SV2MPG: From Surface Validity to Structural Validity via Two-Stage Reinforcement Learning for Olympic-Level Math Problem Generation

Anonymous ACL submission

Abstract

Synthetic data has become a critical scaling strategy in modern LLM training, yet its application to high-difficulty mathematical reasoning remains limited—most methods target on GSM8K, MATH, and at most AIME while automated generation of Olympiad-level problem–solution pairs (e.g., from AIME, HMMT, BRUMO, CMIMC) is underexplored. To bridge this gap, we propose **SV2MPG** (**S**urface **V**alidity and **S**tructural **V**alidity-aligned **M**athematical **P**roblem **G**enerator), a lightweight generator trained via two-stage reinforcement learning to optimize both *surface validity* and *structural validity*. **Inspired by dual-process theory, our training adopts distinct human-like evaluation perspectives: Stage 1 uses System 1 (fast, intuitive) judgments to ensure problems *look right*; Stage 2 uses System 2 (slow, analytical) verification to ensure they *solve right*.** Integrating SV2MPG with a strong open-weight solver (GPT-OSS-120B), we build an end-to-end pipeline for scalable synthesis of verified high-difficulty problems. Building on OMNI-MATH, we generate SV2-MATH—a 3,707-problem dataset of Olympiad-style questions—on which supervised fine-tuning yields substantial improvements: Qwen3-0.6B and Qwen3-4B achieve +2.09% and +8.44% absolute gains in avg@4 (averaged over four 2025 Olympiad-level benchmarks), consistently outperforming models trained on other baseline math datasets.

1 Introduction

Synthetic data has emerged as a pivotal scaling strategy in modern large language model (LLM) training, driving significant advances across code generation, mathematical reasoning, and agentic capabilities (DeepSeek-AI et al., 2025; Tongyi DeepResearch Team, 2025; Zeng et al., 2025). Within mathematical reasoning, while state-of-the-art models now achieve strong performance (70%–90%+ accuracy) on final-answer-based Olympiad benchmarks such as AIME, HMMT, BRUMO, and CMIMC, their capability remains uneven—particularly for smaller-scale models, which typically score below 70% and exhibit substantial benchmark-dependent variance. More critically, although problem–solution pair synthesis is recognized as essential for robust mathematical training, current research remains largely constrained to mid-difficulty datasets like GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), leaving automated generation of high-difficulty competition-level problems—especially those requiring deep multi-step reasoning—substantially underexplored.

To address this gap, we aim to establish a scalable framework for synthesizing high-fidelity Olympiad-level problem–solution datasets, with two concrete objectives: first, to construct a fully automated pipeline capable of generating mathematically sound problems and verified solutions at scale; and second, to leverage the resulting synthetic data as a knowledge resource for systematically enhancing models' competition-level mathematical reasoning.

Our approach centers on **SV2MPG** (**S**urface **V**alidity and **S**tructural **V**alidity-aligned **M**athematical **P**roblem **G**enerator), a lightweight generator trained via a two-stage reinforcement learning pipeline that explicitly separates *how a problem looks* (surface validity: plausibility, creativity, coherence) from *how it solves* (structural validity: solvability, difficulty alignment). This design is grounded in dual-process theory (Kahneman, 2011): **Stage 1 adopts a System 1 evaluation perspective**—fast, intuitive judgments of whether a problem *looks right* (e.g., natural phrasing, no ambiguity)—to optimize surface fluency; **Stage 2 adopts a System 2 evaluation perspective**—slow, analytical verification of whether a problem *solves right* (e.g., consensus solvability, appropriate chal-

lenge)—to ensure structural soundness. Critically, these correspond to *how human experts assess problems*, not how the generator operates internally. By decoupling generation from solving and integrating a high-capacity open-weight solver (GPT-OSS-120B) for answer and chain-of-thought synthesis, we obtain a practical, end-to-end system for producing Olympiad-grade problem–solution pairs without human annotation.

Empirically, training compact models on our synthetic dataset yields substantial gains: Qwen3-0.6B and Qwen3-4B achieve +2.09% and +8.44% absolute improvement in avg@4, respectively—averaged over the four 2025 Olympiad benchmarks (AIME2025, HMMT Feb2025, BRUMO2025, CMIMC2025), which outperforming models trained on OPENR1-MATH-220K (OpenR1 Team, 2025), DAPO-MATH-17K (Yu et al., 2025)-augmented data, or the original OMNI-MATH (Gao et al., 2024). Ablation studies further confirm that both training stages are essential, demonstrating SV2MPG's effectiveness in synthesizing mathematically rigorous, high-utility training data.

2 Related Work

Data Synthesis for Mathematical Reasoning. Synthetic math data has evolved from simple perturbation (e.g., numeric substitution) to structured generation. Early work like META-MATH (Yu et al., 2024) introduces backward self-verification—solving then reconstructing the premise—to enforce logical consistency, but remains confined to algebraic manipulation and struggles with combinatorial or geometric reasoning. IQC (Liu et al., 2024) enhances diversity via iterative constraint injection, yet its reliance on hand-specified templates limits adaptability to open-ended Olympiad-style problems. More recent pipelines (Ding et al., 2025; Seegmiller et al., 2025) modularize generation, difficulty tuning, and filtering, but their LLM-as-a-Judge rewards often prioritize surface fluency (e.g., grammatical polish, narrative plausibility) over genuine solvability—leading to "reward hacking" where problems appear challenging but collapse under solver scrutiny. Semantic-guided methods mitigate this: Zhao et al. (2025b) conditions CoT on core concepts; Huang et al. (2024) leverages topic co-occurrence graphs for knowledge coverage. Yet critically, none decouple *surface validity* (human-intuitive formulation) from *structural validity* (algorithmically verifiable difficulty); instead, they optimize a monolithic objective that conflates intuition and rigor. We resolve this via two-stage RL: Stage 1 targets fluency and creativity (System 1 thinking), while Stage 2 enforces solver-consistent difficulty (System 2 verification)—enabling balanced, high-fidelity synthesis.

Adversarial and Interactive Training with MPG. Interactive frameworks embed MPG in training loops to create adaptive curricula. Dual-play methods (Zhang et al., 2025; Wei et al., 2025) formulate generation as a minimax game, where problem-makers target solver weaknesses—but typically use coarse difficulty buckets (easy/medium/hard), lacking fine-grained calibration. Data-free co-evolution (Wang et al., 2025; Zhao et al., 2025a) eliminates external data via agent self-play, yet suffers from unstable credit assignment in multi-step reasoning. Formal systems like AlphaProof (Hubert et al., 2025) achieve IMO silver-medal performance via PPO in Lean, but require full formalization and incur prohibitive latency (hours per problem), restricting use to final-answer-based competitions. Self-verified CoT (Shao et al., 2025; Pei et al., 2025; Guan et al., 2025) improves step-wise correctness through internal checks or self-questioning, yet tightly couples generation and verification, hindering independent control of diversity and correctness. In contrast, our decoupled, solver-agnostic pipeline separates problem formulation from consensus-based validation—enabling scalable, lightweight synthesis of Olympiad-grade data without human annotation.

Summary. Despite these advances, most pipelines remain confined to mid-difficulty datasets (GSM8K, MATH) that underrepresent competition-level complexity. Olympiad-level synthesis either bears the cost of formal proof reasoning (e.g., AlphaProof) or neglects explicit problem-level validity optimization. We bridge this gap with **SV2MPG**—a decoupled, consensus-verified framework jointly optimizing *surface* and *structural* validity for final-answer competition problems, producing questions that are both human-plausible and solver-calibrated.

3 Method

The overall architecture is illustrated in Figure 1.

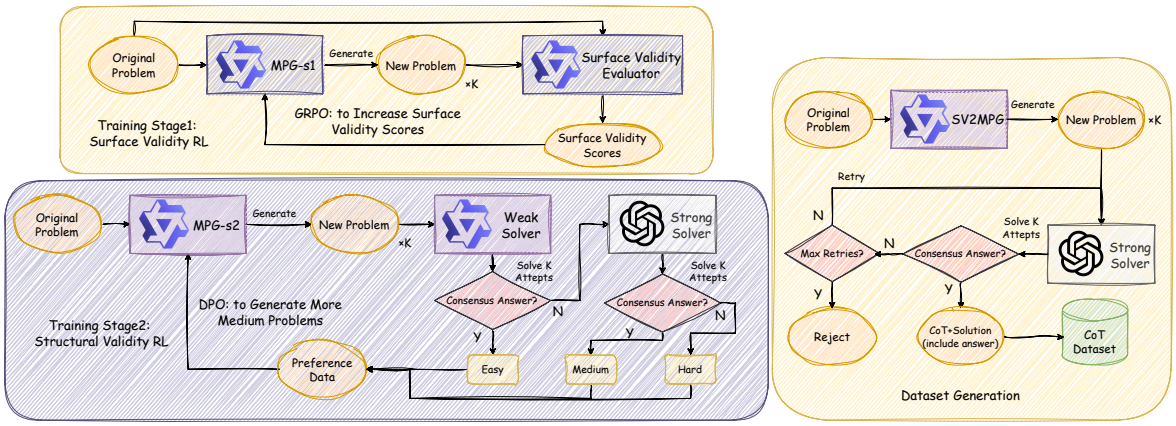

Figure 1: SV2MPG framework: (Left) Two-stage training (GRPO → DPO); (Right) Automated dataset synthesis.

3.1 Definition

To holistically assess the quality of generated problems, we introduce two complementary validity dimensions—inspired by dual-process theory (Kahneman, 2011)—that mirror how humans evaluate problems: *how it looks* (intuitive, surface-level assessment) and *how it solves* (analytical, solver-based verification).

Surface Validity. This metric evaluates problem quality *from the problem-setting perspective*, relying solely on the problem statement without solving—corresponding to **System 1 (fast thinking)**: rapid, intuitive judgments of plausibility and fluency. It ensures generated problems *look right* to human readers, exhibiting sound mathematical formulation. We assess five sub-dimensions (scored 1–5 or via pairwise preference): **Core knowledge retention**: How well does the new problem preserve key concepts from the original? **Innovation**: How creative and non-trivial is the variation (avoiding mechanical paraphrasing)? **Difficulty match**: Is the perceived difficulty aligned with the original? **Reasoning complexity**: Does the problem demand multi-step reasoning of comparable depth? **Coherence**: Is the statement natural, unambiguous, and mathematically self-consistent?

Structural Validity. This metric evaluates quality *from the problem-solving perspective*, focusing on solvability and difficulty appropriateness—corresponding to **System 2 (slow thinking)**: deliberate, stepwise verification of correctness and challenge. It ensures problems *solve right* under solver scrutiny, avoiding ill-posed or trivial variants. We define: **Solvability**: Determined via *consensus solution* @k—a problem is solvable by a model if, among $k = 4$ independent solutions, one answer appears $> k/2 = 2$ times. **Difficulty tiers**: We adopt a dual-solver setup: `Easy`: solvable by a *weak solver* (e.g., Qwen3-4B); `Medium`: unsolvable by the weak solver but solvable by a *strong solver* (e.g., GPT-OSS-120B (Balunović et al., 2025)); `Hard`: unsolvable by the strong solver (potentially ill-posed or beyond current capabilities). Crucially, our optimization target is **maximizing the proportion of Medium problems**, which balance challenge (beyond weak solvers) and reliability (solvable by strong ones)—making them ideal for high-value synthetic data.

3.2 Task Formulation for Problem Generation

Guided by the dual-validity principle, we make three key design choices:

Problem Rewriting vs. Knowledge-Based Generation. Mathematical problems encode not only explicit knowledge but also implicit structural properties—namely, surface and structural validity—which are hard to distill from generic knowledge corpora (e.g., textbooks). As shown in (Chen et al., 2025), problem-solving data (e.g., problem–CoT pairs) yield greater gains in mathematical reasoning than general math texts. Hence, we adopt **problem rewriting** as our core paradigm to preserve and transfer these validity structures.

Decoupled (Problem-only) vs. Integrated (Problem + Solution) Generation. We advocate a **decoupled architecture**: generation and solving are separated to avoid multi-task interference. Empirical evidence (§4.2) shows that models trained to generate *problems only* achieve ∼80% win rate over integrated variants in surface validity evaluation, while the latter's self-generated answers

show $< 40\%$ consistency with GPT-OSS-120B. Thus, SV2MPG focuses solely on high-quality problem generation; solving is delegated to dedicated solvers.

Instruct vs. Thinking Models. Though *thinking* models excel at complex reasoning, their chain-of-thought mechanism is redundant for problem rewriting—a task emphasizing linguistic restructuring over deep inference. Moreover, thinking incurs $\sim 10\times$ longer inference time. We thus select **Qwen3-4B-Instruct-2507** (Yang et al., 2025) as our base model, balancing efficiency and capability.

In summary, SV2MPG follows a three-pronged principle: *problem rewriting*, *decoupled single-task generation*, and *efficient instruct-based modeling*.

3.3 Training of SV2MPG: A Two-Stage Validity-Aligned RL Framework

Our training proceeds in two sequential stages, as depicted in the left panel of Figure 1. Stage 1 begins with `Qwen3-4B-Instruct-2507` and yields the intermediate generator **MPG-s1** (top-left, Figure 1). Stage 2 then initializes from the best checkpoint of MPG-s1 and produces a series of refined models **MPG-s2** (bottom-left, Figure 1); the final model, selected based on validation performance, is designated as **SV2MPG**.

The training data is sourced from `Omni-MATH` (Chen et al., 2025), which contains 4,428 problems. We split it 8:2 into training (3,542 problems) and internal test (886 problems) sets. In **Stage 1**, only the 80% training subset is used for GRPO updates. In **Stage 2**, to maximize coverage and diversity, we utilize the *full* `Omni-MATH` set (4,428 problems) for problem generation and preference construction.

3.3.1 Stage 1: Surface Validity Alignment via On-Policy GRPO

The resulting model is denoted **MPG-s1**. Key components:

Reward Model. We use `Qwen3-30B-A3B-Instruct-2507` as an LLM-as-a-Judge to score candidate problems on the five dimensions defined in §3.1. To reflect their relative importance, we assign non-uniform weights: *Innovation*—our core evaluation objective—is given the highest weight (2.0); *Core knowledge retention*, *Difficulty match*, and *Reasoning complexity* each receive a weight of 1.0; while *Coherence*—though necessary—is treated as a baseline requirement and assigned a lower weight (0.5). The final reward is computed as the weighted sum of the five dimension scores:

$$R_{\text{surf}} = 1.0 \cdot r_1 + 2.0 \cdot r_2 + 1.0 \cdot r_3 + 1.0 \cdot r_4 + 0.5 \cdot r_5,$$

where r_1 to r_5 correspond to coherence, innovation, core knowledge retention, difficulty match, and reasoning complexity, respectively.

Algorithm and Objective. We adopt **GRPO** (Shao et al., 2024), an on-policy algorithm that maximizes the advantage of high-reward candidates within a generation group. For each original problem p_{orig}, MPG-s1 generates $m = 8$ candidates $\{p_{\text{new}}^{(i)}\}_{i=1}^{8}$; the reward for each candidate is computed via the reward model above, and policy updates follow the GRPO objective:

$$\mathcal{L}_{\text{GRPO}} = -E_{p_{\text{orig}}}\left[\sum_{i=1}^{8} \log \pi_\theta(p_{\text{new}}^{(i)} \mid p_{\text{orig}}) \cdot A_i\right],$$

where A_i is the advantage of candidate i relative to the group mean reward. We train for 2 epochs with learning rate 5×10^{-6} and gradient accumulation over 2 steps.

3.3.2 Stage 2: Structural Validity Alignment via Off-Policy DPO

Starting from MPG-s1, we train **MPG-s2**, with the final checkpoint selected as **SV2MPG**.

Difficulty Labeling. To assess the intrinsic difficulty of each generated problem, we adopt a hierarchical solver-consensus strategy leveraging both weak and strong solvers with $k_w = k_s = 4$ attempts. All solver calls use temperature $\tau = 0.7$ and prompt perturbations to promote diversity. Given a problem, we:

1. Query the weak solver (Qwen3-4B) 4 times; if consensus is reached, label `Easy`;
2. Else, query the strong solver (GPT-OSS-120B (Balunović et al., 2025)) 4 times; if consensus is reached, label `Medium`;
3. Else, label `Hard`.

This procedure ensures robust difficulty annotations (see Algorithm 1).

Preference Construction and DPO Training. We use **DPO** (Rafailov et al., 2024), an off-policy algorithm that avoids online sampling and enables full decoupling of generation and solving. Preference pairs are constructed according to two validity-driven criteria: `Medium` problems are preferred over both `Easy` (to avoid triviality) and

Algorithm 1: Difficulty Labeling via Dual-Solver Consensus

Input: Problem p, weak solver $\mathcal{W}$, strong solver $\mathcal{S}$, $k_w = 4$, $k_s = 4$
Output: Difficulty label $d \in \{\texttt{Easy}, \texttt{Medium}, \texttt{Hard}\}$

$\mathcal{A}_w \leftarrow [\mathcal{W}(p)]_{i=1}^{4}$;
if CONSENSUS($\mathcal{A}_w$) **then**
  return `Easy`
$\mathcal{A}_s \leftarrow [\mathcal{S}(p)]_{i=1}^{4}$;
if CONSENSUS($\mathcal{A}_s$) **then**
  return `Medium`
return `Hard`

`Hard` (to ensure solvability), reflecting our core objective of maximizing medium-difficulty, structurally sound problems. Each training triplet takes the form $(\texttt{prompt}, p_{\text{chosen}}, p_{\text{rejected}})$, where `prompt` comprises the fixed instruction template and the original problem p_{orig}, p_{chosen} is a `Medium`-labeled candidate, and p_{rejected} is either an `Easy` or `Hard` counterpart from the same generation batch.

The DPO loss is:

$$\mathcal{L}_{\text{DPO}} = -E_{(x,y_w,y_l)}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} - \beta\log\frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$$

where π_{ref} is the frozen MPG-s1 policy, y_w is a `Medium` problem, y_l is rejected (`Easy`/`Hard`), and we set $\beta = 0.1$. This stage steers SV2MPG toward generating `Medium`-dominant distributions, achieving the desired balance of challenge and reliability.

3.4 SV2MPG for Dataset Generation

Leveraging the dual-validity guarantees of SV2MPG, we construct a scalable pipeline for synthesizing high-fidelity mathematical problem–solution pairs, as illustrated in the right panel of Figure 1. Starting from `Omni-MATH`, we apply SV2MPG to each original problem under a fixed instruction template with temperature $\tau = 0.7$, generating $k_{\text{generate}} = 2$ candidate variants per seed; these candidates are then rigorously deduplicated via exact string matching to eliminate verbatim duplicates. For each surviving synthetic problem, we employ GPT-OSS-120B (Balunović et al., 2025) to produce $k_{\text{strongsolver}} = 4$ independent solutions, from which final answers are extracted in a two-stage manner: first, by parsing XML-style `<answer>...</answer>` tags if present; failing that, by applying regular expressions to capture `\boxed{...}`, terminal numerical expressions, or natural-language answer phrases (e.g., "The answer is ..."). A consensus solution is declared if any normalized answer achieves majority support (i.e., appears in > 2 of 4 trials). Upon consensus, the first complete chain-of-thought trace—including the solver's intermediate reasoning steps (*thinking phase*), final derivation (*solution*), and normalized answer—is preserved as the canonical triple; if no consensus emerges, the solving procedure is repeated up to $T_{\max} = 3$ times with prompt perturbations. The final dataset consists of structured records $(\text{problem}, \text{CoT}, \text{solution}, \text{answer})$, complemented by the raw solver response for reproducibility and auditing, thus forming a closed-loop synthesis framework wherein surface validity secures problem authenticity and structural validity ensures solution correctness.

4 Experiments

4.1 Experimental Setup

All models are deployed using `vLLM` (Kwon et al., 2023), enabling efficient batched inference with PagedAttention and a 32K context window; for deterministic evaluation, temperature is fixed at 0.0. Following the evaluation protocol established in MATHARENA (Balunović et al., 2025), we adopt `avg@4` (i.e., the average of four independent `pass@1` trials) as our primary metric to mitigate stochastic variance, and select the same four 2025 competition suites—`AIME2025`, `BRUMO2025`, `CMIMC2025`, and `HMMT Feb2025`—as our external evaluation benchmark, which collectively represent the current frontier of high-difficulty mathematical reasoning. In addition, we reserve 886 problems from `Omni-MATH` (Chen et al., 2025) (the 20% held-out split from its 4,428-problem corpus) as an internal development set for training analysis.

Usage convention: In §§4.2–4.4, the four competitions are treated as a unified benchmark to ensure statistical robustness in design ablation and training analysis; fine-grained per-competition results are reported in §4.5.

4.2 Analysis of Design Choices

Instruct vs. Thinking Models. Table 1 shows near-identical surface validity reward scores (5-point scale) for `Qwen3-4B-Instruct-2507` and `Qwen3-4B-Thinking-2507` (mean difference < 0.03) under the same prompt and model in §4.3,

Model	Mean	Median	Std. Dev.
Qwen3-4B-Instruct-2507	4.01	4.09	0.62
Qwen3-4B-Thinking-2507	4.03	4.09	0.54

Table 1: Surface validity scores (5-point Likert scale) on the Omni-MATH internal test set ($n = 886$). **Mean**: arithmetic average; **Median**: 50th percentile; **Std. Dev.**: standard deviation—indicating score dispersion around the mean. Scores are assigned by `Qwen3-30B-A3B-Instruct-2507` via the five-dimensional rubric in §3.1.

Model	Win Rate (%)		Answer Consistency (%)	
	test	benchmark	test	benchmark
Instruct	81.90	83.08	31.96	14.62
Thinking	86.58	90.77	38.71	23.85

Table 2: Comparison of decoupled (problem-only) vs. integrated (problem + solution) generation. **Win Rate**: % of problems where decoupled output is preferred by GPT-OSS-120B over integrated output in pairwise judgment ($n = 886$ test, $n = 130$ benchmark). **Answer Consistency**: % of self-generated answers matching GPT-OSS-120B consensus solutions.

generated from Internal Test Set, confirming that chain-of-thought reasoning is redundant for problem rewriting.

Decoupled (Problem-only) vs. Integrated (Problem + Solution) Generation. As shown in Table 2, the decoupled approach achieves win rates of 81.9%–90.8% over integrated generation across both sets. Crucially, answer consistency (vs. GPT-OSS-120B consensus) is low ($\leq 23.9\%$), indicating that joint generation compromises both surface validity and solution reliability.

Collectively, these results validate our three design principles: *problem rewriting*, *decoupled generation*, and *instruct-based modeling*.

4.3 Stage 1: Surface Validity Reinforcement Learning

We train MPG-s1 on the 80% training split of `Omni-MATH` (3,542 problems) using GRPO (Shao et al., 2024). The reward model—`Qwen3-30B-A3B-Instruct-2507`—scores each candidate on the five surface validity dimensions (Core knowledge retention, Innovation, Difficulty match, Reasoning complexity, Coherence) on a 1–5 scale; these scores are then combined via a weighted average with non-uniform weights (Innovation: 2.0; Core retention, Difficulty match, Reasoning complexity: 1.0 each;

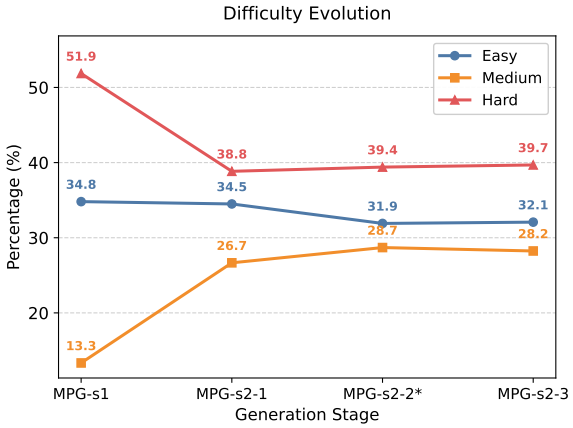

Figure 2: Evolution of problem difficulty distribution across generation stages. Easy (blue), Medium (orange), Hard (red) percentages are shown for MPG-s1, MPG-s2-1, MPG-s2-2*, and MPG-s2-3.

Coherence: 0.5) and normalized by the total weight sum (5.5) to yield the final reward $R_{\text{surf}} \in [1, 5]$. We set the learning rate to 5×10^{-6}, $\beta = 0.01$, train for 2 epochs with gradient accumulation over 2 steps, and generate 8 candidates per original problem during policy sampling.

During the GRPO trainig, the training reward rises steadily in the initial phase and stabilizes after approximately 1,500 optimization steps—indicating stable convergence. The final policy achieves a mean reward of 4.31, a 7.5% absolute improvement over the base model (4.01), confirming effective optimization of the target objective.

4.4 Stage 2: Structural Validity Reinforcement Learning

Building upon MPG-s1, we conduct three iterations of DPO (Rafailov et al., 2024) (yielding MPG-s2-1, MPG-s2-2, and MPG-s2-3) to align structural validity. To maximize preference data coverage and diversity, we generate candidates on the *full* `Omni-MATH` dataset (4,428 problems). For each original problem, we sample $k = 4$ candidates from the current policy, then label them via dual-solver consensus using `Qwen3-4B` (weak) and `GPT-OSS-120B` (strong) as described in §3.1. After collecting all preferences (Medium $\succ$ Easy and Medium $\succ$ Hard)—yielding approximately 1,500 effective preference pairs per iteration—we perform DPO for exactly 3 epochs per iteration with $\beta = 0.01$, and proceed to the next round.

Figure 2 illustrates the evolution of difficulty distributions across stages. Starting from MPG-s1

(Hard: 51.9%, Medium: 13.3%, Easy: 34.8%), the proportion of `Hard` problems declines to 39.7% in MPG-s2-3, while the target `Medium` tier rises to 28.2%, approaching our ideal range of 25–35%. Notably, MPG-s2-2 and MPG-s2-3 exhibit nearly identical distributions (Medium: 28.7% vs. 28.2%), indicating convergence after two DPO rounds. We therefore select **MPG-s2-2** as the final **SV2MPG** model.

4.5 Synthetic Data Quality: Downstream SFT Performance

Building upon the dataset synthesis pipeline in §3.4, we construct SV2-MATH—a purely synthetic corpus of 3,707 problem–CoT–solution triples—by applying SV2MPG to the `Omni-MATH` training set with $k = 2$. To rigorously evaluate its utility, we compare its downstream SFT performance against four baselines: (1) OPENR1 (OpenR1 Team, 2025) (default split, 94,217 problems from Numina-Math 1.5); (2) the original `Omni-MATH` (Gao et al., 2024); (3) DAPO-MATH-17K (Yu et al., 2025); and (4) a strong generator control—GPT-OSS-120B-DATASET, where problems are generated by `GPT-OSS-120B` using the identical rewriting prompt and $k = 2$ protocol as SV2MPG, followed by the same solution synthesis pipeline (GPT-OSS-120B, $k = 4$, consensus verification). This control tests whether raw model scale alone suffices for high-quality generation, isolating the contribution of our two-stage training from mere capacity differences.

Crucially, for `Omni-MATH` and DAPO-MATH-17K, we generate CoT traces using the identical strong solver (GPT-OSS-120B), inference configuration, and consensus-based verification protocol as for SV2-MATH, ensuring fair comparison. Furthermore, to ablate generator quality, we synthesize BASE-DATASET (from untrained `Qwen3-4B-Instruct-2507`) and MPG-S1-DATASET (from **MPG-s1**).

As shown in Table 3, SFT with SV2-MATH yields the strongest gains: Qwen3-0.6B and Qwen3-4B achieve +2.09 and +8.44 percentage points in avg@4, outperforming all baselines. This hierarchy—SV2-MATH > MPG-S1-DATASET > BASE-DATASET—confirms both training stages are essential, with structural validity alignment (Stage 2) providing the decisive boost. Notably, while Qwen3-0.6B benefits less consistently, Qwen3-4B demonstrates robust gains, underscoring compatibility with mid-scale models.

As shown in Table 3, SV2-MATH outperforms even the strong generator control GPT-OSS-120B-DATASET (+1.35 pp for Qwen3-0.6B; +6.21 pp for Qwen3-4B), demonstrating that our validity-aligned training protocol yields higher-quality data than direct generation with a much larger model—confirming that methodological design dominates raw scale in synthetic data quality.

A finer-grained analysis reveals two key patterns: (1) SV2-MATH excels particularly on `AIME2025` and `CMIMC2025`—the two benchmarks with the highest algebraic abstraction and multi-step constraint chaining—where its gains over OPENR1 reach +1.67 pp and +1.00 pp for Qwen3-4B, respectively. This suggests our dual-validity design better preserves the structural depth required for these problems. (2) Baselines like DAPO-SYNTHETIC show degradation (0.47 pp), likely because its RL-based generation prioritizes reward maximization over solvability consensus, leading to borderline-ill-posed problems that confuse mid-scale solvers—a limitation explicitly mitigated in SV2MPG via our two-stage filtering.

Unlike formal RL systems such as AlphaProof (Hubert et al., 2025)—which solve IMO problems via Lean formalization and multi-day inference—our approach targets final-answer-based competitions with lightweight, scalable synthesis, making high-quality data accessible to resource-constrained teams.

All SFT experiments are conducted using **LoRA** (Hu et al., 2021) with rank $r = 16$ (targeting `q_proj`, `v_proj`), under a unified configuration: learning rate 1×10^{-5} with warmup ratio 0.1, constant-with-warmup scheduler, weight decay 0.01, paged AdamW (8-bit), 2 training epochs, per-device batch size 2 with gradient accumulation over 4 steps (effective batch size 8), and max gradient norm 1.0. This lightweight setup enables full training on a single H20 (141GB) for all student models (0.6B–4B), with VRAM usage reduced by $\sim$60% compared to full fine-tuning.

5 Conclusion

We present **SV2MPG**, a lightweight yet highly effective framework for the automated synthesis of high-difficulty mathematical problem–solution datasets—specifically targeting Olympiad-level benchmarks such as AIME2025, BRUMO2025, CMIMC2025, and HMMT Feb2025. Our work begins with a systematic analysis of key design

Model	Method	AIME25	HMMT Feb25	BRUMO25	CMIMC25	avg Δ
Qwen3-0.6B	BaseModel	12.50	8.33	13.33	3.75	–
	+OpenR1	**16.67**	5.83	12.50	**6.25**	+0.84
	+DAPO-Math-17k	8.33	**10.00**	11.67	4.37	-0.89
	+Omni-MATH	14.17	9.17	12.50	4.37	+0.58
	+GPT-OSS-120B-Dataset	13.33	8.33	**17.50**	3.12	+1.09
	+Base-Dataset	12.50	7.50	13.33	3.12	-0.37
	+MPG-s1-Dataset	13.33	7.50	16.67	3.75	+0.84
	+SV2-MATH (Ours)	15.83	7.50	16.67	**6.25**	**+2.09**
Qwen3-4B	BaseModel	52.50	21.67	42.50	17.50	–
	+OpenR1	64.17	23.33	49.17	21.25	+5.94
	+DAPO-Math-17k	40.83	21.67	46.67	23.13	-0.47
	+Omni-MATH	49.17	23.33	41.67	**26.25**	+1.56
	+GPT-OSS-120B-Dataset	65.00	**27.50**	44.17	23.13	+6.40
	+Base-Dataset	46.67	23.33	44.17	16.88	-0.78
	+MPG-s1-Dataset	57.33	22.50	**51.67**	21.88	+4.80
	+SV2-MATH (Ours)	**68.33**	25.00	50.83	23.75	**+8.44**

Table 3: Avg@4(pass@1) results (%) across four benchmarks, with average improvement (Δ) over BaseModel. Bold: best per benchmark. Δ values are percentage points.

dimensions in math problem generation: *problem rewriting* vs. knowledge-based generation, *decoupled* vs. integrated output, and *instruct-based* vs. thinking-based modeling. Guided by preliminary ablation studies (§4.2), we adopt a decoupled, instruct-based rewriting paradigm—enabling specialized modeling where a compact 4B generator focuses solely on problem formulation, while high-capacity solvers (e.g., GPT-OSS-120B) handle solution verification and labeling.

Building on this foundation, we propose a two-stage reinforcement learning pipeline to progressively enhance generation quality: Stage 1 aligns *surface validity* (formal plausibility, creativity, coherence) via GRPO; Stage 2 optimizes *structural validity* (solvability, difficulty controllability) via DPO—steering the generator from *superficial resemblance* to *substantive fidelity*. The resulting model, SV2MPG, powers an end-to-end synthesis system that produces mathematically rigorous, high-utility training data: on OMNI-MATH, it generates SV2-MATH (3,707 problems), on which SFT yields up to +8.44% avg@4 improvement for Qwen3-4B—outperforming baselines such as OPENR1-MATH-220K, original `Omni-MATH`, and DAPO-derived data. Crucially, ablation over generator variants (Base → MPG-s1 → SV2MPG) confirms that both training stages are indispensable, with structural validity alignment providing the decisive boost in downstream utility.

By demonstrating that dual-validity-aware generation enables scalable synthesis of Olympiad-grade data without human annotation, we hope SV2MPG lowers the barrier to high-quality mathematical reasoning research—particularly for resource-constrained teams seeking to advance model capabilities beyond mid-difficulty benchmarks.

Limitations

Our work inherits some constraints typical of current data synthesis paradigms.

First, the reliability of synthesized solutions depends on the capability of the teacher solver (e.g., GPT-OSS-120B). For problems beyond the frontier of current SOTA models—especially those requiring deep insight, novel constructions, or lengthy formal proofs—automated generation remains fragile, often necessitating human expert verification. This creates a *capability ceiling*: synthetic data is most effective for improving sub-SOTA models, while advancing SOTA models themselves remains challenging—a *self-improvement bottleneck*.

Second, we avoid agent-based solvers (e.g., ReAct (Yao et al., 2023) with Python interpreter) in training and evaluation. Although such agents can yield higher reasoning fidelity, their execution incurs $\sim$10$\times$ longer latency and requires extended context windows, cutting throughput in large-scale generation (e.g., vLLM server concurrency drops by $>$70% in pilot tests). Given our throughput targets, the engineering cost currently outweighs marginal gains.

Third, Stage 2 employs an *off-policy* DPO formulation, enabling full decoupling of generation and solving but introducing a distributional shift between the behavior policy (MPG-s1) and target policy (MPG-s2). This can lead to suboptimal updates if the preference dataset insufficiently covers high-reward regions. Future work will explore efficient on-policy implementation to reduce mismatch while sustaining GPU utilization.

Finally, we acknowledge a potential societal risk: synthetic Olympiad problems could be misused to train models that assist in academic dishonesty during math competitions. We emphasize that SV2MPG's primary goal is to democratize access to high-quality training data for research and education, not to enable misuse.

References

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. MathArena: Evaluating LLMs on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*. Version 2, revised on 2 October 2025.

Zui Chen, Tianqiao Liu, Mi Tian, Qing Tong, Weiqi Luo, and Zitao Liu. 2025. Advancing mathematical reasoning in language models: The impact of problem-solving data, data synthesis methods, and training stages. *arXiv preprint arXiv:2501.14002 [cs.CL]*. Accepted at ICLR 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI, Aixin Liu, Aoxue Mei, and 1 others. 2025. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Yujia Ding, Xingyu Shi, Xin Liang, Jiawei Li, Zhiming Tu, Qika Zhu, and Ming Zhang. 2025. Unleashing LLM reasoning capability via scalable question synthesis from scratch. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Bofei Gao, Feifan Song, Zhe Yang, Zhefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2024. Omni-MATH: A universal olympiad-level mathematical benchmark for large language models. *arXiv preprint arXiv:2410.07985*.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rStar-Math: Small LLMs can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, and 1 others. 2021. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Yiming Huang, Xiao Liu, Yeyun Gong, Zhibin Gou, Yelong Shen, Nan Duan, and Weizhu Chen. 2024. Key-point-driven data synthesis with its enhancement on mathematical reasoning. *arXiv preprint arXiv:2403.02333*.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, and 6 others. 2025. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*.

Daniel Kahneman. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language

model serving with PagedAttention. *arXiv preprint arXiv:2309.06180*.

Haoxiong Liu, Yifan Zhang, Yifan Luo, and Andrew Chi-Chih Yao. 2024. Augmenting math word problems via iterative question composing. *arXiv preprint arXiv:2401.09003*.

OpenR1 Team. 2025. Openr1-math-220k: A large-scale dataset for mathematical reasoning. `https://huggingface.co/datasets/open-r1/OpenR1-Math-220k`. Version: default split, 94k problems with verified reasoning traces. Licensed under Apache 2.0.

Jiangbo Pei, Peiyu Liu, Xin Zhao, Aidong Men, and Yang Liu. 2025. Socratic style chain-of-thoughts help LLMs to be a better reasoner. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 12384–12395.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*. Version 3, revised on 29 July 2024.

Parker Seegmiller, Kartik Mehta, Soumya Saha, Chenyang Tao, Shereen Oraby, Arpit Gupta, Tagyoung Chung, Mohit Bansal, and Nanyun Peng. 2025. FLAMES: Improving LLM math reasoning via a fine-grained analysis of the data synthesis pipeline. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhihong Shao, Yuxiang Luo, Chengda Lu, Z.Z. Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. 2025. DeepSeekMath-V2: Towards self-verifiable mathematical reasoning. *arXiv preprint arXiv:2511.22570*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*. Version 3, revised on 27 April 2024.

Tongyi DeepResearch Team. 2025. Tongyi DeepResearch technical report. *arXiv preprint arXiv:2510.24701*.

Shaobo Wang, Zhengbo Jiao, Zifan Zhang, Yilang Peng, Xu Ze, Boyu Yang, Wei Wang, Hu Wei, and Linfeng Zhang. 2025. Socratic-Zero: Bootstrapping reasoning via data-free agent co-evolution. *arXiv preprint arXiv:2509.24726*.

Yongxian Wei, Yilin Zhao, Li Shen, Xinrui Chen, Runxi Cheng, Sinan Du, Hao Yu, Gang Liu, Jiahong Yan, Chun Yuan, and Dian Li. 2025. Learning to pose problems: Reasoning-driven and solver-adaptive data synthesis for large reasoning models. *arXiv preprint arXiv:2511.09907*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*. Version 1, submitted on 14 May 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR 2023)*. Also available as arXiv preprint arXiv:2210.03629v3.

Lili Yu, Weizhe Jiang, Hao Shi, Jing Yu, Zhengxu Liu, Yiran Zhang, James T. Kwok, Zhilin Li, Adrian Weller, and Weizhe Liu. 2024. MetaMath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations (ICLR)*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Aohan Zeng, Xin Lv, Qinkai Zheng, and 1 others. 2025. GLM-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*.

Zhengxin Zhang, Chengyu Huang, Aochong Oliver Li, and Claire Cardie. 2025. Better LLM reasoning via dual-play. *arXiv preprint arXiv:2511.11881*.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2025a. Absolute Zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*.

Xueliang Zhao, Wei Wu, Jian Guan, and Lingpeng Kong. 2025b. PromptCoT: Synthesizing olympiad-level problems for mathematical reasoning in large language models. *arXiv preprint arXiv:2503.02324*.

A Prompt Templates

This appendix documents all prompt templates used in our experimental pipeline. Each prompt is designed for a specific functional module (problem generation, solving, evaluation, verification). Placeholders such as {original_problem}, {problem}, etc., are replaced at runtime with concrete problem statements or answers during experiments. All prompts enforce strict output formatting to facilitate automated parsing and evaluation.

A.1 Problem Generation Prompt

Used by the problem designer module to generate novel math problems from a seed instance while preserving core difficulty and theme.

```
You are an innovative math problem designer.

Your task:
1. Internally analyze the components of the original problem.
   - Identify smallest sub-problems, required knowledge points, and constraints.
   - Consider how to modify them for creativity.
   - DO NOT include this analysis in your final output.

2. Based on your internal analysis, design a new problem that:
   - Keeps the overall theme and difficulty similar.
   - Changes at least one core element meaningfully.
   - Is clear, complete, and solvable.

Final Output format strictly as:
<new_problem>...full text of the new problem statement...</new_problem>

Original problem:
<original_problem>{original_problem}</original_problem>

Requirements:
- Do NOT solve the problem or include answers.
- The output must contain ONLY the <new_problem>...</new_problem> tag with the problem statement.
```

Listing 1: Prompt for problem generation.

A.2 Problem Solving Prompt

Used by solver models to produce step-by-step reasoning and extract final answers. Identical variants were used across solver implementations.

```
You are a math-solving assistant.

1. Carefully read the problem provided.
2. Reason through the problem step-by-step (show your reasoning clearly).
3. Present a concise and structured final solution.
4. Wrap the final numeric or symbolic answer **only** inside <answer> tags.
5. Do not include any extra text inside the <answer> tags.

Problem:
{problem}
```

Listing 2: Prompt for problem solving.

A.3 Problem Quality Evaluation Prompt

Used by the reward model (Section 3.3.1) to score generated problems along five dimensions of surface and structural validity.

```
You are a math problem quality judge.

You will be given:
- The original problem
- A new problem derived from it

Your task:
Rate the new problem in exactly 5 dimensions (each 1  5  points):

1. Core knowledge retention: How well does the new problem keep the key concepts of the original?
2. Innovation: How creative and non-trivial is the variation?
3. Difficulty match: Is the difficulty level similar to the original?
4. Reasoning complexity: Does the problem require a reasoning process of similar depth?
5. Coherence: Is the problem statement natural, clear, and mathematically sound?

Scoring rules (1  5  integers only):
- 1 = very poor / strong reject
- 2 = poor / weak reject
- 3 = acceptable / average
- 4 = good
- 5 = excellent / strong accept

Output requirements:
- Output only a Python-style list of 5 integers in the above order
- Example: [4, 5, 3, 4, 4]
- No explanation, no extra text

Original problem:
{original_problem}

New problem:
{new_problem}

Output:
```

Listing 3: Prompt for problem quality evaluation.

A.4 Answer Verification Prompt

Used by the verification module to judge answer correctness under mathematical equivalence.

```
You are a math expert. Determine whether
    ↪  the predicted answer matches the
    ↪  actual (ground truth) answer in
    ↪ meaning.

Do not perform the calculation again
    ↪ unless necessary.

Actual Answer: {actual}
Predicted Answer: {predicted}

Consider:
- Equivalent forms (e.g., 1/2 vs 0.5,
    ↪ simplified vs unsimplified)
- Case-insensitive string match
- Mathematical equivalence

Think briefly, then respond with:
<result>CORRECT</result> or <result>
    ↪ INCORRECT</result>
```

Listing 4: Prompt for answer verification (Section 4.5).

Implementation note. All prompts were templated using Python f-strings or `str.format()`. Runtime configurations (e.g., temperature, max tokens) were fixed per stage; full training and inference scripts will be released upon publication.

B Training Hyperparameters

This appendix provides the detailed training configurations used in different stages of our experimental pipeline. We list the hyperparameters for: (a) the first-stage GRPO training, (b) the second-stage DPO training, and (c) the SFT training used in verification experiments. All parameters follow YAML format as in our implementation scripts.

B.1 First-stage GRPO Training

```
training:
  learning_rate: 5e-6
  use_vllm: true
  loss_type: grpo
  epsilon: 0.2
  adam_beta1: 0.9
  adam_beta2: 0.99
  beta: 0.01
  weight_decay: 0.1
  warmup_ratio: 0.3
  lr_scheduler_type: cosine
  optim: paged_adamw_8bit
  logging_steps: 1
  per_device_train_batch_size: 1
  gradient_accumulation_steps: 2
  num_generations: 12
  max_prompt_length: 1024
  max_completion_length: 8192
  num_train_epochs: 2
  # max_steps: 1000
  save_total_limit: 2
  save_steps: 500
  max_grad_norm: 0.1
  report_to: swanlab
  output_dir: /tmp/outputs/grpo
  temperature: 0.7
  seed: 42
  fp16: false
  bf16: true
```

Listing 5: GRPO training hyperparameters for the first stage.

B.2 Second-stage DPO Training

```
training:
  beta: 0.1
  learning_rate: 1e-5
  optim: adamw_8bit
  max_prompt_length: 1024
  max_completion_length: 8192
  per_device_train_batch_size: 4
  gradient_accumulation_steps: 4
  num_train_epochs: 3
  warmup_ratio: 0.1

  report_to: swanlab
  mid_run_name: MPG-mid-1209-1
  hard_run_name: MPG-hard-

  save_total_limit: 2
  save_steps: 500
```

Listing 6: DPO training hyperparameters for the second stage.

B.3 SFT Training in Verification Experiment

```
training:
  learning_rate: 5e-5
  use_vllm: true
  adam_beta1: 0.9
  adam_beta2: 0.99
  weight_decay: 0.1
  warmup_ratio: 0.1
  lr_scheduler_type: cosine
  optim: paged_adamw_8bit
  logging_steps: 1
  per_device_train_batch_size: 1
  gradient_accumulation_steps: 2
  max_prompt_length: 1024
  max_completion_length: 8192
  num_train_epochs: 2
  # max_steps: 1000
  save_steps: 500
  max_grad_norm: 0.1
  report_to: swanlab
  output_dir: /tmp/outputs/sft
  temperature: 0.7
  seed: 42
  bf16: true
```

Listing 7: SFT training hyperparameters for verification experiments.