

A APPENDIX

A.1 USAGE OF LLM

We employ LLM primarily as writing assistants to refine and polish the manuscript. Their usage was limited to improving clarity, coherence, and presentation, while all conceptual and experimental contributions remain original.

A.2 DETAILED DERIVATION OF EQUATION (5)

We start from the SFT gradient in Equation (2):

$$\nabla_{\theta} \mathcal{L}_{\text{SFT}}(\theta) = \mathbb{E}_{(x, y^*) \sim \mathcal{D}} [-\nabla_{\theta} \log \pi_{\theta}(y^* | x)]. \quad (1)$$

For each query x , the expectation over expert demonstrations (x, y^*) can be written explicitly as a summation over all possible outputs y :

$$\mathbb{E}_{(x, y^*) \sim \mathcal{D}} [-\nabla_{\theta} \log \pi_{\theta}(y^* | x)] = \mathbb{E}_{x \sim \mathcal{D}_x} \sum_y \mathbf{1}[y = y^*] [-\nabla_{\theta} \log \pi_{\theta}(y | x)]. \quad (2)$$

We insert the model distribution $\pi_{\theta}(y | x)$, which allows us to express the summation in terms of importance weights:

$$\mathbb{E}_{x \sim \mathcal{D}_x} \sum_y \pi_{\theta}(y | x) \cdot \frac{\mathbf{1}[y = y^*]}{\pi_{\theta}(y | x)} [-\nabla_{\theta} \log \pi_{\theta}(y | x)]. \quad (3)$$

Here, the term $\frac{\mathbf{1}[y = y^*]}{\pi_{\theta}(y | x)}$ serves as an importance weight comparing the expert (Dirac delta) distribution with the model’s distribution.

The summation over y can now be rewritten as an expectation under the policy distribution $y \sim \pi_{\theta}(\cdot | x)$:

$$\mathbb{E}_{x \sim \mathcal{D}_x} \mathbb{E}_{y \sim \pi_{\theta}(\cdot | x)} \left[\frac{\mathbf{1}[y = y^*]}{\pi_{\theta}(y | x)} (-\nabla_{\theta} \log \pi_{\theta}(y | x)) \right]. \quad (4)$$

Thus, we obtain Equation (5):

$$\mathbb{E}_{(x, y^*) \sim \mathcal{D}} [-\nabla_{\theta} \log \pi_{\theta}(y^* | x)] = \mathbb{E}_{x \sim \mathcal{D}_x} \mathbb{E}_{y \sim \pi_{\theta}(\cdot | x)} \left[\frac{\mathbf{1}[y = y^*]}{\pi_{\theta}(y | x)} (-\nabla_{\theta} \log \pi_{\theta}(y | x)) \right]. \quad (5)$$

This derivation shows that the SFT gradient can be expressed as an on-policy policy gradient with importance sampling, where the expert demonstration distribution is reweighted relative to the model distribution.

A.3 DISCUSSIONS AND INSIGHTS

Gradient Analysis of DFT. We now analyze the gradient induced by the DFT surrogate loss. Recall the sequence-level definition:

$$\mathcal{L}_{\text{DFT}}(\theta) = -\text{sg}(\pi_{\theta}(y^* | x)) \log \pi_{\theta}(y^* | x), \quad (10)$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator. Since the stop-gradient blocks backpropagation, the detached probability $\text{sg}(\pi_{\theta}(y^* | x))$ is treated as a constant during differentiation. Consequently, the gradient becomes

$$\nabla_{\theta} \mathcal{L}_{\text{DFT}} = -\text{sg}(\pi_{\theta}(y^* | x)) \frac{1}{\pi_{\theta}(y^* | x)} \nabla_{\theta} \pi_{\theta}(y^* | x) \quad (11)$$

$$= -\left(\frac{\text{sg}(\pi_{\theta}(y^* | x))}{\pi_{\theta}(y^* | x)} \right) \nabla_{\theta} \pi_{\theta}(y^* | x). \quad (12)$$

Since $\text{sg}(\pi_\theta(y^* | x))$ equals $\pi_\theta(y^* | x)$ in the forward pass, the prefactor is numerically equal to 1. Therefore,

$$\nabla_\theta \mathcal{L}_{\text{DFT}} = -\nabla_\theta \pi_\theta(y^* | x). \quad (13)$$

This shows that DFT is mathematically equivalent to directly maximizing the model probability of the target token, rather than its log-probability as in cross-entropy.

For cross-entropy, the loss is

$$\mathcal{L}_{\text{CE}}(\theta) = -\log \pi_\theta(y^* | x),$$

yielding gradient

$$\nabla_\theta \mathcal{L}_{\text{CE}} = -\frac{1}{\pi_\theta(y^* | x)} \nabla_\theta \pi_\theta(y^* | x).$$

Thus both CE and DFT share the same gradient direction but differ in scaling: CE amplifies updates for low-probability targets (factor $1/\pi$), while DFT applies a uniform factor 1. As a result, DFT avoids the instability caused by excessively large gradients on unlikely expert tokens, providing more conservative and stable updates.

From the reinforcement learning perspective, the reward of DFT becomes uniformly 1 across all expert trajectories, equivalent to a verification-style objective that treats all correct references equally. From the optimization perspective, DFT trades off aggressive fitting of rare tokens for better stability and calibration. Practically, this explains why DFT often yields smoother training and stronger generalization, while maintaining alignment with the pre-training distribution.

Learning from Noisy Data. DFT offers a simple yet effective approach, prompting us to reflect on why it might actually work. One intuitive explanation lies in its ability to learn from noisy data (Freund, 2009). Sasaki & Yamashina (2020) propose an imitation learning algorithm for learning from noisy demonstrations, based on the core idea of avoiding the fitting of data that is difficult to model, as such data is likely to originate from suboptimal behaviors, i.e., noise. Their method introduces a weighted behavioral cloning objective, where the weights are derived from a previously trained policy’s confidence in each action. Similarly, the weighting mechanism in DFT shares the same intuition, but instead of relying on a fixed old policy model to compute confidence scores, it uses a single policy model to perform confidence-based weighting on-the-fly during training.

A.4 IMPLEMENTATION DETAILS

A.4.1 MAIN EXPERIMENT - MATHEMATICAL REASONING TASK

To efficiently manage computational resources, We randomly sample 100,000 instances from the NuminaMath-CoT dataset (Li et al., 2024) for training. We conduct experiments using multiple models, including Qwen2.5-Math-1.5B, Qwen2.5-Math-7B (Qwen Team et al., 2024a), LLaMA-3.2-3B, LLaMA-3.1-8B (Dubey et al., 2024), and DeepSeekMath-7B (Shao et al., 2024).

Our implementation builds upon the verl framework (Sheng et al., 2025), using recommended SFT hyper-parameters. Specifically, we employ the AdamW optimizer with learning rates of 5×10^{-5} for all models except the LLaMA-3.1-8B-Base, for which we adopt a lower learning rate of 2×10^{-5} . We set the mini-batch size to 256 and the maximum input length to 2048 tokens. The learning rate follows a cosine decay schedule with a warm-up ratio of 0.1. We evaluate on benchmarks including Math500 (Hendrycks et al., 2021), Minerva Math (Lewkowycz et al., 2022), Olympiad Bench (He et al., 2024), AIME 2024 (American Institute of Mathematics, 2024), and AMC 2023 (Mathematical Association of America, 2023) through the official Qwen2.5-Math evaluation pipeline (Qwen Team et al., 2024a). Each model uses the default chat template and Chain-of-Thought (CoT) prompting to stimulate step-by-step reasoning. All reported results represent average accuracy across 16 decoding runs, evaluated with a temperature of 1.0 and maximum generation length of 4096 tokens.

A.4.2 EXPLORATORY EXPERIMENT - OFFLINE RL SETTING

We sample responses for 100,000 math questions using a temperature of 1.0 and generate four responses per question from the base model itself. Correct responses are identified using math verify and retained as training data, resulting in approximately 140,000 examples. For DPO training, we construct 100,000 positive-negative preference pairs from the generated responses.

We compare DFT with representative offline RL methods, including DPO (Rafailov et al., 2023) and RFT (Dong et al., 2023; Ahn et al., 2024), as well as online RL methods PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024). For RFT and DFT, the training setup follows the configuration in Section 4.1. For DPO, we use the ms-swift (Zhao et al., 2024) with a learning rate of 1×10^{-6} , batch size of 128, and a warmup ratio of 0.05. For PPO and GRPO, training is performed using the verl (Sheng et al., 2025) with a learning rate of 1×10^{-6} , batch size of 256, and a warmup ratio of 0.1. We set the number of response $n = 4$ for GRPO.

A.4.3 EXPLORATORY EXPERIMENT - CODE GENERATION TASK

We adopt UltraFeedback (Cui et al., 2024) as the training dataset. From this corpus, we sample 10,000 prompts and, for each prompt, select the response with the highest average score to perform supervised fine-tuning (SFT) (Du et al., 2025). Model performance is assessed on three widely used code generation benchmarks: HumanEval (Chen et al., 2021), HumanEval+ (Liu et al., 2023), and MultiPL-E (Cassano et al., 2023). Training is conducted for one epoch with a learning rate of 5×10^{-5} , a warm-up ratio of 0.05, and a batch size of 16.

A.4.4 EXPLORATORY EXPERIMENT - MULTI-MODAL REASONING

We use the WeThink dataset (Yang et al., 2025a) for training. The model is fine-tuned using LLaMA-Factory (Zheng et al., 2024) and evaluated with VLMEvalKit (Duan et al., 2024). We train the model for 1 epoch with a learning rate of $5e-5$. To comprehensively assess reasoning capabilities, we adopt a suite of multi-modal reasoning benchmarks including MathVerse (Zhang et al., 2024a), MathVision (Wang et al., 2024), and WeMath (Qiao et al., 2024) for evaluation.

A.5 CAN DFT ENHANCE REINFORCEMENT LEARNING?

Mathematical Reasoning. For GRPO training, we sample 50,000 instances from the NuminaMath-CoT dataset (LI et al., 2024). We train for one epoch with a learning rate of 1×10^{-6} , a rollout size of 4, and a batch size of 256.

Code Generation. Following the setup used in Code-R1 (Liu & Zhang, 2025), we train with GRPO for 256 steps using a rollout size of 16, a batch size of 256, and a learning rate of 5×10^{-7} .

Multi-modal Reasoning. Following the training protocol of R1-OneVision (Yang et al., 2025b), we apply GRPO for 25 steps with a learning rate of 1×10^{-6} , a batch size of 128, and a rollout size of 5.

A.6 COMPARISON WITH CONCURRENT WORK IW-SFT

We include a concurrent method, Importance-Weighted SFT (iw-SFT) (Qin & Springenberg, 2025), for comparison. All training settings follow those reported in the original paper, except that we set the number of training epochs to 1.

As shown in Table 9, DFT achieves higher average accuracy than iw-SFT on most model families: LLaMA-3.2-3B (+2.39), LLaMA-3.1-8B (+4.15), DeepSeekMath-7B (+3.34), and Qwen2.5-Math-1.5B (+1.30). Although iw-SFT outperforms our method on Qwen2.5-Math-7B (+2.45), this improvement is not consistent across datasets. In particular, for LLaMA-3.2-3B, iw-SFT underperforms standard SFT on Math500 (5.13 vs. 8.65) and AMC23 (2.03 vs. 3.13). Similarly, for LLaMA-3.1-8B, iw-SFT results in worse performance than SFT on Minerva Math (4.31 vs. 5.78) and AMC23 (7.34 vs. 8.28). In contrast, DFT consistently improves upon both the base model and SFT across nearly all datasets, including those where iw-SFT fails. These results underline better generalization ability of DFT in diverse mathematical reasoning scenarios. Moreover, iw-SFT incurs additional computational overhead by requiring a separate reference model to compute importance weights, whereas DFT dynamically derives its own weighting directly from the token probabilities of model, resulting in a more efficient training procedure.

We also compare against iw-SFT under the offline setting, as shown in Table 10. While iw-SFT performs competitively on certain datasets, achieving 60.80 on Math500 and 44.21 on AMC23, its

Table 9: Comparison with concurrent work iw-SFT on math benchmarks. DFT outperforms the iw-SFT in most settings across model families and benchmarks. The best performance is bold.

	Math500	Minerva Math	Olympiad Bench	AIME24	AMC23	Avg.
LLaMA-3.2-3B w/iw-SFT	5.13	2.63	1.51	0.00	2.03	2.26
LLaMA-3.2-3B w/DFT	12.79	2.84	2.90	0.83	3.91	4.65
LLaMA-3.1-8B w/iw-SFT	18.21	4.31	4.31	0.20	7.34	6.87
LLaMA-3.1-8B w/DFT	27.44	8.26	6.94	0.41	12.03	11.02
DeepSeekMath-7B w/iw-SFT	35.32	8.75	11.11	0.61	18.28	14.81
DeepSeekMath-7B w/DFT	41.46	16.79	15.00	1.24	16.25	18.15
Qwen2.5-Math-1.5B w/iw-SFT	59.38	17.08	26.82	8.13	40.00	30.28
Qwen2.5-Math-1.5B w/DFT	64.89	20.94	27.08	6.87	38.13	31.58
Qwen2.5-Math-7B w/iw-SFT	70.28	25.70	34.46	16.46	51.09	39.60
Qwen2.5-Math-7B w/DFT	68.20	30.16	33.83	8.56	45.00	37.15

Table 10: Evaluation results on five mathematical reasoning benchmarks in the offline reinforcement learning setting using rejection-sampling-based reward signals, compared against iw-SFT.

	Setting	Math500	Minerva Math	Olympiad Bench	AIME24	AMC23	Avg.
Qwen2.5-Math-1.5B w/iw-SFT	SFT	59.38	17.08	26.82	8.13	40.00	30.28
Qwen2.5-Math-1.5B w/DFT	SFT	62.50	22.94	26.87	7.31	33.75	30.67
Qwen2.5-Math-1.5B w/iw-SFT	Offline	60.80	18.13	27.83	8.33	44.21	31.86
Qwen2.5-Math-1.5B w/DFT	Offline	64.71	25.16	30.93	7.93	48.44	35.43

overall average performance (31.86) remains below that of our method by +3.57 points. Moreover, iw-SFT shows only modest improvements compared to its standard SFT counterpart, with an average score of 31.86 in the offline RL setting versus 30.28 with SFT (+1.58). In contrast, DFT achieves a larger gain of +4.76 (from 30.67 to 35.43). These results indicate that iw-SFT provides limited benefits from reward supervision under offline constraints, whereas DFT is able to more effectively incorporate such signals, leading to better generalization and higher task performance.

A.7 EXPLORATORY EXPERIMENT - OPENR1-MATH TRAINING DATASET

Inspired by DeepSeek-R1 [DeepSeek-AI et al. \(2025\)](#), several studies have attempted to train open-source models to reproduce its reasoning capabilities ([Hugging Face, 2025](#)). To this end, a high-quality dataset, OpenR1-Math-220k ([Hugging Face, 2025](#)), was constructed, where the prompts are drawn from NuminaMath 1.5 and the off-policy reasoning traces are generated by DeepSeek-R1. LUFFY ([Yan et al., 2025](#)) further filtered out sequences longer than 8192 tokens as well as those verified incorrect by Math-Verify, resulting in about 45k prompts paired with off-policy reasoning traces. We adopt this dataset as the training corpus for SFT. All training details remain the same as previous experiments, except that the number of epochs is set to 3.

As shown in Table 11, training on OpenR1-Math-220k consistently improves performance, and the use of higher-quality annotations yields additional gains. SFT on this dataset increases the average accuracy of Qwen2.5-Math-1.5B by +13.24 points compared to the base model, while DFT provides a further +9.03 gain, resulting in a total improvement of +22.27. These results suggest that DFT remains effective even when applied on top of high-quality training data, highlighting its potential as a general fine-tuning paradigm.

Table 11: Performance training on the OpenR1-Math-220k dataset. The best score for each benchmark is in bold.

	Math500	Minerva Math	Olympiad Bench	AIME24	AMC23	Avg.
Qwen2.5-Math-1.5B	31.66	8.51	15.88	4.16	19.38	15.92
Qwen2.5-Math-1.5B w/SFT	61.60	20.29	24.27	4.16	35.47	29.16
Qwen2.5-Math-1.5B w/DFT	71.76	27.00	33.48	9.79	48.91	38.19

Table 12: Performance on five mathematical reasoning benchmarks using LoRA for training. The best score of each model across benchmarks is highlighted in bold.

	Math500	Minerva Math	Olympiad Bench	AIME24	AMC23	Avg.
LLaMA-3.2-3B	1.63	1.36	1.01	0.41	1.56	1.19
LLaMA-3.2-3B w/SFT	4.88	1.56	1.68	0.00	2.66	2.56
LLaMA-3.2-3B w/DFT	11.13	5.18	3.87	0.00	2.97	4.63
Qwen2.5-Math-1.5B	31.66	8.51	15.88	4.16	19.38	15.92
Qwen2.5-Math-1.5B w/SFT	41.47	10.85	11.56	1.45	17.03	16.87
Qwen2.5-Math-1.5B w/DFT	64.85	22.58	28.45	5.84	40.78	32.90

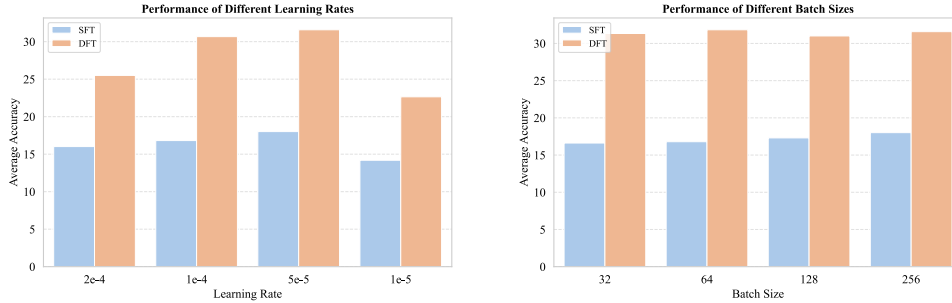


Figure 3: Ablation study of training hyper-parameters, learning rates and batch size, for DFT and SFT on Qwen2.5-Math-1.5B model.

A.8 EXPLORATORY EXPERIMENT - PEFT TRAINING SETTING

To investigate whether DFT remains effective under parameter-efficient fine-tuning (PEFT) settings with limited compute, we apply DFT using LoRA adapters across two model families: LLaMA-3.2-3B and Qwen2.5-Math-1.5B. All training configurations remain identical to previous full-parameter experiments, except that LoRA is enabled with rank=8 and alpha=16.

As shown in Table 12, DFT provides consistent improvements over both base and SFT baselines under LoRA-based PEFT. For Qwen2.5-Math-1.5B, DFT increases the average accuracy from 15.92 (base) and 16.87 (SFT) to 32.90. For LLaMA-3.2-3B, DFT achieves a gain of +3.44 over SFT (from 1.19 to 4.63). These results indicate that DFT can serve as an effective fine-tuning strategy in low-resource or compute-constrained settings, where full model updates are not practical.

A.9 TRAINING HYPER-PARAMETERS ABLATION

To assess the robustness and sensitivity of our approach (DFT) with respect to key training hyper-parameters, we conduct an ablation study focused on learning rate and batch size, using the Qwen2.5-Math-1.5B base model. This analysis aims to answer two central questions: (1) Is the

performance gap between DFT and SFT due to a suboptimal hyperparameter configuration in SFT?
(2) How sensitive are both methods to changes in learning rate and batch size?

We evaluate both DFT and SFT across four learning rates: $2e-4$, $1e-4$, $5e-5$, and $1e-5$. As shown in Figure 3 (left), both methods exhibit a certain degree of sensitivity to the learning rate. DFT consistently outperforms SFT under all configurations, suggesting that the performance gap cannot be attributed solely to suboptimal hyperparameter choices in SFT. For both methods, intermediate learning rates ($1e-4$ and $5e-5$) yield the best results, while both lower ($1e-5$) and higher ($2e-4$) values lead to noticeable degradation.

We further assess the impact of batch size, sweeping values from 32 to 256. As shown in Figure 3 (right), both DFT and SFT exhibit relatively stable performance across the full range of batch sizes. While minor fluctuations are observed, there is no consistent trend indicating that larger or smaller batches significantly affect final accuracy. This suggests that batch size is not a dominant factor for either method in this setup, and that default values may suffice in practice.