GAE Falls Short in Imperfect-Information Self-Play Reinforcement Learning

Zhiyuan Fan
MIT
fanzy@mit.edu

Gabriele Farina
MIT
gfarina@mit.edu

Abstract

Competitive multi-agent reinforcement learning in imperfect-information games requires agents to act under partial observability and against adversarial opponents, necessitating stochastic policies. While self-play reinforcement learning with Proximal Policy Optimization (PPO) has achieved strong empirical success, its standard advantage estimator, generalized advantage estimation, suffers from additional variance due to the sampling of stochastic future actions. This variance is amplified in equilibrium self-play because of the stochastic nature of the equilibrium policy and persists even when the critic is exact. We address this bottleneck by introducing ***Q*-boosting**, a variance-reduced advantage estimator based on a centralized action-value critic, and propose **Variance-Reduced Policy Optimization (VRPO)**, incorporating this new estimator. The algorithm replaces sampled multi-step backups with a multi-step Expected SARSA(λ) trace, computing policy expectations at each step to average out action-sampling noise, while retaining PPO's clipped objective and on-policy actor updates. Empirically, VRPO achieves strong performance in various games: In mid-sized games where exploitability is computable, VRPO lowers the exact exploitability by more than 15% compared to tuned baselines. In Dou Dizhu, it sets a new state of the art over PerfectDou. In Heads-Up No-Limit Texas Hold'em, it posts a positive win rate against Slumbot without any blueprint raise sizes, search, or subgame solving at test time.

1 Introduction

Multi-agent reinforcement learning (MARL) is a central challenge in reinforcement learning, where multiple agents interact competitively under imperfect information. During gameplay, players must take actions based only on the private information they observe. The outcome of the game is determined by both the hidden states and the actions of the opponents.

Learning policies with equilibrium-like robustness is a common goal in such games. This can be formulated as learning a policy that maximizes reward against the most adversarial opponent. However, such equilibrium concepts are generally intractable, and the best response is difficult to evaluate in large-scale games. Therefore, performance is often evaluated using heads-up play against a fixed opponent that is not encountered during training.

This problem is challenging for two main reasons. First, the agent needs to act under partial observability, which requires implicit reasoning about the hidden states of the opponents. Second, we want a policy that is robust to adaptive opponents, meaning that the training process needs to account for adversarial gameplay, even when such strategies may be far from the final policy. These challenges have motivated two broad paradigms: regret minimization and self-play RL.

The first line of work is based on **Counterfactual Regret Minimization (CFR)** [1]. The algorithm provably converges to Nash equilibria in two-player zero-sum games. This approach is considered a leading method for solving extensive-form games such as No-Limit Texas Hold'em [2]. Follow-up

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

methods such as Deep CFR [3] and ReBeL [4] build on neural networks to approximate regrets and values, enabling the algorithms to scale to larger game trees and achieve strong performance in poker.

However, CFR-style methods can still struggle to scale in games such as Stratego or Dou Dizhu due to their large private-information spaces, long gameplay trajectories, and large action sets. These features make it difficult to expand the extensive-form game tree from any position, and thus regret minimization is considered both computationally and statistically inefficient in such settings.

The other, complementary line of work is based on **Self-Play Reinforcement Learning (Self-play RL)**, where policies are directly optimized using policy gradient methods. These algorithms have shown effectiveness in environments with vast private information, including Stratego [5, 6] and Dou Dizhu [7, 8]. A recent line of work suggests that self-play can approximate equilibria in small games where the exact equilibrium can be directly evaluated, both theoretically [9, 10] and empirically [11]. Among policy gradient methods, Proximal Policy Optimization (PPO) [12], deployed under the centralized training with decentralized execution (CTDE) paradigm is a widely used and competitive pipeline, where decentralized actors are trained using centralized critics [13].

However, even with its success, self-play RL still cannot achieve fully end-to-end training in games with a large dynamic range of payoffs, such as No-Limit Texas Hold'em. Existing work [14] has achieved strong results in HUNL, but still relies on restricted betting-action representations to handle the large dynamic range of rewards.

In this paper, we identify a new bottleneck of self-play RL in imperfect-information games: *advantage-estimation variance*. PPO-style algorithms use Generalized Advantage Estimation (GAE) [15] to aggregate multi-step temporal-difference residuals along sampled trajectories, thereby estimating the advantage with a variance–bias trade-off. GAE inherits randomness from future actions sampled in the multi-step trace. While in single-agent settings, policy randomness is largely exploratory and fades with training, it is intrinsic in self-play RL as equilibrium policies are generally stochastic. Different actions lead to different state values, so GAE produces high-variance gradient estimates.

We propose the ***Q*-boosting** estimator to address this problem. This estimator reduces the variance of advantage estimation by removing intrinsic action-sampling noise, replacing GAE's sampled multi-step backup with a multi-step *Expected SARSA*(λ) trace computed from a centralized Q-critic. Instead of bootstrapping through sampled next actions, Q-boosting takes policy expectations at each backup step, averaging out noise from stochastic future actions.

We introduce **Variance-Reduced Policy Optimization (VRPO)** based on this new estimator. VRPO is a PPO-style algorithm with improved stability that achieves strong empirical performance, showing that reducing variance in advantage estimation yields substantial gains in equilibrium-quality self-play.

To summarize, our main contributions are:

- We identify a previously unrecognized source of variance in GAE arising from sampled future actions in stochastic self-play. We show that this variance persists even with an exact state-value critic, and derive Q-boosting: a PPO-compatible advantage estimator that applies an Expected SARSA(λ) trace to a centralized Q-critic to average out future action-sampling noise.
- We propose **Variance-Reduced Policy Optimization (VRPO)**, a PPO-style algorithm that lowers the exact exploitability by more than 15% compared to tuned baselines (IPPO, MAPPO, MMD, and PPG) on mid-sized benchmarks. In Dou Dizhu, VRPO outperforms the previous *state-of-the-art* agent, PerfectDou [8], under a matched training budget. In heads-up no-limit Texas Hold'em, VRPO achieves positive performance against Slumbot, a strong poker baseline, without online search, subgame solving, or blueprint-based raise sizing.

2 Preliminary

2.1 Multiplayer Imperfect Information Game

We consider zero-sum multiplayer games with imperfect information, discrete time, and discrete action sets. Unlike cooperative MARL, we focus on competitive zero-sum settings, where the sum of players' returns is zero and equilibrium quality is measured by unilateral deviation gains. This captures a broad class of competitive decision-making problems, including many board games.

Let n denote the number of players, and let $\mathcal{S}$ be the set of all distinct states. We model simultaneous steps, when present, as sequences of imperfect-information states while preserving the original information structure; players do not observe actions or hidden variables that would be unavailable in the original game. We also explicitly resolve all environmental randomness and model stochastic transitions (e.g., card draws) by introducing a special player, NATURE, who follows a fixed policy. This modeling still allows for a random starting state, such as private hole cards in poker.

Let $\mathcal{E} \subseteq \mathcal{S}$ be the set of terminal states. Each non-terminal state $s \in \mathcal{S} \setminus \mathcal{E}$ in the game is assigned to a player $i(s) \in [\![n]\!]$ or to the NATURE player. When the game transitions to such a state, the assigned player must choose an action a from the set of available actions $\mathcal{A}$.[1] After player $i(s)$ takes action a in state s, the game transitions to a new state s', and each player i receives a reward $r_i(s, a)$. Since we introduce an environment player to handle stochasticity, the transition is considered *deterministic*: $\mathbb{P}(s' \mid s, a) = 1$ for the resulting state s' under the transition kernel $\mathbb{P}$.

Since the information in the game is imperfect, the active player $i(s)$ cannot observe the underlying state s directly. Instead, the player perceives only the private information associated with the state, denoted by $o(s) \in \mathcal{O}$. A player cannot distinguish between states that yield the same private information. Following standard terminology in the literature, the set of states that share the same private information is called an *information set*, denoted by $\mathcal{I}(i, o) \triangleq \{s \in \mathcal{S} : i(s) = i,\ o(s) = o\}$.

A gameplay trajectory is represented as a sequence $h = (s_0, a_0, s_1, \ldots, s_t)$, which starts from an initial state s_0 and ends in a terminal state $s_t \in \mathcal{E}$, with $\mathbb{P}(s_{\tau+1} \mid s_\tau, a_\tau) = 1$ for each timestep $0 \leq \tau < t$. The return of player i is the cumulative reward with discount factor $\gamma \in (0, 1]$:

$$R_i(h) \triangleq \sum_{\tau=0}^{t-1} \gamma^\tau \cdot r_i(s_\tau, a_\tau).$$

In the broader class of games we primarily consider, each player's return is determined solely by the terminal state s_t. To align with the general framework above, we assign the reward to the final action by setting $r_i(s_\tau, a_\tau) = 0$ for every timestep $\tau < t - 1$, and we use a discount factor of $\gamma = 1$. In this case, the return simplifies to: $R_i(h) = r_i(s_{t-1}, a_{t-1})$.

A policy is a mapping from private information to a distribution over available actions, denoted by $\pi_i : \mathcal{O} \to \Delta(\mathcal{A})$ for each player $i \in [\![n]\!]$. In the RL setting, each policy is parameterized by a model parameter θ_i, and we write $\pi_i(\cdot \mid o; \theta_i) \in \Delta(\mathcal{A})$ to denote the action distribution of player i given private information o.

Let $\boldsymbol{\theta} \triangleq (\theta_1, \ldots, \theta_n)$ denote the collection of model parameters. We define the joint policy profile $\boldsymbol{\pi} : \mathcal{S} \to \Delta(\mathcal{A})$ as $\boldsymbol{\pi}(\cdot \mid s; \boldsymbol{\theta}) \triangleq \pi_{i(s)}(\cdot \mid o(s); \theta_{i(s)})$, which represents the action distribution of the player acting at state s. In particular, for states s assigned to the environment player, i.e., those where $i(s) = \text{NATURE}$, the joint policy profile $\boldsymbol{\pi}(\cdot \mid s; \boldsymbol{\theta})$ is fixed and independent of the model parameters $\boldsymbol{\theta}$, corresponding instead to the environment's predefined transition distribution.

2.2 State Value Function and Action Value Function

Given a joint policy $\boldsymbol{\pi}$, we define two quantities. The V-value function, or state-value function, $V_i^{\boldsymbol{\pi}} : \mathcal{S} \to \mathbb{R}$, denotes the expected *return-to-go* for player i at state s, assuming all players follow the joint policy. The Q-value function, $Q_i^{\boldsymbol{\pi}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, denotes the expected return-to-go when a particular action a is taken in state s:

$$V_i^{\boldsymbol{\pi}}(s) \triangleq \mathop{\mathbb{E}}_{h}\left[\sum_{\tau \geq k} \gamma^{\tau-k} \cdot r_i(s_\tau, a_\tau) \;\middle|\; s_k = s\right], Q_i^{\boldsymbol{\pi}}(s, a) \triangleq \mathop{\mathbb{E}}_{h}\left[\sum_{\tau \geq k} \gamma^{\tau-k} \cdot r_i(s_\tau, a_\tau) \;\middle|\; s_k = s,\ a_k = a\right],$$

where the trajectory $h = (s_0, a_0, s_1, \ldots, s_t)$ is generated according to the joint policy $\boldsymbol{\pi}$, and we condition on $s = s_k$ for some timestep k. Because stochasticity is modeled explicitly by the NATURE player, the transition kernel $\mathbb{P}$ is deterministic, and the Q-value function can be computed from the Bellman equation in the style of Expected SARSA [16] as

$$Q_i^{\boldsymbol{\pi}}(s, a) = r_i(s, a) + \gamma \sum_{a' \in \mathcal{A}} \boldsymbol{\pi}(a' \mid s') \cdot Q_i^{\boldsymbol{\pi}}(s', a'), \tag{2.1}$$

[1] Formally, the available action set may depend on the state, i.e., $\mathcal{A} = \mathcal{A}(s)$. Since this dependence is immaterial to our analysis, we suppress it in the notation.

where the next state s' satisfies $\mathbb{P}(s' \mid s, a) = 1$. The V-value function $V_i^{\boldsymbol{\pi}}(s)$ can then be computed by summing the terms $\boldsymbol{\pi}(a \mid s) \cdot Q_i^{\boldsymbol{\pi}}(s, a)$ over the action set $\mathcal{A}$. In this way, backpropagation through this Bellman equation can be performed deterministically in $\mathcal{O}(|\mathcal{A}|)$ time.

2.3 Game-Theoretic Equilibrium

We measure equilibrium quality by the gain available from unilateral deviations. For a joint policy $\boldsymbol{\pi}$, the maximal deviation gain of player i is $\Delta_i(\boldsymbol{\pi}) \triangleq \max_{\pi_i'} \mathbb{E}_{h \sim (\pi_i', \boldsymbol{\pi}_{-i})}[R_i(h)] - \mathbb{E}_{h \sim \boldsymbol{\pi}}[R_i(h)]$. We define exploitability as the average deviation gain across all players:

$$\mathrm{Expl}(\boldsymbol{\pi}) \triangleq \frac{1}{n} \sum_{i=1}^{n} \Delta_i(\boldsymbol{\pi}).$$

A profile is an ε-Nash equilibrium [17] if the deviation gain $\Delta_i(\boldsymbol{\pi}) \leq \varepsilon$ for all players i (hence $\mathrm{Expl}(\boldsymbol{\pi}) \leq \varepsilon$). Conversely, $\mathrm{Expl}(\boldsymbol{\pi}) \leq \varepsilon$ implies the worst-case deviation gain $\max_i \Delta_i(\boldsymbol{\pi}) \leq n\varepsilon$. In two-player zero-sum games, Liu et al. [9, 10] show policy gradient with external regularization converges to an ε-Nash equilibrium in time polynomial in the number of states, and thus guarantees $\mathrm{Expl}(\boldsymbol{\pi}) \leq \varepsilon$ under our definition.

3 Variance Reduction by Q-Boosting

A general framework for multi-agent reinforcement learning (MARL) is centralized training with decentralized execution (CTDE), commonly implemented via a *centralized critic* and decentralized actors [13]. The key asymmetry is that, during training, the critic is allowed to "see more of the game" than any individual actor: while an actor $\pi_i(a_i \mid o_i)$ must act based only on its private information o_i, the critic can condition on centralized information such as the global state, the joint action, or other agents' observations. This additional context often makes the critic substantially easier to train and improves the fidelity of the resulting baseline or advantage estimates used in policy gradient updates.

This distinction is especially important in MARL because gradient estimates inherit additional randomness from other agents' sampled actions, as well as from the non-stationarity introduced when all agents update simultaneously. A centralized critic provides a natural way to average out some of this uncertainty by taking expectations over actions that are not controlled (or not observed) by player i, thereby reducing the variance of the policy gradient estimator while keeping the actors decentralized at execution. In this work, we focus on the CTDE setup, such as MAPPO [13], where a centralized critic is used.

We begin by examining the policy gradient of player i. Let $J_i(\boldsymbol{\theta})$ be the expected cumulative reward object corresponding to the joint policy profile $\boldsymbol{\pi}(\cdot; \boldsymbol{\theta})$, where $\boldsymbol{\theta} \triangleq (\theta_1, \ldots, \theta_n)$ is the collection of models. According to Schulman et al. [15], the policy gradient of player i can be computed as

$$\nabla_{\theta_i} J_i(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{h \sim \boldsymbol{\pi}} \Bigg[\sum_{\tau : i(s_\tau) = i} A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau) \cdot \nabla_{\theta_i} \log \pi_i(a_\tau \mid o_\tau; \theta_i) \Bigg], \tag{3.1}$$

where player i makes a decision at state s_τ based on their private information $o_\tau \triangleq o_i(s_\tau)$, and the advantage function is defined as $A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau) \triangleq Q_i^{\boldsymbol{\pi}}(s_\tau, a_\tau) - V_i^{\boldsymbol{\pi}}(s_\tau)$, with the Q-value function $Q_i^{\boldsymbol{\pi}}$ and the V-value function $V_i^{\boldsymbol{\pi}}$. In policy gradient methods that estimate $\nabla_{\theta_i} J_i(\boldsymbol{\theta})$, the gradient is typically computed by sampling states s_τ from trajectories induced by the current policy and estimating the corresponding advantages $A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau)$.

3.1 The Intrinsic Variance of GAE

An unbiased estimator of $A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau)$ can be derived from the return-to-go of the sampled trajectory. A more sophisticated method is the generalized advantage estimator (GAE) [15] used in PPO [12]. Given a V-critic $V_i(\cdot; \phi_i) : \mathcal{S} \to \mathbb{R}$ parameterized by ϕ_i, the temporal-difference (TD) residual is defined as $\delta_{i,\tau} \triangleq r_i(s_\tau, a_\tau) + \gamma V_i(s_{\tau+1}; \phi_i) - V_i(s_\tau; \phi_i)$. For a trace parameter $\lambda \in [0, 1]$, the generalized advantage estimate is given by

$$\widehat{A}_{i,\tau}^{\mathrm{GAE}} \triangleq \sum_{\tau' \geq \tau} (\lambda\gamma)^{\tau' - \tau} \cdot \delta_{i,\tau'}.$$

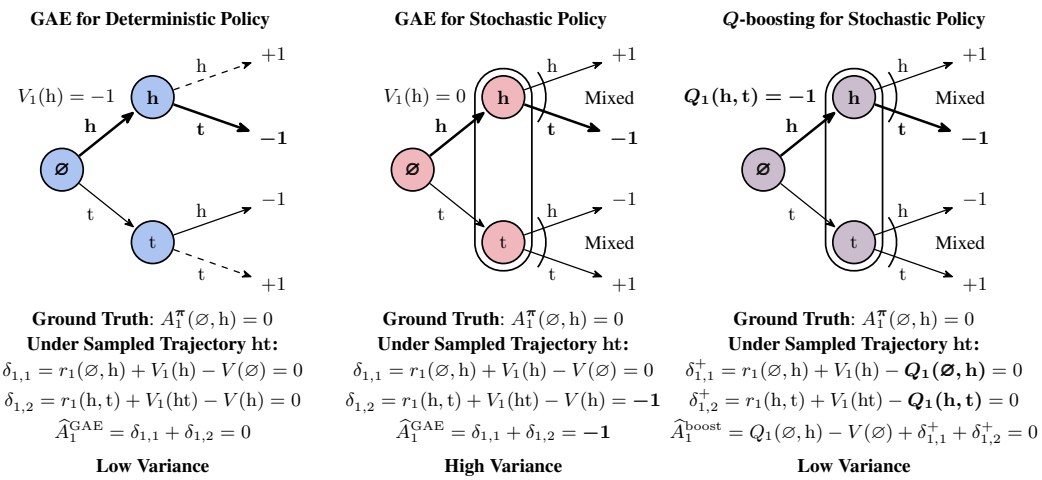

Figure 1: We compare GAE and Q-boosting in the matching-pennies game, where the first player receives a reward of $+1$ if their action matches the second player's and -1 otherwise. With perfect information, the second player's policy is deterministic, and GAE exhibits low variance. However, under imperfect information, the equilibrium policy of the second player mixes uniformly between playing h and t. The action value of taking any action at state h generally differs from the state value $V(\mathrm{h})$. As a result, GAE inherits variance from sampling future actions at state h, leading to high-variance advantage estimates even with an exact critic. In contrast, Q-boosting replaces the state value with the action value, eliminating variance due to action sampling and producing a low-variance estimator. The discount factor is $\gamma = 1$ and trace parameter is $\lambda = 1$ for clarity.

Although GAE typically reduces variance relative to naïve Monte Carlo return-to-go estimation, it cannot eliminate the randomness arising from action sampling. The variance persists even when the V-critic is perfect, i.e., $V_i(\cdot; \phi_i) = V_i^{\boldsymbol{\pi}}$. In this case, the one-step TD residual can be computed as $\delta_{i,\tau'} = r_i(s_{\tau'}, a_{\tau'}) + \gamma V_i^{\boldsymbol{\pi}}(s_{\tau'+1}) - V_i^{\boldsymbol{\pi}}(s_{\tau'})$. For the primary term with $\tau' = \tau$, the expected contribution of the TD residual is $\mathbb{E}[\delta_{i,\tau} \mid s_\tau, a_\tau] = A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau)$, which provides an unbiased estimator of the advantage. For any auxiliary term with $\tau' > \tau$, however, the expected contribution is $\mathbb{E}[\delta_{i,\tau'} \mid s_\tau, a_\tau] = \mathbb{E}[A_i^{\boldsymbol{\pi}}(s_{\tau'}, a_{\tau'}) \mid s_\tau, a_\tau] = 0$. But the TD residuals $\delta_{i,\tau'}$ are generally nonzero at equilibrium in imperfect-information games, which causes these auxiliary terms to be generally nonzero and incur nonzero variance. We illustrate this variance with a concrete example in Figure 1.

Therefore, even with an exact value function, the one-step TD residual remains a non-degenerate random variable unless the policy is deterministic at $s_{\tau'}$ or all actions have identical Q-values; these conditions generally do not hold in imperfect-information games. Since GAE aggregates these residuals across multiple time steps when $\lambda > 0$, this action-sampling noise is propagated through the multi-step advantage estimator rather than eliminated.

3.2 The Q-Boosting Estimator

We introduce ***Q*-boosting** to address the issue mentioned above. The central idea is to replace the sampled multi-step TD trace used in GAE with a multi-step *Expected SARSA* trace computed from a centralized Q-critic. The additional variance of GAE arises because the multi-step trace depends on sampled future actions whose action values may differ. Q-boosting reduces this source of noise by averaging over those future action distributions.

Specifically, given a Q-critic $Q_i(\cdot; \phi_i) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and joint policy profile $\boldsymbol{\pi}(\cdot; \boldsymbol{\theta})$, we define the state value as

$$V_i^{\boldsymbol{\pi}}(s; \phi_i) \triangleq \sum_{a \in \mathcal{A}} \boldsymbol{\pi}(a \mid s; \boldsymbol{\theta}) \cdot Q_i(s, a; \phi_i).$$

With one-step Expected SARSA TD residual $\delta^+_{i,\tau} \triangleq r_i(s_\tau, a_\tau) + \gamma V_i^{\boldsymbol{\pi}}(s_{\tau+1}; \phi_i) - Q_i(s_\tau, a_\tau; \phi_i)$ and trace parameter $\lambda \in [0, 1]$, Q-boosting estimates the advantage as

$$\widehat{A}^{\mathrm{boost}}_{i,\tau} \triangleq Q_i(s_\tau, a_\tau; \phi_i) - V_i^{\boldsymbol{\pi}}(s_\tau; \phi_i) + \sum_{\tau' \geq \tau} (\lambda\gamma)^{\tau' - \tau} \cdot \delta^+_{i,\tau'}. \tag{3.2}$$

The Q-boosting estimator also induces a multi-step Bellman-style update for the Q-critic by accumulating discounted Expected SARSA TD residuals along the trajectory, analogous to GAE:

$$Q_i^{\text{target}}(s_\tau, a_\tau) \leftarrow Q_i(s_\tau, a_\tau; \phi_i) + \sum_{\tau' \geq \tau} (\lambda\gamma)^{\tau'-\tau} \cdot \delta^+_{i,\tau'}. \tag{3.3}$$

This target can be viewed as a multi-step generalization of the Expected SARSA update, analogous to the λ-return in GAE but applied to action-value functions.

Compared to GAE, the residual $\delta^+_{i,\tau'}$ uses the action-conditioned value $Q_i(s_{\tau'}, a_{\tau'}; \phi_i)$ rather than the state value $V_i(s_{\tau'}; \phi_i)$. Consequently, when $Q_i = Q_i^{\boldsymbol{\pi}}$, each Expected-SARSA residual vanishes pathwise, so the multi-step trace no longer carries noise from sampled future actions. With an approximate Q-critic, Q-boosting replaces this action-sampling variance with critic error.

Formally, we have the following result. The theorem is proved in Appendix C.

Theorem 3.1 (Unbiasedness and variance reduction of Q-boosting)**.** Fix a timestep τ at which player $i(s_\tau) = i$ acts, and suppose trajectories are sampled on-policy from $\boldsymbol{\pi}$. With trace parameter $\lambda = 1$, the Q-boosting estimator based on any Q-critic $Q_i(\cdot; \phi_i)$ gives an unbiased estimator of the score-function policy-gradient term:

$$\mathbb{E}[\widehat{A}^{\text{boost}}_{i,\tau} \cdot \nabla_{\theta_i} \log \pi_i(a_\tau \mid o_\tau; \theta_i) \mid s_\tau] = \mathbb{E}[A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau) \cdot \nabla_{\theta_i} \log \pi_i(a_\tau \mid o_\tau; \theta_i) \mid s_\tau].$$

Moreover, if some future state s_u reached with positive probability after (s_τ, a_τ) exhibits non-zero variance in action-value, $\text{Var}_{a \sim \boldsymbol{\pi}(\cdot|s_u)}[Q_i^{\boldsymbol{\pi}}(s_u, a)] > 0$, for any $\lambda \in (0, 1]$, there exists $\xi^\lambda_{i,\tau} > 0$ such that any Q-critic satisfying $\|Q_i(\cdot; \phi_i) - Q_i^{\boldsymbol{\pi}}\|_\infty < \xi^\lambda_{i,\tau}$ yields smaller mean-squared advantage-estimation error than GAE with an exact V-critic and the same trace parameter λ:

$$\mathbb{E}\big[\big(\widehat{A}^{\text{boost}}_{i,\tau} - A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau)\big)^2 \,\big|\, s_\tau, a_\tau\big] < \mathbb{E}\big[\big(\widehat{A}^{\text{GAE}}_{i,\tau} - A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau)\big)^2 \,\big|\, s_\tau, a_\tau\big].$$

This result shows that, for $\lambda = 1$ and on-policy trajectories, the score-function policy-gradient term remains unbiased when the true advantage is replaced by the Q-boosting estimator, even with an arbitrary fixed, possibly inaccurate Q-critic. For $\lambda \in (0, 1]$, whenever exact-V GAE has nonzero future-action noise, a sufficiently accurate Q-critic yields lower conditional mean-squared advantage-estimation error than exact-V GAE with the same trace parameter. In practice, we set $\lambda = 0.95$ to obtain a good bias-variance tradeoff mirroring GAE [15].

We note that the state value $V_i^{\boldsymbol{\pi}}(s; \phi_i)$ is computed by enumerating the available actions in practice. For games with complex decisions, this can be done by decomposing complex decisions into multiple time steps such that the action set at each time step is small: In Dou Dizhu, this reduces the per-step maximum from $27{,}472$ actions [8] to 35. See Appendix G for details.

Relation to prior expected-value estimators. We remark that similar estimators were proposed in the literature. The trace in Q-boosting follows the Expected-SARSA(λ) form [16], but is used here to construct PPO-style advantage estimates for stochastic self-play. Tree-Backup [18] and Retrace [19] provide value-evaluation traces for off-policy policy evaluation, COMA uses a one-step counterfactual baseline [20], and Q-Prop [21] uses an action-value critic as a control variate for single-agent policy gradients. However, our main contribution is to identify that GAE retains future-action sampling noise in self-play even with an exact V-critic, since equilibrium policies remain stochastic at convergence.

4 Variance-Reduced Policy Optimization

In this section, we introduce **Variance-Reduced Policy Optimization (VRPO)**, a variance-reduced variant of *Proximal Policy Optimization* (PPO) [12] that leverages the Q-boosting estimator (3.2). The complete pseudo-code of the algorithm can be found in Appendix B.

Compared to PPO, VRPO retains the same clipped policy gradient surrogate objective and trust-region-style updates, but modifies both the advantage estimation and critic learning procedures to reduce the variance of the policy gradient.

4.1 Objectives and Variance-Reduced Estimation

The core of VRPO replaces the GAE-based advantage used in PPO with the proposed Q-boosting estimator (3.2), while retaining the clipped policy-gradient surrogate objective. Given a trajectory $h = (s_0, a_0, s_1, \ldots, s_t)$ sampled under the joint reference policy $\boldsymbol{\pi}^{\mathrm{ref}} = (\pi_1^{\mathrm{ref}}, \ldots, \pi_n^{\mathrm{ref}})$, the surrogate loss for player i is defined as

$$\mathcal{L}_i^{\mathrm{pg}}(h, \theta_i) \triangleq -\sum_{\tau : i(s_\tau) = i} \min\left\{ \frac{\pi_i(a_\tau \mid o(s_\tau); \theta_i)}{\pi_i^{\mathrm{ref}}(a_\tau \mid o(s_\tau))} \widehat{A}_{i,\tau}^{\mathrm{boost}}, \mathrm{clamp}_{[1-\varepsilon, 1+\varepsilon]}\left(\frac{\pi_i(a_\tau \mid o(s_\tau); \theta_i)}{\pi_i^{\mathrm{ref}}(a_\tau \mid o(s_\tau))} \right) \widehat{A}_{i,\tau}^{\mathrm{boost}} \right\}. \tag{4.1}$$

Following prior work in self-play RL [11, 22], we include an explicit policy-regularization term $\mathcal{L}_i^{\mathrm{reg}}(h, \theta_i)$ to promote exploration and encourage last-iterate convergence. The policy is regularized using a KL-divergence penalty relative to a fixed reference distribution, which is chosen in practice to be uniform over the action space:

$$\mathcal{L}_i^{\mathrm{reg}}(h, \theta_i) \triangleq \sum_{\tau : i(s_\tau) = i} \mathrm{KL}\big(\pi_i(\cdot \mid o(s_\tau); \theta_i) \,\|\, \mathrm{Unif}\big). \tag{4.2}$$

For the critic, VRPO replaces the single centralized V-critic in MAPPO [13] with a single centralized Q-critic with full-information inputs. The critic is trained by minimizing the squared regression loss

$$\mathcal{L}_i^{\mathrm{val}}(h, \phi_i) \triangleq \sum_{\tau \geq 0} \frac{1}{2}\big(Q_i(s_\tau, a_\tau; \phi_i) - Q_i^{\mathrm{target}}(s_\tau, a_\tau)\big)^2, \tag{4.3}$$

where the target Q_i^{target} is defined via the Q-boosting update in (3.3).

4.2 Training Procedure

VRPO follows the standard PPO training schedule, alternating between on-policy actor updates and centralized critic regression. At the beginning of each iteration, the current policies are frozen as reference policies: $\pi_i^{\mathrm{ref}} \leftarrow \pi_i(\cdot; \theta_i)$ for each player $i \in [\![n]\!]$. Using the joint reference policy $(\pi_1^{\mathrm{ref}}, \ldots, \pi_n^{\mathrm{ref}})$, the algorithm collects a batch of B rollout trajectories, denoted by $\mathcal{D}_{\mathrm{rollout}}$.

The actor parameters are optimized using only the on-policy rollout data. In each of the K_{actor} epochs, the rollout batch is partitioned into M minibatches, and each player updates its policy parameters by minimizing the PPO-style objective $\mathcal{L}_i^{\mathrm{pg}}(\cdot, \theta_i) + \alpha \cdot \mathcal{L}_i^{\mathrm{reg}}(\cdot, \theta_i)$, where α controls the strength of regularization, balancing the policy-gradient surrogate (4.1) and policy regularization (4.2). This update follows the standard PPO minibatch training procedure.

During actor optimization, VRPO recomputes the policy-expectation terms in $\widehat{A}_{i,\tau}^{\mathrm{boost}}$ using the current actor probabilities while keeping the critic values fixed, and uses the resulting advantages as ordinary stop-gradient PPO coefficients. This recomputation adds negligible overhead, since evaluating the PPO ratio already requires the current actor probabilities; the additional operation is only a policy-weighted sum over critic values.

The centralized Q-critics are trained for K_{critic} epochs by minimizing the regression loss $\mathcal{L}_i^{\mathrm{val}}(\cdot, \phi_i)$, defined in (4.3).

5 Evaluations

In this section, we evaluate the empirical performance of VRPO across a range of imperfect-information games, spanning from mid-sized benchmarks, where exact exploitability can be computed, to large-scale domains where head-to-head performance is the standard measure. We provide a brief description of each game in Appendix D, training details in Appendix E, evaluation methodology in Appendix F, game-specific encodings in Appendix G, and training dynamics in Appendix H. Notably, we use the *same* network architecture and hyperparameters across all games, adjusting only the number of training iterations and batch sizes to account for differences in game complexity.

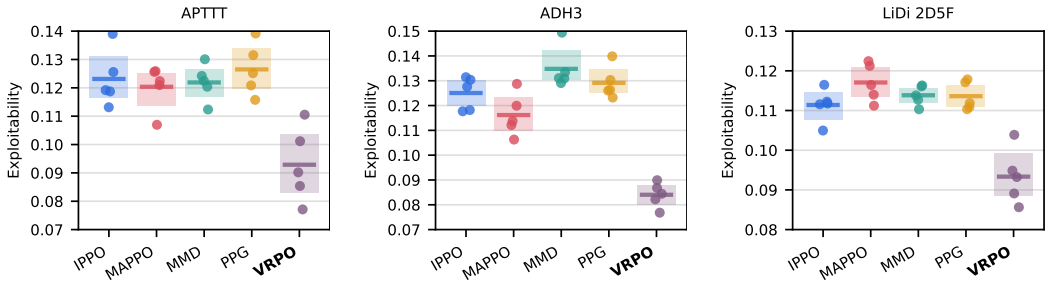

Figure 2: Exact exploitability (lower is better) of agents in various games under a shared training schedule. Each configuration is trained with 5 different seeds.

5.1 Mid-Sized Games with Exact Exploitability

We benchmark VRPO on mid-sized games where exact exploitability can be computed: the number of information sets fits in memory, and full game-tree traversal is computationally feasible. These benchmarks include Abrupt Phantom Tic-Tac-Toe (APTTT), Abrupt Dark Hex 3 (ADH3), and Liar's Dice with 2 Dice and 5 Faces (LiDi 2D5F). The sizes of these games are listed in Table 1. We remark that these games are more than 4 orders of magnitude larger than Leduc Hold'em [23], a common benchmark for CFR-style algorithms.

Game	Infosets	Game States
Leduc Hold'em	936	9,457
APTTT	23.3×10^6	27.1×10^9
ADH3	27.3×10^6	29.3×10^9
LiDi 2D5F	52.4×10^6	1.31×10^9

Table 1: Game sizes in benchmarks

We compare VRPO with several policy-gradient-based baselines, including Independent PPO (IPPO) [12, 24], MAPPO [13], MMD [22], and PPG [25]. All algorithms are trained for 2,000 iterations with a batch size of $B = 2048$ trajectories. Learning rates and other optimization hyperparameters are matched across methods, while algorithm-specific hyperparameters follow the default settings provided by the CleanRL library [26]. To ensure comparable compute budgets, we use $K_{\text{actor}} = 4$ actor epochs and $K_{\text{critic}} = 4$ critic epochs per iteration in VRPO, each with $M = 4$ minibatches. This mirrors the standard PPO training schedule of $K = 4$ epochs and $M = 4$ minibatches for both policy and value updates, as implemented in CleanRL.

We report the results in Figure 2. Baseline algorithms are tuned and outperform the results reported by Rudolph et al. [11]. VRPO consistently achieves more than 15% lower exploitability than all baselines across these games, with an additional computation cost of around 5% (see Appendix I). More hyperparameter-ablation experiments are provided in Appendices K and L, including PPO clipping coefficient ε, regularization coefficient α, and trace parameter λ.

5.2 Dou Dizhu

We apply VRPO to Dou Dizhu, a three-player game between one Landlord and two independent Peasants. The game comprises at least 10^{53} information sets, each averaging a size of 10^{23} [7, 27]. Dou Dizhu is a standard benchmark for self-play RL, with prior agents including DeltaDou [28], DouZero [7], and PerfectDou [8], and implementations in RLCard [27] and OpenSpiel [29].

We train the agent for 40,000 iterations with a batch size of $B = 2048$ trajectories, which takes approximately 55 hours on 4× RTX 5090 GPUs. This corresponds to approximately 2.65×10^9 timesteps, closely matching the 2.5×10^9 timesteps of PerfectDou [8].

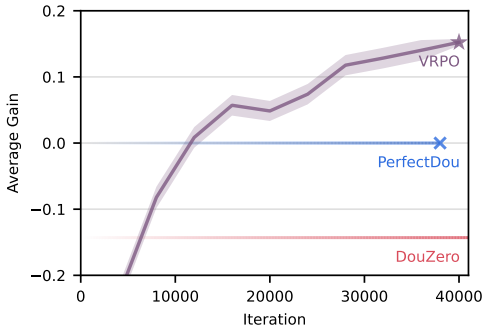

Figure 3: Role-averaged gain (higher is better; values above zero indicate a win) against PerfectDou [8], over the course of VRPO training.

We evaluate the role-averaged performance of the VRPO agent against PerfectDou [8], the previous state-of-the-art trained by MAPPO, in a setting where one agent plays the Landlord and the other controls the two independent Peasants. Figure 3 shows that VRPO surpasses PerfectDou after only about 40% as many training timesteps as PerfectDou, despite relying solely on terminal rewards rather than a manually designed loss function. Each evaluation is averaged over $10{,}000$ duplicate deals, except for the final policy, which is evaluated over $100{,}000$ duplicate deals. We include DouZero [7] as an additional reference point against PerfectDou; DouZero was trained for approximately 10^{10} timesteps, corresponding to more than $150{,}000$ iterations under our training schedule.

We also evaluate the final VRPO checkpoint against prior baselines in Table 2, with each matchup evaluated over $100{,}000$ duplicate deals. The reported uncertainty is $1.0 \times \mathrm{SEM}$. We further report position-specific performance in Appendix J. VRPO achieves consistently strong performance and outperforms PerfectDou and DouZero across all matchups, showing that VRPO scales effectively to large-scale imperfect-information games and establishes a new state of the art.

Hero \ Villain	**VRPO**	PerfectDou	DouZero
VRPO (ours)	—	0.152 ± 0.004	0.286 ± 0.004
PerfectDou	-0.152 ± 0.004	—	0.141 ± 0.004
DouZero	-0.286 ± 0.004	-0.141 ± 0.004	—

Table 2: Average gain of the row agent playing against the column player.

5.3 Heads-Up No-Limit Texas Hold'em

We apply VRPO to Heads-Up No-Limit Texas Hold'em with an initial stack of 200 big blinds. The agent is trained for $40{,}000$ iterations with a batch size of $B = 8192$ trajectories, totaling approximately 1.51×10^9 timesteps. Training takes roughly 63 hours on $4\times$ RTX 5090 GPUs.

We compare training with the IPPO and MAPPO baselines. As shown in Figure 4, VRPO significantly reduces the variance of the advantage compared to the baseline algorithms.

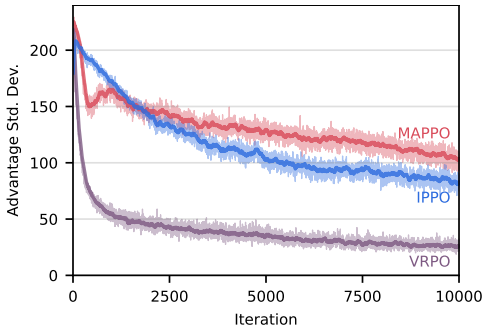

Figure 4: Comparison of the standard deviation of the used advantage $\widehat{A}$ during the first $10{,}000$ steps of training under different methods.

We evaluate the performance of the VRPO agent using the 2022 version of Slumbot [30]. Over a session of $1{,}000{,}000$ hands, the VRPO agent achieves an average gain of 33 ± 19 mBB/hand. For comparison, an always-folding bot loses 750 mBB/hand. While this does not constitute a decisive win, the result is achieved without any form of online search, subgame solving, or blueprint-based raise amounts, demonstrating the scalability of VRPO in imperfect-information domains.

6 Conclusion

In this paper, we identify that sampling future actions in GAE introduces additional variance, which is especially problematic in equilibrium learning, where policies are typically stochastic. We address this with *Q-boosting*, which replaces action-sampled multi-step backups with a multi-step Expected SARSA(λ) trace. Incorporating this new estimator into PPO yields VRPO. Empirically, VRPO achieves strong performance across a range of games at different scales, including a new state of the art in Dou Dizhu.

Limitations and Scope. VRPO targets discrete-action games, where the policy expectation can be computed exactly. With task-specific decomposition, this remains practical at scale, with at most 35 actions per step in Dou Dizhu and 16 in HUNL. Continuous actions are a natural extension, where the exact sum can be replaced by quadrature, sampling, or a DDPG-style deterministic surrogate. As for PPO and MAPPO, no general convergence theorem is available for VRPO under nonlinear function approximation. Our focus is stochastic self-play with centralized critics.

Acknowledgments and Disclosure of Funding

This work was supported in part by National Science Foundation awards CCF-2443068 and IIS-2552046, Office of Naval Research grant N000142512296, and an AI2050 Early Career Fellowship.

References

[1] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. *Advances in neural information processing systems*, 20, 2007.

[2] Noam Brown, Tuomas Sandholm, and Strategic Machine. Libratus: The superhuman AI for no-limit poker. In *IJCAI*, pages 5226–5228, 2017.

[3] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. In *International conference on machine learning*, pages 793–802. PMLR, 2019.

[4] Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. *Advances in neural information processing systems*, 33:17057–17069, 2020.

[5] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

[6] Samuel Sokota, Eugene Vinitsky, Hengyuan Hu, J Zico Kolter, and Gabriele Farina. Superhuman AI for Stratego using self-play reinforcement learning and test-time search. *arXiv preprint arXiv:2511.07312*, 2025.

[7] Daochen Zha, Jingru Xie, Wenye Ma, Sheng Zhang, Xiangru Lian, Xia Hu, and Ji Liu. Douzero: Mastering doudizhu with self-play deep reinforcement learning. In *international conference on machine learning*, pages 12333–12344. PMLR, 2021.

[8] Guan Yang, Minghuan Liu, Weijun Hong, Weinan Zhang, Fei Fang, Guangjun Zeng, and Yue Lin. Perfectdou: Dominating doudizhu with perfect information distillation. *Advances in neural information processing systems*, 35:34954–34965, 2022.

[9] Mingyang Liu, Asuman Ozdaglar, Tiancheng Yu, and Kaiqing Zhang. The power of regularization in solving extensive-form games. *arXiv preprint arXiv:2206.09495*, 2022.

[10] Mingyang Liu, Gabriele Farina, and Asuman Ozdaglar. A policy-gradient approach to solving imperfect-information games with iterate convergence. *arXiv preprint arXiv:2408.00751*, 2024.

[11] Max Rudolph, Nathan Lichtle, Sobhan Mohammadpour, Alexandre Bayen, J Zico Kolter, Amy Zhang, Gabriele Farina, Eugene Vinitsky, and Samuel Sokota. Reevaluating policy gradient methods for imperfect-information games. *arXiv preprint arXiv:2502.08938*, 2025.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[13] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

[14] Enmin Zhao, Renye Yan, Jinqiu Li, Kai Li, and Junliang Xing. AlphaHoldem: High-performance artificial intelligence for heads-up no-limit poker via end-to-end reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 4689–4697, 2022.

[15] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[16] Harm Van Seijen, Hado Van Hasselt, Shimon Whiteson, and Marco Wiering. A theoretical and empirical analysis of expected SARSA. In *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*, pages 177–184. IEEE, 2009.

[17] Tim Roughgarden. Algorithmic game theory. *Communications of the ACM*, 53(7):78–86, 2010.

[18] Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 759–766, 2000.

[19] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

[20] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[21] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations*, 2017.

[22] Samuel Sokota, Ryan D'Orazio, J Zico Kolter, Nicolas Loizou, Marc Lanctot, Ioannis Mitliagkas, Noam Brown, and Christian Kroer. A unified approach to reinforcement learning, quantal response equilibria, and two-player zero-sum games. *arXiv preprint arXiv:2206.05825*, 2022.

[23] Finnegan Southey, Michael P Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. *arXiv preprint arXiv:1207.1411*, 2012.

[24] Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the Starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

[25] Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *International Conference on Machine Learning*, pages 2020–2027. PMLR, 2021.

[26] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL `http://jmlr.org/papers/v23/21-1342.html`.

[27] 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. *arXiv preprint arXiv:1910.04376*, 2019.

[28] Qiqi Jiang, Kuangzheng Li, Boyao Du, Hao Chen, and Hai Fang. Deltadou: Expert-level Doudizhu AI through self-play. In *IJCAI*, pages 1265–1271, 2019.

[29] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

[30] ericgjackson. slumbot2019: Implementations of cfr for solving a variety of holdem-like poker games. GitHub repository, 2023. URL `https://github.com/ericgjackson/slumbot2019`. Version/commit: a74c99d (Sep 18, 2023). Accessed: 2026-01-27.

[31] Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit Texas hold'em. In *IJCAI*, volume 15, pages 645–652, 2015.

[32] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[33] George W Brown. Iterative solution of games by fictitious play. *Act. Anal. Prod Allocation*, 13 (1):374, 1951.

[34] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[35] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

[36] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[37] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering Chess and Shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[38] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[39] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

[40] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Appendix

A Additional Related Work

One of the major breakthroughs in large-scale equilibrium computation for imperfect-information games was Counterfactual Regret Minimization (CFR) [1] and its variants. In particular, CFR+ enabled Cepheus [31] to solve Heads-Up Limit Texas Hold'em. In the No-Limit variant, successor methods such as Libratus [2], which introduced a search-based approach, and DeepStack [32], which employed a learning-based approach, defeated professional players. ReBeL [4] later unified these approaches to achieve state-of-the-art performance.

While regret minimization methods have seen significant success in poker, policy optimization approaches have also made substantial progress in other imperfect-information games. Early foundations trace back to Fictitious Play [33], while modern approaches typically rely on self-play reinforcement learning. Landmark achievements include OpenAI Five [34], defeating the world champion Dota 2 team OG using large-scale self-play with PPO [12], and AlphaStar [35], reaching Grandmaster level in StarCraft II using population-based and self-play methods related to PSRO [36].

In board games, AlphaGo and its successor AlphaZero [37, 38] mastered perfect-information games such as chess, shogi, and Go using Monte Carlo Tree Search. For imperfect-information games, DeepNash [5] reached human-expert-level performance in Stratego using Regularized Nash Dynamics. More recently, Ataraxos [6] combined self-play RL with test-time search to achieve vastly superhuman performance in the game of Stratego.

B Variance-Reduced Policy Optimization

Algorithm 1 Variance-Reduced Policy Optimization

Require: Decentralized actors $\{\pi_i(\cdot;\theta_i)\}_{i=1}^n$
Require: Centralized critics $\{Q_i(\cdot;\phi_i)\}_{i=1}^n$
Require: Batch size B, minibatches M
Require: Actor epochs K_{actor}, critic epochs K_{critic},
Require: Regularization coefficient $\alpha > 0$

for iteration $= 1, 2, \ldots$ **do**
  Set reference policies $\pi_i^{\text{ref}} \leftarrow \pi_i(\cdot;\theta_i)$ for all i
  Collect B trajectories $\mathcal{D}_{\text{rollout}}$ using joint $\boldsymbol{\pi}^{\text{ref}}$
  for K_{actor} epochs $\times$ M minibatches **do**
    Sample $|\mathcal{B}| = B/M$ trajectories $\mathcal{B} \subset \mathcal{D}_{\text{rollout}}$
    Update each θ_i by descending ▷ **Actor update**

$$\frac{1}{|\mathcal{B}|}\sum_{h\in\mathcal{B}}\Big(\mathcal{L}_i^{\text{pg}}(h,\theta_i) + \alpha\cdot\mathcal{L}_i^{\text{reg}}(h,\theta_i)\Big)$$

  end for
  Add $\mathcal{D}_{\text{rollout}}$ to cyclic replay buffer $\mathcal{D}_{\text{replay}}$
  Set $\mathcal{B} \leftarrow \mathcal{D}_{\text{rollout}}$
  for K_{critic} epochs $\times$ M minibatches **do**
    Update each ϕ_i by descending ▷ **Critic update**

$$\frac{1}{|\mathcal{B}|}\sum_{h\in\mathcal{B}}\mathcal{L}_i^{\text{val}}(h,\phi_i)$$

    Resample $|\mathcal{B}| = B/M$ trajectories $\mathcal{B} \subset \mathcal{D}_{\text{replay}}$
  end for
end for

Within the algorithm, the policy gradient term $\mathcal{L}_i^{\text{pg}}(h,\theta_i)$ is given by (4.1), the regularization penalty term $\mathcal{L}_i^{\text{reg}}(h,\theta_i)$ by (4.2), and the value loss term $\mathcal{L}_i^{\text{val}}(h,\phi_i)$ by (4.3).

C Proof of Theorem 3.1

Proof. We consider a finite-horizon trajectory $h = (s_0, a_0, s_1, \ldots, s_T)$ sampled on-policy from the joint policy profile $\boldsymbol{\pi}$. If a trajectory terminates early, we pad it with an absorbing terminal state with zero reward and zero value. Hence the horizon can be treated as fixed. Since all environment randomness is represented by the NATURE player, the transition after conditioning on the current state and action is deterministic.

For compactness, write

$$\bar{Q}_i(s,a) \triangleq Q_i(s,a;\phi_i), \qquad \bar{V}_i(s) \triangleq \sum_{a\in\mathcal{A}(s)} \boldsymbol{\pi}(a \mid s)\bar{Q}_i(s,a).$$

The true state value satisfies

$$V_i^{\boldsymbol{\pi}}(s) = \sum_{a\in\mathcal{A}(s)} \boldsymbol{\pi}(a \mid s)\cdot Q_i^{\boldsymbol{\pi}}(s,a).$$

At terminal states, we set $\bar{V}_i(s_T) = V_i^{\boldsymbol{\pi}}(s_T) = 0$.

Unbiasedness of the score-function term. We first prove the unbiasedness statement for $\lambda = 1$. In this case, the Q-boosting estimator is

$$\widehat{A}_{i,\tau}^{\text{boost}} = \bar{Q}_i(s_\tau,a_\tau) - \bar{V}_i(s_\tau) + \sum_{u=\tau}^{T-1}\gamma^{u-\tau}\big(r_i(s_u,a_u) + \gamma\bar{V}_i(s_{u+1}) - \bar{Q}_i(s_u,a_u)\big).$$

The initial $\bar{Q}_i(s_\tau,a_\tau)$ term cancels with the $-\bar{Q}_i(s_\tau,a_\tau)$ term in the residual at $u=\tau$, so

$$\widehat{A}_{i,\tau}^{\text{boost}} = \sum_{u=\tau}^{T-1}\gamma^{u-\tau}r_i(s_u,a_u) - \bar{V}_i(s_\tau) + \sum_{u=\tau}^{T-1}\gamma^{u-\tau+1}\bar{V}_i(s_{u+1}) - \sum_{u=\tau+1}^{T-1}\gamma^{u-\tau}\bar{Q}_i(s_u,a_u).$$

For any future nonterminal state s_u with $u > \tau$,

$$\mathbb{E}[\bar{Q}_i(s_u,a_u) \mid s_u] = \sum_{a\in\mathcal{A}(s_u)} \boldsymbol{\pi}(a \mid s_u)\cdot\bar{Q}_i(s_u,a) = \bar{V}_i(s_u).$$

Therefore, by the tower property, the future $\bar{Q}_i$ terms become $\bar{V}_i$ terms in conditional expectation and telescope:

$$\mathbb{E}[\widehat{A}_{i,\tau}^{\text{boost}} \mid s_\tau,a_\tau] = \mathbb{E}\Big[\sum_{u=\tau}^{T-1}\gamma^{u-\tau}r_i(s_u,a_u)\,\Big|\,s_\tau,a_\tau\Big] - \bar{V}_i(s_\tau) = Q_i^{\boldsymbol{\pi}}(s_\tau,a_\tau) - \bar{V}_i(s_\tau).$$

Using the tower property again,

$$\mathbb{E}[\widehat{A}_{i,\tau}^{\text{boost}}\cdot\nabla_{\theta_i}\log\pi_i(a_\tau \mid o_\tau;\theta_i) \mid s_\tau] = \mathbb{E}\big[\big(Q_i^{\boldsymbol{\pi}}(s_\tau,a_\tau) - \bar{V}_i(s_\tau)\big)\cdot\nabla_{\theta_i}\log\pi_i(a_\tau \mid o_\tau;\theta_i) \mid s_\tau\big].$$

The second term is a state-dependent baseline. Since $i(s_\tau) = i$, conditioning on s_τ fixes o_τ,

$$\begin{aligned}\mathbb{E}[\bar{V}_i(s_\tau)\cdot\nabla_{\theta_i}\log\pi_i(a_\tau \mid o_\tau;\theta_i) \mid s_\tau] &= \bar{V}_i(s_\tau)\sum_{a\in\mathcal{A}(s_\tau)}\pi_i(a \mid o_\tau;\theta_i)\cdot\nabla_{\theta_i}\log\pi_i(a \mid o_\tau;\theta_i)\\ &= \bar{V}_i(s_\tau)\cdot\nabla_{\theta_i}\sum_{a\in\mathcal{A}(s_\tau)}\pi_i(a \mid o_\tau;\theta_i)\\ &= 0.\end{aligned}$$

Thus

$$\mathbb{E}[\widehat{A}_{i,\tau}^{\text{boost}}\cdot\nabla_{\theta_i}\log\pi_i(a_\tau \mid o_\tau;\theta_i) \mid s_\tau] = \mathbb{E}[Q_i^{\boldsymbol{\pi}}(s_\tau,a_\tau)\cdot\nabla_{\theta_i}\log\pi_i(a_\tau \mid o_\tau;\theta_i) \mid s_\tau].$$

Similarly, $V_i^{\boldsymbol{\pi}}(s_\tau)$ is also a state-dependent baseline, so

$$\begin{aligned}\mathbb{E}[A_i^{\boldsymbol{\pi}}(s_\tau,a_\tau)\cdot\nabla_{\theta_i}\log\pi_i(a_\tau \mid o_\tau;\theta_i) \mid s_\tau] &= \mathbb{E}\big[\big(Q_i^{\boldsymbol{\pi}}(s_\tau,a_\tau) - V_i^{\boldsymbol{\pi}}(s_\tau)\big)\cdot\nabla_{\theta_i}\log\pi_i(a_\tau \mid o_\tau;\theta_i)\,\big|\,s_\tau\big]\\ &= \mathbb{E}\big[Q_i^{\boldsymbol{\pi}}(s_\tau,a_\tau)\cdot\nabla_{\theta_i}\log\pi_i(a_\tau \mid o_\tau;\theta_i)\,\big|\,s_\tau\big].\end{aligned}$$

Combining the two displays proves

$$\mathbb{E}[\widehat{A}_{i,\tau}^{\text{boost}}\cdot\nabla_{\theta_i}\log\pi_i(a_\tau \mid o_\tau;\theta_i) \mid s_\tau] = \mathbb{E}[A_i^{\boldsymbol{\pi}}(s_\tau,a_\tau)\cdot\nabla_{\theta_i}\log\pi_i(a_\tau \mid o_\tau;\theta_i) \mid s_\tau].$$

Mean-squared error of exact-V GAE. We next prove the variance-reduction statement. Fix trace parameter $\lambda \in (0, 1]$ and define $\rho \triangleq \lambda\gamma$. Consider GAE with the exact value critic $V_i(\cdot) = V_i^{\boldsymbol{\pi}}(\cdot)$. Its one-step TD residual is

$$\delta_{i,u} = r_i(s_u, a_u) + \gamma V_i^{\boldsymbol{\pi}}(s_{u+1}) - V_i^{\boldsymbol{\pi}}(s_u).$$

By the Bellman equation for $Q_i^{\boldsymbol{\pi}}$ and the deterministic-transition model,

$$Q_i^{\boldsymbol{\pi}}(s_u, a_u) = r_i(s_u, a_u) + \gamma V_i^{\boldsymbol{\pi}}(s_{u+1}).$$

Hence

$$\delta_{i,u} = Q_i^{\boldsymbol{\pi}}(s_u, a_u) - V_i^{\boldsymbol{\pi}}(s_u) = A_i^{\boldsymbol{\pi}}(s_u, a_u).$$

Therefore,

$$\widehat{A}_{i,\tau}^{\mathrm{GAE}} - A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau) = \sum_{u=\tau}^{T-1} \rho^{u-\tau} A_i^{\boldsymbol{\pi}}(s_u, a_u) - A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau) = \sum_{u=\tau+1}^{T-1} \rho^{u-\tau} A_i^{\boldsymbol{\pi}}(s_u, a_u).$$

Define the conditional future-action noise level

$$\Gamma_{i,\tau}^{\lambda} \triangleq \mathbb{E}\Big[\Big(\sum_{u=\tau+1}^{T-1} \rho^{u-\tau} A_i^{\boldsymbol{\pi}}(s_u, a_u)\Big)^2 \,\Big|\, s_\tau, a_\tau\Big].$$

Then

$$\Gamma_{i,\tau}^{\lambda} = \mathbb{E}\big[\big(\widehat{A}_{i,\tau}^{\mathrm{GAE}} - A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau)\big)^2 \,\big|\, s_\tau, a_\tau\big].$$

We now show that the non-value-equivalence condition implies $\Gamma_{i,\tau}^{\lambda} > 0$. We interpret this condition as follows: after (s_τ, a_τ), there exists a future timestep $u > \tau$ and a nonterminal state s reached with positive probability under $\boldsymbol{\pi}$ such that

$$\Pr_{\boldsymbol{\pi}}(s_u = s \,|\, s_\tau, a_\tau) > 0, \qquad \operatorname*{Var}_{a\sim\boldsymbol{\pi}(\cdot|s)}[Q_i^{\boldsymbol{\pi}}(s, a)] > 0.$$

Equivalently, two actions in the support of $\boldsymbol{\pi}(\cdot \mid s)$ have different $Q_i^{\boldsymbol{\pi}}$-values.

Let $\mathcal{F}_u$ denote the trajectory history up to state s_u, before sampling a_u. For every $u > \tau$,

$$\mathbb{E}[A_i^{\boldsymbol{\pi}}(s_u, a_u) \mid \mathcal{F}_u] = \sum_{a\in\mathcal{A}(s_u)} \boldsymbol{\pi}(a \mid s_u) \cdot \big(Q_i^{\boldsymbol{\pi}}(s_u, a) - V_i^{\boldsymbol{\pi}}(s_u)\big) = 0.$$

Thus the future advantage terms form a martingale-difference sequence. For $u < v$,

$$\mathbb{E}[A_i^{\boldsymbol{\pi}}(s_u, a_u) \cdot A_i^{\boldsymbol{\pi}}(s_v, a_v) \mid s_\tau, a_\tau] = \mathbb{E}\big[A_i^{\boldsymbol{\pi}}(s_u, a_u)\, \mathbb{E}[A_i^{\boldsymbol{\pi}}(s_v, a_v) \mid \mathcal{F}_v] \,\big|\, s_\tau, a_\tau\big] = 0.$$

Therefore the cross terms vanish, and

$$\Gamma_{i,\tau}^{\lambda} = \sum_{u=\tau+1}^{T-1} \rho^{2(u-\tau)}\, \mathbb{E}\big[\big(A_i^{\boldsymbol{\pi}}(s_u, a_u)\big)^2 \,\big|\, s_\tau, a_\tau\big].$$

At any state s satisfying the non-value-equivalence condition,

$$\mathbb{E}\big[\big(A_i^{\boldsymbol{\pi}}(s_u, a_u)\big)^2 \,\big|\, s_u = s\big] = \operatorname*{Var}_{a\sim\boldsymbol{\pi}(\cdot|s)}[Q_i^{\boldsymbol{\pi}}(s, a)] > 0.$$

Since such a state is reached with positive probability and $\rho = \lambda\gamma > 0$, at least one term in the above sum is strictly positive. Hence $\Gamma_{i,\tau}^{\lambda} > 0$.

Mean-squared error of Q-boosting. We now bound the error of the Q-boosting estimator with an approximate Q-critic. Define

$$e_Q(s, a) \triangleq \bar{Q}_i(s, a) - Q_i^{\boldsymbol{\pi}}(s, a), \qquad e_V(s) \triangleq \bar{V}_i(s) - V_i^{\boldsymbol{\pi}}(s).$$

Since $\bar{V}_i$ and $V_i^{\boldsymbol{\pi}}$ are policy averages of $\bar{Q}_i$ and $Q_i^{\boldsymbol{\pi}}$, respectively,

$$|e_V(s)| = \Big|\sum_{a\in\mathcal{A}(s)} \boldsymbol{\pi}(a \mid s) \cdot e_Q(s, a)\Big| \le \|\bar{Q}_i - Q_i^{\boldsymbol{\pi}}\|_\infty.$$

Let

$$\epsilon_Q \triangleq \|\bar{Q}_i - Q_i^{\boldsymbol{\pi}}\|_\infty.$$

Using the Bellman identity

$$Q_i^{\boldsymbol{\pi}}(s_u, a_u) = r_i(s_u, a_u) + \gamma V_i^{\boldsymbol{\pi}}(s_{u+1}),$$

the Expected-SARSA residual in Q-boosting satisfies

$$\delta_{i,u}^{+} = r_i(s_u, a_u) + \gamma \bar{V}_i(s_{u+1}) - \bar{Q}_i(s_u, a_u) = \gamma e_V(s_{u+1}) - e_Q(s_u, a_u).$$

Subtracting the true advantage from the Q-boosting estimator gives

$$\widehat{A}_{i,\tau}^{\text{boost}} - A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau) = e_Q(s_\tau, a_\tau) - e_V(s_\tau) + \sum_{u=\tau}^{T-1} \rho^{u-\tau}\big(\gamma e_V(s_{u+1}) - e_Q(s_u, a_u)\big).$$

The $e_Q(s_\tau, a_\tau)$ term cancels with the $-e_Q(s_\tau, a_\tau)$ term inside the residual at $u = \tau$, so

$$\widehat{A}_{i,\tau}^{\text{boost}} - A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau) = -e_V(s_\tau) + \gamma e_V(s_{\tau+1}) + \sum_{u=\tau+1}^{T-1} \rho^{u-\tau}\big(\gamma e_V(s_{u+1}) - e_Q(s_u, a_u)\big).$$

Let $L_\tau \triangleq T - \tau$ be the remaining number of action steps. Using $|e_Q(s, a)| \le \epsilon_Q$ and $|e_V(s)| \le \epsilon_Q$, we obtain the pathwise bound

$$\big|\widehat{A}_{i,\tau}^{\text{boost}} - A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau)\big| \le (1+\gamma)\epsilon_Q + \sum_{u=\tau+1}^{T-1} \rho^{u-\tau}(1+\gamma)\epsilon_Q = (1+\gamma)\epsilon_Q \sum_{k=0}^{L_\tau - 1} \rho^k.$$

Consequently,

$$\mathbb{E}\big[\big(\widehat{A}_{i,\tau}^{\text{boost}} - A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau)\big)^2 \,\big|\, s_\tau, a_\tau\big] \le (1+\gamma)^2 \epsilon_Q^2 \cdot \left(\sum_{k=0}^{L_\tau - 1} \rho^k\right)^2.$$

Since $\Gamma_{i,\tau}^{\lambda} > 0$, define

$$\xi_{i,\tau}^{\lambda} \triangleq \frac{\sqrt{\Gamma_{i,\tau}^{\lambda}}}{(1+\gamma)\sum_{k=0}^{L_\tau - 1} (\lambda\gamma)^k}.$$

This quantity is strictly positive. If

$$\|Q_i(\cdot; \phi_i) - Q_i^{\boldsymbol{\pi}}\|_\infty = \epsilon_Q < \xi_{i,\tau}^{\lambda},$$

then

$$\mathbb{E}\big[\big(\widehat{A}_{i,\tau}^{\text{boost}} - A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau)\big)^2 \,\big|\, s_\tau, a_\tau\big] < \Gamma_{i,\tau}^{\lambda} = \mathbb{E}\big[\big(\widehat{A}_{i,\tau}^{\text{GAE}} - A_i^{\boldsymbol{\pi}}(s_\tau, a_\tau)\big)^2 \,\big|\, s_\tau, a_\tau\big].$$

Thus, for any $\lambda \in (0, 1]$, a sufficiently accurate Q-critic yields strictly smaller conditional mean-squared advantage-estimation error than exact-V GAE whenever the reachable future actions are not all $Q_i^{\boldsymbol{\pi}}$-value-equivalent. The theorem follows. □

Remarks. We remark on the role of the trace parameter. For $\lambda = 1$, on-policy Q-boosting gives an unbiased score-function term for any fixed Q-critic. With an exact Q-critic, Q-boosting equals the exact advantage for every λ, since every Expected SARSA residual vanishes pathwise. With an approximate critic and $\lambda < 1$, bias may remain, and the theorem gives the lower conditional mean-squared error for every $\lambda \in (0, 1]$.

The radius $\xi_{i,\tau}^{\lambda}$ is sufficient but not necessary, and it scales with the future-action noise that Q-boosting removes. In the matching-pennies example of Figure 1, we have $\gamma = \lambda = 1$, $\Gamma_{i,\tau}^{\lambda} = 1$, and $L_\tau = 2$, which gives $\xi_{i,\tau}^{\lambda} = 1/4$ in reward units.

D Games Description

Abrupt Phantom Tic-Tac-Toe (APTTT). Phantom Tic-Tac-Toe is an imperfect-information version of standard Tic-Tac-Toe on a 3×3 grid: players alternately attempt to place their mark, aiming to achieve three in a row horizontally, vertically, or diagonally. If the grid fills without a winning line, the game is a draw. In Phantom Tic-Tac-Toe, imperfect information arises because the opponent's marks are hidden, players do not directly observe the opponent's moves, and a player only learns whether their own attempted placement succeeds or fails. The *abrupt* variant differs from the classical ruleset in how failed placements are handled: when a player attempts to place a mark on a square occupied by the opponent, the attempt fails and the player *loses their turn* (in the classical ruleset, the player must choose a different square) [11, 29].

Abrupt Dark Hex. Abrupt Dark Hex is an imperfect-information variant of Hex. As in Hex, the first player aims to connect the top and bottom edges, while the second player aims to connect the left and right edges; draws are impossible (Hex theorem). The game becomes imperfect-information because actions are *hidden*: each player observes only their own stones, and the opponent's placements are not directly visible. If a player attempts to place a stone on a cell already occupied by the opponent, the move fails. In the *abrupt* ruleset, the player *loses their turn* (instead of retrying a different cell, as in the classical ruleset) [11, 29]. We use **Abrupt Dark Hex 3 (ADH3)**, the variant played on a 3×3 board. This is the largest size for which we can evaluate exact policy exploitability within a reasonable computational budget.

Liar's Dice. In Liar's Dice, each of two players privately rolls dice. Players then alternate making bids of the form "(q, f)," claiming that across both players there are at least q dice showing face f. Under the OpenSpiel ruleset, a legal bid must either increase the quantity q, or increase the face value f without decreasing q. Instead of bidding, a player may call *liar*, which ends the round by revealing all dice: if the previous bid does not exceed the actual count of face f, the bidder wins; otherwise, the caller wins. Liar's Dice (including closely related parameterizations) is a standard benchmark for imperfect-information games in prior equilibrium-finding and RL+search work [4, 29]. We use **Liar's Dice with 2 Dice of 5 Faces (LiDi 2D5F)**, a variant in which both players roll two dice with five faces each. This is the largest size for which we can compute exact exploitability within the available computational budget.

Dou Dizhu. Dou Dizhu (*Fighting the Landlord*) is a three-player imperfect-information shedding game played with a 54-card poker deck, between one landlord and two peasants. Players take turns playing legal card combinations (e.g., singles, pairs, triples, sequences) that must beat the previous play or pass. The goal is to be the first to empty one's hand. The landlord wins if they go out first, while the peasants win if *either* peasant empties their hand first. Rewards are computed using a base score of 2 for the landlord and 1 for each peasant if they win, with additional bonuses based on the number of special combinations played: *bombs* (four of a kind) and *rocket* (the two jokers). Dou Dizhu is widely used as a benchmark for self-play RL and imperfect-information decision-making, including end-to-end systems such as DouZero [7] and toolkits/environments such as RLCard [27].

Heads-Up No-Limit Texas Hold'em. Heads-Up No-Limit Texas hold'em is a two-player imperfect-information betting game. Each hand begins with forced blinds and a shuffle deal: each player receives two private *hole cards*, and up to five *community cards* are revealed over four betting rounds (preflop, flop, turn, river). In each round, players act sequentially and may fold, call/check, or bet/raise any amount up to their remaining stack (the *no-limit* betting structure). The hand ends either when a player folds, awarding the current pot to the opponent, or at showdown after the final betting round, where private cards are revealed and the stronger five-card poker hand wins the pot. Imperfect information arises from the hidden hole cards and stochastic dealing, making HUNL a standard benchmark for equilibrium-finding and decision-making under uncertainty in poker agent [2, 29, 32]. We use **HUNL with a 200 big-blind initial stack (HUNL200)**, where each player starts each hand with a stack of 200 big blinds of 20,000 chips, with small blind and big blind equal to 50 chips and 100 chips respectively. Specifically, to prevent excessively long trajectories in the RL environment, the fifth raise within a betting street is forced to be an all-in.

E Training Recipe

RL Environment Overview. The RL environment provides, at each timestep:

- Per-player tokens describing the differences from the previous step. Notably, the full sequence of tokens over timesteps should contain all information observed by the player. For APTTT or ADH3, this encodes the last move made by the player, as well as whether the placement was successful. For Dou Dizhu games, this encodes the most recently drawn card during the card drawing phase, or the selected card combinations of the active player.
- Per-player binary feature channels representing the current view of the board. These channels do not encode the full board state. Although the sequence of channels across timesteps contains the complete information, we only use the current timestep's channels for efficiency. For APTTT or ADH3, this indicates whether each square has been played successfully, unsuccessfully, or remains untouched. For Dou Dizhu games, this encodes the player's current hand as well as the cards discarded by each player.
- Candidate action sets for the active player, each encoded as a variable-length token sequence describing a possible move. We use a similar format as the observation tokens. For complex decision spaces, such as in Dou Dizhu (which may have more than a thousand valid moves), actions are decomposed over multiple timesteps to reduce the candidate set size. For APTTT or ADH3, this consists of all unplayed squares encoded by their positions. For Dou Dizhu, each action is represented as a *prefix* of a valid card combination, breaking the decision into multiple steps.

This design is based on the observation that changes between timesteps are relatively small compared to the full game state. Therefore, we avoid encoding the entire game state at each step and instead encode only the changes, supplemented by a current board description to help the model reconstruct the full state. Ideally, this design can be seen as a backtrace of the game state. We use token-based descriptions since the changes are often sparse, making binary channels inefficient.

Training Hyperparameters. We use the Muon optimizer [39] with Nesterov momentum $\beta = 0.95$ and a base learning rate of $\eta_{\text{base}} = 4 \times 10^{-4}$. We apply fixed weight decay 0.01 to all projection matrices from the Transformer decoder and the MLPs, where RMSNorm is attached to the output of each layer, making the architecture homogeneous.

The algorithm performs $K_{\text{actor}} = 4$ actor epochs and $K_{\text{critic}} = 4$ critic epochs per iteration, each consisting of $M = 4$ minibatches. It employs a base PPO clipping coefficient of $\varepsilon_{\text{base}} = 0.02$ and a base regularization coefficient of $\alpha_{\text{base}} = 0.1$. The rollout batch size is denoted as $|\mathcal{D}_{\text{rollout}}| = B$, and the replay buffer has a capacity of $|\mathcal{D}_{\text{replay}}| = 64 \cdot B$, sufficient to store all trajectories from the past 64 iterations. We present an ablation study of VRPO-specific hyperparameters in Appendix L. The discount factor of all games is fixed at $\gamma = 1$, and the Q-boosting trace parameter is set to $\lambda = 0.95$.

In addition to the total number of iterations T_{total}, the learning rate and regularization schedules are controlled by two decay coefficients: the stable learning rate iteration count T_η and the stable regularization coefficient iteration count T_α. The schedule hyperparameters are game-specific and listed in Table 3.

Game	Iteration T_{total}	Iteration T_η	Iteration T_α	Batch Size B
APTTT	2000	500	500	2048
ADH3	2000	500	500	2048
LiDi 2D5F	2000	500	500	2048
Dou Dizhu	40000	5000	500	2048
HUNL200	40000	5000	500	8192

Table 3: Schedule hyperparameters by game.

At iteration T, the actor learning rate, critic learning rate, and PPO clip coefficient are decayed according to:

$$\eta_{\text{actor}} = \eta_{\text{base}} \cdot \min\left\{1, \frac{T_\eta}{T}\right\}, \qquad \eta_{\text{critic}} = \eta_{\text{base}} \cdot \min\left\{1, \frac{T_\eta}{T}\right\}^{0.5}, \qquad \varepsilon = \varepsilon_{\text{base}} \cdot \min\left\{1, \frac{T_\eta}{T}\right\}.$$

The regularization coefficient decays according to:

$$\alpha = \alpha_{\text{base}} \cdot \min\left\{1, \frac{T_\alpha}{T}\right\}^{0.5}.$$

For Dou Dizhu and HUNL200, both of which require many training iterations, we follow Sokota et al. [6] by using an exponential moving average of the policy with $\beta = 0.999$ as the evaluation policy, effectively merging the policies from the final several thousand iterations. This model-merged policy is not used for training rollouts; it is used only for evaluation. This averaging smooths out the noise introduced by rollout batches at individual gradient steps.

Model Architecture. We use the same model architecture across all games and algorithms. At each timestep, observation tokens are embedded into a 128-dimensional space, summed over all available tokens, and combined with a learned player-identity embedding. The resulting temporal sequence is processed by a small LLaMA-style causal (decoder-only) Transformer [40] with 4 layers, 4 attention heads, and a feedforward expansion ratio of 2. The decoder outputs are then concatenated with the channel features and passed through a 3-layer MLP with hidden dimension 256, equipped with RMSNorm, to produce a per-player state feature.

Each candidate action is encoded independently by summing its token embeddings (each of dimension 256) and applying an action MLP, similar to the state MLP but with only 2 layer. Action scores are computed by summing the elementwise product between the 256-dimensional state feature and each action feature, followed by a learned linear projection. The actor architecture is shown in Figure 5.

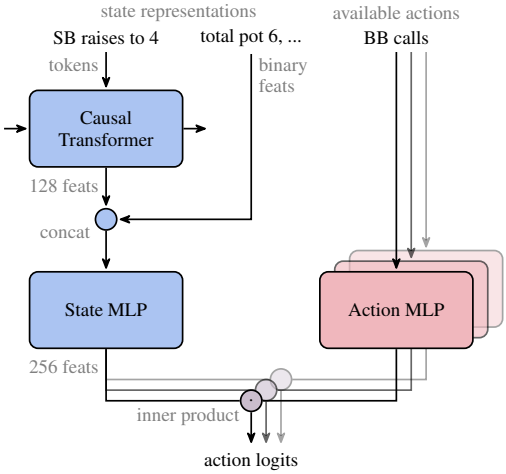

Figure 5: Actor architecture. The observation history is encoded into features by a decoder-only Transformer, combined with binary channel features, and scored against independently encoded action features via inner product to produce action logits. The critic is built on the actor architecture, with an additional residual MLP after the state MLP to fuse state features from all players. The architecture is shared across all games; see Appendix E for details.

The critic mirrors the per-player history/state encoder for all players (with separate parameters), fuses the concatenated per-player state features using a 2 layer residual MLP mixer of dimension 256, and outputs per-player values for each candidate action using the same state-action interaction and a linear output head. In models where only a value head is required, the elementwise product is replaced by the state feature alone.

F Evaluation Details

F.1 Abrupt Phantom Tic-Tac-Toe and Abrupt Dark Hex 3

We use the `exp-a-spiel` library [11], which employs a traverser to compute the exact exploitability from the tabular strategy. The tabular strategy is generated by enumerating all information sets and evaluating the actor network to obtain the policy distribution from model logits at temperature $\tau = 1$.

F.2 Liar's Dice

We implement a traverser routine to compute the exact exploitability from the tabular strategy. This routine is cross-verified against the `OpenSpiel` library [29] on smaller Liar's Dice games. The tabular strategy is generated by enumerating all information sets and evaluating the actor network to obtain the policy distribution from model logits at temperature $\tau = 1$.

F.3 Dou Dizhu

We integrate our model with the evaluation routine of PerfectDou [8] by converting their encoding to a semantic string representation that describes the information set. This is then converted into the game encoding presented in Appendix D. The semantic string encoding creates an information barrier that prevents potential information leakage. We compute our policy by evaluating the actor network to obtain the policy distribution from model logits at temperature $\tau = 0$.

F.4 No Limit Texas Hold'em

We evaluate the performance by querying the Slumbot API.[2] We compute our policy by evaluating the actor network to obtain the policy distribution from model logits at temperature $\tau = 0.1$.

We remark that the external regularization term $\mathcal{L}^{\text{reg}}$ introduces additional exploration. For example, the training process must handle all-in opens, even though these actions are far from equilibrium play. Reducing the inference temperature is the simplest way to eliminate such actions from the agent's behavior. Clean-up using human knowledge [5] or test-time search [6] can refine this step further.

[2] https://slumbot.com/

G Game Encoding

In this section, we present the game encoding used in the RL environment. An overview of the hyperparameters is shown in Table 4, where **Tokens** refers to the number of tokens in the dictionary, including the special padding token `Token 0`; **Channels** refers to the number of binary channels used to encode the current timestep; **Actions** refers to the maximum number of available actions in a single timestep; **Horizon** represents the maximum possible environment steps according to the encoding; **O tokens** indicates the maximum number of observation tokens shown in a single timestep; and **A tokens** denotes the maximum number of tokens used in an action.

Game	Tokens	Channels	Horizon	Actions	O Tokens	A Tokens
APTTT	10	29	18	9	3	2
ADH3	10	29	18	9	3	2
LiDi 2D5F	16	23	13	11	4	4
Dou Dizhu	91	288	177	35	5	5
HUNL200	77	297	100	16	8	8

Table 4: Game encoding parameters.

Importantly, for games where the number of available actions is large in the original game logic, such as Dou Dizhu or HUNL, we divide a single decision into multiple environment steps to accommodate the maximum number of available actions within the game horizon.

G.1 Abrupt Phantom Tic-Tac-Toe and Abrupt Dark Hex 3

Token Dictionary.

- `Token 1`: Initial token
- `Token 2-4`: Row token $0 \dots 2$
- `Token 5-7`: Column token $0 \dots 2$
- `Token 8-9`: Player token $0 \dots 1$

Binary Channels.

- `Channel 0-8`: Unobserved cells: $0 \dots 8$
- `Channel 9-17`: Cells $0 \dots 8$ occupied by player 0
- `Channel 18-26`: Cells $0 \dots 8$ occupied by player 1
- `Channel 27-28`: Turn indicator for player $0 \dots 1$

Observation Tokens. The game starts with an initial token (`Token 1`). Then, after each move made by the player, they observe three tokens: a row token `r`, a column token `c`, and a player token `i`. These tokens indicate that playing at the corresponding cell was successful if the player token matches the current player, or that the cell was already occupied by the other player if the player token belongs to them. After each opponent's move, the player does not observe anything.

Action Tokens. Each available action corresponds to a placement at an untried cell and is encoded by the corresponding row token `r` and column token `c`.

Rewards. The winning agent is awarded $+1$, the losing agent is awarded -1, and both receive 0 in the case of a draw.

G.2 Liar's Dice with 2 Dice of 5 Faces

Token Dictionary.

- `Token 1`: Initial token

- `Token 2-3`: Player token $0 \ldots 1$
- `Token 4`: Bid token
- `Token 5`: Call token
- `Token 6-10`: Face tokens for $1 \ldots 5$
- `Token 11-15`: Quantity tokens for $0 \ldots 4$

Binary Channels.

- `Channel 0-10`: Last bid: $1 \ldots 4$ of face $1 \ldots 5$, or no bid yet
- `Channel 11-15`: Private die 0 shows face $1 \ldots 5$
- `Channel 16-20`: Private die 1 shows face $1 \ldots 5$
- `Channel 21-22`: Turn indicator for player $0 \ldots 1$

Observation Tokens. The game starts with an initial token (`Token 1`). For each dice roll, the player observes two tokens: a face token `f` and a player token `i`, indicating the result of each die. After any player's bid, the player observes four tokens: a player token `i`, the bid token (`Token 4`), a quantity token `c`, and a face token `f`. These tokens indicate that the corresponding player is making a bid for a certain quantity of the specified face.

Action Tokens. Each bidding action corresponds to four tokens: a player token `i`, the bid token (`Token 4`), a quantity token `c`, and a face token `f`, which exactly match the observation tokens. The call action consists of two tokens: a player token `i` and the call token (`Token 5`).

Rewards. The winning agent is awarded $+1$, the losing agent is awarded -1.

G.3 Dou Dizhu

Token Dictionary.

- `Token 1`: Initial token
- `Token 2-4`: Player tokens – Landlord, Peasant 1, Peasant 2
- `Token 5-8`: Discard Class – Solo, Pair, Trio, or Quad
- `Token 9-11`: Discard Class – Straight, Pair Chain (Pair Sisters), or Trio Chain
- `Token 12`: Discard Class – Rocket
- `Token 13-27`: Rank tokens – 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A, 2, B, R
- `Token 28-38`: Sequence starting point – 3, 4, 5, 6, 7, 8, 9, T, J, Q, K
- `Token 40-50`: Sequence end point – 4, 5, 6, 7, 8, 9, T, J, Q, K, A
- `Token 51`: No kickers
- `Token 52-56`: Solo kickers with $1 \ldots 5$ cards
- `Token 57-60`: Pair kickers with $1 \ldots 4$ cards
- `Token 89`: Pass token
- `Token 90`: Hold token[3]

Binary Channels.

- `Channel 0-53`: Self-remaining cards for each card
- `Channel 54-107`: Landlord public cards for each card
- `Channel 108-161`: Landlord discarded cards for each card
- `Channel 162-215`: Peasant 1 discarded cards for each card

[3]Remark: Some token indices are left empty due to internal backward compatibility.

- `Channel 216-269`: Peasant 2 discarded cards for each card
- `Channel 270-284`: Prefix-filling bomb count
- `Channel 285-287`: Turn indicators for Landlord, Peasant 1, and Peasant 2

Observation Tokens. The game starts with an initial token (`Token 1`). In the next 17 timesteps, each of the 17 private cards is revealed to each player privately using three tokens: a player token `i`, the hold token (`Token 90`), and a rank token r. In the following 3 timesteps, the three Landlord public cards from the remaining of the deck are announced to all players using three tokens: the Landlord token (`Token 2`), the hold token (`Token 90`), and a rank token r.

During gameplay, the card combinations are encoded across multiple timesteps, depending on the complexity of the card combination.

- **Solo/Pair** combinations are encoded in one timestep with three tokens: a player token i, the Solo token (`Token 5`) or the Pair token (`Token 6`), and the rank token r.
- **Trio/Quad** combinations are encoded in multiple timesteps. The first timestep contains three tokens: a player token i, the Trio token (`Token 7`) or the Quad token (`Token 8`), and the major rank token r. In the second timestep, the kicker details are specified, and the token count increases to four by including the kicker information. This can be no kicker (`Token 51`), one solo kicker (`Token 52`) or one pair kicker (`Token 57`) for a trio, or two solo kickers (`Token 53`) or two pair kickers (`Token 58`) for a quad. The following timesteps encode each kicker as if they were discarded individually as a solo or pair, in increasing order.
- **Straight/Pair Chain** combinations are encoded in two timesteps. The first timestep contains three tokens: a player token i, the Straight token (`Token 9`) or the Pair Chain token (`Token 10`), and the sequence starting point token r_1. In the next timestep, the end point of the sequence is determined, and four tokens are presented, including the sequence end point token r_2.
- **Trio Chain** combinations are encoded in multiple timesteps. The first timestep contains three tokens: a player token i, the Trio Chain token (`Token 11`), and the sequence starting point token r_1. In the second timestep, both the kicker information and the sequence end point are specified together. For a Trio Chain with kickers (airplane with wings), the token sequence contains five tokens, including the sequence end point token r_2 and the kicker information. The following timesteps encode each kicker as if it were discarded individually as a solo or pair, in increasing order.
- **Rocket** is encoded in one timestep with two tokens: a player token i with Rocket token (`Token 12`).
- **PASS** is encoded in one timestep with two tokens: a player token i and the pass token (`Token 89`).

Under this multi-timestep decision encoding, the maximum number of actions required in one timestep is significantly reduced to 35 actions, compared to 27,472 in DouZero [7]. Notably, the channels are updated as if the kickers were treated as continuous actions by the player: a Trio Chain with three pair kickers is treated as first discarding a Trio Chain with kickers, followed by three additional timesteps for discarding each kicker, which are limited to pairs.

Action Tokens. The action tokens are identical to the observation tokens as the actions are public.

Rewards. Let B be the total number of quads and rockets played during gameplay. The Landlord is awarded $+2 \times 2^B$ if they win, and -2×2^B if they lose. Each Peasant is awarded $+1 \times 2^B$ if they win, and -1×2^B if they lose.

G.4 Heads-Up No-Limit Texas Hold'em

Token Dictionary.

- `Token 1`: Initial token
- `Token 2-14`: Rank tokens – 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A
- `Token 15-18`: Suit tokens – c, d, h, s
- `Token 19-21`: Action tokens – Fold, Call, Raise

- `Token 22-25`: Street tokens – Preflop, Flop, Turn, River
- `Token 26-27`: Player tokens – Small Blind, Big Blind
- `Token 28-75`: Amount tokens – Raise to $i \times 16^j \times 10$ chips (0.1 big blinds)
- `Token 76`: Raise all-in

The raise tokens encode 3 base-16 digits, specifically following amounts: 0, 10, 20, 30, ..., 150, 160, 320, 480, ..., 2560, 5120, 7680, ..., 40960 chips.

Binary Channels.

- `Channel 0-33`: 2×17 channels for the two hole cards: suits c, d, h, s then ranks 2..A
- `Channel 34-118`: 5×17 channels for the five board cards, same as private hands
- `Channel 119-170`: self hole cards, 52-card multi-hot
- `Channel 171-222`: visible board cards, 52-card multi-hot
- `Channel 223-246`: Total pot, 6 base-4 digits
- `Channel 247-270`: Current player's bet on this street, 6 base-4 digits
- `Channel 271-294`: Opponent's player's bet on this street, 6 base-4 digits
- `Channel 295-296`: Turn indicator for Small Blind and Big Blind

Observation Tokens. In the encoding, each card dealt is encoded over two timesteps: the rank token is presented in the first timestep, while the suit token is presented in the second timestep. This reduces the number of branches from 52 cards to 13 ranks, as we treat the board dealer as a special player for deterministic transitions.

The game starts with an initial token (`Token 1`). The next 2×2 timesteps reveal the two hole cards to each player *simultaneously*. After each betting round, the corresponding board cards are revealed, with each card taking two timesteps.

For a board card, the first timestep contains one token, the rank token r, while the second timestep contains two tokens: the rank token r and the suit token s. For a private hole card, the player token i is also attached to the encoding, so the number of tokens becomes two and three respectively.

In the betting rounds, actions are encoded over multiple timesteps depending on the complexity of the action. For a raise action, the sequence is encoded using three timesteps, each timestep decides one digit of the base-16 numbers. In an all-in raise, the raise sequence is replaced by the all-in token.

The other actions, Call/Check and Fold, are encoded using three tokens in one timestep, containing a street token, a player token i and either the Call token (`Token 20`) or the Fold token (`Token 19`).

The environment faithfully represents all trajectories, since actions are never aggregated. The only exception is that, within each betting street, the fifth raise is constrained to be an all-in raise to avoid excessively long training episodes.

Action Tokens. The action tokens are identical to the observation tokens as the actions are public.

Rewards. The players are awarded a chip change measured in big blinds. Since there are $20{,}000$ chips in the stack, the maximum reward is 200.

H Training Dynamics

In this section, we present the training dynamics of VRPO on Dou Dizhu and HUNL200, using the default hyperparameters from the training recipe in Appendix E. The solid curve represents an exponential moving average of the metric over training iterations, while the faint shaded trace corresponds to the raw, unsmoothed values.

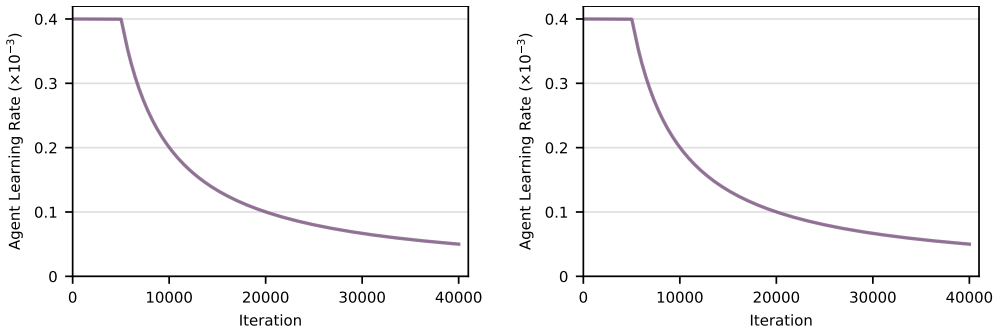

Figure 6: Actor learning rate schedule η_{actor} (Left: Dou Dizhu; Right: HUNL200).

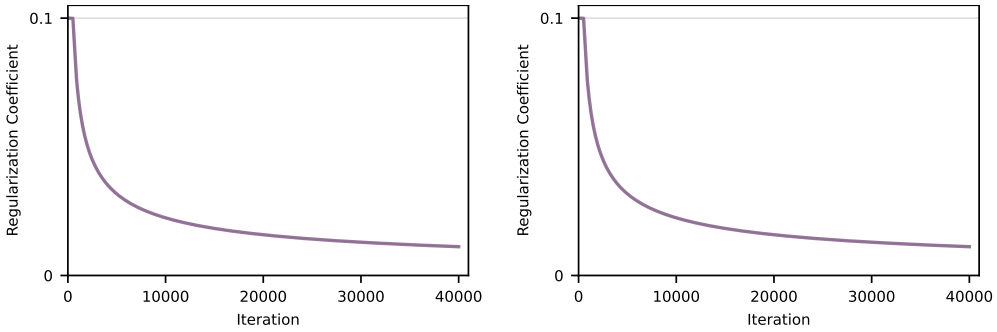

Figure 7: Regularization coefficient schedule α (Left: Dou Dizhu; Right: HUNL200).

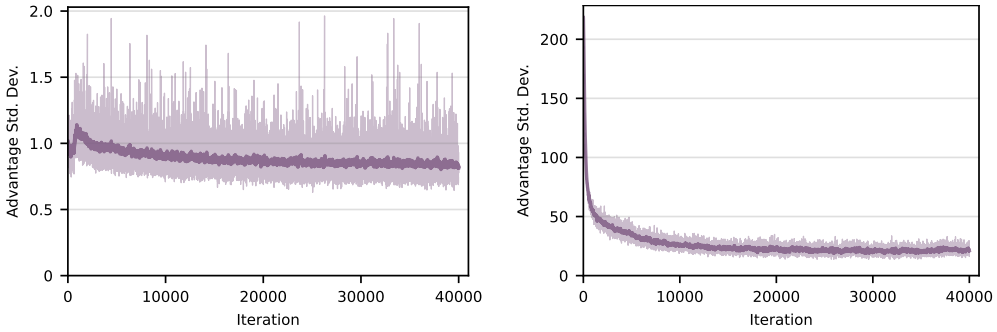

Figure 8: Standard deviation of the estimated advantage $\widehat{A}$ (Left: Dou Dizhu; Right: HUNL200).

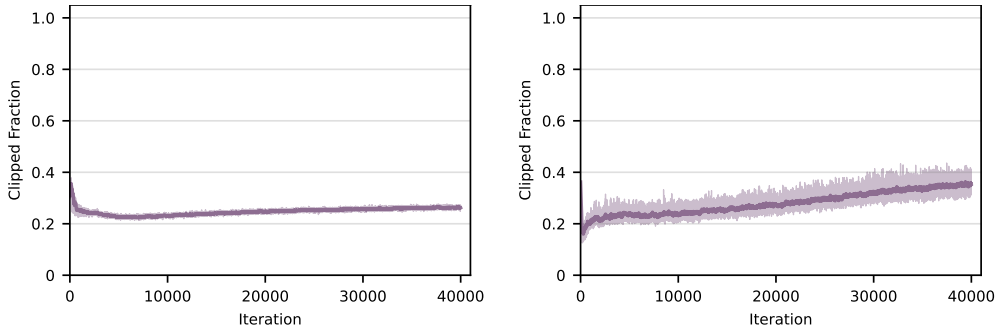

Figure 9: Clipped fraction under the PPO threshold (Left: Dou Dizhu; Right: HUNL200).

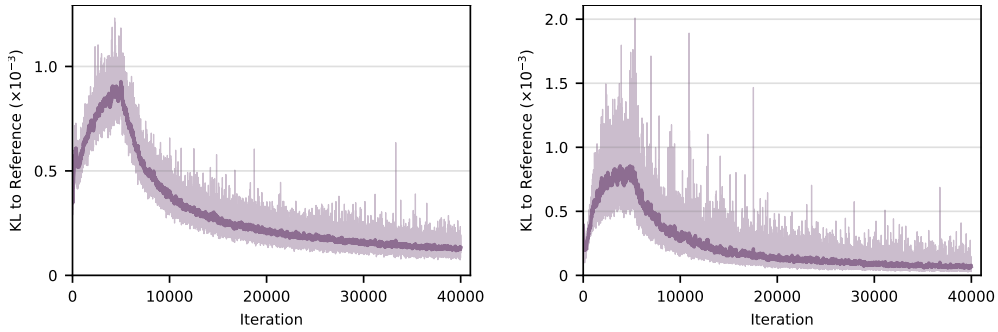

Figure 10: KL divergence to the reference policy $\mathrm{KL}(\boldsymbol{\pi} \,\|\, \boldsymbol{\pi}^{\mathrm{ref}})$ (Left: Dou Dizhu; Right: HUNL200).

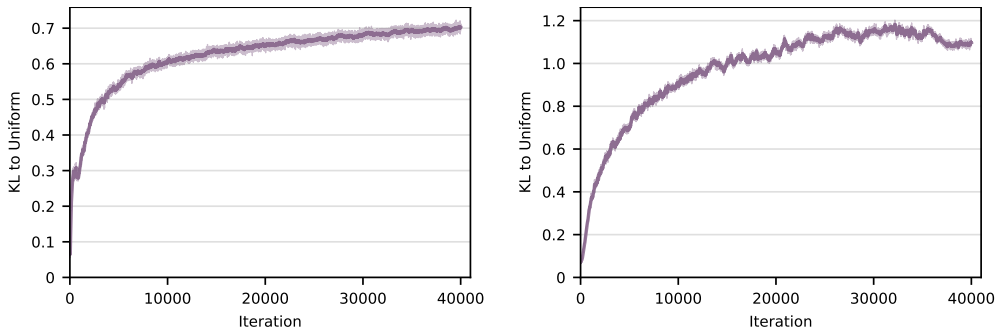

Figure 11: KL divergence to the uniform policy $\mathrm{KL}(\boldsymbol{\pi} \,\|\, \mathrm{Unif})$ (Left: Dou Dizhu; Right: HUNL200).

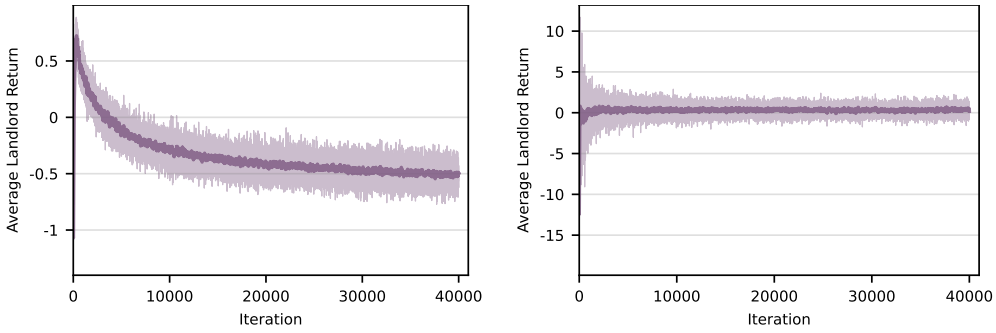

Figure 12: Average return of the first player under policy π (Left: Dou Dizhu; Right: HUNL200).

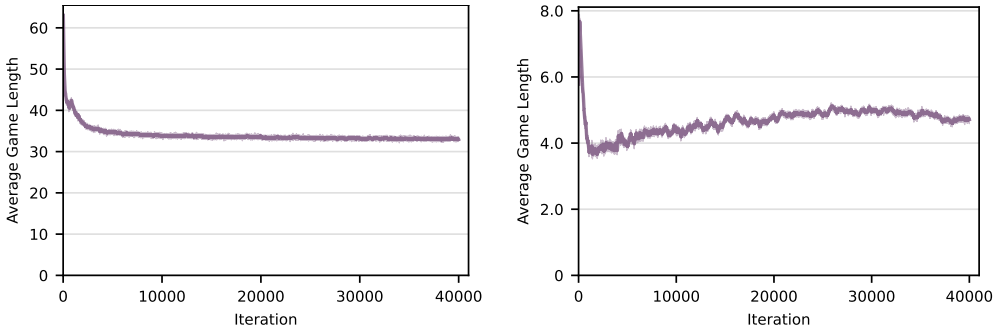

Figure 13: Average gameplay length under the current policy π (Left: Dou Dizhu; Right: HUNL200).

I Training Time

Table 5 reports the measured per-iteration training time of VRPO and MAPPO. Since VRPO replaces the centralized V-critic with a centralized Q-critic and only changes its output layer, the overhead is 3.6–9.2% per iteration. The critic is discarded at execution, so inference cost matches MAPPO.

	APTTT	ADH3	LiDi 2D5F	Dou Dizhu	HUNL200
MAPPO (s/iter)	1.32	1.31	1.01	4.65	5.19
VRPO (s/iter)	1.37	1.36	1.08	4.95	5.67
Overhead	3.6%	3.7%	6.5%	6.5%	9.2%

Table 5: Per-iteration training time. Overheads are computed from unrounded timings.

J More Experiment Results on Dou Dizhu

Peasants \ Landlord	**VRPO**	PerfectDou	DouZero
VRPO (ours)	0.707 ± 0.011	0.811 ± 0.011	0.877 ± 0.011
PerfectDou	0.507 ± 0.011	0.601 ± 0.011	0.656 ± 0.011
DouZero	0.304 ± 0.011	0.373 ± 0.011	0.453 ± 0.011

Table 6: Average total team gain of the row agent playing as two independent Peasants against the column playing agent as the Landlord.

K Ablation Study on General Hyperparameters

We study the effect of several PPO-style optimization hyperparameters on exact exploitability. For each sweep, we report the mean exact exploitability over random seeds, with uncertainty shown as $1.0 \times \mathrm{SEM}$. Lower values indicate stronger policies. We bold the best mean within each game block.

K.1 Trace Parameter

We first study the trace parameter λ used in the multi-step advantage estimation. We sweep the parameter over $\lambda \in \{0, 0.9, 0.95, 1.0\}$ and report the results in Table 7. Lower values reduce variance by using shorter traces, while larger values incorporate longer-horizon return information. VRPO remains the best-performing algorithm across the sweep.

ADH3					
λ	**VRPO**	MAPPO	IPPO	MMD	PPG
0	0.080 ± 0.002	0.106 ± 0.011	0.180 ± 0.014	0.180 ± 0.006	0.154 ± 0.009
0.9	$\mathbf{0.070 \pm 0.001}$	0.124 ± 0.014	0.128 ± 0.004	0.132 ± 0.011	0.120 ± 0.002
0.95	0.079 ± 0.004	0.125 ± 0.010	0.138 ± 0.012	0.139 ± 0.006	0.124 ± 0.004
1	0.078 ± 0.004	0.116 ± 0.008	0.139 ± 0.002	0.144 ± 0.003	0.137 ± 0.005
APTTT					
λ	**VRPO**	MAPPO	IPPO	MMD	PPG
0	$\mathbf{0.090 \pm 0.006}$	0.106 ± 0.002	0.220 ± 0.013	0.225 ± 0.014	0.197 ± 0.016
0.9	0.094 ± 0.004	0.117 ± 0.003	0.126 ± 0.005	0.117 ± 0.001	0.126 ± 0.003
0.95	0.090 ± 0.002	0.117 ± 0.002	0.120 ± 0.002	0.120 ± 0.001	0.126 ± 0.004
1	0.091 ± 0.004	0.116 ± 0.003	0.126 ± 0.005	0.128 ± 0.002	0.129 ± 0.004
LiDi 2D5F					
λ	**VRPO**	MAPPO	IPPO	MMD	PPG
0	0.092 ± 0.002	0.104 ± 0.002	0.119 ± 0.006	0.141 ± 0.006	0.120 ± 0.006
0.9	0.087 ± 0.002	0.116 ± 0.002	0.108 ± 0.004	0.110 ± 0.005	0.111 ± 0.002
0.95	0.092 ± 0.002	0.117 ± 0.003	0.111 ± 0.002	0.115 ± 0.002	0.116 ± 0.002
1	$\mathbf{0.084 \pm 0.001}$	0.121 ± 0.003	0.112 ± 0.002	0.115 ± 0.001	0.117 ± 0.002

Table 7: Exact exploitability (lower is better) under the trace parameter λ sweep. Each entry reports mean $\pm$ SEM over seeds. Bold indicates the best mean in each game block.

K.2 Regularization Coefficient

We next vary the base regularization coefficient α_{base} over $\alpha_{\text{base}} \in \{0.02, 0.05, 0.1, 0.2\}$, with results shown in Table 8. This coefficient controls the strength of the regularization term in the policy update, trading off conservative updates against the ability to move toward improved policies.

The best VRPO setting is consistent across all three games: 0.1 achieves the lowest exploitability. Both smaller and larger values are less reliable, indicating that this regularization term requires a moderate setting to balance optimization stability and policy improvement.

ADH3					
α_{base}	**VRPO**	MAPPO	IPPO	MMD	PPG
0.02	0.232 ± 0.014	0.134 ± 0.008	0.141 ± 0.007	0.185 ± 0.019	0.152 ± 0.007
0.05	0.102 ± 0.005	0.095 ± 0.003	0.110 ± 0.007	0.115 ± 0.007	0.110 ± 0.010
0.1	$\mathbf{0.086 \pm 0.005}$	0.115 ± 0.006	0.126 ± 0.007	0.146 ± 0.004	0.136 ± 0.010
0.2	0.107 ± 0.005	0.203 ± 0.002	0.226 ± 0.002	0.219 ± 0.003	0.224 ± 0.007
APTTT					
α_{base}	**VRPO**	MAPPO	IPPO	MMD	PPG
0.02	0.325 ± 0.022	0.173 ± 0.009	0.148 ± 0.003	0.142 ± 0.011	0.147 ± 0.014
0.05	0.140 ± 0.012	0.103 ± 0.007	0.105 ± 0.006	0.113 ± 0.004	0.120 ± 0.007
0.1	$\mathbf{0.089 \pm 0.002}$	0.122 ± 0.003	0.132 ± 0.004	0.129 ± 0.003	0.121 ± 0.004
0.2	0.107 ± 0.004	0.162 ± 0.002	0.179 ± 0.006	0.173 ± 0.004	0.169 ± 0.003
LiDi 2D5F					
α_{base}	**VRPO**	MAPPO	IPPO	MMD	PPG
0.02	0.330 ± 0.017	0.209 ± 0.016	0.173 ± 0.009	0.211 ± 0.024	0.143 ± 0.008
0.05	0.127 ± 0.005	0.135 ± 0.002	0.102 ± 0.005	0.103 ± 0.001	0.108 ± 0.002
0.1	$\mathbf{0.099 \pm 0.004}$	0.114 ± 0.002	0.115 ± 0.003	0.114 ± 0.001	0.111 ± 0.003
0.2	0.129 ± 0.002	0.184 ± 0.003	0.182 ± 0.001	0.179 ± 0.003	0.184 ± 0.002

Table 8: Exact exploitability (lower is better) under the base regularization coefficient α_{base} sweep. Each entry reports mean $\pm$ SEM over seeds. Bold indicates the best mean in each game block.

K.3 PPO Clipping Coefficient

We finally inspect the effect of the base clipping coefficient $\varepsilon_{\text{base}}$ used in the PPO-style policy update. We sweep the coefficient over $\varepsilon_{\text{base}} \in \{0.02, 0.05, 0.1, 0.2\}$ and present the results in Table 9. Smaller clipping coefficients impose a tighter constraint on policy updates, while larger values allow more aggressive updates.

Across all three games, VRPO achieves its best performance at a clipping coefficient of 0.02. Increasing the coefficient generally degrades VRPO, especially at 0.2, suggesting that overly loose policy updates can hurt stability in these exact-exploitability benchmarks.

ADH3					
$\varepsilon_{\text{base}}$	**VRPO**	MAPPO	IPPO	MMD	PPG
0.02	$\mathbf{0.081 \pm 0.004}$	0.121 ± 0.008	0.123 ± 0.004	0.131 ± 0.005	0.140 ± 0.006
0.05	0.091 ± 0.008	0.125 ± 0.002	0.122 ± 0.008	0.133 ± 0.006	0.133 ± 0.007
0.1	0.125 ± 0.012	0.135 ± 0.005	0.134 ± 0.008	0.171 ± 0.013	0.138 ± 0.013
0.2	0.238 ± 0.012	0.218 ± 0.019	0.209 ± 0.013	0.233 ± 0.012	0.161 ± 0.007
APTTT					
$\varepsilon_{\text{base}}$	**VRPO**	MAPPO	IPPO	MMD	PPG
0.02	$\mathbf{0.088 \pm 0.002}$	0.119 ± 0.003	0.116 ± 0.002	0.128 ± 0.005	0.126 ± 0.003
0.05	0.103 ± 0.010	0.118 ± 0.005	0.116 ± 0.004	0.123 ± 0.002	0.117 ± 0.004
0.1	0.116 ± 0.005	0.129 ± 0.003	0.134 ± 0.004	0.126 ± 0.005	0.108 ± 0.002
0.2	0.205 ± 0.001	0.157 ± 0.012	0.152 ± 0.004	0.145 ± 0.008	0.108 ± 0.007
LiDi 2D5F					
$\varepsilon_{\text{base}}$	**VRPO**	MAPPO	IPPO	MMD	PPG
0.02	$\mathbf{0.092 \pm 0.001}$	0.117 ± 0.002	0.112 ± 0.004	0.107 ± 0.003	0.112 ± 0.002
0.05	0.098 ± 0.003	0.110 ± 0.002	0.108 ± 0.006	0.107 ± 0.001	0.103 ± 0.003
0.1	0.125 ± 0.002	0.130 ± 0.005	0.116 ± 0.002	0.124 ± 0.006	0.105 ± 0.004
0.2	0.173 ± 0.019	0.155 ± 0.004	0.138 ± 0.006	0.146 ± 0.026	0.132 ± 0.005

Table 9: Exact exploitability (lower is better) under the base clipping coefficient $\varepsilon_{\text{base}}$ sweep. Each entry reports mean $\pm$ SEM over seeds. Bold indicates the best mean in each game block.

L Ablation Study on VRPO-specific Hyperparameters

We study the effect of VRPO-specific hyperparameters on the overall performance of VRPO.

L.1 Replay Buffer Capacity

We inspect the effect of changing the replay buffer capacity $|\mathcal{D}_{\text{replay}}|$. We sweep the ratio between the capacity of replay buffer $|\mathcal{D}_{\text{replay}}|$ and the rollout batch size $|\mathcal{D}_{\text{rollout}}|$ over $\{1, 4, 16, 64\}$, and present the results in Figure 14. While a higher ratio means the critic is trained on more diverse samples, it also introduces additional lag between the stored trajectories and the current policy. When the ratio is 1, VRPO reduces to using only the current rollout to update the critic without replay, following the standard PPO training scheme.

As shown in the figure, increasing the replay buffer capacity does not degrade performance. In our experiments, we set the replay buffer capacity to $64 \times |\mathcal{D}_{\text{rollout}}|$ to balance sample divergence and total memory consumption.

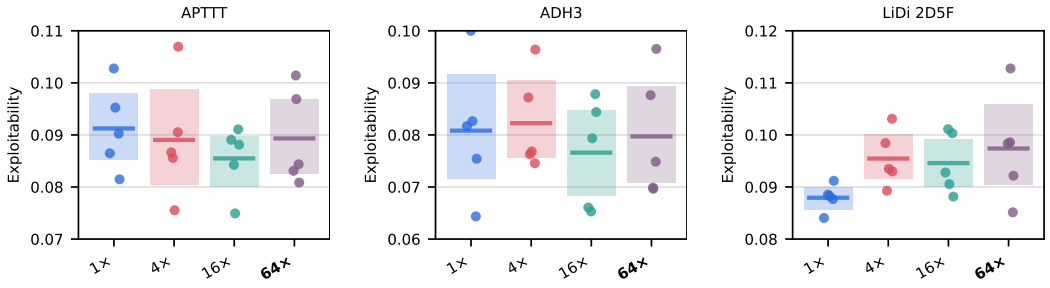

Figure 14: Exact exploitability (lower is better) of agents in APTTT (Abrupt Phantom Tic-Tac-Toe), ADH3 (Abrupt Dark Hex 3), and LiDi 2D5F (Liar's Dice with 2 Dice and 5 Faces), trained with VRPO under different replay buffer capacity.

L.2 Number of Critic Epochs

We inspect the effect of changing the number of critic epochs K_{critic} per iteration. We sweep $K_{\text{critic}} \in \{1, 2, 4, 8\}$ and present the results in Figure 15. As shown in the figure, a larger number of critic epochs leads to a more accurate critic, resulting in more stable performance, as expected. In our main experiments, we set $K_{\text{critic}} = 4$ to balance stability and computational cost, matching the PPO baselines used in CleanRL [26].

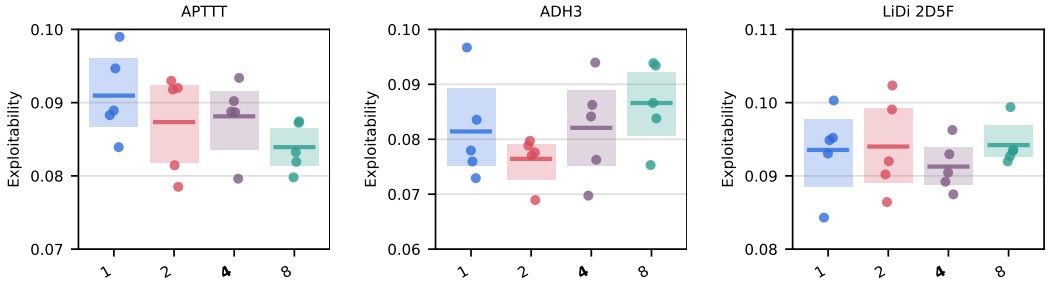

Figure 15: Exact exploitability (lower is better) of agents in APTTT (Abrupt Phantom Tic-Tac-Toe), ADH3 (Abrupt Dark Hex 3), and LiDi 2D5F (Liar's Dice with 2 Dice and 5 Faces), trained with VRPO under different numbers of critic epochs.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have a separate paragraph that discusses the limitations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The theoretical result is proved in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the detailed information in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the codebase needed to reproduce the results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all the training and test details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report error bars about the statistical significance of the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide sufficient information on the computer resources.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This work studies self-play RL in games. Potential negative uses include stronger automated agents for online gambling or unfair play.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We credit the license and terms for the used assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.