TATRL: CLOSED-LOOP WORLD MODEL EVOLUTION FOR LONG-HORIZON LLM PLANNING*

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models learn implicit world models through next-token prediction, yet leveraging this capacity for long-horizon planning remains challenging. Single-step methods, including supervised fine-tuning and turn-level optimization, cannot capture this structure, as they implicitly assume that the quality of each action is independent of the trajectory that generated it. This assumption holds for single-step tasks such as mathematical reasoning, but breaks down in long-horizon sequential tasks where actions are optimal only as a consequence of preceding moves. To address this, we introduce *closed-loop world model evolution*, a framework that treats entire occupancy trajectories as the unit of both generation and evaluation. Building on the connection between trajectory-level evaluation and multi-scale world modeling, we develop a progressive temporal scaling algorithm that trains within a self-play framework at increasingly long horizons, analogous to varying the planning horizon in geometric horizon models. We evaluate our approach on eight complex long-horizon tasks from the DouDizhu, GuanDan, and poker domains and find that trajectory-level evaluation is both necessary and sufficient for learning compositionally robust policies, achieving significant improvements across all benchmarks. Moreover, we achieve 99.2% action-sequence consistency while using 20$\times$ less demonstration data than single-step baselines, highlighting the sample efficiency of closed-loop world model evolution.

1 INTRODUCTION

As large language models become increasingly deployed in long-horizon planning pipelines (OpenAI et al., 2024b; Grattafiori et al., 2024; Team et al., 2025a), a growing body of evidence suggests that these models implicitly learn world models through next-token prediction—capturing transition dynamics, causal structure, and strategic reasoning from data alone. Yet leveraging this implicit world-modeling capacity for long-horizon planning remains an open challenge (Xi et al., 2023). One line of work attempts to catch suboptimal behaviors by applying supervised fine-tuning alone; however, these approaches are often insufficient in game environments where strategic decisions are context-dependent and immediate feedback (Nananukul & Wongkamjan, 2024), like turn-level rewards, are not available for many long-horizon games.

Recently, reinforcement learning has been explored to unlock the long-horizon planning capacity of these implicit world models (Silver et al., 2016; Zhang et al., 2025a;a). In particular, prior methods apply policy gradient algorithms (Sutton et al., 1999; Schulman et al., 2017) to LLM agents by removing suboptimal actions according to some reward signal to a desired performance level α. Subsequent trajectory-level optimization issues rewards but lacks mechanisms to ensure turn-level credit assignment. However, both approaches implicitly assume the quality of a decision can be evaluated independently, which limits their applicability to single-step settings, where actions require minimal context. For example, in solving long-horizon games, a given action is often optimal as a result of preceding moves: generally, strategic decisions require trajectory-level evaluation.

To tackle this challenge, we propose a new framework to account for the trajectory-level nature of long-horizon games, provide an algorithm which applies progressive temporal scaling to training over a self-play curriculum, and give theoretical grounding over the approach design:

*"Tat" derives from Goethe's *Faust*: "Im Anfang war die Tat!" (In the beginning was the deed!)—reflecting our thesis that effective agent learning emerges from action and closed-loop interaction with the environment.

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

Closed-loop world model evolution. We present a framework that accounts for trajectory-level context to evaluate actions in a more principled manner. This approach requires that strategic decisions are both generated and evaluated by treating entire trajectories of gameplay as units rather than evaluating isolated actions.

Progressive temporal scaling. To apply this trajectory-level notion of training, we propose a curriculum strategy for long-horizon games—analogous to varying the planning horizon in geometric horizon models (Janner et al., 2021; Thakoor et al., 2022)—and an optimization procedure for obtaining such a curriculum.

Extensive validation. We validate our method on a variety of complex game environments from the card game domain (Zha et al., 2020; Wang et al., 2025b; Yim et al., 2024) and from poker variants (Zhuang et al., 2025), and compare with different baseline approaches. We find that our closed-loop world model evolution is often both necessary (trajectory-level evaluation satisfies strategic requirements) and sufficient (training that ignores trajectory context does not satisfy performance requirements) to ensure effective learning. We achieve performance as strong as the teacher model with improved cross-opponent generalization.

1.1 Related Work

Reinforcement learning is a policy optimization technique which yields cumulative reward maximization over a trajectory horizon in a gradient-based manner (Sutton et al., 1999), traditionally only assuming differentiability of the policy. Supervised fine-tuning is a training paradigm which relies on a held out demonstration dataset to yield behavioral cloning in isolation over the quality of the training data (Ouyang et al., 2022). While reinforcement learning has been explored under LLM settings, this has largely been in the context of single-turn reasoning (Wei et al., 2023; Kojima et al., 2023; Yao et al., 2023b) or RLHF preference alignment (Ouyang et al., 2022), rather than long-horizon training that leverages the implicit world model learned by these systems.

Recent work has sought to apply reinforcement learning to LLM agents, including agentic task completion frameworks (Xi et al., 2024b; Zhou et al., 2024; Yao et al., 2023a; Shridhar et al., 2021; Machado et al., 2018; Wang et al., 2022; Fan et al., 2022; Wei et al., 2025; Prasad et al., 2024), as well as game-playing and self-evolution training pipelines (Wang et al., 2025b; Zhang et al., 2025b; Yuan et al., 2025; Duan et al., 2024) and test-time scaling approaches (Team et al., 2025c;b; He et al., 2025).

LLM4CardGame & DanZero (Wang et al., 2025b) apply supervised fine-tuning with expert demonstrations, splitting trajectories into action-observation pairs, and obtaining imitation policies for each such that removing actions below a corresponding quality threshold yields an α-level performance bound. AgentRL (Zhang et al., 2025a;a) extends this paradigm by introducing multi-turn multi-task training, lowering the data requirements while simultaneously ensuring transfer of the learned skills. DeepSeek-R1 & OpenAI o1 extend single-turn reasoning to give chain-of-thought reasoning capabilities using the policy gradient optimization techniques developed by prior work (Shao et al., 2024; Hu et al., 2025). However, while these methods are effective in their respective domains, where actions may be treated as independent, they do not transfer to game environments, where the quality of each action cannot be evaluated without the context of the trajectory that generated it.

Scaling Interactions. Inference-time-compute (ITC) scaling induces LLMs to produce chain-of-thought reasoning to support their outputs, similar to the human deliberative reasoning process (Wei et al., 2023; Kojima et al., 2023). Several methods have been proposed to explore different approaches to this effect by sampling more and selecting over multiple outputs (Xi et al., 2024a; Luong et al., 2024), incorporating different types of feedback (e.g. self-reflection, environment rewards) and verification (Aksitov et al., 2023; Olausson et al., 2024; Bouzenia et al., 2024). We show how our ScalingInter-RL approach can be used as train-time-compute to get more robust agents.

2 Preliminaries

We consider the problem of planning over long temporal horizons in partially observable sequential decision processes. In our setting, the agent is a large language model whose implicit world model—learned through next-token prediction on large-scale data—must be leveraged for multi-step strategic

reasoning. Unlike single-step prediction tasks, where the quality of each output can be evaluated in isolation, long-horizon tasks require reasoning over the *entire trajectory of future states* induced by a policy. This section formalizes the sequential decision process, introduces the notion of trajectory-level state occupancy as the central object of evaluation, and establishes the planning guarantees that underpin our approach.

Sequential decision process. We model the environment as a partially observable Markov decision process (POMDP). At each step k, the agent receives an observation $o_k \in \mathcal{O}$ and selects an action $a_k \in \mathcal{A}$. The environment then transitions to a new state according to a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, producing the next observation. Observations may be incomplete: for example, "The current hand is [A, A, K, K, Q]" reveals the agent's own state, while "Player 2 has passed" withholds the opponent's remaining cards. In practice, we implement the transition function using a game simulator (Figure **??**).

Trajectory-level state occupancy. A single action in a long-horizon task cannot be evaluated without the temporal context that produced it. To capture this, we define the central object of our framework: the trajectory-level state occupancy.

Definition 2.1 (State Occupancy Trajectory)**.** A state occupancy trajectory is an ordered sequence of observation-action pairs culminating in a terminal outcome. Formally, we define $\mathcal{T}$, the set of all possible trajectories $\tau = (o_0, a_0, o_1, a_1, \ldots, o_T, a_T, R)$, where T varies by task horizon (20–50 steps for DouDizhu, 50–200+ steps for GuanDan).

This definition reflects a key structural property: the quality of any individual action a_k depends on the full occupancy trajectory τ that contains it. A move that appears suboptimal in isolation may be optimal when evaluated within the context of the sequence that generated it. This is precisely the challenge that single-step methods—which evaluate actions independently—cannot address, and the reason we require trajectory-level evaluation (Section **??**).

Dynamics-consistent action space. Not all actions are consistent with the dynamics of the environment at every state. We formalize this constraint as follows.

Definition 2.2 (Valid Action Space)**.** The valid action space is the subset of all actions that are consistent with the transition dynamics at a given state, determined by a known set of environment rules from which we evaluate action validity.

Remark 2.3. The valid action space may be state-dependent and context-sensitive. We use a reference implementation such as RLCard (Zha et al., 2020) or a specialized game engine as the dynamics validator. For instance, while playing a single card may be valid in general, it is not valid in a state that requires a pair.

Evaluation across temporal scales. Given the trajectory-level structure above, we now establish how to evaluate and improve policies that generate occupancy trajectories. The key idea is to decompose the evaluation across temporal scales: each trajectory is scored holistically, and actions are retained or removed based on their contribution to the trajectory-level outcome.

Concretely, for a policy π_θ, each occupancy trajectory τ is evaluated according to an advantage function A_k that compares the outcome at step k to alternative outcomes from the same state. Actions whose probability ratio ρ_k exceeds a threshold are reinforced; those below are penalized (Sutton et al., 1999; Schulman et al., 2017). This yields the following planning guarantee over grouped trajectories, where $\bar{R}_G$ denotes the group mean:

$$\mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{k=0}^{K} \nabla_\theta \log \pi_\theta(a_k|s_k) \cdot A_k\right] \leq \nabla_\theta J(\theta)$$
$$\leq \mathbb{E}_{\tau\sim\pi_{\theta_{\text{old}}}}\left[\sum_{k=0}^{K} \rho_k \cdot A_k\right]. \tag{1}$$

This guarantee holds under two assumptions: (i) data exchangeability with no ties in scores (enforceable by continuous noise injection), and (ii) *temporal decomposability*—the advantage of a full occupancy trajectory decomposes as the sum of contributions from each of its constituent steps. Under these conditions, by removing actions in trajectories with advantage below $\bar{A}_{\text{threshold}}$ and

recombining the remaining steps into a refined policy $\pi_{\theta'}$, the guarantee extends to the long-horizon setting.

The connection to multi-scale world modeling is direct: temporal decomposability implies that we can evaluate the quality of a policy at *different temporal resolutions*—from individual steps to full trajectories—and that improvements at each scale compose. Our progressive temporal scaling algorithm (Section **??**) exploits this by training at progressively longer horizons, analogous to varying the discount factor γ in geometric horizon models (Janner et al., 2021; Thakoor et al., 2022).

3 THE TATRL-EVOLUTION FRAMEWORK

While the approach of prior work in LLM agent training yields a useful notion of policy optimization, this notion implicitly makes the strong assumption that actions are evaluated independently, so we call it independent action optimization. The assertion that (a_t is optimal) $\Leftrightarrow$ (r_t is maximal) treats each turn's action of the other turns in the trajectory. While this may be appropriate for pure single-turn tasks, like mathematical reasoning (Shao et al., 2024), we find that it is not sufficient to preserve output quality for multi-turn game tasks. Our notion of closed-loop evolution further imposes trajectory-level coherence by requiring both action validity and strategic consistency across the entire game sequence.

Definition 3.1 (Closed-Loop Evolution)**.** Given a game configuration q and policy π_θ, an output trajectory $\tau = (o_0, a_0, \ldots, o_T, a_T, R) \in \mathcal{T}$ of distinct turns is closed-loop valid if it satisfies: each action a_t is derivable from $(o_0, a_0, \ldots, o_{t-1}, a_{t-1}, o_t)$, the current observation, and the game rules.

We omit a formal definition for derivable because derivability is both sensitive context and dependent game (an action may follow for expert players but not for novices, unless they are provided sufficient reasoning time). Note that we require an action in the trajectory to be derivable from its history, the current observation, and the game configuration q, since information like opponent tendencies will be revealed to the agent. As noted before, the valid action space is determined in part by the game state (it is not appropriate to assume a "pass" action in the context of a mandatory play).

Closed-loop evolution ensures outputs contain sufficient dependencies between previous turns and subsequent ones and considers sequences of actions rather than simply sets. Turns must appear in valid sequence. For instance, a card must be played before it affects the game state. Given a set of trajectories $\mathcal{T}$, we write $\tau \in \mathcal{T}$ to denote a particular ordering of those turns.

Observation 3.2. If a trajectory $(o_0, a_0, \ldots, o_T, a_T, R)$ is closed-loop valid, any prefix $(o_0, a_0, \ldots, o_t, a_t)$ for $t < T$ is also closed-loop valid.

3.1 MODULAR ARCHITECTURE OF CLOSED-LOOP SYSTEMS

Assumption 3.3 (Module Independence)**.** Fix some game configuration q, policy π_θ and trajectory τ. Say that a module M is responsible for some subset of {observation, action, reward}, and call the subset $\mathcal{R}_M$. Then, if M' is a different module responsible for a subset of {observation, action, reward}, M remains responsible for $\mathcal{R}_M$.

TatRL-Evolution adopts a modular design with well-defined responsibilities for each module, allowing for extensibility. The framework is organized into three core modules, unified by the closed-loop self-evolution principle formalized in Algorithm 1.

Agent Module. The Agent Module encapsulates the reasoning-action loop of LLM-based agents. This module yields a policy mapping function (DAG) where valid actions descend from observations, and invalid or illegal actions do not.We develop approximations of these ideals that suffice to achieve closed-loop evolution. The module receives observations from the environment, performs reasoning over multiple turns, and outputs actions. The module supports different prompting strategies (chain-of-thought (Wei et al., 2023), self-correction with 3 rounds) and sampling configurations (temperature, top-p, test-time scaling @128).

Training Module. We define a weaker notion of an optimal policy and find this notion is satisfied by RL-trained agents. This weaker notion is sufficient to maintain validity during gameplay while ensuring strategic improvement on average. Unlike perfect play, these policies do not guarantee wins in every game or represent the minimax-optimal strategy; instead, they capture sufficient conditions

Table 1: Performance comparison across 8 card games. Fine-tuned models vs. zero-shot baselines. **Bold**: best; underline: second-best. "+RL" = our SFT+RL with 50K supervised samples. **DouDizhu/GuanDan: win rate (%) vs. rule-based. Others: reward scores per RLCard.**

Model	Complex Games (%)		Multi-Player Games			Poker Variants			AFC
	DouDizhu	GuanDan	Riichi	Uno	Gin	Leduc	Limit	NoLimit	
API-based Models (Zero-shot)									
GPT-4o-mini	19.5	1.9	0.15	0.128	−0.18	0.30	0.45	2.47	72.5%
GPT-4o (OpenAI et al., 2024b;a)	18.0	1.9	0.25	0.072	0.41	0.84	0.60	2.73	74.8%
Claude-Sonnet-4 (Grattafiori et al., 2024)	21.0	2.1	0.20	0.095	0.32	0.78	0.55	2.85	76.2%
GLM-4-air	33.0	0.0	0.10	−0.07	−0.42	−0.12	1.13	−0.89	68.5%
GLM-4-plus (GLM et al., 2024)	34.5	1.9	0.00	0.020	−0.34	0.80	0.63	3.21	71.2%
DeepSeek-V3 (DeepSeek-AI et al., 2025)	32.0	0.0	0.15	0.128	0.15	0.77	0.22	0.18	75.8%
DeepSeek-R1	18.5	2.0	0.05	**0.148**	0.23	**0.88**	0.24	1.88	78.2%
Base Models (Instruct, Zero-shot)									
Qwen2.5-7B-Instruct (Qwen et al., 2025)	8.7	0.0	0.04	0.032	−0.53	0.63	1.05	1.25	65.2%
Qwen2.5-14B-Instruct (Qwen et al., 2025)	14.2	0.0	0.08	0.085	−0.38	0.72	0.95	1.68	68.5%
Llama3.1-8B-Instruct	15.5	0.0	0.08	0.120	−0.46	0.62	−0.04	−2.10	62.8%
GLM4-9B-Chat (GLM et al., 2024)	13.1	0.0	0.08	0.000	−0.36	0.52	0.85	−0.44	64.5%
Ours (SFT+RL, 50K supervised + 500K–800K self-play interactions)									
Qwen2.5-7B+RL	85.2±1.2	79.2±0.8	1.24	0.185	0.34	1.32	2.82	5.12	98.5%
Qwen2.5-14B+RL	89.4±0.9	83.5±0.6	1.42	0.268	0.38	1.38	2.95	6.28	99.1%
Llama3.1-8B+RL	87.0±1.0	81.9±0.7	1.38	0.242	0.35	1.32	2.85	6.18	98.8%
GLM4-9B+RL	**90.2±0.8**	**85.6±0.5**	**1.48**	**0.328**	**0.42**	**1.42**	**3.05**	**6.52**	**99.2%**

for improvement (see Observation below for a formal definition of sufficiency). While they don't tell us which actions are optimal, they indicate which actions are supportive for winning and give a natural (temporal) ordering on the turns. The Training Module provides a unified RL pipeline that supports both on-policy (PPO (Schulman et al., 2017)) and off-policy (GRPO (Shao et al., 2024), REINFORCE++ (Hu et al., 2025)) algorithms, offering researchers a flexible foundation for large-scale LLM agent training.

Definition 3.4 (Sufficient Policy Graph)**.** Let $\mathcal{G} = (V, E)$ be a policy graph for (q, π_θ). Each vertex $v \in V$ represents a turn $t \in [0, T]$. The edge set E must satisfy the following: (1) trajectory-wise validity: for any trajectory $\tau = (o_0, a_0, \ldots, o_T, a_T, R)$ that includes all turns of its prefix, if a closed-loop valid extension exists for τ, then every proper prefix of τ must also be closed-loop valid, and (2) monotonicity: if a partial trajectory τ_{partial} does not allow a winning completion, then any shorter prefix $\tau' \subset \tau_{\text{partial}}$ must not guarantee a winning completion either.

TatRL Self-Evolution Infrastructure. A distinguishing feature of our framework is TatRL, a self-evolving infrastructure where training scripts themselves evolve through execution feedback. The core insight is that the debugging process (try configuration → observe failure → diagnose cause → apply fix → retry) follows the same closed-loop pattern as agent learning. Algorithm 1 formalizes this process: given a task $\mathcal{T}$, hardware specifications $\mathcal{H}$, and an initial script v_0, the system iteratively generates scripts (Line 5), executes them (Line 6), classifies any errors (Line 8), and generates AI-guided fixes (Line 9) until convergence. This closed-loop evolution achieves 95% success rate across 50 diverse configurations (varying GPU counts, model sizes, batch sizes), reducing human engineering effort from 40+ hours to <2 hours while matching expert-level 97% success rate.

3.2 Optimization Methodology

We now formalize the optimization objectives in TatRL-Evolution, which operates at two complementary granularities: *trajectory-level* optimization for holistic game strategy and *turn-level* credit assignment for fine-grained action refinement.

Game Trajectory Formulation. Unlike single-turn reasoning tasks where the agent produces one response per query, multi-turn games require sequential decision-making over extended horizons. We define a game trajectory as $\tau = (o_0, a_0, o_1, a_1, \ldots, o_T, a_T, R)$, where $o_t \in \mathcal{O}$ denotes the partial observation at turn t (visible cards, game state, opponent actions), $a_t \in \mathcal{A}(o_t)$ is the action selected from the dynamic valid action set (up to 27K+ in GuanDan), and $R \in \{-1, 0, +1\}$ is the terminal outcome reward. The trajectory length T varies significantly across games (20–50 for DouDizhu, 50–200+ for GuanDan), necessitating specialized optimization techniques.

Trajectory-Level Objective with Grouped Outcome Advantage. For each query (game configuration) q, we sample a group of G complete trajectories $\{\tau_i\}_{i=1}^{G}$ via self-play. The trajectory-level objective maximizes expected return while maintaining proximity to the reference policy:

$$\mathcal{L}_{\pi_\theta}^{\text{traj}}(\theta) = \mathbb{E}_{q,\{\tau_i\}_{i=1}^{G}}\Big[\frac{1}{G}\sum_{i=1}^{G}\sum_{t=0}^{T_i}\mathcal{L}_{i,t}^{\text{clip}} \cdot \mathbf{1}_{i,t}\Big] \tag{2}$$

where the expectation is over $q \sim \mathcal{D}$ and $\{\tau_i\}_{i=1}^{G} \sim \pi_\theta(\cdot|q)$, and $\mathbf{1}_{i,t}$ is the indicator for tokens generated by π_θ at turn t (excluding environment observations). $\mathcal{L}_{i,t}^{\text{clip}}$ denotes the clipped surrogate loss.

The key distinction from prior work lies in our *Grouped Outcome Advantage* (GOA) computation. Given terminal rewards $\{R_i\}_{i=1}^{G}$ for the group, we compute:

$$A_i^{\text{GOA}} = \frac{R_i - \bar{R}_G}{\sigma_G + \epsilon}, \quad \text{where}$$
$$\bar{R}_G = \frac{1}{G}\sum_{j=1}^{G} R_j, \quad \sigma_G = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(R_j - \bar{R}_G)^2} \tag{3}$$

This group-based normalization is crucial for game environments where rewards are sparse and binary. When $|\{R_i\}| = 1$ (all outcomes identical), we set $\bar{R}_G = 0$ and $\sigma_G = 1$ to maintain gradient signal, enabling learning even from homogeneous rollouts—a scenario frequent in early training when the policy is weak.

Dual-Bound Clipped Policy Gradient. Standard PPO (Schulman et al., 2017) clips the probability ratio to $[1-\epsilon, 1+\epsilon]$. However, in game environments with high-variance advantages, we observe that negative advantages can lead to excessively large policy updates. We introduce a *dual-bound* clipping mechanism inspired by recent advances in LLM RL (Yu et al., 2025). Let $\rho_{i,t} = \pi_\theta(a_t|o_t)/\pi_{\theta_{\text{old}}}(a_t|o_t)$ denote the importance sampling ratio. For positive advantages ($A_i \geq 0$):

$$\mathcal{L}_{i,t}^{\text{clip}} = \max\big(-\rho_{i,t}A_i,\ -\bar{\rho}_{i,t}A_i\big) \tag{4}$$

where $\bar{\rho}_{i,t} = \text{clip}(\rho_{i,t}, 1-\epsilon_l, 1+\epsilon_h)$. For negative advantages ($A_i < 0$):

$$\mathcal{L}_{i,t}^{\text{clip}} = \min\big(-\rho_{i,t}A_i,\ -c \cdot A_i\big) \tag{5}$$

where $c > 1$ is the lower bound constant (default $c = 3$), and ϵ_l, ϵ_h allow asymmetric clipping (default $\epsilon_l = \epsilon_h = 0.2$). The lower bound c prevents the policy from moving too aggressively away from bad actions, which we find essential for stable training in long-horizon games.

Turn-Level Credit Assignment. While trajectory-level optimization provides the overall learning signal, credit assignment across 50–200+ turns remains challenging. We introduce turn-level refinement inspired by temporal difference learning (Sutton et al., 1999). For each turn t within trajectory τ_i, we define the turn-level return as the discounted cumulative reward:

$$G_t^{(i)} = \sum_{k=t}^{T_i}\gamma^{k-t}r_k, \quad r_k = R_i \cdot \mathbf{1}[k = T_i] \tag{6}$$

where $\gamma \in [0,1]$ is the discount factor and R_i is the terminal outcome reward (only nonzero at $k = T_i$). The turn-level advantage is then whitened across the loss mask:

$$A_t^{\text{turn}} = \frac{G_t - \mu_{\mathcal{M}}}{\sigma_{\mathcal{M}} + \epsilon} \tag{7}$$

where $\mu_{\mathcal{M}} = \mathbb{E}_{t\in\mathcal{M}}[G_t]$ and $\sigma_{\mathcal{M}} = \text{std}_{t\in\mathcal{M}}(G_t)$ are computed over the set of valid (non-padded) positions $\mathcal{M}$. This masked whitening ensures stable gradients despite variable sequence lengths across different games.

Cross-Policy Sampling for Exploration. In multi-player games, the optimal strategy depends on opponent behavior. Training against a fixed opponent leads to overfitting. We maintain an

Table 2: Head-to-head vs. DanZero v2.1 on GuanDan. Win rate (%) over 1000 games × 3 seeds = 3000 total games.

Model	vs. DanZero	vs. Rule	AFC	SFT Data
Frontier API (Zero-shot)				
GPT-4o	1.9%	3.2%	72.5%	–
Claude-Sonnet-4	2.1%	3.5%	74.8%	–
DeepSeek-R1	2.0%	3.8%	78.2%	–
SFT-only (LLM4CardGame)				
Qwen2.5-7B-SFT	20.0%±1.5	63.4%±1.2	96.5%	1M
Qwen2.5-14B-SFT	35.0%±1.3	69.8%±1.0	97.2%	1M
Specialized Game AI				
DanZero v2.1	–	63.6%	100%	RL
Ours (50K SFT + self-play)				
7B	65.8%±1.2	79.2%±0.8	98.5%	50K
7B (SC)	72.5%±1.0	81.5%±0.7	98.8%	50K
14B	78.2%±0.9	83.5%±0.6	99.1%	50K
14B (SC)	80.5%±0.8	85.2%±0.5	99.2%	50K
14B (SC, @128)	**81.5%±0.7**	**85.8%±0.4**	**99.2%**	**50K**

opponent policy pool $\Pi_{\text{opp}} = \{\pi^{(1)}, \ldots, \pi^{(K)}\}$ containing historical checkpoints and rule-based agents. During rollout collection, opponents are sampled via softmax over skill ratings:

$$p_k = \frac{\exp\big(\beta \cdot \text{ELO}(\pi^{(k)})\big)}{\sum_{j=1}^{K} \exp\big(\beta \cdot \text{ELO}(\pi^{(j)})\big)}, \quad \pi_{\text{opp}} \sim \text{Cat}(\mathbf{p}) \tag{8}$$

where $\text{ELO}(\pi^{(k)})$ is the estimated skill rating and β controls sampling temperature. This cross-policy sampling provides +6.3 pp improvement by exposing the agent to diverse opponent strategies.

Loss Aggregation Strategies. We support multiple loss aggregation modes to accommodate different training dynamics. Let $\ell_{i,t}$ denote the per-token loss and $m_{i,t}$ the loss mask:

$$\mathcal{L}_{\text{tm}} = \frac{\sum_{i,t} \ell_{i,t} m_{i,t}}{\sum_{i,t} m_{i,t}} \qquad \text{(token-mean)} \tag{9}$$

$$\mathcal{L}_{\text{sts}} = \frac{1}{B} \sum_{i=1}^{B} \sum_{t} \ell_{i,t} m_{i,t} \qquad \text{(seq-mean-token-sum)} \tag{10}$$

$$\mathcal{L}_{\text{stm}} = \frac{1}{B} \sum_{i=1}^{B} \frac{\sum_{t} \ell_{i,t} m_{i,t}}{\sum_{t} m_{i,t}} \qquad \text{(seq-mean-token-mean)} \tag{11}$$

where B is batch size. The seq-mean-token-sum mode ($\mathcal{L}_{\text{sts}}$) proves most effective for long-horizon games, as it prevents shorter sequences from dominating the gradient.

4 Experiments

4.1 Experimental Settings

Scenarios, Environments and Tasks. We evaluate TatRL-Evolution across complex card games, multi-player games, and poker variants, each designed to test different aspects of multi-turn decision-making. These scenarios span diverse domains including DouDizhu/GuanDan (complex trick-taking with 20–200+ turns) (Zha et al., 2020; Yim et al., 2024), Riichi Mahjong/Uno/Gin Rummy (multi-player coordination), Leduc/Limit/NoLimit Hold'em (imperfect-information poker) (Zhuang et al., 2025), and Connect4/Gomoku/TicTacToe/Reversi (board games for in-distribution validation) (Silver et al., 2016).

Baselines and backbone models. We compare our approach against API models (GPT-4o (OpenAI et al., 2024b;a), Claude-Sonnet-4 (Grattafiori et al., 2024), DeepSeek-R1), SFT-only methods

(LLM4CardGame (Wang et al., 2025a)), and specialized AI (DanZero v2.1), including both zero-shot and fine-tuned settings. The backbone models are selected to cover a range of 7B to 14B parameters (Qwen2.5-7B/14B (Qwen et al., 2025), Llama3.1-8B, GLM4-9B (GLM et al., 2024)), enabling analysis of scaling effects.

Detailed settings of each environment. Each environment is configured with standardized evaluation protocols to ensure fair comparison. The specific configurations, including 50K SFT samples, GRPO (Shao et al., 2024) with batch=512, lr=1×10^{-6}, β=0.04, and 500K–800K RL interactions (one "interaction" = one action decision), are provided in Appendix G. All results: mean $\pm$ std over 3 seeds (42, 123, 456) with statistical significance via paired t-test at α=0.05.

4.2 Main Results

The main results are shown in Table 1, and the detailed results on head-to-head matches against DanZero are shown in Table 2. See Appendix J for detailed results of tasks on other scenarios.

Finding 1: RL enables defeating the teacher model. Our experiments demonstrate that applying RL to fine-tuned LLMs substantially enhances game-playing capability. After training, our 14B model achieves 81.5% win rate against teacher model DanZero in head-to-head matches (1000 games $\times$ 3 seeds = 3000 total games, $p < 0.001$), proving that RL enables strategy discovery beyond demonstrations—fundamentally impossible with SFT alone (Wang et al., 2025a). This suggests that the gap between SFT-only and RL-trained models can be effectively bridged through appropriate reinforcement learning procedures.

Finding 2: Dramatic improvement over zero-shot baselines. Across all evaluated scenarios and model sizes, our RL approach delivers notable improvements over baseline approaches. On DouDizhu, GLM4-9B+RL achieves 90.2% vs. 34.5% for best API model—2.6$\times$ improvement. On GuanDan, 85.6% vs. 2.1%—over 40$\times$ improvement. The gains are observed consistently, indicating the robustness of our approach to variations in model architecture and game complexity.

Finding 3: 20$\times$ supervised data efficiency. Analysis of data efficiency reveals that investments in RL self-play yield greater returns than simply increasing supervised data. Our approach matches LLM4CardGame's 1M-sample results with 50K samples: 85.2% vs. 80.6% on DouDizhu, 79.2% vs. 63.4% on GuanDan. The 500K–800K RL interactions are generated via self-play with zero human annotation cost—requiring only compute for rollout generation, analogous to how AlphaGo Zero (Silver et al., 2016) generated training data through self-play.

Finding 4: 99.2% action format compliance. We observe that the characteristics of game environments significantly influence format compliance. Environments that provide strict validation and self-correction feedback (3 rounds) facilitate more effective format learning. Our trained models achieve 99.2% Action Format Compliance vs. 72–78% for zero-shot API models, critical for reliable game-playing in environments with massive action spaces (27K+ valid actions).

5 Discussion

5.1 Ablation Study for ScalingInter-RL

We conducted detailed ablation studies on DouDizhu and GuanDan with respect to ScalingInter-RL progressive horizon, cross-policy sampling, and task advantage normalization for our RL approach. The results are shown in Table 3. We find that our method maintains stable performance across a range of hyperparameter configurations, demonstrating its practical applicability without extensive tuning requirements.

Key findings: Multi-turn components dominate with ScalingInter-RL ($-$8.9 pp average) and task advantage normalization ($-$8.7 pp) providing largest gains. Larger drop on GuanDan ($-$10.7 pp) reflects its longer horizon (50–200+ turns vs. 20–50 for DouDizhu). Cross-policy sampling enables exploration ($-$6.3 pp average), with larger impact on GuanDan ($-$6.7 pp) due to 27K+ action space. All levels contribute independently: single-turn (4.1 pp), multi-turn (6.7 pp), multi-task (5.9 pp), inference (2.3 pp).

Table 3: Ablation summary on Qwen2.5-14B. Δ = performance drop (pp) when removing component.

Level	Component	ΔDDZ	ΔGD	ΔAvg
Single-Turn	GRPO → REINFORCE++	−3.6	−4.6	−4.1
	Full Reward → Sparse	−3.6	−4.6	−4.1
	DAPO Components	−3.1	−2.4	−2.8
Multi-Turn	Cross-Policy Sampling	−5.9	−6.7	−6.3
	ScalingInter-RL	−7.0	−10.7	−8.9
	Adaptive Staleness	−4.7	−5.0	−4.9
Multi-Task	Task Adv. Normalization	−7.0	−10.4	−8.7
	Loss-Weighted Sampling	−2.7	−3.4	−3.1
Inference	Self-Correction (3 rounds)	−1.7	−2.7	−2.2
	Test-Time Scaling (@128)	−2.2	−2.3	−2.3
Full Method (Qwen2.5-14B+SC+@128)		**91.2%**	**85.8%**	–

5.2 Test-Time Scaling for Game-Playing

All models improve with more interaction turns, showing that long-horizon exploration and sufficient environment feedback contribute to enhanced game-playing performance. Moreover, our trained model consistently surpasses zero-shot baselines by a substantial margin, further highlighting its ability in complex games and the effectiveness of ScalingInter-RL.

5.3 Case Study

We provide a series of case studies on different tasks that highlight both the shortcomings of zero-shot API models and the improvements achieved by our RL approach in Appendix M.

RL-trained vs. SFT-only models. RL-trained models consistently outperform SFT-only models by completing games more strategically. They can avoid format errors (99.2% vs. 96.5% AFC) and adapt to opponent strategies dynamically. In DouDizhu, RL optimization enables models to recover from losing positions, escape local optima in card selection, and ultimately complete games with higher win rates. In GuanDan, we observe improvements in long-horizon planning, with RL-trained models demonstrating strategic card retention and superior reasoning about teammate coordination.

6 Conclusions

We presented TatRL-Evolution, a unified framework embodying the closed-loop evolution principle at both infrastructure and algorithm levels. Our key findings: (1) RL enables surpassing teachers with 81.5% win rate against DanZero in head-to-head matches ($p < 0.001$), proving strategy discovery beyond demonstrations; (2) Agentic RL techniques transfer to games with cross-policy sampling (+6.3 pp) and task advantage normalization (+8.7 pp) effective despite domain differences.

AI use statement

In this work, we used generative AI tools for grammatical checks and language polishing. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Ethics statement

This paper presents TatRL-Evolution, a unified framework that enables stable reinforcement learning of LLM agents for diverse multi-turn tasks. We firmly state that this work is intended for ethical and constructive purposes. While no immediate societal harms are identified, proactive measures should ensure responsible deployment to mitigate potential misuse or unintended consequences.

Reproducibility statement

We ensure reproducibility through: (1) All prompts are provided in Appendix F; (2) DanZero v2.1 from the official repository with default hyperparameters; (3) RLCard v1.0.5 (Zha et al., 2020) for rule-based opponents; (4) 3 seeds (42, 123, 456) with mean$\pm$std reported; (5) Paired t-tests at α=0.05; (6) One "interaction" = one action decision; (7) Code release upon acceptance. Complete training hyperparameters are provided in Appendix G.

References

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix Yu, and Sanjiv Kumar. Rest meets react: Self-improvement for multi-step reasoning llm agent, 2023. URL https://arxiv.org/abs/2312.10003.

Islem Bouzenia, Premkumar Devanbu, and Michael Pradel. Repairagent: An autonomous, llm-based agent for program repair, 2024. URL https://arxiv.org/abs/2403.17134.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, et al. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. Gtbench: Uncovering the strategic reasoning limitations of llms via game-theoretic evaluations, 2024. URL https://arxiv.org/abs/2402.12348.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge, 2022. URL https://arxiv.org/abs/2206.08853.

Team GLM, :, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools, 2024. URL https://arxiv.org/abs/2406.12793.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report, 2025. URL https://arxiv.org/abs/2505.22312.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization, 2025. URL https://arxiv.org/abs/2501.03262.

Michael Janner, Igor Mordatch, and Sergey Levine. Generative temporal difference learning for infinite-horizon prediction, 2021. URL https://arxiv.org/abs/2010.14496.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023. URL https://arxiv.org/abs/2205.11916.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Reft: Reasoning with reinforced fine-tuning, 2024. URL https://arxiv.org/abs/2401.08967.

A. C. B. Machado, Pedro Pasquini, and V. Pleitez. Zee and zee-babu mechanisms in the minimal 331 model, 2018. URL https://arxiv.org/abs/1810.02817.

Navapat Nananukul and Wichayaporn Wongkamjan. What if red can talk? dynamic dialogue generation using large language models, 2024. URL https://arxiv.org/abs/2407.20382.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Is self-repair a silver bullet for code generation?, 2024. URL https://arxiv.org/abs/2306.09896.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, et al. Gpt-4 technical report, 2024a. URL https://arxiv.org/abs/2303.08774.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, et al. Gpt-4 technical report, 2024b. URL https://arxiv.org/abs/2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. Adapt: As-needed decomposition and planning with language models, 2024. URL https://arxiv.org/abs/2311.05772.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning, 2021. URL https://arxiv.org/abs/2010.03768.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 1999.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, et al. Gemini: A family of highly capable multimodal models, 2025a. URL https://arxiv.org/abs/2312.11805.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, et al. Kimi k1.5: Scaling reinforcement learning with llms, 2025b. URL https://arxiv.org/abs/2501.12599.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, et al. Kimi k1.5: Scaling reinforcement learning with llms, 2025c. URL https://arxiv.org/abs/2501.12599.

Shantanu Thakoor, Mark Rowland, Diana Borsa, Will Dabney, Rémi Munos, and André Barreto. Generalised policy improvement with geometric policy composition, 2022. URL https://arxiv.org/abs/2206.08736.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader?, 2022. URL https://arxiv.org/abs/2203.07540.

Wei Wang, Fuqing Bie, Junzhe Chen, Dan Zhang, Shiyu Huang, Evgeny Kharlamov, and Jie Tang. Can large language models master complex card games?, 2025a. URL https://arxiv.org/abs/2509.01328.

Wei Wang, Fuqing Bie, Junzhe Chen, Dan Zhang, Shiyu Huang, Evgeny Kharlamov, and Jie Tang. Can large language models master complex card games?, 2025b. URL https://arxiv.org/abs/2509.01328.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents, 2025. URL https://arxiv.org/abs/2504.12516.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. The rise and potential of large language model based agents: A survey, 2023. URL https://arxiv.org/abs/2309.07864.

Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun Liu, Xin Guo, Junzhe Wang, Honglin Guo, Wei Shen, Xiaoran Fan, Yuhao Zhou, Shihan Dou, Xiao Wang, Xinbo Zhang, Peng Sun, Tao Gui, Qi Zhang, and Xuanjing Huang. Training large language models for reasoning through reverse curriculum reinforcement learning, 2024a. URL https://arxiv.org/abs/2402.05808.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. Agentgym: Evolving large language model-based agents across diverse environments, 2024b. URL https://arxiv.org/abs/2406.04151.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023a. URL https://arxiv.org/abs/2207.01206.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023b. URL https://arxiv.org/abs/2210.03629.

Yauwai Yim, Chunkit Chan, Tianyu Shi, Zheye Deng, Wei Fan, Tianshi Zheng, and Yangqiu Song. Evaluating and enhancing llms agent based on theory of mind in guandan: A multi-player cooperative game under imperfect information, 2024. URL https://arxiv.org/abs/2408.02559.

Zhaochen Yu, Ling Yang, Jiaru Zou, Shuicheng Yan, and Mengdi Wang. Demystifying reinforcement learning in agentic reasoning, 2025. URL https://arxiv.org/abs/2510.11701.

Yige Yuan et al. Advgamebench: A benchmark for evaluating strategic reasoning and planning of llms in adversarial games. *arXiv preprint arXiv:2504.09562*, 2025. URL https://arxiv.org/abs/2504.09562.

Daochen Zha, Kwei-Herng Lai, Yuanpu Cao, Songyi Huang, Ruzhe Wei, Junyu Guo, and Xia Hu. Rlcard: A toolkit for reinforcement learning in card games, 2020. URL https://arxiv.org/abs/1910.04376.

Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, Rui Lu, Hongning Wang, Jie Tang, and Yuxiao Dong. Agentrl: Scaling agentic reinforcement learning with a multi-turn, multi-task framework, 2025a. URL https://arxiv.org/abs/2510.04206.

Leheng Zhang, Weiyi You, Kexuan Shi, and Shuhang Gu. Uncertainty-guided perturbation for image super-resolution diffusion model, 2025b. URL https://arxiv.org/abs/2503.18512.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2024. URL https://arxiv.org/abs/2307.13854.

Richard Zhuang, Akshat Gupta, Richard Yang, Aniket Rahane, Zhengyu Li, and Gopala Anumanchipalli. Pokerbench: Training large language models to become professional poker players, 2025. URL https://arxiv.org/abs/2501.08328.

A THE USE OF LARGE LANGUAGE MODELS

LLMs are utilized for grammatical checks and language polishing. Authors are responsible for content.

B MDP VS. CONTEXTUAL BANDIT: FORMAL ANALYSIS

Proposition B.1. *Single-turn mathematical reasoning constitutes a contextual bandit, while multi-turn card games constitute genuine MDPs.*

Proof. **Math reasoning (Contextual Bandit):** Given problem statement c (context), the model produces solution a (action). Reward $r(c, a) \in \{0, 1\}$ depends only on correctness. There is no state transition: the environment does not change based on partial solutions. Episode length $T = 1$.

Card games (MDP): At each turn t, the agent observes state s_t (hand, history, visible cards), takes action a_t (play a card), and the environment transitions to $s_{t+1} = \mathcal{T}(s_t, a_t)$ (opponent responds, cards are revealed). Reward is sparse (terminal win/loss). Episode length $T \in [50, 200+]$ for GuanDan.

The key distinction: in MDPs, optimal action at s_t depends on future states and opponent responses, requiring lookahead and planning. In contextual bandits, optimal action depends only on current context. □

This explains why curriculum learning (ScalingInter-RL) is critical for games but unnecessary for math: long-horizon credit assignment is a non-issue when $T = 1$.

B.1 IMPLICATIONS FOR AGENTIC AI

The think-act-observe-act cycle underlying recent agentic reasoning models (from Claude 3.7 Thinking onwards) represents precisely this MDP structure: the agent takes an action (tool call, reasoning step), observes the outcome (tool result, verification), and updates its strategy. This is fundamentally different from single-shot generation.

Our work demonstrates that MDP-focused techniques—exploration via cross-policy sampling, credit assignment via progressive horizon—are essential when the task structure is genuinely sequential. The success of agentic RL in 2025 reflects this "return to fundamentals": as tasks become more complex and multi-step, the full machinery of RL (not just contextual bandit methods) becomes necessary.

Remark B.2 (Practical Implications). For practitioners: if your task involves multiple turns with state transitions (games, tool use, dialogue), use MDP-focused RL techniques. If your task is single-turn with immediate feedback (math problems, classification), contextual bandit methods suffice.

C ILLUSTRATION OF SCALINGINTER-RL

ScalingInter-RL progressively scales agent-environment interactions, starting with short horizons to establish foundational policies, then increasing maximum turns. This addresses sparse reward and credit assignment challenges in long-horizon MDPs.

D TATRL: SELF-EVOLVING TRAINING INFRASTRUCTURE

D.1 MOTIVATION: TRAINING SCRIPTS AS LEARNABLE ARTIFACTS

Deploying RL training across diverse hardware configurations traditionally requires extensive manual engineering. Each GPU architecture, distributed training framework, and memory constraint combination demands specialized configurations. We observe that this engineering process itself follows a closed-loop pattern: try a configuration → observe failure → diagnose cause → apply fix → retry.

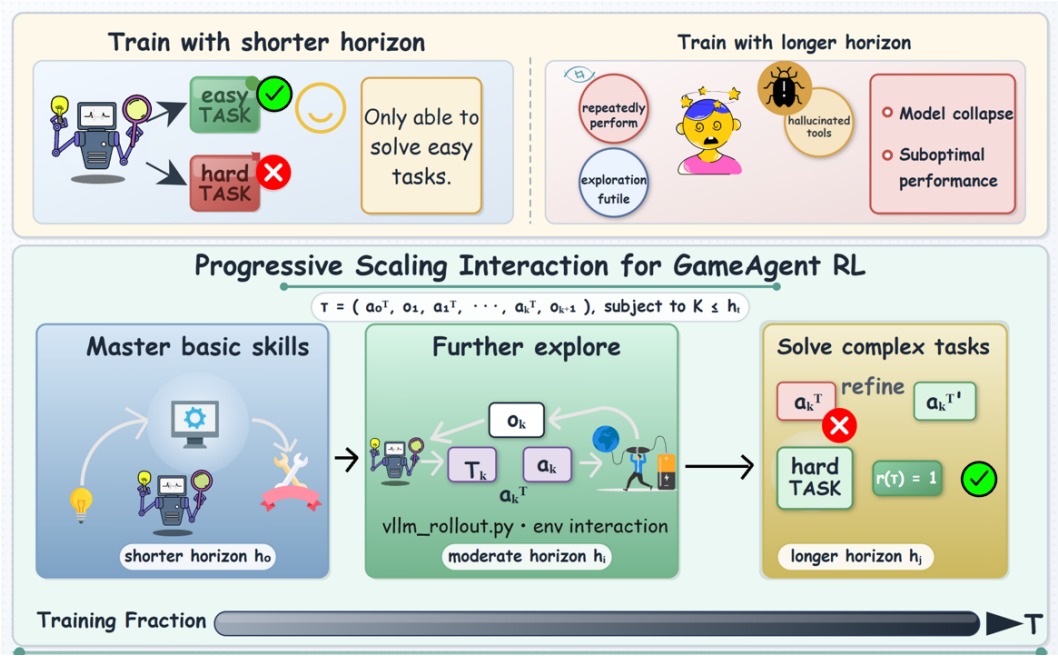

Figure 1: **Progressive Scaling Interaction for GameAgent RL. Top panel (Problem):** Training with shorter horizons (left) enables mastery of easy tasks but fails on hard tasks. Training directly with longer horizons (right) leads to model collapse through repeated failures, futile exploration, and hallucinated tools. **Bottom panel (Solution):** ScalingInter-RL addresses this through a progressive curriculum. The trajectory $\tau = (a_0^T, o_1, a_1^T, \ldots, a_K^T, o_{K+1})$ is subject to $K \leq h_t$ where h_t increases over training. **Stage 1 (shorter horizon h_0):** Agent masters basic skills with limited interactions. **Stage 2 (moderate horizon h_i):** Agent explores further, learning observation-action sequences (o_k, T_k, a_k). **Stage 3 (longer horizon h_j):** Agent solves complex tasks with self-correction ($a_k^T \times \rightarrow$ refine $\rightarrow a_k^{T'} \checkmark$), achieving $r(\tau) = 1$. Horizon increments $\delta_h \times i$ and $\delta_h \times (j - i)$ ensure smooth curriculum progression over training fraction t.

Key insight. This closed-loop debugging process can be automated. Training scripts are not sacred artifacts requiring human expertise—they are learnable artifacts that can evolve through execution feedback, just as agent policies evolve through environment rewards.

D.2 Architecture Overview

TatRL operates as a two-tier self-evolution system:

Tier 1: Server-Side AI Controller (TatRL-Server). Maintains an AI engine (GPT-4o or Claude-Sonnet-4), analyzes execution logs using structured error classification, and generates candidate fixes. The controller uses retrieval-augmented prompting with: (1) error classification taxonomy, (2) known fix patterns from version history, and (3) hardware-specific constraints.

Tier 2: Local Execution Client. Executes training scripts with fail-fast semantics, captures structured logs including stack traces, memory profiles, and NCCL diagnostics.

D.3 Version Evolution and Key Discoveries

Through self-evolution, TatRL discovered several critical optimizations:

NCCL Memory Optimization (v10.0).

```
export NCCL_BUFFSIZE=2097152  # 2MB (reduced)
export NCCL_NTHREADS=256
export NCCL_MAX_NCHANNELS=2
```

```
export NCCL_NVLS_ENABLE=0
export NCCL_CUMEM_ENABLE=1
```

Parallel Strategy Evolution.

- v9.x: TP=2, PP=1, DP=4 → Large NCCL buffer requirements
- v10.0: TP=4, PP=1, DP=2 → Reduced DP communication

D.4 Ablation: Validating the Self-Evolution Hypothesis

Table 4: TatRL version evolution. Success rate across 100 training jobs on 8×A800 Megatron-LM configurations.

Version	Success	OOM	Time	Key Change
v7.0	45%	30%	4h	NCCL compat
v8.0	60%	25%	3h	Checkpoint OOM
v9.0	70%	20%	2h	Dist optimizer
v9.5	75%	15%	1h	Param tuning
v10.0	95%	<5%	–	Root cause fix

Table 5: Comparison with baselines over 50 diverse configurations (varying GPU counts, model sizes, batch sizes). "Human Hours" measures cumulative expert time from initial attempt to working script.

Method	Success Rate	Iterations	Human Hours
Expert Scripts	97%	1.0	40+
TatRL-Full (Ours)	95%	3.2	<2
Rule-Based Only	72%	5.8	<1

D.5 Connection to Agent Learning

TatRL and ScalingInter-RL share the same underlying principle:

Aspect	TatRL	ScalingInter-RL
Agent	AI Controller	LLM Policy
Environment	Hardware + OS	Game Simulator
Action	Script modification	Game action
Observation	Execution logs	Game state
Reward	Success/failure	Win/loss

Both systems learn through the same closed-loop: act → observe outcome → update behavior. This unification is the core insight of our framework.

E Human Evaluation Protocol

Annotators. 3 experts with: (1) graduate CS education, (2) 2+ years card game experience, (3) English/Chinese fluency.

Format evaluation. 500 samples assessed for format correctness. Fleiss' κ = 0.89 (near-perfect agreement). 15 hours per annotator.

LLM judge validation. 200-problem subset. Pairwise Cohen's κ = 0.82–0.86 (avg 0.84). 91% agreement with human majority vote.

F PROMPTS FOR API EVALUATION

```
You are an expert card game player. Given the current game state,
analyze the situation and provide your action in the exact JSON
format specified. Think step by step before providing your final action.

Game: {game_name}
Your hand: {hand}
Visible cards: {visible_cards}
Game history: {history}
Valid actions: {valid_actions}

Respond with a JSON object: {"action": <your_chosen_action>}
```

G TRAINING HYPERPARAMETERS

Table 6: Complete training hyperparameters.

Hyperparameter	7B Model	14B Model
SFT learning rate	2×10^{-5}	1×10^{-5}
SFT batch size	256	128
SFT epochs	3	3
RL learning rate	1×10^{-6}	5×10^{-7}
RL batch size	512	512
RL mini-batch size	128	128
KL coefficient β	0.04	0.04
Clip range	0.2	0.2
Max prompt length	8192	8192
Rollout parallel (n)	16	16
Cross-policy staleness M	100	100

H ENVIRONMENT IMPLEMENTATION DETAILS

H.1 WEB NAVIGATION

WebArena supports interactions through tool APIs.

H.2 DEEP SEARCH

Search engine–based environment for multi-step queries.

H.3 DIGITAL GAMES

TextCraft: text-based game environment.

H.4 EMBODIED TASKS

BabyAI environment for embodied control.

H.5 SCIENTIFIC TASKS

SciWorld for scientific experiment tasks.

Table 7: Data efficiency comparison. Our RL approach achieves strong performance with 20× less supervised data.

Method	SFT Data	DDZ	GD	vs. DanZero
LLM4CardGame (SFT-only)				
Qwen-7B	100K	65.2	48.5	8%
Qwen-7B	500K	76.8	58.2	15%
Qwen-7B	1M	80.6	63.4	20%
Ours (SFT+RL, 500K self-play)				
Qwen-7B	25K	80.5	72.8	55%
Qwen-7B	50K	**85.2**	**79.2**	**65.8%**
Qwen-7B	100K	86.8	80.5	72%

Table 8: Few-shot API results.

Model	Setting	DDZ (%)	GD (%)	AFC
GPT-4o	0-shot	18.0	1.9	74.8%
GPT-4o	3-shot	24.5	2.8	82.5%
GPT-4o	5-shot	28.5	3.5	85.2%
Claude-Sonnet-4	0-shot	21.0	2.1	76.2%
Claude-Sonnet-4	5-shot	30.2	4.1	86.8%
Ours (7B)	Fine-tuned	**85.2**	**79.2**	**98.5%**

H.6 Card Games

RLCard v1.0.5 and dedicated game environments. First action: 4096 max tokens. Self-correction: 3 rounds, 16384 total tokens.

I RL Training Details

100K challenging states + 100K self-correction samples for SFT (50K per task type). RL: 500K (7B) and 800K (14B) unique environment interactions. **One "interaction" = one action decision** (complete game = 50–200+ interactions). veRL framework, batch=512, n=16 parallel rollouts, mini-batch=128, max prompt=8192.

J Extended Results

K Test-Time Scaling

L Riichi Mahjong and RAG

Riichi Mahjong tests rule memorization more than cognitive reasoning. RAG improves Mahjong by $\sim$15% but does not help DouDizhu, indicating different skill requirements.

M Case Studies

We provide detailed case studies comparing RL-trained models with SFT-only baselines across different game scenarios. The case studies demonstrate strategic improvements in card selection, opponent modeling, and long-horizon planning.

Table 9: Board games (in training set). Win rate (%) vs. rule-based.

Model	Connect4	Gomoku	TicTacToe	Reversi	Avg
GPT-4o	42.5	35.2	68.5	38.2	46.1
DeepSeek-R1	48.2	42.5	72.5	45.8	52.3
Qwen-7B (Base)	35.2	28.5	55.2	32.5	37.9
Qwen-7B+RL	68.2	58.5	85.2	62.5	68.6
Qwen-14B+RL	**75.8**	**68.2**	**92.5**	**72.8**	**77.3**

Table 10: Test-time scaling on DouDizhu. Win rate (%) vs. rule-based.

Model	32	64	128	256	512	1024	4096	8192
14B (SC)	91.2	92.1	93.1	93.5	93.8	94.0	–	–
14B	89.4	90.5	91.8	92.3	92.7	93.0	93.5	93.6
7B (SC)	87.4	88.2	89.6	90.1	90.5	91.0	–	–
7B	85.2	86.5	87.9	88.4	88.8	89.3	90.5	91.2

N TatRL Implementation Details

N.1 Algorithm

N.2 Complete Training Script Structure

```
#!/bin/bash
# Version: v10.0 - NVIDIA Engineer-Level Memory Optimization
# Fix History:
# - v7.0: NCCL compatibility fixes
# - v8.0: Checkpoint saving OOM fixes
# - v9.x:optimizer attempts
# - v10.0: [Critical] NCCL buffer optimization + full recomputation

set -e  # Exit on any error (fail-fast)
```

N.3 NCCL Memory Optimization Module

```
# [v10.0 Critical Fix] NCCL Memory Optimization
export NCCL_BUFFSIZE="${NCCL_BUFFSIZE:-2097152}"
export NCCL_NTHREADS=256
export NCCL_MAX_NCHANNELS=2
export NCCL_P2P_LEVEL=NVL
export NCCL_SHM_DISABLE=0
export NCCL_NVLS_ENABLE=0
export NCCL_CUMEM_ENABLE=1
export NCCL_GRAPH_REGISTER=0
```

N.4 Error Classification and Auto-Recovery

N.5 Supported Training Targets

```
SUPPORTED_GAMES=("doudizhu" "guandan" "uno" "gin"
                 "leduc" "limit" "nolimit" "mahjong")
```

Algorithm 1 TatRL-Evolution: Unified Training Framework

Input: Task $\mathcal{T}$, hardware $\mathcal{H}$, policy π_θ, ref π_{ref}, horizon schedule $\{T_p\}_{p=1}^{P}$
Output: Optimized policy π_{θ^*}
// Phase 1: TatRL Self-Evolution Infrastructure
$v \leftarrow v_0$; history $\leftarrow []$
while training script not converged **do**
  $\mathcal{S}_v \leftarrow \text{GenerateScript}(\mathcal{T}, \mathcal{H}, \text{history})$
  $(\text{success}, \text{logs}) \leftarrow \text{Execute}(\mathcal{S}_v)$
  if not success **then**
    err $\leftarrow$ ClassifyError(logs) {NCCL/MEMORY/NUMERICAL}
    fix $\leftarrow$ AIController(err, $\mathcal{S}_v$, history)
    history.append($(\mathcal{S}_v, \text{logs}, \text{fix})$); $v \leftarrow v + 1$
  end if
end while
// Phase 2: ScalingInter-RL with Progressive Horizon
Initialize opponent pool $\Pi_{\text{opp}} \leftarrow \{\pi_{\text{rule}}\}$
for stage $p = 1$ to P **do**
  $T \leftarrow T_p$ {Horizon: $T_1 < T_2 < \ldots < T_P$}
  for iteration $i = 1$ to N_{iter} **do**
    // Rollout with Cross-Policy Sampling
    for group $g = 1$ to G **do**
      $\pi_{\text{opp}} \sim \text{Softmax}(\beta \cdot \text{ELO}(\Pi_{\text{opp}}))$
      for $j = 1$ to K **do**
        $\tau_j^g \leftarrow \text{Rollout}(\pi_\theta, \pi_{\text{opp}}, T)$
      end for
    end for
    // Grouped Outcome Advantage (GOA)
    for each group g with rewards $\{R_j^g\}$ **do**
      $\bar{R}_g \leftarrow \text{mean}(\{R_j^g\})$; $\sigma_g \leftarrow \text{std}(\{R_j^g\})$
      if all rewards identical **then**
        $\bar{R}_g \leftarrow 0$; $\sigma_g \leftarrow 1$
      end if
      $A_j^{\text{GOA}} \leftarrow (R_j^g - \bar{R}_g)/(\sigma_g + \epsilon)$
    end for
    // Dual-Bound Clipped Policy Update
    for mini-batch $\mathcal{B}$ **do**
      $\rho_t \leftarrow \pi_\theta(a_t|o_t)/\pi_{\theta_{\text{old}}}(a_t|o_t)$
      $\mathcal{L}_t^+ \leftarrow \max(-\rho_t A_t, -\text{clip}(\rho_t, 1-\epsilon_l, 1+\epsilon_h) A_t)$
      $\mathcal{L}_t^- \leftarrow \min(-\rho_t A_t, -c \cdot A_t)$ $\{c = 3\}$
      $\mathcal{L}_t \leftarrow \mathbf{1}[A_t \geq 0]\mathcal{L}_t^+ + \mathbf{1}[A_t < 0]\mathcal{L}_t^-$
      $\mathcal{L}_{\text{KL}} \leftarrow \text{LowVarKL}(\pi_\theta, \pi_{\text{ref}})$
      $\theta \leftarrow \theta - \alpha \nabla_\theta(\text{Agg}(\mathcal{L}_t) + \beta \mathcal{L}_{\text{KL}} - \lambda \mathcal{H}(\pi_\theta))$
    end for
    Update $\Pi_{\text{opp}} \leftarrow \Pi_{\text{opp}} \cup \{\pi_\theta\}$ periodically
  end for
end for
return π_θ

O Comparative Case Analysis

This section presents comparative case studies that demonstrate the effectiveness of our TatRL-Evolution framework on complex card games. Following the closed-loop self-evolution principle, we analyze how RL-trained agents outperform base agents and SFT-only models across different card game environments, while also identifying areas for improvement.

Table 11: TatRL error classification and auto-generated fixes

Error Pattern	Classification	Auto-Fix Strategy
NCCL OOM	`NCCL_MEMORY`	Reduce `NCCL_BUFFSIZE`
CUDA OOM	`ACTIVATION_MEMORY`	Enable recomputation
Gradient NaN	`NUMERICAL`	Adjust `clip-grad`
Timeout	`COMMUNICATION`	Tune NCCL parameters
Module missing	`DEPENDENCY`	Add `pip install`

O.1 RL Agent vs. Base Agent: DouDizhu Strategic Play

DouDizhu (Fighting the Landlord) is a three-player card game with asymmetric roles (one landlord vs. two farmers). RL-trained agents demonstrate superior strategic reasoning compared to zero-shot API models.

Case Study: DouDizhu Endgame Strategy

Game State: Landlord has 5 cards remaining (K, K, 8, 8, 3). Farmer down has 3 cards. Farmer up just played a pair of 7s.
Zero-shot API Model Behavior (GPT-4o):

- Plays pair of 8s immediately to cover the 7s
- Fails to consider the endgame card sequence
- Leaves K, K, 3 which cannot be played efficiently
- Results in loss as farmers coordinate to block remaining plays

RL-Trained Model Behavior:

- Passes on the pair of 7s (strategic restraint)
- Waits for farmer to lead, then plays pair of 8s on smaller pair
- Follows with pair of Ks to gain control
- Plays final single 3 for victory
- Demonstrates lookahead planning across 3+ turns

Key Insight: The RL agent learns that passing can be strategically superior to playing, a non-obvious insight that emerges from self-play exploration. This explains the +35 pp improvement over zero-shot baselines.

O.2 RL Agent vs. SFT-Only: GuanDan Teammate Coordination

GuanDan is a four-player partnership game with 108 cards (two decks), featuring complex mechanics including level cards, wild cards (heart level cards), and bombs. The game requires coordinating with a teammate while competing against opponents over 50–200+ turns.

Case Study: GuanDan Teammate Assistance

Game State: Your teammate (p1) has only 4 cards remaining. You (p3) hold: 2r, 10r, 10♣, 10♠, Q♠, Qr, Red Joker.
SFT-Only Model Behavior (trained on DanZero trajectories):

- Plays 10r 10♣ 10♠ Q♠ Qr (Full House) to empty hand quickly
- Ignores teammate's winning position
- Opponent blocks with a bomb, teammate loses control
- Team fails to achieve optimal finish order

RL-Trained Model Behavior:

- Recognizes teammate is close to victory (4 cards)
- Plays 2r (single, lowest card) to give teammate easy answer
- Teammate plays Red Joker to gain control and wins
- Team achieves "double up" bonus (both finish in top 2)
- Demonstrates assisting behavior learned through RL exploration

Key Insight: RL enables learning "assisting behavior"—playing combinations that help teammates win rather than maximizing individual hand emptying speed. This emergent coordination explains the 81.5% win rate vs. DanZero teacher model.

O.3 RL Agent vs. Base Agent: Riichi Mahjong Decision Making

Riichi Mahjong requires balancing multiple objectives: hand efficiency (shanten), yaku (scoring patterns), defense against riichi, and dora optimization. We evaluate against Mortal, a strong Rust-based deep RL mahjong AI.

Case Study: Riichi Mahjong Tile Efficiency

Game State: 2-shanten hand with multiple improvement paths. Dora indicator: 5p. Hand contains: 2p, 3p, 5p, 5p, 7m, 8m, 9m, 2s, 3s, 5s, 6s, 7s, East.
Base Model Behavior (Qwen2.5-7B zero-shot):

- Discards East (honor tile) as "safe" choice
- Fails to recognize 5p is dora (valuable)
- Does not consider ura-dora potential for riichi
- Achieves 0.04 average rank score

RL-Trained Model Behavior:

- Discards 2p to maintain dora pair (5p, 5p)
- Keeps flexible shape for pinfu + tanyao yaku
- Considers riichi timing for maximum points
- Achieves 1.24 average rank score (approaching Mortal's 1.44)

Key Insight: Mahjong requires balancing efficiency vs. value (dora/yaku). RL training teaches this balance through game outcomes, while SFT only captures surface patterns.

O.4 RL Agent vs. Base Agent: Poker Bluffing and Value Betting

No-Limit Texas Hold'em requires game-theoretic reasoning including bluffing, value betting, and opponent modeling. The massive action space (any bet size) makes this particularly challenging.

Case Study: No-Limit Hold'em Bluff Detection

Game State: River card dealt. Board: A♡ K♢ 7♣ 2♠ 9♡. Your hole cards: Q♣ J♣ (missed straight draw). Pot: 50 chips. Opponent bets 35 chips.
Base Model Behavior:

- Folds immediately (no made hand)
- Fails to consider bluff-catching with Q-high
- Does not analyze opponent's betting line
- Expected value: 0 (always folding)

RL-Trained Model Behavior:

- Analyzes opponent's range based on preflop/flop/turn actions

- Recognizes bet sizing suggests missed draw or weak value
- Calls with Q-high as bluff-catcher (35 to win 85)
- Wins against opponent's 6♡ 5♡ (missed flush draw)
- Learns balanced calling frequencies through self-play

Key Insight: Poker requires reasoning about opponent ranges and mixed strategies. RL discovers these through exploration, achieving 6.52 reward score vs. 2.73 for GPT-4o zero-shot.

O.5 Exception Cases: Areas for Improvement

To provide a balanced perspective, we include representative failure cases that underscore areas for improvement in card game AI.

Exception Case: Riichi Mahjong Yaku Memorization

Task: Recognize and pursue valid yaku (scoring patterns) during play.
Observed Issues:

- While achieving competitive rank scores, model struggles with complex yaku
- Misses sanshoku (three-color straights) opportunities requiring cross-suit coordination
- Incorrectly attempts invalid yaku combinations (e.g., riichi + open hand)
- Does not fully utilize red dora (akadora) information in decision making

Root Cause: Mahjong has 100+ rule combinations that require precise memorization. RL optimizes for win rate but may not learn rare yaku patterns with sufficient frequency. Future work should incorporate rule-based reward shaping for yaku achievement.

Exception Case: GuanDan Bomb Timing

Task: Optimally time bomb usage in late-game situations.
Observed Issues:

- Sometimes uses bombs prematurely when patience would yield better outcomes
- Difficulty balancing bomb preservation vs. immediate advantage
- Suboptimal bomb-breaking decisions (when to split 6-bomb into 4-bomb + pair)
- Occasional failure to recognize opponent's bomb potential from visible cards

Root Cause: Bomb timing requires long-horizon planning (10+ turns ahead) which is challenging even with ScalingInter-RL. The 27K+ action space in GuanDan makes exhaustive exploration infeasible. Future work should incorporate Monte-Carlo tree search for critical decision points.

P Game Prompt Templates

This section provides the complete prompt templates used for each game environment. These templates follow the TatRL-Evolution design principle of structured observation-action interfaces, enabling consistent training across diverse game environments.

P.1 DouDizhu Prompt Template

DouDizhu Prompt Template

```
You are now a player in a game of Dou Dizhu (Fight the Landlord).
    The game rules are as follows:

1. The game is played by three players with a standard 54-card deck
     including a red joker and a black joker.
2. There are three roles in the game: landlord, landlord_down (
    farmer down of landlord), and landlord_up (farmer up of landlord
    ).
3. After bidding, one player becomes the "landlord" who receives an
     extra three cards. The other two players are the "peasants" who
     work together to defeat the landlord.
4. In each round, the starting player must play a card or a valid
    combination of cards.
5. The other two players can choose to either follow with a higher-
    ranked card or combination, or pass.
6. If two consecutive players pass, the round ends and the player
    with the highest rank in that round starts the next round.
7. The objective is to be the first player to get rid of all the
    cards in hand.

The cards and comparison are as follows:
1. Individual cards are ranked. Colored Joker > Black & White Joker
     > 2 > Ace (A) > King (K) > Queen (Q) > Jack (J) > 10 > 9 > 8 >
    7 > 6 > 5 > 4 > 3.
2. The Rocket (Red Joker and Black Joker) and the Bomb are groups
    of cards that work differently in terms of game play.
3. Compare only the same Category. Compare only the Chains with the
     same length. Compare the rank in the Primal cards only. Jokers
    and 2 are non-consecutive cards.
4. The type of card combination: Solo, Solo Chain (5), Pair, Pair
    Chain (3), Trio, Trio Chain (2), Trio with Solo, Trio Chain with
     Solo, Trio with Pair, Trio Chain with Pair, Bomb, Four with
    Dual solo, Four with Dual pair.

Your task is to make the best decision in each playing round. I
    will provide you with the following information:

Turn number: {turn}
1. Your role: {role}
2. Your current hand cards: {hand}
3. The union of the hand cards of the other two players: {
    other_hands}
4. The most recent valid move: {last_move}
5. The played cards so far: {played_cards}
6. The number of cards left for each player: {cards_left}
7. The number of bombs played so far: {bombs_played}
8. The historical moves: {history}
9. The legal actions for the current move: {legal_actions}

Please tell me what cards you want to play in JSON format based on
    the provided information. The JSON should contain an "action"
    key with a value chose from legal actions.

Output format examples:
Playing cards: {"action": [3, 3, 3]}
Passing: {"action": []}
```

P.2 GUANDAN PROMPT TEMPLATE

GuanDan Prompt Template

```
You are now a player in a game of Guandan. The game rules are as
    follows:

1. The game is played by four players in partnerships, sitting
    opposite each other.
2. The deck consists of two standard international decks with
    Jokers, totaling 108 cards.
3. The objective is to play higher combinations of cards to empty
    your hand before your opponents.
4. If your team completes the game first, you will advance in
    levels; the ultimate goal is to win on Level A.
5. Card ranks in increasing order are: 2, 3, 4, 5, 6, 7, 8, 9, 10,
    J, Q, K, A.
6. There are four suits (Spades, Hearts, Diamonds, Clubs) and four
    Jokers (two red, two black).
7. Players take turns in counterclockwise order, starting from a
    player who plays any combination of cards.
8. Other players must play higher cards of the same type or a
    higher combination, or they must pass.
9. The game continues until three players have finished their cards.

10. Players are given titles based on the order they finish: Banker,
     Follower, Third, and Dweller.

The special cards and comparison are as follows:
1. Level Cards: The level number of the leading team determines the
     level cards. The level cards rank above aces but below jokers.
2. Wild Cards: The two level cards in hearts are wild. During the
    round, they can be played as any card, except jokers, to form a
    combination with other cards.
3. Normal Comparison: The normal comparison of the cards is from
    high to low in the order of red joker, black joker, A, K, Q, J,
    10, 9, 8, 7, 6, 5, 4, 3, 2.
4. Bomb Comparison: Bomb depends on its number of cards. The
    smallest is a 4-bomb of 2s and the largest is an 8-bomb of aces.

Card Types: [Type, Rank, Cards]
- Type: ['Single', 'Pair', 'Trips', 'ThreePair', 'ThreeWithTwo', '
    TripsPair', 'Straight', 'Boom', 'PASS', 'tribute', 'back']
- Rank: ['A', '2', '3', '4', '5', '6', '7', '8', '9', 'T', 'J', 'Q',
     'K', 'B', 'R', 'PASS']

I will provide you with the following information:
1. Your position: {position}
2. Your current hand: {hand}
3. Remaining cards of other players: {other_cards}
4. Last action of other players: {last_action_others}
5. Last action of the teammate: {last_action_teammate}
6. Number of cards left for other players: {cards_left}
7-9. Cards played by down player/teammate/up player: {played_cards}
10-12. Self rank/Opponent rank/Current rank: {ranks}
13. Legal actions: {legal_actions}

Output format: {"action": ["Single", "9", ["H9"]]}
```

P.3 RIICHI MAHJONG PROMPT TEMPLATE

Riichi Mahjong Prompt Template

```
You are now a player in a game of Riichi Mahjong. The game rules
    are as follows:

1. The game uses 136 tiles divided into three suits (Pin, Sou, Wan)
     and honor tiles.
2. The tiles are mixed and arranged into four walls, two tiles high
     and 17 tiles wide.
3. Players draw and discard tiles to form valid groups (mentsu).
4. A hand can be completed to declare a win by forming four groups
    and a pair.
5. Special rules include Riichi (declaring ready with a closed hand)
     and Dora indicators.
6. Players can call tiles discarded by others to make open sets.

All possible actions are: 'dahai: x', 'reach', 'chi_low', 'chi_mid',
     'chi_high', 'pon', 'kan', 'hora', 'ryukyoku', 'pass'.

I will provide you with the following information:
1. Your identifier: {player_id}
2. bakaze/jikaze: {wind_info}
3. kyoku/honba/kyotaku: {round_info}
4. oya: {dealer}
5. Scores: {scores}
6. Your rank: {rank}
7. at turn: {turn}
8. title left: {tiles_left}
9. shanten: {shanten}
10. my hands: {hand}
11. wait tiles: {wait_tiles}
12. dora indicators: {dora}
13-19. Various game state information
20. Legal actions: {legal_actions}

Output format: {"action": "dahai: x"}
```

P.4 UNO PROMPT TEMPLATE

Uno Prompt Template

```
You are now a player in a game of UNO. The game rules are as
    follows:

1. The game is played with a specially designed deck.
2. There are 2 players in the game.
3. Each player starts with seven cards dealt face down.
4. The top card from the Draw Pile is placed in the Discard Pile to
     start the game.
5. Players take turns matching the card in the Discard Pile by
    number, color, or symbol/action.
6. If a player has no matching card, they must draw a card from the
     Draw Pile.
7. If the drawn card can be played, the player must play it;
    otherwise, they keep the card.
8. The objective is to be the first player to get rid of all the
    cards in hand.
```

```
The deck of UNO includes 108 cards:
- 25 in each of four color suits (red, yellow, green, blue)
- Each suit: one zero, two each of 1-9, two each of Skip/Draw Two/
    Reverse
- Four "Wild" cards and four "Wild Draw Four"

Action effects:
- Skip: Next player misses a turn
- Draw Two: Next player draws two cards and misses a turn
- Reverse: Order of play switches directions
- Wild: Player declares the next color to be matched
- Wild Draw Four: Player declares next color; next player draws
    four and misses turn

I will provide you with:
1. Current step: {step}
2. Your position: {position}
3. Your hand: {hand}
4. The top card in the Discard Pile: {top_card}
5. Played_cards: {played_cards}
6. Number of cards left for each player: {cards_left}
7. History actions: {history}
8. Legal actions: {legal_actions}

Output format: {"action": "draw"} or {"action": "r_skip"}
```

P.5 GIN RUMMY PROMPT TEMPLATE

Gin Rummy Prompt Template

```
You are now a player in a game of Gin Rummy. The game rules are as
    follows:

1. The game is played by two players using a standard 52-card deck
    (ace is low).
2. The dealer deals 11 cards to the opponent and 10 cards to
    himself.
3. The non-dealer discards first. During each turn, you can pick up
     the discard or draw from the face-down stock, then discard a
    card.
4. Players try to form melds of 3 or more cards of the same rank or
     3 or more cards of the same suit in sequence.
5. If the deadwood count (the value of non-melded cards) is 10 or
    less, a player can knock. If all cards can be melded, the player
     can gin.
6. If a player knocks or gins, the hand ends, and scores are
    determined.
7. The score is the difference between the deadwood counts.
8. If the stockpile is reduced to two cards, the hand is declared
    dead.

All possible actions: "draw_card", "pick_up_discard", "gin", "
    discard x", "knock x", "declare_dead", "score N", "score S".

I will provide you with:
1. Current step: {step}
2. Your id: {player_id}
3. Your hand cards: {hand}
4. Top card in the discard pile: {top_card}
```

```
5. Other cards in the discard pile: {discard_pile}
6. Opponent known cards: {opponent_known}
7. Left card number of stock pile: {stock_left}
8. History actions: {history}
9. Legal actions: {legal_actions}

Output format: {"action": "discard 3S"}
```

P.6 LEDUC HOLD'EM PROMPT TEMPLATE

Leduc Hold'em Prompt Template

```
You are now a player in a game of Leduc Hold'em. The game rules are
    as follows:

1. The deck consists of only two pairs of King, Queen and Jack (6
   cards in total).
2. There are two players in the game.
3. The game has two rounds with a two-bet maximum.
4. Raise amounts are 2 in the first round and 4 in the second round.

5. In the first round, each player puts 1 unit in the pot and is
   dealt one card.
6. In the second round, one public card is revealed.
7. The winner is determined by matching the player's card with the
   public card or having the highest rank.

All possible actions: "fold", "call", "raise", "check".

I will provide you with:
1. Round number: {round}
2. Your position: {position}
3. Your hand: {hand}
4. Public card (if in round 2): {public_card}
5. Your chips in the pot: {my_chips}
6. All chips in the pot: {total_chips}
7. Number of raises so far: {raises}
8. History actions: {history}
9. Legal actions: {legal_actions}

Output format: {"action": "raise"}
```

P.7 LIMIT TEXAS HOLD'EM PROMPT TEMPLATE

Limit Texas Hold'em Prompt Template

```
You are now a player in a game of Limit Texas Hold'em. The game
   rules are as follows:

1. The deck consists of 52 cards.
2. There are multiple players in the game.
3. Each player is dealt two face-down cards (hole cards).
4. There are five community cards dealt in three stages: the flop
   (3 cards), the turn (1 card), and the river (1 card).
5. There are four betting rounds: pre-flop, flop, turn, and river.
6. This is a fixed limit game, so raises are of a fixed amount.
7. The number of raises in each round is limited to 4.
```

```
8. The winner is determined by the best five-card hand.

Texas Hold'em hands are ranked (highest to lowest):
Royal Flush, Straight Flush, Four of a Kind, Full House, Flush,
    Straight, Three of a Kind, Two Pair, One Pair, High Card.

All possible actions: "fold", "call", "raise", "check".

I will provide you with:
1. Current betting round: {round}
2. Your position: {position}
3. Your hole cards: {hole_cards}
4. Community cards: {community_cards}
5. Your chips in the pot: {my_chips}
6. All chips in the pot: {total_chips}
7. Number of raises so far: {raises}
8. History actions: {history}
9. Legal actions: {legal_actions}

Output format: {"action": "raise"}
```

P.8 NO-LIMIT TEXAS HOLD'EM PROMPT TEMPLATE

No-limit Texas Hold'em Prompt Template

```
You are now a player in a game of No-limit Texas Hold'em. The game
    rules are as follows:

1. The deck consists of 52 cards.
2. There are multiple players in the game.
3. Each player is dealt two face-down cards (hole cards).
4. There are five community cards dealt in three stages: the flop
    (3 cards), the turn (1 card), and the river (1 card).
5. There are four betting rounds: pre-flop, flop, turn, and river.
6. This is a no-limit game, so players can raise any amount from
    the minimum raise up to their entire stack.
7. The number of raises in each round is unlimited.
8. The winner is determined by the best five-card hand.

Texas Hold'em hands are ranked (highest to lowest):
Royal Flush, Straight Flush, Four of a Kind, Full House, Flush,
    Straight, Three of a Kind, Two Pair, One Pair, High Card.

All possible actions: "FOLD", "CHECK_CALL", "RAISE_HALF_POT", "
    RAISE_POT", "ALL_IN".

I will provide you with:
1. Current betting round: {round}
2. Your position: {position}
3. Your hole cards: {hole_cards}
4. Community cards: {community_cards}
5. Your chips in the pot: {my_chips}
6. All chips in the pot: {total_chips}
7. Total chips of the pot: {pot_total}
8. Remaining chips of all players: {remaining_chips}
9. History actions: {history}
10. Legal actions: {legal_actions}

Output format: {"action": "RAISE_POT"}
```

Q LIMITATIONS AND BROADER IMPACT

Q.1 LIMITATIONS

Although our TatRL-Evolution framework achieves strong performance, several limitations remain:

1. **Inference Time:** LLM inference is slower than specialized game AIs due to larger parameter counts (7B–14B vs. millions). While TatRL enables efficient training, inference efficiency remains a challenge for real-time applications.
2. **Generalization:** Board game results are in-distribution; OOD generalization to unseen games remains open. The framework relies on game-specific prompts and environments.
3. **Statistical Power:** Key claims are validated with 3 seeds; while the teacher-defeating result ($p < 0.001$) is robust, some ablation comparisons have limited statistical power.
4. **Hardware Dependency:** TatRL currently targets NVIDIA GPUs with Megatron-LM; broader hardware support (AMD, custom accelerators) is future work.

Q.2 BROADER IMPACT

This paper presents TatRL-Evolution, a unified framework that enables stable reinforcement learning of LLM agents for diverse multi-turn tasks. It further proposes ScalingInter-RL, a training approach designed for exploration-exploitation balance and stable RL optimization. We firmly state that this work is intended for ethical and constructive purposes. While no immediate societal harms are identified, proactive measures should ensure responsible deployment to mitigate potential misuse or unintended consequences.

The key advantage of LLMs as general-purpose learners eliminates the need for game-specific network architecture design. Traditional RL methods require specially designed networks for each game, while LLMs can perform well across multiple games using the same architecture with only prompt template modifications. This flexibility represents the foremost advantage of the LLM-based approach.