Team-Based Self-Play With Dual Adaptive Weighting for Fine-Tuning LLMs

Wu Li[1] **Yigeng Zhou**[1] **Zesheng Shi**[1] **Yequan Wang**[2] **Min Zhang**[1] **Jing Li**[1]✉
[1]Harbin Institute of Technology, Shenzhen, China
[2]Beijing Academy of Artificial Intelligence, Beijing, China
li-555@outlook.com jingli.phd@hotmail.com

Abstract

While recent self-training approaches have reduced reliance on human-labeled data for aligning LLMs, they still face critical limitations: (i) sensitivity to synthetic data quality, leading to instability and bias amplification in iterative training; (ii) ineffective optimization due to a diminishing gap between positive and negative responses over successive training iterations. In this paper, we propose Team-based self-Play with dual Adaptive Weighting (TPAW), a novel self-play algorithm designed to improve alignment in a fully self-supervised setting. TPAW adopts a team-based framework in which the current policy model both collaborates with and competes against historical checkpoints, promoting more stable and efficient optimization. To further enhance learning, we design two adaptive weighting mechanisms: (i) a response reweighting scheme that adjusts the importance of target responses, and (ii) a player weighting strategy that dynamically modulates each team member's contribution during training. Initialized from a SFT model, TPAW iteratively refines alignment without requiring additional human supervision. Experimental results demonstrate that TPAW consistently outperforms existing baselines across various base models and LLM benchmarks.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of NLP tasks (Radford et al., 2019; Achiam et al., 2023), driven by pre-training on massive datasets and further refined through alignment techniques such as Supervised Fine-Tuning (SFT), Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022; Christiano et al., 2017; Stiennon et al., 2020), and Direct Preference Optimization (DPO) (Rafailov et al., 2023). Despite their success, these alignment methods heavily depend on human-annotated data, which limits their scalability. To address this, recent research has explored self-training paradigms (Wang et al., 2023; Yuan et al., 2024; Chen et al., 2024) that utilize model-generated data. Among them, methods such as Self-Play Fine-Tuning (SPIN) (Chen et al., 2024) have shown promise by repurposing existing SFT datasets to iteratively align models without the need for additional human annotations.

However, these self-training approaches still face several limitations that may hinder their alignment performance. First, based on current model's performance, self-rewarding or self-play mechanisms fail to fully utilize the model's accumulated training trajectory throughout the iterative training process. In addition, these approaches exhibit training instability and high sensitivity to the quality of synthetic data, often leading to the accumulation and amplification of errors and biases during self-training (Ren et al., 2024; Xu et al., 2024). Second, most self-training methods use DPO-like objectives that jointly optimize positive and negative samples. However, studies have shown that the probabilities of both y^+ and y^- often decrease during training (Pang et al., 2024; Yuan et al., 2025; Rafailov et al., 2024; Tajwar et al., 2024), possibly due to the high similarity between them (Pal et al., 2024; Razin et al., 2025). This problem is exacerbated in self-training methods, where the distinction between positive and negative responses diminishes over iterations (Song et al., 2025; Huang et al., 2025), resulting in noisy preference data and suboptimal policy updates (Wang et al., 2025).

To address these challenges, we propose **T**eam-based self-**P**lay with dual **A**daptive **W**eighting (TPAW), as illustrated in Figure 1. TPAW consists of two key components: (1) a team-based self-play framework that optimizes policy model by leveraging historical checkpoints, leading to more stable and efficient training; and (2) two adaptive weighting mechanisms—one for assess-

✉ Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 18617–18632

ing the importance of target responses and the other for adjusting the influence of each player in team-based interactions—both of which contribute to improved model performance and better alignment with the target response distribution. Starting from a SFT model, TPAW iteratively trains the LLM using SFT dataset. In each iteration, historical policy checkpoints act as opponents, generating responses aligned with the target distribution. Concurrently, the current policy model, in collaboration with these historical checkpoints, forms the reward model, which acts as the main player team that works to distinguish authentic target responses from LLM-generated content. During this process, the two adaptive weighting mechanisms dynamically adjust (i) the contribution of each main player and (ii) the weight of target responses, ensuring effective learning and adaptation.

In summary, the key contributions of our paper are as follows:

- **Team-Based Self-Play Framework**: We propose the first self-play method that unifies historical policy checkpoints into both the opponent and main player teams, overcoming the limitations of multi-iteration self-training methods by fully optimizing on historical training trajectories.
- **Dual Adaptive Weighting Mechanism**: We propose a dual adaptive weighting scheme that jointly optimizes two key objectives: (i) balancing the main players' discriminative ability with LLM's alignment to the target response distribution through adaptive response reweighting, and (ii) adaptively assigning influence weights to the main players based on their discriminative strengths.
- **Theoretical and Empirical Advancements**: Empirically, TPAW significantly enhances the well-aligned SFT model, outperforming baselines trained on similar datasets across multiple benchmarks. Further analysis confirms that TPAW better exploits the SFT dataset and achieves closer alignment with the target response distribution.

2 Related Work

Self-play. Self-play (Samuel, 1959; Tesauro et al., 1995) is a training methodology in which a player repeatedly interacts with copies of itself to improve performance through exploration, evaluation, and refinement of strategies. This iterative process enables the discovery of increasingly effective solutions within a given environment. A landmark example of self-play research is AlphaGo Zero (Silver et al., 2017), which, unlike previous models, does not rely on human data but instead leverages self-play combined with Monte Carlo Tree Search, ultimately achieving performance that exceeds human capabilities. Recent studies have demonstrated the application of self-play in LLMs (Wu et al., 2024; Zhao et al., 2025; Cheng et al., 2024; Swamy et al., 2024). SPIN (Chen et al., 2024) enables LLMs to iteratively generate training data and refine their strategies by distinguishing between self-generated and human-annotated responses. This approach yields substantial performance gains without requiring additional human-labeled data. AutoIF (Dong et al., 2024) improves the performance of LLMs on complex tasks by automatically generating instruction-following data and utilizing code verification and execution feedback-based rejection sampling. However, these methods fail to fully utilize historical checkpoints and are highly sensitive to the quality of synthetic data, which makes it difficult for their outputs to fit the target response distribution and amplifies errors and biases during the iteration process.

Alignment. Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022; Lee et al., 2024) aims to align LLMs with human preferences. This method first learns a reward model that reflects human preferences and then uses reinforcement learning algorithms to maximize the reward. The RLHF method involves reward modeling, distributed training of multiple models, and expert annotations, presenting operational challenges. Therefore, recent research has focused on reducing or eliminating reliance on reward models to simplify the RLHF process (Rafailov et al., 2023; Ethayarajh et al., 2024; Meng et al., 2024). Alignment research has also extended to safety settings (Shi et al., 2025). DPO (Direct Policy Optimization) (Rafailov et al., 2023) is an improved method that directly optimizes the policy instead of relying on traditional reward models, effectively reducing the dependence on complex reward modeling and enhancing training efficiency and stability. Building upon similar ideas, TPAW directly trains the model to distinguish "good" responses from the SFT dataset and "bad" responses generated by the model, significantly reducing the reliance on human-annotated data.

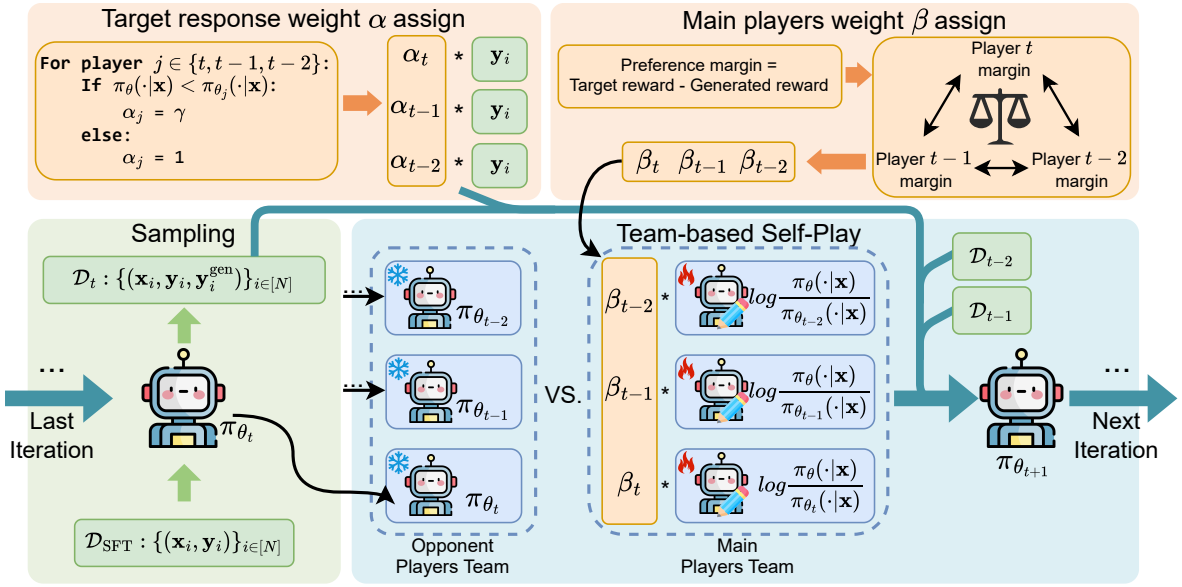


Figure 1: The workflow of TPAW. During response sampling, the model $\pi_{\boldsymbol{\theta}_t}$ generates responses $\mathbf{y}^{\text{gen}}$ from SFT prompts $\mathbf{x}$, forming a triple dataset $\mathcal{D}_t$. Next, TPAW adaptively assigns target response weights $\boldsymbol{\alpha}$ based on the target response likelihood, and main player weights $\boldsymbol{\beta}$ based on the reward margin between the target and generated responses. Finally, the policy model $\pi_{\boldsymbol{\theta}}$ is fine-tuned using the overall Team-based Self-Play training objective, resulting in model $\pi_{\boldsymbol{\theta}_{t+1}}$ for next iteration.

3 Methodology

In this section, we present TPAW, a novel approach that enhances LLM alignment through team-based self-play with dual adaptive weighting, as shown in Figure 1. First, we formulate the Team-Based Self-Play framework, establishing its theoretical foundations and operational structure. Next, we introduce the Dual Adaptive Weighting mechanism, which dynamically adjusts both the importance of training target responses and the influence of main players during the team-based self-play process. Finally, we provide a detailed algorithm for the practical implementation of TPAW, including pseudocode and optimization strategies.

3.1 Team-Based Self-Play

Conventional self-training methods typically rely on a single model's performance, suffering from training instability and are highly sensitive to the quality of synthetic data, which can lead to the propagation and amplification of errors and biases over successive iterations (Ren et al., 2024; Xu et al., 2024). To overcome these limitations, we revisit the self-play paradigm by effectively leveraging the historical checkpoints accumulated across iterative training.

Based on this idea, we propose a Team-Based Self-Play framework that reframes self-play as a competitive game between two teams. In this setting, the main players are responsible for distinguishing between human-written and LLM-generated responses, while the opponents aim to generate responses that are as human-like as possible. Both teams are composed of instances of the same LLM, but at different training stages: the opponents are drawn from earlier checkpoints, whereas the main players are composites built from the current model and its previous iterations.

This team-based dynamic not only introduces diversity and adversarial robustness into the training process but also allows for targeted optimization—since some main players may struggle to distinguish responses effectively, we can adaptively refine their learning to improve overall training efficiency. Furthermore, the inclusion of the training trajectory serves as an implicit form of regularization, helping to stabilize the learning process and mitigate error accumulation over time.

Our Team-Based Self-Play framework proceeds in each iteration with two key steps: (1) sampling from opponents, and (2) updating the main players.

Sampling from Opponents. At iteration $t+1$, the opponent team is composed of the LLMs from the three most recent iterations, denoted as $(\pi_{\boldsymbol{\theta}_t}, \pi_{\boldsymbol{\theta}_{t-1}}, \pi_{\boldsymbol{\theta}_{t-2}})$. This ensures a dynamic balance between recent and historical strategies during training. For the newly added opponent $\pi_{\boldsymbol{\theta}_t}$ in this iteration, we use prompts $\mathbf{x}$ from the SFT

dataset $\mathcal{D}_{\text{SFT}}$ to generate responses $\mathbf{y}^{\text{gen}}$ according to the $\pi_{\boldsymbol{\theta}_t}(\cdot|\mathbf{x})$. Then we pair these generated responses with the original dataset entries $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_{\text{SFT}}$ to construct a new triple-form dataset: $\mathcal{D}_t = \{(\mathbf{x}_i, \mathbf{y}_i, \mathbf{y}_i^{\text{gen}})\}_{i=1}^{N}$. In addition, to construct the complete sampling dataset for the opponent team, denoted as $\mathcal{D}_O$, we aggregate the triple datasets from the most recent three iterations: $\mathcal{D}_O = \mathcal{D}_t \cup \mathcal{D}_{t-1} \cup \mathcal{D}_{t-2}$.

Forming Main Players Team. The main players are designed to be trained to distinguish between responses generated by LLMs and those written by humans. Inspired by SPIN (Chen et al., 2024) and DPO (Rafailov et al., 2023)'s implicit reward:

$$r(\mathbf{x}, \mathbf{y}) = \lambda \cdot \log \frac{\pi_r(\mathbf{y}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})} + \lambda \cdot \log Z(x), \quad (1)$$

we define each main player as:

$$P_j(\mathbf{x}, \mathbf{y}) = \lambda \cdot \log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{\pi_{\boldsymbol{\theta}_j}(\mathbf{y}|\mathbf{x})}, \; j \in \{t, t-1, t-2\}, \quad (2)$$

where $\lambda > 0$ is a regularization parameter used to constrain the deviation of $\pi_{\boldsymbol{\theta}_{t+1}}$ from $\pi_{\boldsymbol{\theta}_t}$, thereby stabilizing the self-play process.

In this context, as we are only concerned with different responses corresponding to the same prompt $\mathbf{x}$, the normalization term $\lambda \cdot \log Z(x)$ in Eq. 1 can be omitted. Consequently, Eq. 2 can be interpreted as an implicit reward score, jointly inferred by the current model and a historical model, conditioned on the given prompt $\mathbf{x}$ and response $\mathbf{y}$. Within our framework, this score—produced by the main player—quantifies how well the response $\mathbf{y}$ aligns with the target distribution. Ideally, main players should assign higher scores to responses from the supervised fine-tuning dataset ($\mathbf{y} \in \mathcal{D}_{\text{SFT}}$) and lower scores to model-generated responses ($\mathbf{y}^{\text{gen}} \sim \pi_{\boldsymbol{\theta}_j}(\cdot \mid \mathbf{x})$).

Furthermore, to enable each player P_j to learn the distinction between $\mathbf{y}$ and $\mathbf{y}^{\text{gen}}$, the single main player's training objective becomes:

$$\mathcal{L}_j = -\mathbb{E}_{\mathcal{D}_O}\left[\ell\big(P_j(\mathbf{x}, \mathbf{y}) - P_j(\mathbf{x}, \mathbf{y}^{\text{gen}})\big)\right]. \quad (3)$$

Following SPIN (Chen et al., 2024), we use the logistic loss $\ell(t) := \log(1 + \exp(-t))$ due to its non-negativity, smoothness, and exponential tail decay as $t \to \infty$. This choice helps prevent excessive growth in the magnitude of P_j.

In subsequent subsections, we will detail how to further optimize each player's objective based on Eq. 3, and describe how to aggregate these individual objectives into a unified team objective using a dual adaptive weighting mechanism.

3.2 Dual Adaptive Weighting

To further enhance the robustness and alignment of the team-based self-play framework, we introduce a dual adaptive weighting mechanism that dynamically adjusts the importance of training target responses and the influence of main players during team-based self-play.

Adaptive Target Response Weighting. Most existing self-training methods adopt optimization objectives similar to DPO, simultaneously optimizing both positive and negative samples. However, recent studies have shown that when the similarity between positive and negative responses is high (Pal et al., 2024; Razin et al., 2025), or the probability of generating negative responses is low (Ren and Sutherland, 2025), the model's likelihood of producing preferred (i.e., target) responses can deteriorate during training. We also find the decrease phenomenon in our ablation experiment and SPIN baseline in Section 4.4.

To address this issue, we propose Adaptive Target Response Weighting, a method designed to balance the discriminative capacity of the main players with the alignment of the LLM policy to the target distribution. Specifically, based on each player P_j's optimization objective in Eq. 3, our method adaptively increases the reward score $P_j(\mathbf{x}, \mathbf{y})$ for the target response $\mathbf{y}$ when the model's likelihood for $\mathbf{y}$ decreases relative to a reference model. This adjustment effectively increases the weight of the target response during training, encouraging the model to better refit to the target distribution.

We define the adaptive target response weight for player P_j as:

$$\boldsymbol{\alpha}_j = \begin{cases} \boldsymbol{\eta} & \text{if } P_j(\mathbf{x}, \mathbf{y}) \le 0, (\mathbf{x}, \mathbf{y}) \in \mathcal{D}_O, \\ 1 & \text{else}, \end{cases} \quad (4)$$

where $\boldsymbol{\eta} > 1$ is a hyperparameter that controls the degree of weight growth when the target response score is low.

Incorporating this adaptive weighting into the player's objective, we modify Eq. 3 as follows:

$$\mathcal{L}_j = -\mathbb{E}_{\mathcal{D}_O}\left[\ell\big(\boldsymbol{\alpha}_j \cdot P_j(\mathbf{x}, \mathbf{y}) - P_j(\mathbf{x}, \mathbf{y}^{\text{gen}})\big)\right], \quad (5)$$

Adaptive Main Players Weighting. In the Team-Based Self-Play framework, multiple main players are introduced to fully leverage the potential of historical model checkpoints in enhancing the discriminative capabilities of the main players. However, in iteration $t+1$, since P_{t-1} and P_{t-2} have already undergone optimization in previous iterations, statically assigning weights to the main players can lead to suboptimal training. Specifically, P_t may be undertrained, while P_{t-1} and P_{t-2} risk overfitting.

To address this, we propose Adaptive Main Player Weighting, which dynamically assigns greater weight to the main player when it exhibits weaker discriminative performance in distinguishing between human-written and LLM-generated responses. Specifically, for a given player P_j, we define the preference margin m_j as:

$$m_j = P_j(\mathbf{x}, \mathbf{y}) - P_j(\mathbf{x}, \mathbf{y}^{\text{gen}}). \tag{6}$$

As previously discussed, P_j provides a reward score indicating how well the response $\mathbf{y}$ aligns with the target distribution. Thus, the preference margin m_j quantifies the main player's effectiveness in discriminating between human-written and LLM-generated responses for a given sample.

We then apply the softmax function to compute each player's weight $\boldsymbol{\beta}_j$ based on their margin value:

$$\boldsymbol{\beta}_j = \frac{e^{-\boldsymbol{\gamma} \cdot m_j}}{\sum_{k \in \text{main players}} e^{-\boldsymbol{\gamma} \cdot m_k}}. \tag{7}$$

For each main player, a smaller margin—indicating poorer judgment on a sample—results in a higher weight for that sample. The hyperparameter $\boldsymbol{\gamma}$ controls the sensitivity of the weight distribution to the margin values. A larger $\boldsymbol{\gamma}$ amplifies the influence of small margins, allowing them to dominate the softmax output. Conversely, as $\boldsymbol{\gamma}$ approaches zero, the weights converge to a uniform distribution, effectively ignoring differences in margin. To suppress the influence of extremely small values and unstable noise, we set weights smaller than 0.01 to zero and re-normalize the remaining weights.

Overall Training Objective. By assigning weights to each main player, we can aggregate their individual objectives from Eq. 5 into a unified overall training objective for the entire main player team:

Algorithm 1 Team-based self-Play with dual Adaptive Weighting (TPAW)

Input: SFT dataset $\mathcal{D}_{\text{SFT}} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$; Initial SFT policy $\pi_{\boldsymbol{\theta}_0}$; Number of iterations T.
for $t = 0$ to $T-1$ **do**
  Generate responses: $\{\mathbf{y}_i^{\text{gen}}\}_{i=1}^{N} \sim \pi_{\boldsymbol{\theta}_t}(\cdot \mid \mathbf{x}_i),\ \forall \mathbf{x}_i \in \mathcal{D}_{\text{SFT}}$
  Construct pairwise preference dataset:
  $\mathcal{D}_t = \{(\mathbf{x}_i, \mathbf{y}_i, \mathbf{y}_i^{\text{gen}})\}_{i=1}^{N},\ (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_{\text{SFT}}$
  Aggregate past opponents: $\mathcal{D}_O = \mathcal{D}_t \cup \mathcal{D}_{t-1} \cup \mathcal{D}_{t-2}$
  for $j \in \{t, t-1, t-2\}$ **do**
    Get target response weights $\boldsymbol{\alpha}_j$ according to Eq. 4
    Compute each player's loss for each sample:
    $\boldsymbol{l}_{ij}(\pi_{\boldsymbol{\theta}}; \pi_{\boldsymbol{\theta}_j}) = \ell\left(\boldsymbol{\alpha}_j \log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_i \mid \mathbf{x}_i)}{\pi_{\boldsymbol{\theta}_j}(\mathbf{y}_i \mid \mathbf{x}_i)} - \log \frac{\pi_{\boldsymbol{\theta}}(\mathbf{y}_i^{\text{gen}} \mid \mathbf{x}_i)}{\pi_{\boldsymbol{\theta}_j}(\mathbf{y}_i^{\text{gen}} \mid \mathbf{x}_i)}\right)$,
    where $(\mathbf{x}_i, \mathbf{y}_i, \mathbf{y}_i^{\text{gen}}) \in \mathcal{D}_O$.
  end for
  Determine policy weighting coefficients: $\boldsymbol{\beta}_t, \boldsymbol{\beta}_{t-1}, \boldsymbol{\beta}_{t-2}$ according to Eq. 7
  Update policy parameters:
  $\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_{i=1}^{3N} \lambda \cdot (\boldsymbol{\beta}_t \boldsymbol{l}_{i,t} + \boldsymbol{\beta}_{t-1} \boldsymbol{l}_{i,t-1} + \boldsymbol{\beta}_{t-2} \boldsymbol{l}_{i,t-2})$
end for
Output: Final aligned policy $\pi_{\boldsymbol{\theta}_T}$.

$$\mathcal{L}_{\text{ours}} = \sum_{i=t-2}^{t} \boldsymbol{\beta}_i \cdot \mathcal{L}_i(\pi; \pi_i) \tag{8}$$

3.3 Summary of TPAW

With the proposed Team-Based Self-Play framework and Dual Adaptive Weighting mechanism in place, we now present the implementation of TPAW in Algorithm 1. An overview of the entire framework is shown in Figure 1.

At each iteration, the current policy generates responses on the SFT dataset and forms preference pairs with the target responses. These are combined with data from previous iterations to construct an opponent dataset. For each past policy, we compute adaptive response weights $\boldsymbol{\alpha}_j$ to guide learning at the sample level, and player weights $\boldsymbol{\beta}_j$ to balance the influence of different main players also at the sample level. The policy is then updated by minimizing a weighted sum of losses against each opponent, enabling alignment through iterative, adaptive team-based self-play.

Specifically, at $t = 0$, we ignore the historical models π_{t-1} and π_{t-2}, at $t = 1$, we ignore π_{t-2}. To better illustrate how policies evolve over time, Appendix B summarizes the training process for each policy, including initialization, training data, opponents, and main players.

Model	Method	Open LLM Leaderboard V1						V1 Avg.	Open LLM Leaderboard V2						V2 Avg.
		Arc	TQA	Wino	GSM8k	HS	MMLU		IFEval	BBH	Math	GPQA	MUSR	MMLU-p	
Qwen2.5-1.5B	SFT	50.94	46.83	64.80	51.25	64.91	58.93	56.28	31.73	15.15	6.65	3.47	**6.02**	17.38	13.40
	DPO	54.52	49.39	64.96	56.71	66.29	59.41	58.55	31.79	16.25	9.89	2.24	2.57	18.24	13.50
	SPIN Iter-1	51.79	47.82	64.33	53.83	65.76	59.26	57.13	32.44	15.47	8.23	4.03	5.55	17.57	13.88
	SPIN Iter-2	52.05	48.25	64.40	53.45	66.00	59.43	57.26	33.97	15.22	8.99	3.69	5.65	17.52	14.17
	SPIN Iter-3	52.65	48.38	64.48	53.53	66.25	59.56	57.47	35.34	14.44	9.06	**4.47**	5.16	17.58	14.34
	SPIN Iter-4	53.58	**48.41**	64.72	53.30	66.05	59.58	57.61	33.82	14.33	9.21	4.25	5.70	17.54	14.14
	TPAW Iter-1 (Ours)	52.99	47.64	64.96	52.77	65.77	59.50	57.27	32.35	**15.60**	8.69	3.24	5.66	17.65	13.87
	TPAW Iter-2 (Ours)	53.41	47.91	64.17	53.15	66.08	59.58	57.38	35.52	15.37	8.69	4.36	5.98	17.72	14.61
	TPAW Iter-3 (Ours)	**53.67**	47.91	65.04	53.60	66.18	59.52	57.65	**36.10**	15.03	**10.20**	4.14	5.43	**18.03**	**14.82**
	TPAW Iter-4 (Ours)	52.56	47.90	**65.11**	**55.04**	**66.30**	**59.64**	**57.76**	36.04	15.51	9.37	4.36	4.73	17.82	14.64
Llama3.1-8B	SFT	60.75	51.65	74.82	53.15	79.48	63.72	63.93	36.34	26.28	4.61	6.15	11.77	21.01	17.69
	DPO	63.14	57.42	74.35	50.27	80.46	63.08	64.79	36.83	25.83	4.53	6.60	11.78	21.91	17.91
	SPIN Iter-1	61.69	53.62	75.3	53.9	80.44	63.94	64.82	42.91	28.05	4.83	6.26	13.17	22.84	19.68
	SPIN Iter-2	62.03	53.81	75.61	52.31	80.63	63.97	64.73	43.21	28.17	5.14	4.81	13.28	22.93	19.59
	SPIN Iter-3	62.2	53.98	76.01	51.71	80.74	64.15	64.80	44.13	**28.46**	5.21	3.36	12.96	22.92	19.51
	SPIN Iter-4	62.03	53.92	76.01	52.54	80.78	63.97	64.88	44.01	28.36	4.98	3.8	13.44	23.02	19.60
	TPAW Iter-1 (Ours)	62.88	53.71	75.93	55.19	80.66	64.29	65.44	38.76	28.16	5.21	6.15	13.65	23.17	19.18
	TPAW Iter-2 (Ours)	62.97	54.11	76.16	**55.50**	80.63	64.45	65.64	42.43	27.85	6.19	**6.38**	**16.22**	23.02	20.35
	TPAW Iter-3 (Ours)	**65.54**	**54.46**	**76.32**	55.19	80.77	**64.56**	**66.14**	45.04	27.82	5.89	6.26	16.16	23.34	20.75
	TPAW Iter-4 (Ours)	63.23	54.44	76.24	55.34	**80.91**	64.46	65.77	**45.12**	28.14	**6.34**	6.15	15.73	**23.56**	**20.84**

Table 1: Performance comparison across Open LLM Leaderboard V1 and V2. **Bold** indicates best, underline indicates second-best (excluding the DPO method). Some datasets are abbreviated, see Appendix A for details.

4 Experiments

4.1 Experiment Setup

Model and Baseline. We employ Qwen2.5-1.5B (Yang et al., 2024) and Llama3.1-8B (Dubey et al., 2024) as our base pretrained models. To ensure a fair comparison under similar training data conditions, we adopt supervised fine-tuning (SFT) and SPIN (Chen et al., 2024) as baseline methods. For SPIN, we follow the official implementation, noting that "SPIN iter-1" refers to the model after the first iteration, which differs slightly from its original definition. Additionally, we perform supplementary experiments using DPO trained on the UltraFeedback dataset as an extended baseline.

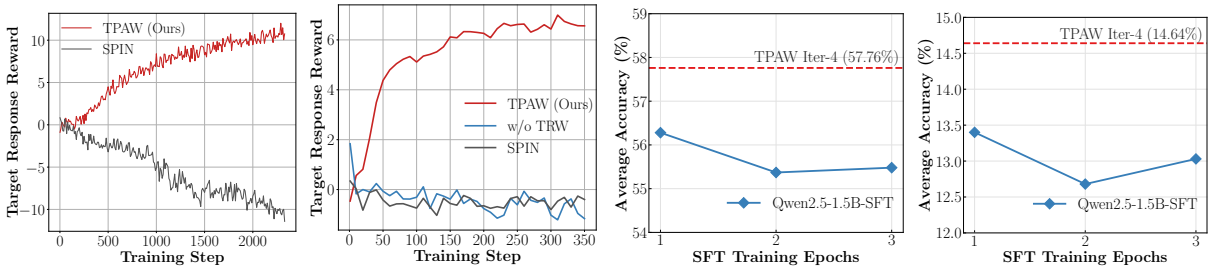
Datasets. For SFT, we use the full Ultrachat200k dataset, a filtered version of UltraChat (Ding et al., 2023). For both TPAW and SPIN, we train on a 50k subset of Ultrachat200k. To assess the TPAW's performance on domain-specific tasks, we additionally train it using the GSM8K (Cobbe et al., 2021).

Evaluation. We evaluate models on both versions of the Open LLM Leaderboard (V1 (Beeching et al., 2023) and V2 (Fourrier et al., 2024)) using standardized metrics from the Language Model Evaluation Harness (Gao et al., 2024). V2 is more challenging than V1, and together they cover 12 benchmarks across knowledge, reasoning, math, and instruction following. Further experimental setup details are provided in Appendix A.

4.2 Main Results

Results on Open LLM Leaderboard. Table 1 reports the evaluation results of TPAW on both versions of the Open LLM Leaderboard, compared with the base model and the SPIN baseline over multiple iterations. TPAW consistently outperforms both SPIN and the SFT baseline by utilizing the SFT dataset more effectively. Although trained only on a general-purpose dialogue dataset, TPAW achieves improvements across all 12 benchmarks, showing strong generalization beyond its training domain. Even with just one-quarter of the original SFT data, it delivers notable gains—for example, on Qwen, up to 4.37% on IFEval, 3.55% on Math, and 3.79% on GSM8k, and on Llama, up to 4.79% on Arc, 8.78% on IFEval, and 4.45% on MUSR. The performance peaks around the third or fourth iteration and then converges. Furthermore, compared with DPO, TPAW is trained using only one-quarter of their data volume and does not rely on any additional preference annotations. Despite these more restrictive training conditions, our method still achieves superior performance, particularly on the more challenging Leaderboard V2, highlighting its strong data efficiency and robustness.

These results underscore the effectiveness and robustness of our iterative approach. Building upon



(a) Target response reward, trained on Ultrachat200k.

(b) Target response reward, trained on GSM8k.

(c) Average accuracy of leaderboard v1.

(d) Average accuracy of leaderboard v2.

Figure 2: Subfigures (a) and (b) show the target responses reward curves from the iteration 4 training process on Ultrachat200k and GSM8k, respectively. Subfigures (c) and (d) present the average accuracy on the leaderboard for models trained with additional epochs using SFT, evaluating the effect of extended training.

Model	Acc (%)
Qwen2.5-1.5B-SFT	51.25
SPIN-gsm8k Iter-1	53.75 (+2.65)
SPIN-gsm8k Iter-2	54.36 (+0.61)
SPIN-gsm8k Iter-3	53.75 (-0.61)
SPIN-gsm8k Iter-4	54.59 (+0.84)
TPAW-gsm8k Iter-1	54.21 (+3.11)
TPAW-gsm8k Iter-2	54.81 (+0.60)
TPAW-gsm8k Iter-3	<u>56.56</u> (+1.75)
TPAW-gsm8k Iter-4	**56.94** (+0.38)

Table 2: Performance of TPAW on GSM8k. Both SPIN and TPAW are trained on the GSM8k training set.

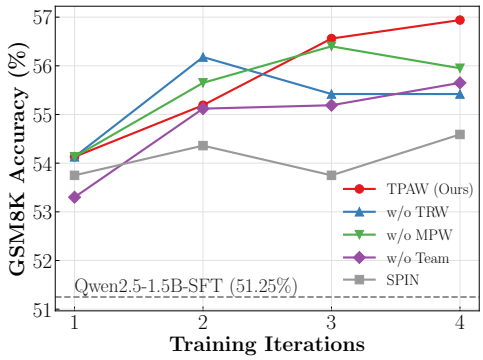


Figure 3: Ablation study on GSM8k. We evaluate TPAW by removing key components: without Target Response Weighting (**w/o TRW**); without Main Player Weighting (**w/o MPW**); without Team-based Mechanism (**w/o Team**).

the SFT model, we are able to leverage training data more efficiently, producing outputs that align more closely with the target responses—and in some cases, even surpassing previous performance limits—without requiring additional data or external assistance. The performance boost over SPIN stems from our novel Team-Based Self-Play Framework and Dual Adaptive Weighting Mechanism, which work together to guide learning more efficiently and maintain high-quality outputs throughout the iterative process.

Results on GSM8K. We applied our proposed TPAW method for domain-specific fine-tuning on the GSM8K dataset, starting from the Qwen2.5-1.5B-SFT model. Table 2 compares the accuracy of TPAW against the SPIN and standard SFT baselines across multiple iterations.

Both iterative approaches outperform the SFT model, but TPAW consistently achieves superior performance at every stage. The method demonstrates steady improvement across iterations, reaching 56.94% accuracy by the fourth iteration—a 5.69% gain over the initial SFT model. These results underscore the effectiveness and stability of TPAW in enhancing mathematical reasoning capabilities. The greater performance gains observed on domain-specific reasoning and mathematical tasks may stem from a larger initial discrepancy between the model's outputs and the target responses. Compared to the baseline, our method achieves a deeper alignment with the target responses, particularly on such challenging reasoning tasks. Appendix D.1 further provides representative GSM8K case studies, showing that these gains mainly come from improved structural reasoning consistency, while token-level arithmetic errors can still remain.

4.3 Ablation Studies

To further evaluate the effectiveness of the team-based self-play framework and the dual adaptive weighting mechanism, we conducted additional ablation experiments on GSM8k. In Figure 3, we present the results of ablating each key component of TPAW:

- Without adaptive Target Response Weighting (**w/o TRW**): Remove adaptive weighting for target responses, with weight fixed at 1;

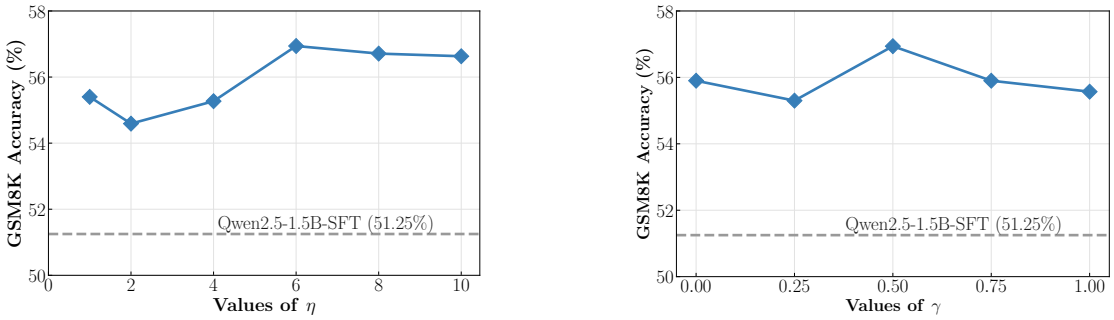


Figure 4: Impact of hyperparameters on GSM8K accuracy. Performance data is from the fourth iteration.

- Without adaptive Main Player Weighting (**w/o MPW**): Remove adaptive weighting for main players, all players assigned equal static weights;
- Without Team based Mechanism (**w/o Team**): Remove team structure, using only a single player for both opponent and main player teams.

The ablation study results demonstrate that both the team-based self-play framework and the dual adaptive weighting mechanism in TPAW are critical to achieving optimal performance: removing either component prevents the model from converging to the optimal solution. Overall, across the four iterations, ablating the team-based mechanism leads to the most significant performance degradation. Notably, even in configurations where these components are removed, our method consistently surpasses baseline performance. See more ablation experiments and detailed results in Appendix D.2.

4.4 Further Analyses

Adaptive Target Response Weighting Prevents Decline in Target Reward. As shown in Figure 2a and Figure 2b, training methods that do not employ the Adaptive Target Response Weighting mechanism exhibit a continuous decline in the reward value for the target response, which remains in the negative range. This suggests that during self-play training, the model reduces the output probabilities of both y^+ and y^-, causing a drift in the output distribution away from the intended target distribution.

In contrast, when our Adaptive Target Response Weighting mechanism is applied, this issue is substantially alleviated. During the training process of TPAW, the reward value consistently increases and eventually converges to a positive value, indicating that the model effectively learns to align with the target distribution.

TPAW Makes Fuller Use of Datasets Than More Epochs of SFT. As shown in Figure 2c and Figure 2d, training for more epochs on the dataset using SFT does not lead to further performance improvements. This stems from inherent limitations of the SFT training objective, causing the model to overfit after multiple epochs by mechanically memorizing answers in the dataset rather than genuinely learning generalizable capabilities. In contrast, by continuing training on the same dataset, TPAW successfully surpasses the performance ceiling of SFT, enabling the model not only to fit the target data distribution but also to retain its generalization ability. See more detail results in Appendix D.3.

Analysis and Ablation on Hyperparameters η and γ. Figure 4 presents the results of a numerical analysis investigating their impact. η and γ are core components of TPAW's Dual Adaptive Weighting mechanism, with their practical values set to 6 and 0.5. η controls how strongly the weight increases when the target response reward is low. The results show that relatively higher values of η encourage greater alignment with the target response, resulting in improved performance. γ adjusts the sensitivity of the main players' weight distribution to their judgment ability. A larger γ amplifies the impact of weaker performance, assigning more weight to poorly performing plyers during optimization. In contrast, when γ is close to zero, weights become nearly uniform, ignoring differences among players. The results show that a moderate value of 0.5 yields the most balanced weight distribution. See more experiments and detailed result in Appendix D.5.

5 Conclusion

In this work, we introduced TPAW, a novel self-training algorithm that enhances LLM alignment through a team-based self-play framework augmented with dual adaptive weighting mechanisms. TPAW leverages past policy checkpoints both as opponents and main players, enabling more ro-

bust and stable policy optimization. By adaptively adjusting the influence of individual team members and reweighting target responses, TPAW effectively mitigates issues related to noisy synthetic data and output offset from target response distribution. Extensive empirical results demonstrate that TPAW significantly improves model alignment and generalization across various benchmarks, validating its ability to better exploit SFT datasets and align with the target response distribution. We believe our team-based, adaptively weighted self-play paradigm paves the way for scalable and effective LLM alignment with minimal human supervision.

Acknowledgements

This work was supported in part by National Key R&D Program of China (SQ2024YFE0200592), National Natural Science Foundation of China (62476070), Shenzhen Science and Technology Program (JCYJ20241202123503005, GXWD 20231128103232001, ZDSYS20230626091203008, KQTD20240729102154066), Department of Science and Technology of Guangdong (2024A1515011540) and Suzhou Science and Technology Program (SYG2025072).

Limitations

While our work demonstrates the effectiveness of TPAW in aligning model outputs to target distributions, its performance is inherently bounded by the quality of the SFT datasets. This highlights both a limitation and a potential opportunity for future research in model distillation task. Additionally, our experiments on GSM8K suggest that TPAW holds promise for domain-specific fine-tuning, highlighting the need for further fine-tuning and evaluation across a broader range of specialized tasks. Overall, addressing these limitations will be crucial for realizing the full potential of our approach in broader and more challenging settings. In addition, our ablations suggest a trade-off in team size $N_{\max}$: while a larger team can improve diversity, setting $N_{\max} > 3$ may introduce weaker early checkpoints and lower-quality preference signals, leading to slight performance degradation.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard (2023-2024). https://huggingface.co/spaces/open-llm-leaderboard-old/open_llm_leaderboard.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning convertsweak language models to strong language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pages 6621–6642.

Pengyu Cheng, Yong Dai, Tianhao Hu, Han Xu, Zhisong Zhang, Lei Han, Nan Du, and Xiaolong Li. 2024. Self-playing adversarial language game enhances llm reasoning. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:126515–126543.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems (NeurIPS)*, 30.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3029–3051.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. 2024. Self-play with execution feedback: Improving instruction-following capabilities of large language models. *arXiv preprint arXiv:2406.13542*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. 2024. Open llm leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2.

Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T. Ash, and Akshay Krishnamurthy. 2025. Self-improvement in language models: The sharpening mechanism. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2024. RLAIF vs. RLHF: scaling reinforcement learning from human feedback with AI feedback. In *Forty-first International Conference on Machine Learning (ICML)*).

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems (NeurIPS)*, 35:27730–27744.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*.

Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:116617–116637.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. 2024. From r to q^*: Your language model is secretly a q-function. In *First Conference on Language Modeling*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:53728–53741.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Noam Razin, Sadhika Malladi, Adithya Bhaskar, Danqi Chen, Sanjeev Arora, and Boris Hanin. 2025. Unintentional unalignment: Likelihood displacement in direct preference optimization. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Yi Ren, Shangmin Guo, Linlu Qiu, Bailin Wang, and Danica J. Sutherland. 2024. Bias amplification in language model evolution: An iterated learning perspective. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Yi Ren and Danica J. Sutherland. 2025. Learning dynamics of LLM finetuning. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Arthur L Samuel. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.

Zesheng Shi, Yigeng Zhou, and Jing Li. 2025. Safety alignment via constrained knowledge unlearning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, and 1 others. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham M. Kakade, Dean Foster, and Udaya Ghai. 2025. Mind the gap: Examining the self-improvement capabilities of large language models. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in neural information processing systems (NeurIPS)*, 33:3008–3021.

Gokul Swamy, Christoph Dann, Rahul Kidambi, Zhiwei Steven Wu, and Alekh Agarwal. 2024. A minimaximalist approach to reinforcement learning from human feedback. *arXiv preprint arXiv:2401.04056*.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. 2024. Preference fine-tuning of llms should leverage suboptimal, on-policy data. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pages 47441–47474.

Gerald Tesauro and 1 others. 1995. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 13484–13508.

Zhaoyang Wang, Weilei He, Zhiyuan Liang, Xuchao Zhang, Chetan Bansal, Ying Wei, Weitong Zhang, and Huaxiu Yao. 2025. CREAM: Consistency regularized self-rewarding language models. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. 2024. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. 2024. Pride and prejudice: Llm amplifies self-bias in self-refinement. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 15474–15492.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Boji Shan, Zeyuan Liu, Jia Deng, Huimin Chen, Ruobing Xie, Yankai Lin, Zhenghao Liu, Bowen Zhou, Hao Peng, Zhiyuan Liu, and Maosong Sun. 2025. Advancing LLM reasoning generalists with preference trees. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. In *Forty-first International Conference on Machine Learning (ICML)*.

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

A Further Details about Experiment Set-Up

A.1 Hyperparameters and Implementation Details

We employ DeepSpeed ZeRO-3 (Rajbhandari et al., 2020) and FlashAttention-2 (Dao, 2023) across all training stages to improve efficiency and reduce computational cost. Qwen2.5-1.5B-SFT and Llama3.1-8B-SFT were trained on the UltraChat200k dataset via LLaMA-Factory, with the following shared settings: batch size of 64, maximum sequence length of 2048, a cosine learning rate schedule (10% warmup over 1 epoch), and the Adam optimizer (Kingma, 2014). Their distinct learning rates are 5e-5 (for Qwen) and 1e-5 (for Llama). These SFT models serve as the starting point for all subsequent baseline training, including DPO, SPIN, and TPAW.

For DPO training, we conducted additional experiments on the UltraFeedback dataset as a supplementary baseline. Specifically, we used LLaMA-Factory with a learning rate of 1e-6 for 2 epochs.

For TPAW, we use a 50k subset of UltraChat200k and adopt the same batch size and sequence length. The model is trained for 2 epochs using RMSProp (Hinton et al., 2012) optimizer, a cosine learning rate schedule with 10% warmup, and a regularization coefficient $\boldsymbol{\lambda} = 0.1$. As UltraChat200k contains multi-round conversations, we only retain the first round as prompt-completion pairs. For the newly introduced hyperparameters in this method, we set $\boldsymbol{\gamma} = 0.5$, $\boldsymbol{\eta} = 6$.

During the four-stage training of TPAW for Qwen, the learning rate was set to 1e-6 for iterations 1–2, then decayed to 1e-7 for iterations 3–4 as the model converged. For Llama, the learning rate was also 1e-6 in iterations 1–2, but decayed to 5e-7 in iterations 3–4 during convergence. In iteration 4, we further reduce the training to 1 epoch and increase $\boldsymbol{\lambda}$ to 5.0. For TPAW on GSM8k, we follow the same setup, except that iteration 3 is trained for only 1 epoch.

Method	Iter-1	Iter-2	Iter-3	Iter-4
SPIN	6:13	14:42	14:42	14:42
TPAW	3:20	7:02	6:10	6:10

Table 3: Total training time per iteration (hours:minutes). Compared to SPIN, TPAW's Iter3 and 4 were trained for only one epoch.

A.2 Computational cost analysis

We analyze the computational and memory overhead of TPAW and clarify its efficiency relative to existing self-play methods.

Importantly, only the current policy model is updated via backpropagation during training. All historical checkpoints involved in the main-player team are used exclusively for forward inference, incurring no additional gradient-related memory cost compared to standard fine-tuning.

To further reduce runtime and memory usage, TPAW adopts a log-probability precomputation strategy: log-probabilities of all reference models are computed once on the training set prior to training, after which the reference models are fully offloaded from GPU memory. During training, each step computes log-probabilities only for the current policy model. On 4×L40S GPUs, precomputing 50k samples per reference model requires approximately 30 minutes, which is negligible relative to total training time.

In practice, this design leads to higher training throughput. We observe an increase in train_steps_per_second from 0.021 (SPIN) to 0.044 (TPAW), corresponding to a 110% speedup per step. Moreover, even after accounting for precomputation, TPAW consistently achieves shorter wall-clock training time across all iterations. Table 3 reports total training time per iteration on 8×A100 80GB GPUs with Llama-3.1-8B.

Overall, TPAW 's forward-only use of historical checkpoints combined with log-probability precomputation ensures a lower effective memory footprint and improved training efficiency, demonstrating superior computational performance compared to prior self-play approaches.

A.3 Evaluation Datasets

A.3.1 Open LLM Leaderboard V1.

Leaderboard v1 evaluates models on 6 key benchmarks using the Eleuther AI Language Model Evaluation Harness, a unified framework designed to test generative language models on a wide range of different evaluation tasks.

- **AI2 Reasoning Challenge (25-shot)**: A set of grade-school science questions. Abbreviated as Arc in Table 1.
- **HellaSwag (10-shot)**: A commonsense reasoning test, which is easy for humans (95% accuracy) but challenging for state-of-the-art models. Abbreviated as HS in Table 1.
- **MMLU (5-shot)**: A test to measure a text model's multitask accuracy. The test covers 57 tasks, including elementary mathematics, U.S. history, computer science, law, and more.
- **TruthfulQA (0-shot)**: A test to measure a model's propensity to reproduce falsehoods commonly found online. Note: TruthfulQA is technically a 6-shot task in the Harness because each example is prepended with 6 Q/A pairs, even in the 0-shot setting. Abbreviated as TQA in Table 1.
- **Winogrande (5-shot)**: An adversarial and difficult Winograd benchmark at scale, for commonsense reasoning. Abbreviated as Wino in Table 1.
- **GSM8k (5-shot)**: Diverse grade school math word problems to measure a model's ability to solve multi-step mathematical reasoning problems.

A.3.2 Open LLM Leaderboard V2.

Leaderboard v2 seeks new benchmarks using high-quality, uncontaminated datasets and evaluates models on key tasks. They chose the following tasks: knowledge testing, reasoning abilities, complex mathematical abilities, and tasks related to human preferences, such as instruction following. They cover these tasks with six benchmarks:

- **MMLU-Pro**: A refined version of the MMLU dataset with higher difficulty, with expert-reviewed questions to reduce noise. Abbreviated as MMLU-p in Table 1.
- **GPQA**: An extremely challenging knowledge dataset, designed by domain experts to ensure difficulty and factual accuracy.
- **MuSR**: A dataset with complex reasoning problems, such as murder mysteries and team allocation issues, requiring long-range reasoning.
- **MATH**: A compilation of high school-level competition problems, requiring precise output formatting, focusing on the hardest questions.

- **IFEval**: A test of a model's ability to strictly follow instructions, focusing on format execution.
- **BBH**: A subset of 23 challenging tasks from the BigBench dataset, involving multi-step reasoning and commonsense knowledge, with performance highly correlated with human preferences.

A.4 Baselines

A.4.1 SFT

Supervised Fine-Tuning (SFT) is a widely used technique to adapt pre-trained language models (LMs) to specific downstream tasks by fine-tuning on labeled datasets. The goal of SFT is to reduce task-related errors and align the model's predictions with human-labeled ground truth.

Given a pre-trained model f_θ, the SFT process involves the following:

Pre-trained Model. The model $f_\theta(x)$ is initially pre-trained on a large corpus using language modeling objectives, allowing the model to learn useful language representations.

Fine-Tuning on Task-Specific Data. The pre-trained model is fine-tuned on a supervised dataset $D = \{(x_i, y_i)\}_{i=1}^N$, where x_i is the input and y_i is the corresponding label. The objective is to minimize the loss function:

$$\mathcal{L}_{\text{SFT}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(f_\theta(x_i), y_i)$$

where $\mathcal{L}$ measures the discrepancy between the model's output and the true label.

Gradient Descent Optimization. The model's parameters θ are updated using gradient-based methods like Stochastic Gradient Descent (SGD) or Adam:

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta\mathcal{L}_{\text{SFT}}(\theta_t)$$

SFT is effective for adapting pre-trained models to specific tasks, though it relies on large labeled datasets for fine-tuning.

A.4.2 SPIN.

SPIN centers on a self-play mechanism where the LLM simultaneously acts as both the main player and the opponent. During fine-tuning, the main player (the current LLM) is trained to differentiate between data distributions produced by the opponent (the LLM from the prior iteration) and human-annotated target responses, iteratively aligning the LLM with the target data distribution.

B Clear View of Model Sequence.

We define the models and their training schemes as follows:

$\boldsymbol{M_0}$: Initial supervised fine-tuning (SFT) model.

$\boldsymbol{M_1}$: Initialized from M_0, then fine-tuned on $\mathcal{D}_0$ with:

$$\text{Opponents: } \pi_0(\cdot|\mathbf{x})$$
$$\text{Main players: } \lambda\cdot\log\frac{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}{\pi_{\boldsymbol{\theta}_0}(\cdot|\mathbf{x})}$$

$\boldsymbol{M_2}$: Initialized from M_1, then fine-tuned on $\mathcal{D}_1, \mathcal{D}_0$ with:

$$\text{Opponents: } \pi_1(\cdot|\mathbf{x}), \pi_0(\cdot|\mathbf{x})$$
$$\text{Main players: } \lambda\cdot\log\frac{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}{\pi_{\boldsymbol{\theta}_1}(\cdot|\mathbf{x})},\ \lambda\cdot\log\frac{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}{\pi_{\boldsymbol{\theta}_0}(\cdot|\mathbf{x})}$$

$\boldsymbol{M_3}$: Initialized from M_2, then fine-tuned on $\mathcal{D}_2, \mathcal{D}_1, \mathcal{D}_0$ with:

$$\text{Opponents: } \pi_2(\cdot|\mathbf{x}), \pi_1(\cdot|\mathbf{x}), \pi_0(\cdot|\mathbf{x})$$
$$\text{Main players: } \lambda\cdot\log\frac{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}{\pi_{\boldsymbol{\theta}_2}(\cdot|\mathbf{x})},\ \lambda\cdot\log\frac{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}{\pi_{\boldsymbol{\theta}_1}(\cdot|\mathbf{x})},\ \lambda\cdot\log\frac{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}{\pi_{\boldsymbol{\theta}_0}(\cdot|\mathbf{x})}$$

$\boldsymbol{M_4}$: Initialized from M_3, then fine-tuned on $\mathcal{D}_3, \mathcal{D}_2, \mathcal{D}_1$ with:

$$\text{Opponents: } \pi_3(\cdot|\mathbf{x}),\ \pi_2(\cdot|\mathbf{x}),\ \pi_1(\cdot|\mathbf{x})$$
$$\text{Main players: } \lambda\cdot\log\frac{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}{\pi_{\boldsymbol{\theta}_3}(\cdot|\mathbf{x})},\ \lambda\cdot\log\frac{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}{\pi_{\boldsymbol{\theta}_2}(\cdot|\mathbf{x})},\ \lambda\cdot\log\frac{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{x})}{\pi_{\boldsymbol{\theta}_1}(\cdot|\mathbf{x})}$$

C Notation Summary

Table 4 summarizes the main symbols used throughout the paper, especially the policy notation across iterations.

Notation	Meaning
$\pi_{\boldsymbol{\theta}}$	Current policy model being optimized at the present training step. Only this parameter set is updated by backpropagation.
$\pi_{\boldsymbol{\theta}_t}$	Policy checkpoint obtained after completing iteration t; also used as the newest opponent model at iteration $t+1$.
$\pi_{\boldsymbol{\theta}_j}$	A generic historical checkpoint indexed by $j \in \{t, t-1, t-2\}$, used as a fixed reference model in main-player scoring.
t	Iteration index of team-based self-play.
$\mathbf{x}, \mathbf{y}$	Prompt and target response from the SFT dataset.
$\mathbf{y}^{\text{gen}}$	Model-generated response sampled from an opponent policy for prompt $\mathbf{x}$.
$\mathcal{D}_{\text{SFT}}$	Original supervised fine-tuning dataset of prompt-response pairs.
$\mathcal{D}_t$	Triple-form dataset collected at iteration t, i.e., $\{(\mathbf{x}, \mathbf{y}, \mathbf{y}^{\text{gen}})\}$.
$\mathcal{D}_O$	Aggregated opponent dataset formed by recent triple datasets, e.g., $\mathcal{D}_t \cup \mathcal{D}_{t-1} \cup \mathcal{D}_{t-2}$.
$P_j(\mathbf{x}, \mathbf{y})$	Main-player score defined by the log-ratio between the current policy $\pi_{\boldsymbol{\theta}}$ and a historical checkpoint $\pi_{\boldsymbol{\theta}_j}$.
$\mathcal{L}_j$	Training objective of the j-th main player.
$\boldsymbol{\alpha}_j$	Adaptive target-response weight for player P_j.
$\boldsymbol{\beta}_j$	Adaptive player weight assigned to the j-th main player.
λ	Regularization coefficient controlling deviation between the updated policy and the previous checkpoint.
$\boldsymbol{\eta}$	Hyperparameter controlling the strength of target-response reweighting.
$\boldsymbol{\gamma}$	Hyperparameter controlling the sharpness of player-weight assignment.
$N_{\max}$	Maximum team size, i.e., the number of historical checkpoints retained in the team.

Table 4: Summary of the main notations used in TPAW.

Iteration	Model	Acc (%)
-	Qwen2.5-1.5B-SFT	51.10
Iter 1	SPIN	53.75 (+2.65)
	TPAW	54.13 (+3.11)
	w/o TRW	54.13 (+3.11)
	w/o MPW	54.13 (+3.11)
	w/o TRW+MPW	54.36 (+3.24)
	w/o Team	53.30 (+2.20)
Iter 2	SPIN	54.36 (+0.61)
	TPAW	55.19 (+1.06)
	w/o TRW	56.18 (+2.05)
	w/o MPW	55.65 (+1.52)
	w/o TRW+MPW	55.95 (+1.59)
	w/o Team	55.12 (+1.82)
Iter 3	SPIN	53.75 (-0.61)
	TPAW	56.56 (+1.37)
	w/o TRW	55.42 (-0.76)
	w/o MPW	56.40 (+0.75)
	w/o TRW+MPW	55.42 (-0.53)
	w/o Team	55.19 (-0.62)
Iter 4	SPIN	54.59 (+0.84)
	TPAW	**56.94** (+0.38)
	w/o TRW	55.42 (±0.00)
	w/o MPW	55.95 (-0.45)
	w/o TRW+MPW	55.65 (+0.23)
	w/o Team	55.65 (+0.46)

Table 5: Performance of TPAW based on Qwen2.5-1.5B-SFT across GSM8k datasets. Both SPIN and TPAW are trained on the GSM8k training set. The numbers in the subscripts represent the accuracy changes compared to the corresponding methods in the previous iteration.

D Additional Experiment Results

D.1 Qualitative GSM8K Case Study

To complement the quantitative GSM8K results in the main text, we provide two representative case studies with verbatim model outputs. The first example illustrates a success case where TPAW corrects a structural reasoning error made by SPIN. The second example highlights a remaining limitation: TPAW improves global reasoning consistency, but does not completely eliminate token-level arithmetic mistakes.

Case 1: Multi-Entity Aggregation Failure. We observe a common failure mode in SPIN-style iterative self-training: the model may lose track of earlier entities during the final aggregation step in multi-step reasoning problems.

Model	Open LLM Leaderboard V1						Avg.
	Arc	TruthfulQA	Winogrande	GSM8k	HellaSwag	MMLU	
Qwen2.5-1.5B-SFT	50.94	46.83	64.80	51.25	64.91	58.93	56.28
Qwen2.5-1.5B-SFT-epoch2	51.79	45.93	62.35	49.36	65.64	57.17	55.37
Qwen2.5-1.5B-SFT-epoch3	52.47	44.87	62.12	49.43	66.70	57.31	55.48
TPAW Iter-1 (Ours)	52.99	47.64	64.96	52.77	65.77	59.50	57.27
TPAW Iter-2 (Ours)	53.41	47.91	64.17	53.15	66.08	59.58	57.38
TPAW Iter-3 (Ours)	53.67	47.91	65.04	53.60	66.18	59.52	57.65
TPAW Iter-4 (Ours)	52.56	47.90	65.11	55.04	66.30	59.64	**57.76**
Model	**Open LLM Leaderboard V2**						**Avg.**
	IFEval	BBH	Math	GPQA	MUSR	MMLU-pro	
Qwen2.5-1.5B-SFT	31.73	15.15	6.65	3.47	6.02	17.38	13.40
Qwen2.5-1.5B-SFT-epoch2	32.81	12.79	8.23	2.13	3.69	16.44	12.68
Qwen2.5-1.5B-SFT-epoch3	33.12	14.26	7.25	4.92	2.08	16.56	13.03
TPAW Iter-1 (Ours)	32.35	15.60	8.69	3.24	5.66	17.65	13.87
TPAW Iter-2 (Ours)	35.52	15.37	8.69	4.36	5.98	17.72	14.61
TPAW Iter-3 (Ours)	36.10	15.03	10.20	4.14	5.43	18.03	**14.82**
TPAW Iter-4 (Ours)	36.04	15.51	9.37	4.36	4.73	17.82	14.64

Table 6: Detailed evaluation data of More Epochs of SFT.

Model	Open LLM Leaderboard V1						Avg.
	Arc	TruthfulQA	Winogrande	GSM8k	HellaSwag	MMLU	
Qwen2.5-1.5B-SFT	50.94±1.46	46.83±1.52	64.80±1.34	51.25±1.38	64.91±0.48	58.93±0.40	56.28
TPAW Iter-4 (Ours)	52.56±1.46	47.90±1.52	65.11±1.34	55.04±1.37	66.30±0.47	59.64±0.40	57.76
Model	**Open LLM Leaderboard V2**						**Avg.**
	IFEval	BBH	Math	GPQA	MUSR	MMLU-pro	
Qwen2.5-1.5B-SFT	31.73±1.83	15.15±0.60	6.65±0.67	3.47±1.30	6.02±1.70	17.38±0.40	13.40
TPAW Iter-4 (Ours)	36.04±1.96	15.51±0.60	9.37±0.77	4.36±1.30	4.73±1.66	17.82±0.40	14.64

Table 7: Example Experiment Results with Standard Error.

Problem.

In a neighborhood, the number of rabbit pets is twelve less than the combined number of pet dogs and cats. If there are two cats for every dog, and the number of dogs is 60, how many pets in total are in the neighborhood?

Ground truth (abridged).

Cats $= 2 \times 60 = 120$
Dogs + Cats $= 180$
Rabbits $= 180 - 12 = 168$
Total $= 168 + 180 =$ **348**

SPIN output (verbatim).

If there are two cats for every dog, then there are $60 \times 2 = 120$ cats.
The combined number of pet dogs and cats is $120 + 60 = 180$ pets.
The number of rabbit pets is $180 - 12 = 168$ pets.
The total number of pets in the neighborhood is $168 + 60 = 228$ pets.
228

TPAW output (verbatim).

If there are two cats for every dog, then there are $60 \times 2 = 120$ cats.
The combined number of pet dogs and cats is $120 + 60 = 180$.
The number of rabbit pets is $180 - 12 = 168$.
The total number of pets in the neighborhood is $168 + 60 + 120 = 348$.
348

Analysis. SPIN correctly computes all intermediate quantities, but drops the cat count during the final aggregation step, summing only rabbits and dogs. This reflects a context-dropout phenomenon in iterative self-training, where the model over-relies on the most recent intermediate variable.

Why TPAW helps. TPAW's team-based self-play introduces historical checkpoints as main players. When the current policy begins to neglect earlier entities, gradients from team members that still correctly score the full aggregation provide a cor-

rective signal. This acts as a regularizer against partial forgetting in long reasoning chains, which is difficult to achieve with a self-play trajectory.

Case 2: Token-Level Arithmetic Slip. We also note that TPAW does not eliminate all GSM8K failure modes.

Problem.

> Lloyd has an egg farm. His chickens produce 252 eggs per day and he sells them for $2 per dozen. How much does Lloyd make on eggs per week?

Ground truth (abridged).

> $252 \times 7 = \mathbf{1764}$ eggs per week ...

TPAW output (verbatim).

> There are 7 days in a week, so Lloyd sells $252 \times 7 = \mathbf{1784}$ eggs per week ...

Interpretation. Here, TPAW makes a token-level arithmetic slip ($1764 \rightarrow 1784$) despite following the correct reasoning structure. This suggests that TPAW mainly improves global reasoning consistency and entity tracking, but does not directly address low-level arithmetic precision. Addressing such errors likely requires orthogonal techniques such as tool use or arithmetic verification.

Summary. These case studies suggest that the GSM8K gains of TPAW primarily stem from reducing structural reasoning failures, such as entity omission and aggregation inconsistency, rather than uniformly improving all aspects of numerical accuracy.

D.2 Further Ablation Study

- Without adaptive Target Response Weighting and adaptive Main Player Weighting (**w/o TRW+MPW**): both the target response weight and main players weight are fixed.

See Table 5 for detailed evaluation data of ablation study. The experimental results on w/o TRW+MPW are basically consistent with the conclusions of the main text, demonstrate that the dual adaptive weighting mechanism in TPAW are critical to achieving optimal performance.

D.3 Further Experiment Result for More Epochs of SFT

As we showed in the main text training for more epochs on the dataset using SFT does not lead to further performance improvements, and may even lead to a decrease in performance. See Table 6 for detailed evaluation data.

D.4 Error Bars of Experiment Result

Here we present a subset of the Standard Error data obtained from our harness-based evaluations in Table 7. The relatively low Standard Error values indicate that the evaluation process is stable. Moreover, the Standard Error does not increase noticeably after training, suggesting that the performance improvements are not due to chance but are statistically significant and consistently stable.

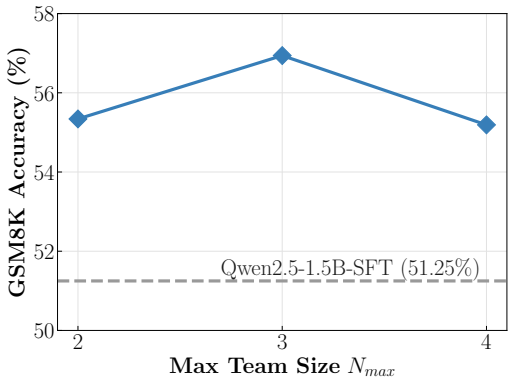


Figure 5: Impact of $N_{\max}$ on GSM8K accuracy.

γ	0.00	0.25	0.50	0.75	1.00
Iter-1	54.13	55.57	54.13	54.28	55.57
Iter-2	55.65	55.19	55.19	55.12	55.95
Iter-3	56.40	55.19	56.56	55.34	56.03
Iter-4	55.95	55.27	**56.94**	55.95	55.57

(a) Effect of γ.

η	1	2	4	6	8	10
Iter-1	54.13	54.06	54.21	54.13	54.21	54.13
Iter-2	56.18	54.82	55.95	55.19	56.53	56.33
Iter-3	55.42	54.28	55.27	56.56	56.41	56.79
Iter-4	55.42	54.59	55.27	**56.94**	56.71	56.63

(b) Effect of η.

N_{max}	2	3	4
Iter-3	54.89	56.56	56.56
Iter-4	55.34	**56.94**	55.19

(c) Effect of N_{max}.

Table 8: Detailed GSM8K results for the hyperparameter study. Panels (a)–(c) show the effects of γ, η, and N_{max}, respectively.

D.5 Further Experiment Result for Impact of Hyperparameters

We further investigate the impact of the team size hyperparameter $N_{\max}$, which denotes the final number of team members and corresponds to the size of the historical checkpoint window considered throughout the game. The experimental results are presented in Figure 5. Notably, when $N_{\max}$ is set to 2 or 4, the observed suboptimal performance may indicate the effects of underfitting and overfitting, respectively.

Additionally, we report comprehensive experimental results for the hyperparameter study in Table 8. These results further demonstrate the soundness of our hyperparameter choices and the robustness of the proposed method.