

CPMÖBIUS: ITERATIVE COACH-PLAYER REASONING FOR DATA-FREE REINFORCEMENT LEARNING

Anonymous authors

Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) have demonstrated strong potential in complex reasoning, yet their progress remains fundamentally constrained by reliance on massive high-quality human-curated tasks and labels, either through supervised fine-tuning (SFT) or reinforcement learning (RL) on reasoning-specific data. This dependence renders supervision-heavy training paradigms increasingly unsustainable, with signs of diminishing scalability already evident in practice. To overcome this limitation, we introduce **CPMöbius**, a collaborative **Coach-Player** paradigm for data-free reinforcement learning of reasoning models. Unlike traditional adversarial self-play frameworks, **CPMöbius** inspired by multi-agent collaboration treats the Coach and Player as independent but cooperative roles. The Coach proposes instructions targeted at the Player’s capability and receives rewards based on changes in the Player’s performance, while the Player is rewarded for solving the increasingly instructive tasks generated by the Coach. This cooperative optimization loop is designed to directly enhance the Player’s mathematical reasoning ability. Remarkably, **CPMöbius** achieves substantial improvement without relying on any external training data, outperforming existing unsupervised approaches. For example, on Qwen2.5-Math-7B-Instruct, our method improves accuracy by overall average +4.9 and out-of-distribution average +5.4, which exceed RENT for +1.5 on overall accuracy and R-zero for +4.2 on OOD accuracy.

1 INTRODUCTION

Large Language Models (LLMs) (OpenAI, 2025a; Yang et al., 2024a; Touvron et al., 2023) have demonstrated remarkable capabilities in complex reasoning tasks, from mathematical reasoning, problem solving (Wei et al., 2022) to code generation (Chen et al., 2021). The dominant paradigm for enhancing these abilities involves post-training on domain-specific data, typically through supervised fine-tuning (SFT) (Ouyang et al., 2022; Tunstall et al., 2023) followed by reinforcement learning (RL) (Christiano et al., 2017; Schulman et al., 2017). While effective, these approaches are fundamentally constrained by its reliance on massive, high-quality, human-curated datasets. The scarcity of such expert-produced examples means this highly supervision-dependent paradigm is showing signs of strain, raising concerns about its long-term scalability.

To break free from this dependency, a promising frontier has emerged in data-free learning, where models improve through autonomous interaction. Self-play, a concept inspired by game-playing AI (Silver et al., 2017), has been adapted for LLM reasoning to achieve self-evolving. Recent self-play frameworks in RL (Huang et al., 2025; Zhao et al., 2025) have shown that a model can generate its own training data and learn from solving them, entirely removing the need for external datasets. However, these pioneering methods are often built on an adversarial or competitive dynamic, where the model in one role generates challenges to stump another. Such an adversarial setup is prone to instability, collapsing into nonsensical or unlearnable proposed tasks for RL training.

In this work, we propose **CPMöbius**, a Coach-Player paradigm for data-free reinforcement learning, inspired by multi-agent collaboration (Chen et al., 2024; Qian & Cong, 2023). Instead of casting the Player model as competitors, the Coach is responsible for adapting the task difficulty to the Player’s capabilities. **CPMöbius** treat the Coach and Player model as independent but cooperative partners in a symbiotic learning process. Shown in Fig. 1, the Coach and Player model are optimized through a cooperative loop:

- The **Coach** model acts as a curriculum designer, proposing maximally instructive tasks targeted at the Player’s current capability.
- The **Player** model focuses on solving these tasks to enhance its reasoning skills.
- The **reward signals** for both Coach and Player are designed to foster cooperation. The Coach is rewarded based on the **changes in the Player’s validation accuracy**, directly incentivizing it to generate instructions that lead to tangible learning progress. Simultaneously, the Player is rewarded via a **standard verifiable outcome** for correctly solving tasks provided by the Coach.

This collaborative dynamic allows **CPMöbius** to generate a highly targeted and adaptive curriculum from scratch, tailored specifically to the Player’s evolving needs throughout the training process. Our experiments show that this data-free, cooperative approach is not only viable but remarkably effective. Without relying on any external training data, **CPMöbius** achieves substantial improvements and outperforms existing unsupervised methods. For instance, on the Qwen2.5-Math-7B-Instruct, our method improves accuracy by overall average +4.9 and out-of-distribution average +5.4, a significant leap compared to the +1.5 from RENT, a method of reinforcement learning via entropy minimization (Prabhudesai et al., 2025) and +4.2 from R-zero. The details of these baseline’s method are provided in Section 4.1 These results demonstrate the effectiveness and scalability of collaborative paradigm as a new pathway for advancing mathematical reasoning in LLMs, decoupling their progress from the constraints of human supervision.

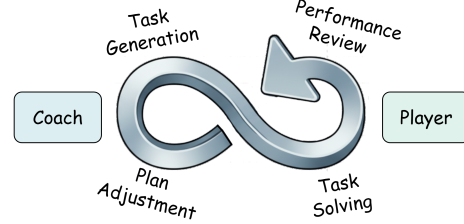


Figure 1: **CPMöbius** starts with the coach proposing tasks of suitable difficulty. The player learns by solving these tasks, then reviews on a predefined validation set. Finally, the coach adjusts the next training plan based on the player’s performance.

2 PRELIMINARIES

In this section, we briefly review two key RL methods for LLM that are relevant to our framework.

2.1 GROUP RELATIVE POLICY OPTIMIZATION

Group Relative Policy Optimization (GRPO) (Shao et al., 2024), proposed by DeepSeek, is a *critic-free* reinforcement learning algorithm. Given a query q , GRPO samples G candidate outputs $\{o_1, \dots, o_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$, and defines the normalized advantage function using the corresponding rewards $\{r_1, \dots, r_G\}$:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \dots, r_G\})}{\text{std}(\{r_1, r_2, \dots, r_G\})} \quad (1)$$

The policy π_{θ} is then updated by maximizing the following objective:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q, \{o_i\}} \left[\frac{1}{G} \sum_{i=1}^G \min(r_i(\theta) A_i, \text{clip}(r_i(\theta), 1 - \epsilon, 1 + \epsilon) A_i) \right] - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \quad (2)$$

where ϵ and β are hyperparameters, $r_i(\theta) = \frac{\pi_{\theta}(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}$ is the importance sampling ratio, and $D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}})$ is the KL divergence regularization with respect to a reference model.

2.2 REINFORCEMENT LEARNING WITH VERIFIABLE REWARDS

Reinforcement Learning with Verifiable Rewards (RLVR) is a framework that trains models using verifiable reward functions without relying on human feedback (Lambert et al., 2024). In RLVR, the reward function is typically defined by deterministic rules that automatically assess the correctness of model outputs, providing binary signals (1 for correct, 0 for incorrect):

$$r(y) = \text{verify}(y), \quad (3)$$

where $\text{verify}(\cdot)$ is a verifiable function determining whether the output y is correct.

Depending on the verification source, rewards can be obtained in different ways. When ground truth labels y^* are available, the accuracy is determined by direct comparison $r(y) = \mathbb{1}[y = y^*]$, as in Group Relative Policy Optimization (GRPO), where rules-based rewards check both the accuracy of the solutions and the required output format. In the absence of labels, verification can be performed in an unsupervised manner using self-consistency (Wang et al., 2023; Zuo et al., 2025), where the majority-voted answer from a set of candidate outputs is treated as the correct answer and rewards are assigned accordingly. This formulation highlights that verifiable rewards can be constructed either with or without supervision, enabling reinforcement learning to be applied even in data-scarce or fully unsupervised reasoning scenarios.

3 FRAMEWORK

In this section, we present a comprehensive overview of **CPMöbius**, a collaborative Coach–Player paradigm for data-free reinforcement learning. **CPMöbius** introduces a symbiotic learning loop between two independent language models: the **Coach**, a curriculum designer, and the **Player**, a reasoning solver.

The core objective is to maximize learning progress without human-curated data. To achieve this, the Coach generates mathematical tasks tailored to the Player’s current capability, while the Player attempts to solve them. The key innovation lies in the cooperative reward mechanism: the Coach is optimized not to stump the Player, but to maximize the Player’s capability based on Coach-proposed tasks. This ensures that the curriculum remains instructive, learnable, and adaptive.

We illustrate the main framework in Fig. 2, and the pseudo-code of algorithm can be found in Appendix A.2. Formally, let π_θ^C denote the Coach policy and π_ϕ^P the Player policy. At each round t :

1. **Coach designs training plan.** The Coach generates a batch of m task instruction $\{x_i\}_{i=1}^m \sim \pi_{\theta_t}^C(\cdot)$, where $\pi_{\theta_t}^C$ is the current Coach policy.
2. **Player trains.** For every x_i the current Player produces n independent answers $\{y_{i,j}\}_{j=1}^n \sim \pi_{\phi_t}^P(\cdot|x_i)$. Majority voting over the n answers yields a *pseudo-label* y_i^* . Then each answer receives a verifiable reward $r_{i,j} = \mathbb{I}[y_{i,j} = y_i^*]$ as well as a GRPO advantage $A_{i,j}$ computed w.r.t. the n samples for question i . The *instruction-level train reward* is obtained by averaging: $R_i^{\text{Player}} = \frac{1}{n} \sum_{j=1}^n r_{i,j}$. The set $\{(x_i, \{y_{i,j}\}_{j=1}^n)\}_{i=1}^m$ constitutes one GRPO batch, and Player parameters ϕ_t are updated using GRPO method while keeping KL within a trust-region.
3. **Player evaluate results.** The updated Player is evaluated on a fixed held-out validation set D_{val} , yielding a *progress reward* $\Delta_t = \text{Acc}_{\text{val}}(\pi_{\phi_{t+1}}^P; D_{\text{val}}) - \text{Acc}_{\text{val}}(\pi_{\phi_t}^P; D_{\text{val}})$, which measures the Player’s accuracy difference on the validation set D_{val} .
4. **Coach adjusts plan.** Each instruction x_i is assigned an *instruction reward* $R_i^{\text{Coach}} = R_i^{\text{Player}} \cdot \Delta_t$, i.e., instructions that produced high Player rewards and coincided with a global accuracy improvement are reinforced. A group of m instruction-level REINFORCE steps update Coach parameters θ_t using each instance in the batch $\{(x_i, R_i^{\text{Coach}})\}_{i=1}^m$.

The entire loop is trained end-to-end with separate policy optimization for Coach and Player using the REINFORCE and GRPO, respectively. Critically, no human prompts and no external curricula are ever used. The Coach learns to teach, and the Player learns to solve, purely through interaction with each other. This cooperative design sidesteps the instability of adversarial self-play while retaining the benefits of open-ended, adaptive curriculum generation. In the following subsections, we detail the architecture, reward design, and training procedure of both the Coach and the Player.

3.1 COACH

The Coach serves as an adaptive *curriculum designer*, fundamentally responsible for generating tasks that improve the Player’s current reasoning capabilities. Unlike traditional static curriculum approaches, our Coach operates as a dynamic learning policy that continuously refines its task generation strategy based on the Player’s learning trajectory. The Coach never sees ground-truth

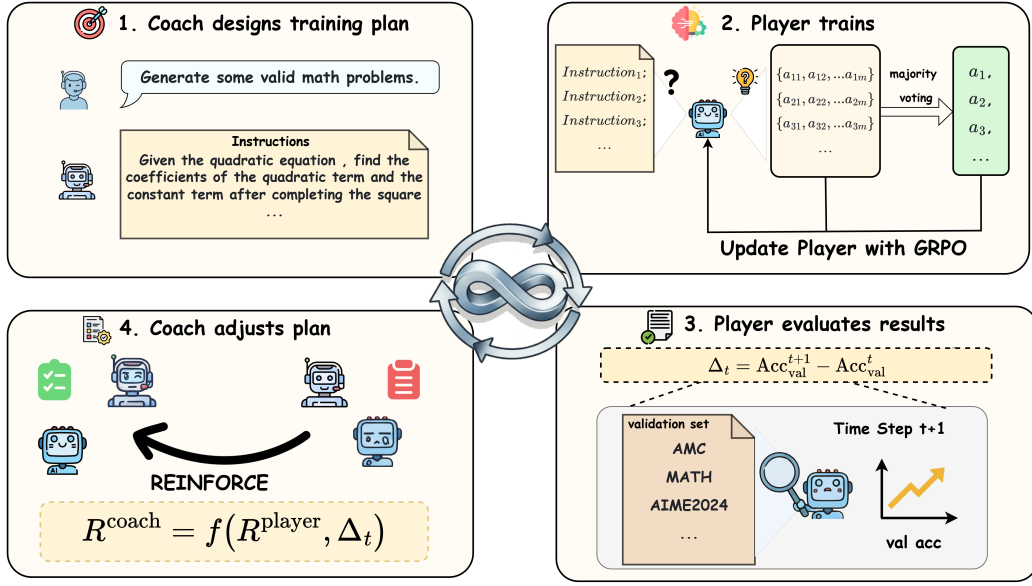


Figure 2: The illustration on the conceptual layered architecture on the design of CPMöbius. The iterative process includes four stages. **Coach designs training plan:** The coach gives instructions of suitable difficulty based on the player’s current ability. **Player trains:** The player executes each instruction multiple times, uses majority voting to get pseudo-labels, and updates with GRPO. **Player evaluates results:** The updated player is tested on a prepared validation set, and the accuracy of validation is recorded. **Coach adjusts plan:** The coach updates with REINFORCE, using the player’s performance on both the proposed instructions and the validation set as rewards.

solutions, while its only feedback sign is the scalar Δ_t , the process on the held-out validation set after the Player has been updated.

Difficulty-Filtered Batching To ensure every proposal task is *learnable yet non-trivial*, we use a lightweight difficulty check during the task-generation phase. For each candidate task x_i sampled from π_θ^C , we rollout n Player answers $\{y_{i,j}\}_{j=1}^n \sim \pi_\phi^P(\cdot|x_i)$, obtain the majority-voted pseudo-label y_i^* , and compute the rollout-dependent accuracy score of the instruction.

$$\text{acc}_i = \frac{1}{n} \sum_{j=1}^n \mathbb{I}[y_{i,j} = y_i^*]. \quad (4)$$

This score effectively measures the problem’s alignment with the Player’s current capability frontier. The Coach then applies a principled filtering criterion, retaining only problems whose accuracy scores fall within the pedagogically optimal zone of $0.2 \leq \text{acc}_i \leq 0.8$. Problems outside this range are immediately discarded and replaced through on-the-fly resampling. This online filter guarantees that the final mini-batch of m questions is challenging enough to promote skill development yet solvable enough to avoid frustration, providing a natural curriculum ramp.

Design Objectives The Coach embodies a learner-centered educational philosophy, where its primary objective is to optimize the constructiveness of proposed-task for the Player. Formally, the Coach policy π_θ^C is optimized using instruction-level rewards that combine local training effectiveness with global educational outcomes.

$$R_i^{\text{Coach}} = R_i^{\text{Player}} \cdot \Delta_t \quad (5)$$

where $R_i^{\text{Player}} = \frac{1}{n} \sum_{j=1}^n r_{i,j}$ represents the average training reward achieved by the Player on instruction x_i , and $\Delta_t = \text{Acc}_{\text{val}}(\pi_{\phi_{t+1}}^P) - \text{Acc}_{\text{val}}(\pi_{\phi_t}^P)$ measures the Player’s accuracy improvement on the validation set.

This multiplicative reward embodies a pedagogical principle: proposed tasks receive positive reinforcement only when they **simultaneously achieve high Player performance during training**

(high R_i^{Player}) and contribute to measurable learning progress (positive Δ_t). The Coach parameters are updated through REINFORCE using the batch of instruction-reward pairs $\{(x_i, R_i^{\text{Coach}})\}_{i=1}^m$:

$$\nabla_{\theta} J(\theta) = \frac{1}{m} \sum_{i=1}^m R_i^{\text{Coach}} \nabla_{\theta} \log \pi_{\theta}^{\text{C}}(x_i). \quad (6)$$

3.2 PLAYER

The Player functions as the primary reasoning model, designed to develop robust mathematical problem-solving capabilities through iterative interaction with the Coach-generated curriculum.

Design Objectives and Collaborative Dynamics The Player’s core objective is to maximize solving accuracy on mathematical problems while developing generalizable reasoning strategies. The Player operates within a collaborative learning framework where its performance directly influences curriculum adaptation through a sophisticated feedback mechanism. The Player’s learning process is also inherently adaptive, continuously calibrating its problem-solving strategies based on feedback from the Coach-generated curriculum.

The Player employs multi-sample reasoning for each problem x_i , generating n independent solution attempts $\{y_{i,j}\}_{j=1}^n$ to enable robust pseudo-label generation through majority voting. This approach mitigates individual reasoning errors, provides confidence estimates for generated solutions, and creates multiple learning signals from each instructional instance.

The interaction protocol between the Coach and Player establishes a dynamic feedback loop that drives mutual improvement. This ensures the curriculum remains at an optimal difficulty, maintaining learning momentum and continuously pushing the frontier of the Player’s capabilities.

Training and Optimization The Player is optimized using GRPO, which enables stable learning from the pseudo-labels generated through majority voting. For each problem instance x_i , the Player receives rewards

$$r_{i,j} = \mathbb{I}[y_{i,j} = y_i^*], \quad (7)$$

where y_i^* is the majority-voted pseudo-label. The GRPO advantage computation considers the relative performance across the n samples for each problem:

$$A_{i,j} = \frac{r_{i,j} - \text{mean}(\{r_{i,1}, r_{i,2}, \dots, r_{i,n}\})}{\text{std}(\{r_{i,1}, r_{i,2}, \dots, r_{i,n}\})} \quad (8)$$

This collaborative process completes the **CPMöbius** training loop: the Coach designs training curriculum, the Player explores potential solutions, and the Player’s consequent capability guides the curriculum’s evolution. The process is inherently curriculum-aware, prioritizing challenging yet solvable problems to ensure the Player’s skill development remains aligned with the Coach’s adaptive strategy. Through this orchestrated interaction, the framework achieves data-free mathematical reasoning development, where both models co-evolve to maximize learning efficiency without reliance on human-curated data or a pre-defined curriculum.

4 EXPERIMENTS

4.1 EXPERIMENT SETUP

Models Selection We select four base models for our training experiments, representing the three main stages of a typical LLM training lifecycle: pre-training, supervised fine-tuning (SFT), and reinforcement learning:

- **Qwen2.5-Math-1.5B** (Yang et al., 2024b): mathematical pre-training model.
- **OpenMath-Nemotron-1.5B** (Moshkov et al., 2025): large-scale SFT enhanced model based on Qwen2.5-Math-1.5B.
- **Qwen2.5-Math-7B-Instruct** (Yang et al., 2024b) and **OctoThinker-3B-Hybrid-Zero** (Wang et al., 2025): models optimized via reinforcement learning.

Table 1: Performance comparison of **CPMöbius** against baseline methods across mathematical reasoning benchmarks. Overall Average represents the mean performance across all benchmarks. OOD Average denotes the out-of-distribution performance, calculated as the mean across all benchmarks excluding AMC datasets, since RENT was trained on AMC and **CPMöbius** validation utilized AMC. This separation ensures fair comparison by distinguishing in-distribution (AMC) from out-of-distribution generalization performance. Bold values indicate best performance for each metric.

Models	Average	OOD Average	AMC	AIME 2024	AIME 2025	Minerva	MATH	Olympiad
<i>Qwen2.5-Math-1.5B</i>								
Base Model	23.3	19.8	34.6	6.2	2.8	16.3	56.2	23.4
R-Zero (Iter 3)	27.1	24.7	39.2	9.8	5.0	19.3	62.4	26.8
RENT	27.1	24.7	39.3	10.0	5.0	19.0	62.2	27.1
CPMöbius	28.8	26.8	39.4	9.8	5.4	28.0	63.1	26.9
<i>OpenMath-Nemotron-1.5B</i>								
Base Model	59.5	54.9	82.3	55.6	43.3	25.1	89.4	61.0
R-Zero (Iter 3)	-	-	-	-	-	-	-	-
RENT	61.7	56.5	87.7	55.0	46.0	24.2	90.7	66.7
CPMöbius	62.1	57.0	87.5	54.9	46.9	24.3	91.2	67.9
<i>OctoThinker-3B-Hybrid-Zero</i>								
Base Model	21.3	20.6	24.6	3.9	1.7	16.3	57.9	23.4
R-Zero (Iter 3)	20.5	19.5	25.9	2.0	0.3	14.6	58.1	22.3
RENT	23.0	21.7	29.2	7.3	2.1	15.0	60.2	24.1
CPMöbius	23.6	22.0	28.0	4.8	1.7	22.1	60.4	24.7
<i>Qwen2.5-Math-7B-Instruct</i>								
Base Model	35.8	33.0	49.2	9.0	6.3	34.6	78.0	37.4
R-Zero (Iter 3)	36.9	34.2	50.5	9.5	7.4	32.7	83.3	38.1
RENT	39.2	37.6	53.1	10.8	9.9	38.8	83.8	38.8
CPMöbius	40.7	38.4	55.6	11.8	9.6	44.9	84.2	38.3

More details about these model are introduced in Appendix A.3.

Training Details. All experiments were conducted within the verl (Sheng et al., 2025). We use AMC as the fixed held-out validation D_{val} during training. The Coach model was initialized using Qwen2.5-7B-Instruct with a preliminary cold-start phase on mathematical problems sourced from the PRIME Eurys-2-RL-Data (Cui et al., 2025). All experiments were conducted using 4 to 8 NVIDIA A800-80GB GPUs per setting. We set the batch size as 16 and the number of rollout samples for each prompt as 16, ensuring that each training round involves the Coach generating 16 questions and the Player producing 16 candidate solutions for majority voting-based pseudo-label generation. More hyperparameter configurations and prompt templates are provided in Appendix A.4.

Evaluation Details. We evaluate the Player models on six established mathematical reasoning benchmarks spanning diverse difficulty levels: AMC, Minerva (Lewkowycz et al., 2022), MATH-500 (Hendrycks et al., 2021), Olympiad-Bench (He et al., 2024), and AIME 2024 and AIME 2025. To ensure robustness, we employ benchmark-specific sampling strategies calibrated to each benchmark’s difficulty: mean@32 for AIME benchmarks, mean@10 for AMC, mean@6 for Minerva, mean@5 for MATH-500, and mean@3 for Olympiad-Bench. Since AMC is used as the validation set during training, we compute both the average score on all six datasets and the OOD average score on the other five datasets except for AMC. All sampling settings are kept consistent with the training configuration, as illustrated in Appendix A.4.

Baselines. For our main experiments, beyond the selected base models, we considered two representative unsupervised training paradigms as baselines. The first is RENT (Prabhudesai et al., 2025), which employs entropy minimization: the model’s own confidence in its generated answers is treated as a reward signal, without relying on external feedback. The second is R-Zero (Huang et al., 2025), which initializes two roles of the same model that interact adversarially, with the challenger generating tasks and the solver attempting to solve them.

4.2 RESULTS

We present the main results in Table 1. We have the following observations:

CPMöbius outperforms other unsupervised RL methods: The results show that **CPMöbius** achieves superior performance compared to other unsupervised RL baselines, consistently achiev-

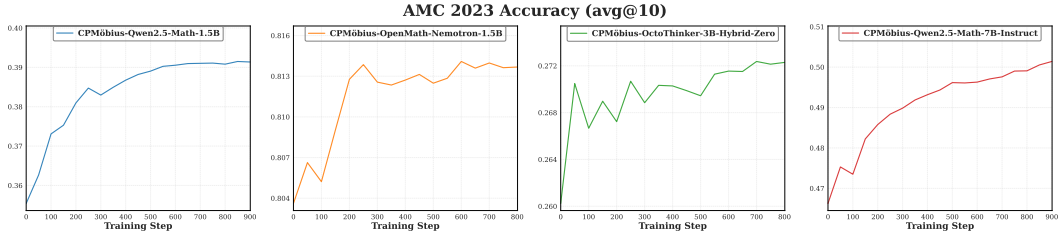


Figure 3: Visualization of the training dynamics of **CPMöbius** using validation results on AMC dataset. The curves are smoothed with Time Weighted EMA, where **CPMöbius** shows consistent performance improvement for different base models.

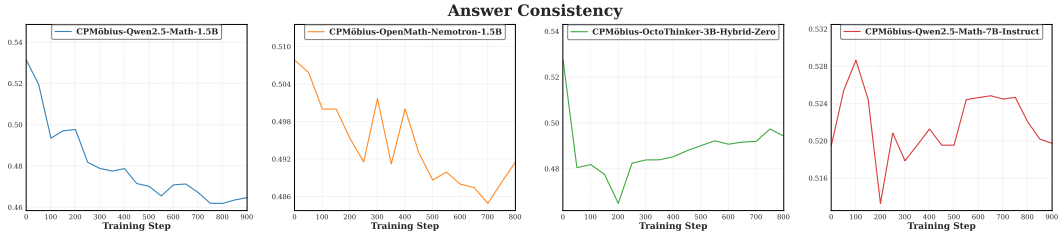


Figure 4: Visualization of the Player’s answer consistency on Coach proposed tasks during training. A lower value indicates higher difficulty of the instructions.

ing the highest overall average and OOD average scores across all four base model. Impressively, **CPMöbius** successfully improves high-performing base models OpenMath-Nemotron-1.5B (from 59.5 to 62.1). Notably, we found that the method from R-Zero failed on OpenMath-Nemotron-1.5B, failing to be trained as a Challenger as required by R-Zero. This demonstrates **CPMöbius**’s ability to push models beyond their apparent performance ceiling, a critical advantage for practical applications where starting from pre-optimized models is common.

Strong out-of-distribution generalization: **CPMöbius** achieves better OOD average scores across all four tested models, demonstrating that the reasoning capabilities learned from AMC competition problems effectively transfer to diverse mathematical domains. On MATH, **CPMöbius** consistently outperforms other methods with improvements ranging from 1.8 to 6.9 points over base models. The most striking OOD generalization occurs on the Minerva benchmark, where **CPMöbius** achieves obvious improvements: from 16.3 to 28.0 (71.8%) on Qwen2.5-Math-1.5B and 34.6 to 44.9 (29.8%) on Qwen2.5-Math-7B-Instruct.

Performance analysis for different initial models: The experimental results reveal distinct performance patterns that correlate with initial model characteristics. (1) Foundation models demonstrate high improvement potential: Qwen2.5-Math-1.5B achieves an overall 5.5 points improvement (23.6% relative gain), suggesting that models with domain-specific pre-training provide strong foundations for **CPMöbius**’s optimization approach. (2) SFT-enhanced models show diminishing but meaningful returns: Despite starting from a high 59.5 points baseline after extensive SFT on 5.5 million instances, OpenMath-Nemotron-1.5B still achieves an overall 2.6 points improvement, demonstrating **CPMöbius**’s ability to push beyond traditional SFT limits. (3) RL-optimized models exhibit varied enhancement: Qwen2.5-Math-7B-Instruct shows remarkable 4.9 points improvement despite instruction tuning, while OctoThinker-3B-Hybrid-Zero shows modest 2.3 points gains.

4.3 TRAINING DYNAMICS

We analyze the training dynamics of **CPMöbius** by tracking both validation accuracy on AMC and the consistency of the Player’s responses throughout training steps. As shown in Fig 3, **CPMöbius** steadily improves the Player’s performance across all four base models, indicating that the cooperative Coach–Player optimization loop enables stable and continual reasoning enhancement. The

Table 2: Ablation study results are based on the Qwen2.5-Math-1.5B base model. **w/o Coach Update**: disables training of the Coach. **w/o Coach Cold Start**: uses the base model as the Coach. **w/o Instruction Filter**: disables difficulty filtering by the Coach.

Models	Average	OOD Average	AMC	AIME 2024	AIME 2025	Minerva	MATH	Olympiad
<i>Qwen2.5-Math-1.5B</i>								
Base Model	23.3	19.8	34.6	6.2	2.8	16.3	56.2	23.4
CPMöbius	28.8	26.8	39.4	9.8	5.4	28.0	63.1	26.9
Ablation								
└ w/o Coach Update	25.3	23.1	36.7	8.7	4.8	17.2	58.4	26.3
└ w/o Coach warm-up	23.7	21.2	36.1	9.2	3.6	13.8	54.4	24.8
└ w/o Instruction Filter	24.9	22.5	37.3	9.0	3.5	16.6	58.4	24.9

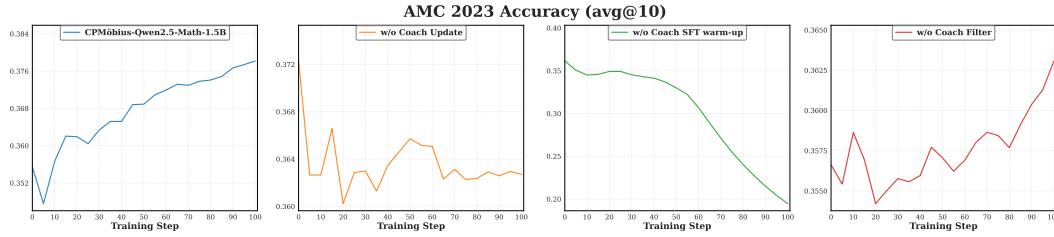


Figure 5: Visualization of the training dynamics on **CPMöbius** and different ablation experiments using validation results on AMC dataset.

performance gains are gradual yet consistent, demonstrating that the curriculum adapts effectively to the Player’s evolving capabilities.

Fig 4 illustrates the evolution of answer consistency, where lower values correspond to more challenging tasks proposed by the Coach. Notably, for Qwen2.5-Math-1.5B and OpenMath-Memotron-1.5B, two reasoning models without previous RL training, the downward trends in consistency indicates that the Coach progressively generated questions of increasing difficulty, maintaining the Player within an optimal learning zone. For OctoThinker-3B-Hybrid-Zero and Qwen2.5-Math-7B-Instruct, with better performance benefiting from previous RL training, the difficulty maintains a reasonably range.

Additionally, we found that the length of problems proposed by the Coach is increasing, indicating that the Coach gradually generates more complex tasks to adapt to the Player’s growing capabilities. Meanwhile, the Player’s response length is decreasing, suggesting that the Player is generating increasingly efficient answers. Details can be found in Appendix A.5 and Appendix A.6. Together, these results highlight that **CPMöbius** not only drives performance improvement but also naturally induces a self-adjusting curriculum based on the Player’s performance.

4.4 ABLATION STUDY

To systematically evaluate the individual contributions of each core component within **CPMöbius**, we conduct a thorough ablation study on the Qwen2.5-MATH-1.5B model. We examine the relative importance of three critical modules (i.e., Coach update, Coach SFT warm-up, and instruction filter) by selectively removing each component and measuring the resulting performance degradation across multiple mathematical reasoning benchmarks. The comprehensive results of this ablation analysis are presented in Table 2, and training dynamics of different ablations are shown in Fig 5.

Coach Update. ablation fixes the Coach model throughout training instead of adapting it to the Player’s evolving performance. This dynamic adaptation mechanism enables personalized curriculum generation tailored to the Player’s current capabilities, creating a co-evolutionary learning dynamic. Removing Coach updates degrades average accuracy from 28.8% to 25.3%, with out-of-distribution (OOD) performance dropping from 26.8% to 23.1%, demonstrating the critical importance of adaptive instruction.

Coach Cold Start. ablation eliminates the initial cold start phase. This initialization ensures that the Coach can generate high-quality math problems from the outset, establishing a strong foundation

for subsequent cooperative training. Without warm-up, average accuracy drops to 23.7% (OOD: 21.2%), indicating that proper Coach initialization is essential for effective curriculum generation.

Instruction Filter. ablation removes the difficulty calibration mechanism that maintains problems within the optimal learning zone, where the accuracy is between 0.2 and 0.8. This filter ensures generated problems remain challenging yet solvable, maintaining the Player at its capability frontier. Disabling this mechanism reduces average accuracy to 24.9% (OOD: 22.5%), confirming that appropriate difficulty calibration is crucial for efficient learning.

5 RELATED WORK

Reinforcement Learning with Verifiable Rewards. Recent advances in language model reasoning have leveraged Reinforcement Learning with Verifiable Rewards (RLVR), in which models are trained using binary feedback derived from programmatic verifiers that check correctness against ground truth (Lambert et al., 2024; Guo et al., 2025). By replacing learned reward models with rule-based verifiers, RLVR enables reliable optimization and mitigates reward hacking. Leading systems (Jaech et al., 2024; OpenAI, 2025b;a; Agarwal et al., 2025a; Comanici et al., 2025; Seed et al., 2025) demonstrate that RLVR can substantially improve reasoning and problem-solving abilities. Typical rule-based rewards include accuracy checks for deterministic outcomes and format constraints for structured outputs, both of which enhance the reliability and reproducibility of large-scale RL training pipelines. Despite their effectiveness, RLVR is fundamentally limited by the availability of verifiable supervision, which becomes increasingly costly as models surpass human-level expertise in specialized domains (Burns et al., 2023).

Self-Play and Co-Evolving Policy-Rewards. Self-play has emerged as a powerful paradigm for improving LLMs without relying solely on external supervision. In this approach, a model either generates its own training signals or interacts with a counterpart to refine both policy and reward (Yuan et al., 2024; Jiang et al., 2025). Techniques include self-rewarding, where a model critiques or corrects its own outputs (Xiong et al., 2025; Zhang et al., 2025; Team, 2025), and co-optimization, where the policy and a separate reward model are trained jointly to enhance robustness and reduce reward hacking (Zha et al., 2025; Hong et al., 2025; Lu et al., 2025). By unifying the roles of generator and verifier, self-play enables dynamic adaptation and continuous improvement, offering a scalable alternative to purely supervised or static reward schemes.

Data-Free Reinforcement Learning. To address the limitations of human-generated rewards, recent work has explored data-free RL methods that generate training signals automatically. Some approaches leverage a model’s own outputs or internal states, using consistency, confidence, or self-evaluation to guide learning (Zuo et al., 2025; Agarwal et al., 2025b; Li et al., 2025; Yuan et al., 2024). Others rely on external, automated signals, such as heuristics or the structure of large unlabeled corpora (Dong et al., 2025; Zweiger et al., 2025). More sophisticated methods combine these ideas, allowing models to generate problems for themselves, evaluate solutions, and iteratively refine both policy and reward (Zhao et al., 2025; Huang et al., 2025; Chen et al., 2025). Together, these data-free approaches provide scalable training for LLMs, enabling self-improvement without human labels, though they remain sensitive to reward misalignment and can exhibit failure modes such as collapse or repetitive behavior.

6 CONCLUSION

In this work, we introduced **CPMöbius**, a novel Coach-Player paradigm inspired by multi-agent collaboration to foster reasoning abilities in a completely data-free manner. Our paradigm’s core innovation lies in its cooperative optimization loop, where a coach model generates a targeted curriculum rewarded by the Player’s learning progress. This creates a dynamic that autonomously discovers a curriculum tailored to the Player’s evolving capabilities, successfully decoupling reasoning enhancement from any reliance on pre-existing tasks or human-curated labels. Our work validates that a collaborative, data-free reinforcement learning approach can be a powerful and efficient alternative to a supervision-heavy training mechanism. Future work may explore extending this collaborative paradigm to other complex domains. Furthermore, investigating the emergent properties and long-term stability of the co-evolution between models presents a compelling direction for future research.

7 ETHICS STATEMENT

This work introduces CPMöbius, a data-free reinforcement learning framework that enhances reasoning in large language models through a cooperative Coach–Player paradigm. Because our method does not require human-annotated data or human feedback during training, it avoids risks associated with large-scale human data collection, such as privacy concerns, labor exploitation, or biased supervision. All experiments were conducted on publicly available benchmark datasets (e.g., AMC, AIME, MATH, OlympiadBench), which are widely used in the research community for evaluating mathematical reasoning models. No personally identifiable, sensitive, or private data was used. Potential societal impacts include both positive applications, such as advancing safe autonomous reasoning systems, and risks, such as misuse for harmful automated problem-solving. We emphasize that CPMöbius is designed to improve verifiable mathematical reasoning, not to generate unverified or harmful content. Nonetheless, as with any reinforcement learning system, safeguards should be considered in future deployments to mitigate unintended misuse.

8 REPRODUCIBILITY STATEMENT

We have made every effort to ensure reproducibility of our results.

- Framework description: Section 3 of the paper details the CPMöbius training loop, including the cooperative optimization of Coach and Player policies, reward definitions, and curriculum adaptation.
- Baselines: We compare against RENT and R-Zero under identical evaluation settings to ensure fair benchmarking.
- Datasets: We evaluate on publicly available reasoning benchmarks (AMC, AIME 2024/2025, MATH, Minerva, OlympiadBench), clearly separating in-distribution and out-of-distribution splits.
- Hyperparameters: Appendix A.2 provides complete training hyperparameters for both Coach and Player models across all tested architectures, including batch sizes, learning rates, KL penalties, rollout numbers, and entropy coefficients.
- Model selection: We specify all base models used (Qwen2.5-Math-1.5B, Qwen2.5-Math-7B-Instruct, OpenMath-Nemotron-1.5B, OctoThinker-3B-Hybrid-Zero) along with their training histories.
- Evaluation protocol: All results are averaged across multiple benchmarks, with OOD generalization explicitly reported to ensure transparency.

Together, these details should allow independent researchers to reproduce our experiments and validate our findings.

REFERENCES

- Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv: 2508.10925*, 2025a.
- Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. *arXiv preprint arXiv:2505.15134*, 2025b.
- Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.
- Lili Chen, Mihir Prabhudesai, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Self-questioning language models. *arXiv preprint arXiv:2508.03682*, 2025.

- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Weize Chen, Ziming You, Ran Li, Yitong Guan, Chen Qian, Chenyang Zhao, Cheng Yang, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Internet of agents: Weaving a web of heterogeneous agents for collaborative intelligence. *arXiv preprint arXiv:2407.07061*, 2024.
- Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arxiv preprint arXiv: 2507.06261*, 2025.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.
- Qingxiu Dong, Li Dong, Yao Tang, Tianzhu Ye, Yutao Sun, Zhifang Sui, and Furu Wei. Reinforcement pre-training. *arXiv preprint arXiv:2506.08007*, 2025.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhatta, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Celebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Rapparthi, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collet, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, and Tobias Speckbacher. The Llama 3 Herd of Models. *arXiv e-prints*, art. arXiv:2407.21783, July 2024. doi: 10.48550/arXiv.2407.21783.

- 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.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL <https://aclanthology.org/2024.acl-long.211/>.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
- Haitao Hong, Yuchen Yan, Xingyu Wu, Guiyang Hou, Wenqi Zhang, Weiming Lu, Yongliang Shen, and Jun Xiao. Cooper: Co-optimizing policy and reward models in reinforcement learning for large language models. *arXiv preprint arXiv:2508.05613*, 2025.
- Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data, 2025. URL <https://arxiv.org/abs/2508.05004>.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- Yuhua Jiang, Yuwen Xiong, Yufeng Yuan, Chao Xin, Wenyuan Xu, Yu Yue, Qianchuan Zhao, and Lin Yan. Pag: Multi-turn reinforced llm self-correction with policy as generative verifier. *arXiv preprint arXiv:2506.10406*, 2025.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.
- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. *ArXiv*, abs/2206.14858, 2022. URL <https://api.semanticscholar.org/CorpusID:250144408>.
- Pengyi Li, Matvey Skripkin, Alexander Zubrey, Andrey Kuznetsov, and Ivan Oseledets. Confidence is all you need: Few-shot rl fine-tuning of language models. *arXiv preprint arXiv:2506.06395*, 2025.
- Songshuo Lu, Hua Wang, Zhi Chen, and Yaohua Tang. Urpo: A unified reward & policy optimization framework for large language models. *arXiv preprint arXiv:2507.17515*, 2025.
- Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.
- OpenAI. Gpt-5 system card. *Blog*, 2025a.
- OpenAI. Openai o3 and o4-mini system card. *Blog*, 2025b.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

- Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.
- Chen Qian and Xin Cong. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 6(3):1, 2023.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, et al. Seed1. 5-thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.
- 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.
- Guangming Sheng, Chi Zhang, Zilinfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.
- David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharmashan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.
- Kimi Team. Kimi k2: Open agentic intelligence, 2025. URL <https://arxiv.org/abs/2507.20534>.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro Von Werra, Clémentine Fourrier, Nathan Habib, et al. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*, 2023.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=1PLlNIMMrw>.
- Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025. URL <https://arxiv.org/abs/2506.20512>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. Self-rewarding correction for mathematical reasoning. *arXiv preprint arXiv:2502.19613*, 2025.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024a.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024b.
- Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 3, 2024.

- Kaiwen Zha, Zhengqi Gao, Maohao Shen, Zhang-Wei Hong, Duane S Boning, and Dina Katabi. Rl tango: Reinforcing generator and verifier together for language reasoning. *arXiv preprint arXiv:2505.15034*, 2025.
- Xiaoying Zhang, Hao Sun, Yipeng Zhang, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. Critique-grpo: Advancing llm reasoning with natural language and numerical feedback. *arXiv preprint arXiv:2506.03106*, 2025.
- Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.
- Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.
- Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. *arXiv preprint arXiv:2506.10943*, 2025.

A APPENDIX

A.1 USE OF LLM

In preparing this work, we use large language models (LLMs) to assist in writing and editing. Specifically, LLMs were employed to help refine the clarity and readability of certain sections and check for consistency in terminology. All technical content, experimental design, implementation details, and results were produced, verified, and analyzed independently by the authors. No part of the experimental process, including model training, data handling, or evaluation, relied on external LLM outputs.

A.2 PSEUDO-CODE FOR CPMÖBIUS

Algorithm 1 CoachPlayer Framework for Data-Free Reinforcement Learning

Require: Pretrained Coach LLM $\pi_{\theta_0}^C$; Player LLM $\pi_{\phi_0}^P$; validation set $\mathcal{D}_{val}$
Require: Batch size m ; samples per task n ; iterations T ; learning rates α_C, α_P

- 1: $\theta \leftarrow \theta_0, \phi \leftarrow \phi_0$ ▷ Initialize parameters
- 2: **for** $t \leftarrow 1$ **to** T **do**
- 3: $\mathcal{B} \leftarrow \emptyset$ ▷ COACH GENERATION PHASE
- 4: **while** $|\mathcal{B}| < m$ **do**
- 5: $x_{cand} \sim \pi_{\theta}^C(\cdot)$ ▷ Coach proposes candidate task
- 6: $\{y_j\}_{j=1}^n \sim \pi_{\phi}^P(\cdot | x_{cand})$ ▷ Player attempts task
- 7: $y^* \leftarrow \text{MajorityVote}(\{y_j\}_{j=1}^n)$ ▷ Compute pseudo-label
- 8: $acc \leftarrow \frac{1}{n} \sum_{j=1}^n \mathbb{I}[y_j = y^*]$ ▷ Calculate accuracy
- 9: **if** $0.2 \leq acc \leq 0.8$ **then**
- 10: $\mathcal{B} \leftarrow \mathcal{B} \cup \{x_{cand}\}$ ▷ Accept task if difficulty appropriate
- 11: **end if**
- 12: **end while** ▷ PLAYER TRAINING PHASE
- 13: **for** $i \leftarrow 1$ **to** m **do**
- 14: $\{y_{i,j}\}_{j=1}^n \sim \pi_{\phi}^P(\cdot | x_i)$ where $x_i \in \mathcal{B}$ ▷ Generate responses
- 15: $y_i^* \leftarrow \text{MajorityVote}(\{y_{i,j}\}_{j=1}^n)$ ▷ Pseudo-label
- 16: $r_{i,j} \leftarrow \mathbb{I}[y_{i,j} = y_i^*]$ for $j = 1, \dots, n$ ▷ Assign rewards
- 17: $A_{i,j} \leftarrow \frac{r_{i,j} - \bar{r}_i}{\sigma_i + \epsilon}$ ▷ GRPO advantages
- 18: $R_i^{Player} \leftarrow \frac{1}{n} \sum_{j=1}^n r_{i,j}$ ▷ Instruction-level reward
- 19: **end for**
- 20: $\phi \leftarrow \phi + \alpha_P \cdot \nabla_{\phi} \mathcal{L}_{GRPO}$ ▷ Update Player via GRPO
- 21: $\Delta_t \leftarrow \text{Acc}_{val}(\pi_{\phi}^P; \mathcal{D}_{val}) - \text{Acc}_{val}(\pi_{\phi_{old}}^P; \mathcal{D}_{val})$ ▷ EVALUATION PHASE
- 22: **for** $i \leftarrow 1$ **to** m **do**
- 23: $R_i^{Coach} \leftarrow R_i^{Player} \cdot \Delta_t$ ▷ Coach instruction reward
- 24: **end for** ▷ COACH UPDATE PHASE
- 25: $\theta \leftarrow \theta + \alpha_C \cdot \frac{1}{m} \sum_{i=1}^m R_i^{Coach} \nabla_{\theta} \log \pi_{\theta}^C(x_i)$ ▷ REINFORCE update
- 26: **end for**
- 27: **return** $\pi_{\theta}^C, \pi_{\phi}^P$ ▷ Trained Coach and Player policies

A.3 DETAILS OF BASED MODEL SELECTIONS

We select Qwen2.5-Math-1.5B, OpenMath-Nemotron-1.5B, Qwen2.5-Math-7B-Instruct and OctoThinker-3B-Hybrid-Zero as base models for our training experiments, representing the three main stages of a typical LLM training lifecycle: pre-training, supervised fine-tuning (SFT), and reinforcement learning.

Specifically, OpenMath-Nemotron-1.5B, which builds upon the Qwen2.5-Math-1.5B backbone with SFT on 5.5 million task instances, allows us to examine the impact of large-scale supervised training. In contrast, OctoThinker-3B-Hybrid-Zero, derived from Llama-3.2-3B-Base Grattafiori et al. (2024) through R1-Zero-style RL training, represents a fundamentally different approach to mathematical reasoning acquisition. Together, these models span a spectrum from mathematical foundation models to extensively fine-tuned variants to RL-optimized architectures, providing comprehensive coverage of contemporary approaches to mathematical reasoning in language models.

A.4 DETAILS OF TRAINING HYPERPARAMETER

This section summarizes training hyperparameters for the Coach and the Player.

A.4.1 COACH TRAINING

- **Train Batch Size:** 16
- **Learning Rate:** 1×10^{-6}
- **Temperature:** 0.7
- **Top-p:** 1.0
- **Number of Rollout:** 1
- **KL Penalty Coefficient:** 1×10^{-3}
- **Entropy Coefficient:** 1×10^{-2}
- **Total Steps:** 1000

A.4.2 PLAYER TRAINING

Qwen2.5-Math-1.5B

- **Train Batch Size:** 16
- **Learning Rate:** 1×10^{-6}
- **Response Length:** 2048
- **Temperature:** 0.6
- **Top-p:** 1.0
- **Number of Rollout:** 16
- **Repetition Penalty:** 1
- **KL Penalty Coefficient:** 1×10^{-3}
- **Entropy Coefficient:** -1×10^{-2}
- **Max Steps:** 1000

Qwen2.5-Math-7B-Instruct

- **Train Batch Size:** 16
- **Learning Rate:** 1×10^{-6}
- **Response Length:** 3300
- **Temperature:** 0.7
- **Top-p:** 0.9
- **Number of Rollout:** 16

- **Repetition Penalty:** 1.05
- **KL Penalty Coefficient:** 1×10^{-3}
- **Entropy Coefficient:** -1×10^{-2}
- **Max Steps:** 1000

OpenMath-Nemotron-1.5B

- **Train Batch Size:** 16
- **Learning Rate:** 1×10^{-6}
- **Response Length:** 18000
- **Temperature:** 0.6
- **Top-p:** 1.0
- **Number of Rollout:** 16
- **Repetition Penalty:** 1
- **KL Penalty Coefficient:** 1×10^{-3}
- **Entropy Coefficient:** -1×10^{-2}
- **Max Steps:** 1000

OctoThinker-3B-Hybrid-Zero

- **Train Batch Size:** 16
- **Learning Rate:** 1×10^{-6}
- **Response Length:** 8192
- **Temperature:** 0.7
- **Top-p:** 0.9
- **Number of Rollout:** 16
- **Repetition Penalty:** 1.05
- **KL Penalty Coefficient:** 1×10^{-3}
- **Entropy Coefficient:** -1×10^{-2}
- **Max Steps:** 1000

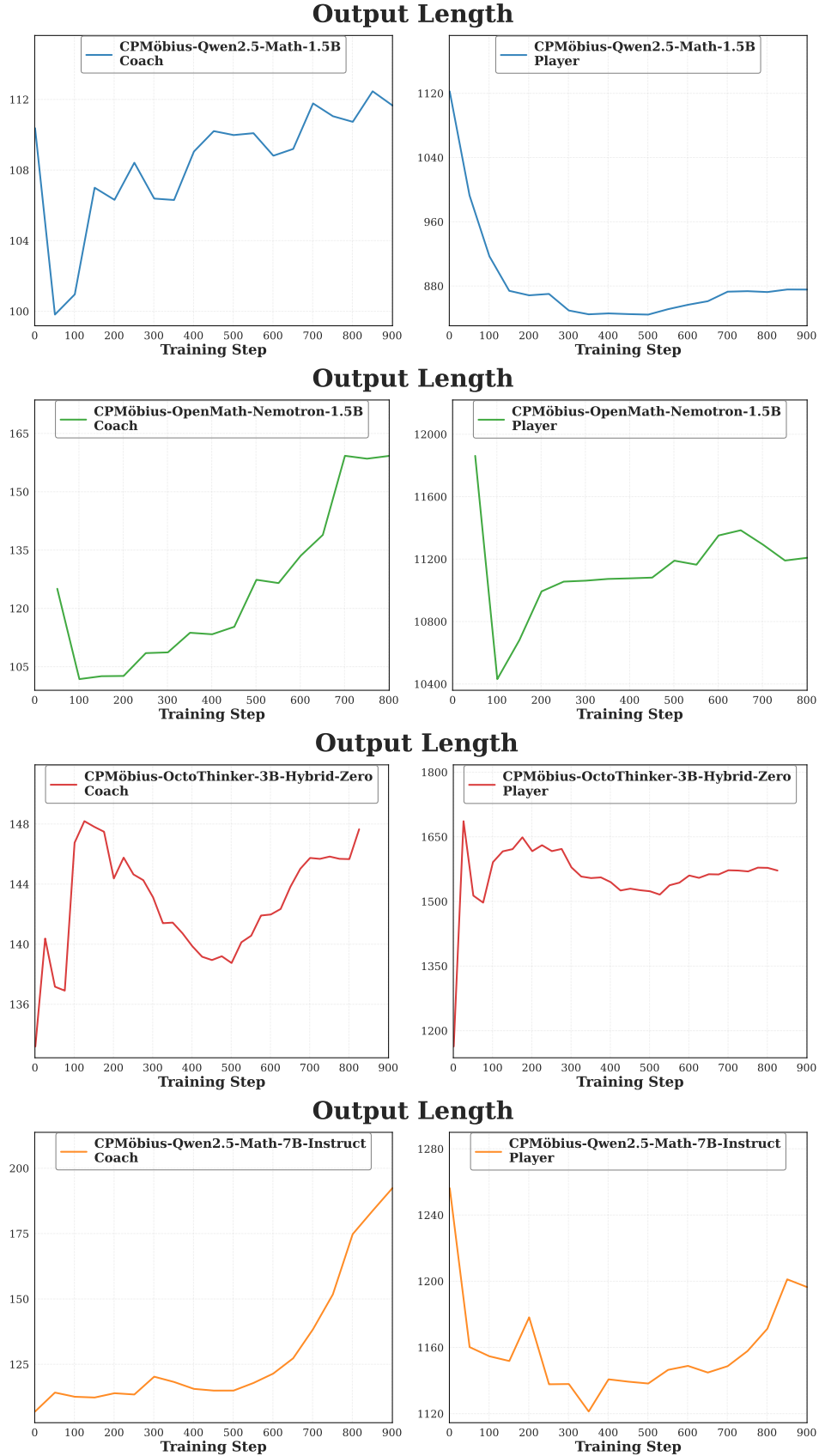


Figure 6: Visualization of the growing trend of output length of different models.

A.5 DIFFERENT TREND OF OUTPUT LENGTH ON COACH AND PLAYER MODEL

As shown in Figure 6, the output length of the Coach model tends to increase, while that of the Player model tends to decrease. We speculate that the Coach is spontaneously generating more challenging problems, thereby creating a form of curriculum learning for the Player. Meanwhile, the Player appears to refine its responses to be more concise, reflecting a "long-to-short" learning trend.

A.6 EXAMPLES OF PROBLEMS

Below are examples of problems and its corresponding reference answers proposed by the Coach along the training process.

Step 1 Easy Problem

Question:

Given that Lisa walks 3 miles more than John every day, and John walks a total of 8 miles, determine how many miles Lisa walks each day.

Reference Answers:

The answer is 11 miles.

Explanation:

Since John walks 8 miles and Lisa walks 3 miles more than John each day, we can calculate Lisa's daily walk as follows:

John's daily walk = 8 miles

Lisa's daily walk = John's daily walk + 3 miles

Lisa's daily walk = 8 miles + 3 miles

Lisa's daily walk = 11 miles

Therefore, Lisa walks 11 miles each day. The final answer is 11.

Step 5 Easy Geometry Problem

Question:

The lengths of the two diagonals of a rhombus are 6 and 8 respectively. The area of this rhombus is

Reference Answers:

To find the area of a rhombus when the lengths of its diagonals are known, we can use the formula:

$$\text{Area} = \frac{1}{2} \times d_1 \times d_2$$

where d_1 and d_2 are the lengths of the diagonals. In this problem, the lengths of the diagonals are 6 and 8. So we can substitute these values into the formula to get the area.

Let's calculate this using Python to ensure accuracy.

```
# Lengths of the diagonals
```

```
d1 = 6
```

```
d2 = 8
```

```
# Area of the rhombus
```

```
area = 0.5 * d1 * d2
```

```
print(area)
```

```
""output
```

```
24.0
```

```
""
```

The area of the rhombus with diagonals of lengths 6 and 8 is 24.

Step 10 Brain Teaser with Highlight that Sibling Relationships are Reciprocal**Question:**

Each of the three people in the Martin family also has a sibling in their family. If they need to divide a package of cookies among themselves and their siblings, and the package contains 120 cookies, how many cookies will they each get?

Reference Answers: If you get this right, you can get a special prize at the end of the week. Great job, smart cookie!

Step 1: Understand the Problem Correctly

The problem states "Each of the three people in the Martin family also has a sibling in their family." This means:

The Martin family consists of 3 people total

Each of these 3 people has a sibling relationship with others in the family

The siblings are among these same 3 people (not additional people)

Step 2: Interpret the Family Structure

Since there are only 3 people in the Martin family, and each person has a sibling, this means:

These 3 people are siblings to each other

No additional family members need to be counted

Total people sharing the cookies = 3

Step 3: Calculate Cookies per Person

Total cookies: 120

Total people: 3

Cookies per person: $120 \div 3 = 40$

The number of cookies each person will get is 40.

Step 100 Complex Combinatorics Problem that Requires Case Analysis**Question:**

A child builds towers using identically shaped cubes of different colors. Determine the number of different towers with a height 8, built with 4 red cubes, 3 blue cubes, and 2 green cubes.

Reference Answers: Let me solve this problem step by step.

****Step 1: Analyze the Problem****

We have:

- 4 red cubes
- 3 blue cubes
- 2 green cubes
- Total: $4 + 3 + 2 = 9$ cubes

But we need to build a tower with height 8, so we must choose 8 cubes from the 9 available.

****Step 2: Identify the Cases****

Since we have 9 cubes but only need 8, we must exclude exactly 1 cube. This gives us three cases:

- Case 1: Exclude 1 red cube (use 3 red, 3 blue, 2 green)
- Case 2: Exclude 1 blue cube (use 4 red, 2 blue, 2 green)
- Case 3: Exclude 1 green cube (use 4 red, 3 blue, 1 green)

****Step 3: Apply Multiset Permutation Formula****

For each case, we use the formula: $\frac{n!}{n_1! \times n_2! \times n_3!}$
where $n = 8$ (tower height) and n_1, n_2, n_3 are the numbers of each color cube.

****Case 1:**** Use 3 red, 3 blue, 2 green cubes

$$\frac{8!}{3! \times 3! \times 2!} = \frac{40320}{6 \times 6 \times 2} = \frac{40320}{72} = 560$$

****Case 2:**** Use 4 red, 2 blue, 2 green cubes

$$\frac{8!}{4! \times 2! \times 2!} = \frac{40320}{24 \times 2 \times 2} = \frac{40320}{96} = 420$$

****Case 3:**** Use 4 red, 3 blue, 1 green cube

$$\frac{8!}{4! \times 3! \times 1!} = \frac{40320}{24 \times 6 \times 1} = \frac{40320}{144} = 280$$

****Step 4: Sum All Cases****

Total number of different towers = $560 + 420 + 280 = 1260$

Therefore, the number of different towers with height 8 is 1260.

Step 200 More Challenging Problem that Needs Algebraic Identity Transformations**Question:**

For an m -element subset A of the set $\{1, 2, \dots, 10\}$, define $T(A)$ as the number of unordered pairs $\{x, y\}$, $x < y$, such that $x, y \in A$. Find $\sum_{A \subseteq \{1, 2, \dots, 10\}} T(A)$.

Reference Answers: To solve the problem, we need to calculate the sum of the number of unordered pairs $\{x, y\}$ such that $x < y$ and both x and y belong to each m -element subset A of the set $\{1, 2, \dots, 10\}$, for all possible subsets A .

To break down the problem:

1. ****Understanding the Problem:****

- We need to find the sum of the number of unordered pairs $\{x, y\}$ for all possible m -element subsets A of $\{1, 2, \dots, 10\}$.

- For each subset A , the number of unordered pairs is given by the binomial coefficient $\binom{m}{2} = \frac{m(m-1)}{2}$.

- We need to sum this value over all possible m -element subsets of $\{1, 2, \dots, 10\}$.

2. ****Counting the Subsets:****

- There are $\binom{10}{m}$ subsets of $\{1, 2, \dots, 10\}$ with m elements.

- Therefore, the total sum is $\sum_{m=0}^{10} \binom{10}{m} \cdot \binom{m}{2}$.

3. ****Simplifying the Sum:****

- We can use the identity $\binom{m}{2} = \frac{m(m-1)}{2}$ to rewrite the sum.

- The sum becomes $\sum_{m=0}^{10} \binom{10}{m} \cdot \frac{m(m-1)}{2} = \frac{1}{2} \sum_{m=0}^{10} \binom{10}{m} m(m-1)$.

- We can simplify the sum further by noting that $\sum_{m=0}^{10} \binom{10}{m} m(m-1) = 10 \cdot 9 \cdot 2^8$ (since $\sum_{m=0}^{10} \binom{10}{m} m(m-1) = 10 \cdot 9 \cdot 2^{10-2}$).

4. ****Final Calculation:****

- The sum is $\frac{1}{2} \cdot 10 \cdot 9 \cdot 2^8 = 45 \cdot 256 = 11520$.

Let's implement this in Python using sympy to verify our steps:

```
import sympy as sp

# Define the sum
total_sum = 0
for m in range(11):
    total_sum += sp.binomial(10, m) * sp.binomial(m, 2)

print(total_sum)
```

““output

11520

““

The sum of the number of unordered pairs $\{x, y\}$ such that $x < y$ and both x and y belong to each m -element subset A of the set $\{1, 2, \dots, 10\}$ is 11520.