GENERALIZABLE OPPONENT EXPLOITATION IN LLM POKER AGENTS VIA MIXED BEST-RESPONSE TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Opponent exploitation, adapting to and profiting from an opponent's weaknesses, is a crucial capability in competitive games, yet existing LLM-based poker agents are mostly prompted or fine-tuned toward equilibrium play and fail to exploit suboptimal opponents. We introduce GOE-LLM (Generalizable Opponent Exploitation with LLMs), a framework for learning opponent exploitation in two-player zero-sum poker. A lightweight MLP profiler summarizes an opponent's recent behavior into a natural-language description, and an LLM exploiter conditioned on this description is trained with group relative policy optimization on a curated mixture of best-response strategies. We propose the Mixture-of-Best-Responses Principle, which excludes non-transitive dominance cycles from the mixture, to keep training stable while preventing collapse to a single equilibrium strategy. In Kuhn poker and Leduc Hold'em, GOE-LLM outperforms prompting and equilibrium-fine-tuned baselines across three model sizes and approaches the best-response upper bound, including against opponents unseen during training. Moreover, GOE-LLM trained only on Kuhn poker carries over zero-shot to unseen imperfect-information games in GTBench, improving the win rate by 18.0 points on average almost entirely through better strategy rather than fewer illegal moves, while bringing no gain in perfect-information games. These results suggest that mixed best-response training teaches LLM agents exploitation skills that generalize beyond the specific opponents and the single game seen in training.

1 INTRODUCTION

Opponent exploitation, i.e., adapting to a specific opponent and profiting from its weaknesses, is a critical skill for agents in competitive scenarios (Hoehn et al., 2005; Ganzfried & Sandholm, 2015; Liu et al., 2022). Large Language Models (LLMs) have recently shown strong strategic reasoning as autonomous agents (Wang et al., 2024) in interactive domains such as multi-agent games (Zhang et al., 2024b; Duan et al., 2024; Huang et al., 2025; Light et al., 2023; 2025; Guo et al., 2024; Hu et al., 2024), yet their potential for *opponent exploitation* remains underexplored. Existing LLM-based game agents mainly follow two approaches: (1) in-context learning with carefully engineered prompts (Brown et al., 2020; Zhang et al., 2024a; Guan et al., 2024; Karten et al., 2025; Cui et al., 2025; Xu et al., 2025b), and (2) direct policy learning methods using data generated from equilibrium or optimal strategies (Huang et al., 2024; Zhuang et al., 2025; Feng et al., 2023; Zhang et al., 2025; Wang et al., 2025; Xu et al., 2025a). Both prioritize general strategic competence or equilibrium play (Zhuang et al., 2025) rather than explicitly adapting to, and exploiting, the weaknesses of diverse opponents. This raises a central question: ***what would it take for LLM-based agents to move beyond equilibrium play and achieve effective and generalizable opponent exploitation?*** Answering it requires two capabilities: recognizing an opponent's tendencies from a few interactions, and learning in a single policy how to respond to many opponent types. The latter is non-trivial, because best responses to different opponents may counter one another in non-transitive cycles, which assigns conflicting action targets to the same game state and destabilizes LLM fine-tuning.

Motivated by this question, we propose GOE-LLM (Generalizable Opponent Exploitation with LLMs), an LLM-centered framework for opponent exploitation (Figure 1). GOE-LLM integrates (i) a lightweight profiler that identifies the opponent's strategic tendencies and (ii) an LLM exploiter

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

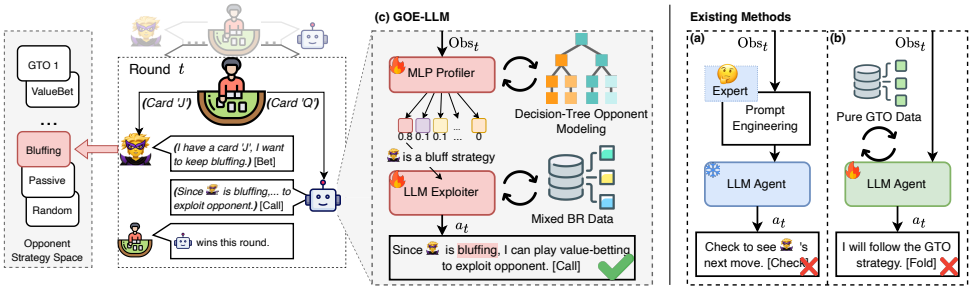


Figure 1: GOE-LLM overview. In two-player zero-sum poker, opponents that deviate from optimal strategies can be exploited to gain more chips. Existing work relies on (a) in-context learning with expert knowledge, which suffers from hallucinations, or (b) training on optimal-strategy data only, which fails to exploit opponent weaknesses. (c) In GOE-LLM, a profiler classifies the opponent and an exploiter is trained on a mixture of best responses to adapt to diverse opponents.

fine-tuned to best-respond to diverse opponents. Inspired by mixture-of-opponents learning (Smith et al., 2023), we train the exploiter on a balanced mixture of best-response strategies rather than on equilibrium play alone, and we construct the mixture to exclude non-transitive dominance cycles (Tenney & Foster, 1976), which preserves training stability and generalization. We then fine-tune the exploiter with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and an opponent-aware reward. In Kuhn poker and Leduc Hold'em, GOE-LLM achieves the best average return among non-oracle methods and generalizes to out-of-distribution opponents across LLM sizes; in Kuhn poker it captures 37.9% of the exploitable surplus, versus -11.4% for equilibrium fine-tuning. The exploiter trained only on Kuhn poker also transfers zero-shot to unseen imperfect-information games in GTBench (Duan et al., 2024), raising the win rate by 18.0 points on average. Compared with self-consistency, a test-time-computation baseline that takes a majority vote over sampled reasoning chains (Wang et al., 2023), the two methods improve complementary skills: self-consistency helps mainly in perfect-information games, while GOE-LLM helps in imperfect-information games and, over all five games, matches self-consistency (54.8% vs. 55.1%) at $1/15$ of the inference cost.

Our contributions are threefold: (1) an LLM-based framework explicitly targeting opponent exploitation; (2) a mixed best-response training principle that balances stability and diversity, with a matching opponent-aware policy optimization; and (3) extensive experiments in Kuhn poker and Leduc Hold'em with zero-shot transfer beyond poker, together with analyses of each component.

2 Related Work

Opponent Exploitation. Opponent exploitation has traditionally combined game theory and reinforcement learning. Classical methods such as Independent RL, Iterated Best Response, Double Oracle (DO), and Fictitious Play (FP) compute responses to opponents' strategies (Lanctot et al., 2017), but often overfit to specific equilibria and generalize poorly to unseen adversaries. Rectified Nash Response maintains diversity under non-transitive strategy cycles (Balduzzi et al., 2019), and Policy Space Response Oracles (PSRO) unify DO and FP by iteratively expanding restricted games (Bighashdel et al., 2024), with extensions for population robustness (Yao et al., 2023a) and adaptive meta-solvers (Li et al., 2024a). In StarCraft II, league training integrates opponent modeling with goal-conditioned exploiters for real-time adaptation (Huang et al., 2023).

LLM-based Agents. LLMs have recently been used in games for opponent modeling and analysis (Hu et al., 2024), mainly through prompting or lightweight fine-tuning, but without systematically addressing opponent exploitation. Carefully designed prompts let LLM agents follow rules and act coherently (Zhang et al., 2024a; Karten et al., 2025), and fine-tuning on curated game data improves domain-specific play: PokerBench shows that fine-tuning yields modest gains in poker but remains far from expert play (Zhuang et al., 2025), Mastermind-Dou/Go fine-tune on structured game data (Wang et al., 2025), and PokerGPT (Huang et al., 2024) adapts lightweight LLMs to Texas Hold'em with RLHF. ICE attempts in-context opponent exploitation via trajectory-based RL, but struggles with dynamic opponents and requires full historical trajectories (Li et al., 2024b; Shi et al., 2024). These methods, based on trajectory imitation or reward optimization, fail to capture opponent-specific weaknesses, a limitation most acute in imperfect-information games such as poker.

3 METHODOLOGY

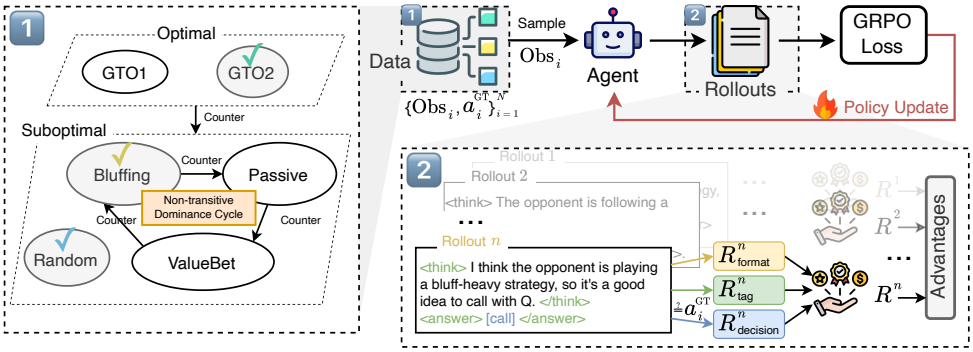


Figure 2: LLM exploiter training. (1) The Mixture-of-Best-Responses Principle selects training data with diverse yet non-cyclic counter relations among strategies; (2) Opponent-Aware Policy Optimization trains the exploiter with GRPO and an opponent-aware reward.

GOE-LLM has two layers: opponent modeling followed by opponent exploitation. An MLP profiler classifies the opponent from its recent behavior, and the classification is translated into a language description that is passed to the LLM exploiter. We first formalize the game setting.

3.1 PRELIMINARIES

Two-player Zero-sum IIEFG. An imperfect-information extensive-form game (IIEFG) $G = (N, A, H, Z, \chi, \rho, \sigma, u, \mathcal{I})$ describes a sequential interaction among n players (Liu et al., 2022). $N = \{1, \ldots, n\}$ is the set of players and c is the chance player with a fixed, commonly known policy σ_c. H and Z are the non-terminal decision nodes and terminal nodes; $\chi : H \to 2^A$ gives the legal actions $\chi(h)$ at node h, and $\rho : H \to N \cup \{c\}$ gives the acting player. $u = (u_1, \ldots, u_n)$ with $u_i : Z \to \mathbb{R}$ is the utility function. $\mathcal{I} = (I_1, \ldots, I_n)$ collects the information sets: I_i partitions player i's decision nodes, nodes in one infoset share the acting player and legal actions, and $I(h)$ denotes the infoset containing h. A strategy $\sigma_i : I_i \to \Delta(A)$ maps infosets to action distributions, and $u_i(\sigma) = u_i(\sigma_i, \sigma_{-i})$ is player i's expected utility under profile $\sigma = (\sigma_1, \ldots, \sigma_n)$.

In two-player zero-sum IIEFGs, $n = 2$ and $u_1 + u_2 = 0$; the game value is $v = \max_{\sigma_1} \min_{\sigma_2} u_1(\sigma_1, \sigma_2)$, and a best response to σ_2 is $\text{BR}_1(\sigma_2) = \arg\max_{\sigma_1} u_1(\sigma_1, \sigma_2)$. A profile $\sigma^* = (\sigma_1^*, \sigma_2^*)$ is a Nash equilibrium if neither player gains by deviating unilaterally; in two-player zero-sum games it is also a minimax profile, in which each strategy best-responds to the other:

$$\sigma_1^* = \text{BR}_1(\sigma_2^*), \qquad \sigma_2^* = \text{BR}_2(\sigma_1^*). \tag{1}$$

Equilibrium play guarantees the game value v against any opponent but forgoes the extra payoff against a suboptimal σ_2, which $\text{BR}_1(\sigma_2)$ captures and opponent exploitation targets.

Game-playing with LLM agents. We put the game rules and the current infoset into the prompt, so that the LLM can make valid decisions without explicitly modeling the game state. At decision node h, the LLM receives prompt_h containing the rules, the infoset $I(h)$, and the action history, and outputs an action $a \sim \sigma_{\text{LLM}}(I(h)) = f(\pi_\theta(\text{prompt}_h))$ with $a \in \chi(I(h))$, where π_θ is the LLM with parameters θ and f is a deterministic parser that extracts the action from its output.

3.2 MLP OPPONENT PROFILER

The MLP profiler is pre-trained on data from predefined opponent types to map the opponent's behavior tree over its last k games to a discrete opponent type, which is then translated into a language description for the exploiter. A small k may miss the opponent's patterns, while a large k may overlook recent weaknesses; we set $k = 10$ for Kuhn poker and $k = 50$ for Leduc Hold'em. At test time, the profiler observes the opponent's actions in the most recent k games, predicts its type, and inserts the corresponding description (e.g., the opponent's tendency to bluff or to play passively)

Algorithm 1 GOE-LLM training and inference

Require: predefined opponent types, candidate matchups $\{(\sigma^i_{\text{oppo}}, \sigma^i_{\text{agent}})\}$, LLM π_θ, history length k
1: Train the MLP profiler to map the opponent's behavior over k games to its type
2: Keep matchups with $\sigma^i_{\text{agent}} \in \text{Counter}(\sigma^i_{\text{oppo}}) \cup \text{BR}(\sigma^i_{\text{oppo}})$ whose agent strategies form no counter cycle
3: Build $\mathcal{D}$ of (prompt with opponent description, agent action) pairs from the kept matchups
4: **for** each training step **do**
5: Sample prompts $q \sim \mathcal{D}$ and G responses $\{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$
6: Compute rewards R_i and advantages $\hat{A}_i$; update π_θ by maximizing Eq. (3)
7: **end for**
8: **Inference:** every k games, profile the opponent and insert its description into the prompt; act with π_θ

into the exploiter's prompt; the exploiter then conditions its reasoning and action on this description. Because the interface between the two components is natural language, the exploiter can in principle receive opponent descriptions from any source, which we exploit in the cross-game experiments of Section 4.4. Training details, the choice of k, and visualizations are in Appendix A.4.

3.3 LLM Opponent Exploiter

The exploiter should learn a robust policy that approximates best responses to different opponent types. We propose a Mixture-of-Best-Responses Principle that balances training stability and generalization, and extend GRPO (Shao et al., 2024) with an opponent-aware reward (Figure 2); Algorithm 1 summarizes the pipeline from profiler training to exploiter training and inference.

3.3.1 Mixture-of-Best-Responses Principle

Training LLM agents only on equilibrium strategies tends to overfit (Lanctot et al., 2017) and adapts poorly to diverse opponents, whereas learning to counter all possible opponents can make training unstable. Let the strategy space of two-player zero-sum IIEFG be Σ. For two strategies $\sigma, \sigma' \in \Sigma$, we say that σ *counters* σ', written as $\sigma \succ \sigma'$, if $\tilde{u}(\sigma, \sigma') > 0$, where $\tilde{u}(\sigma, \sigma')$ denotes the symmetrized payoff of strategy σ against σ', averaged over both player positions. Let $\Sigma_i, \Sigma_j \subseteq \Sigma$ denote two disjoint subsets (strategy clusters). We say that Σ_i *counters* Σ_j, written as $\Sigma_i \succ \Sigma_j$, if

$$\forall \sigma \in \Sigma_i,\ \forall \sigma' \in \Sigma_j : \quad \sigma \succ \sigma' \Leftrightarrow \tilde{u}(\sigma, \sigma') > 0.$$

By the definition of best response, $\tilde{u}(\text{BR}(\sigma), \sigma) > 0$, i.e., $\text{BR}(\sigma)$ counters σ; we define the *counter set* of σ as $\text{Counter}(\sigma) := \{\, \sigma' \in \Sigma \mid \sigma' \succ \sigma \,\}$, which thus contains $\text{BR}(\sigma)$.

Cyclic counter relation. A strategy set $\Sigma' \subseteq \Sigma$ exhibits a *cyclic counter relation* if some subset $\{\sigma^1, \ldots, \sigma^k\} \subseteq \Sigma'$ forms a *non-transitive dominance cycle* $\sigma^1 \succ \sigma^2 \succ \cdots \succ \sigma^k \succ \sigma^1$. The training dataset $\mathcal{D}$ is generated from profiles $\{(\sigma^i_{\text{oppo}}, \sigma^i_{\text{agent}})\}_{i=1}^m$, where the agent's strategy satisfies $\sigma^i_{\text{agent}} \in \text{Counter}(\sigma^i_{\text{oppo}}) \cup \text{BR}(\sigma^i_{\text{oppo}})$. $\mathcal{D}$ satisfies the *Mixture-of-Best-Responses Principle* if the agent strategies $\{\sigma^1_{\text{agent}}, \ldots, \sigma^m_{\text{agent}}\}$ exhibit no cyclic counter relation. Intuitively, a cycle-free mixture of best responses keeps training stable while retaining diversity for generalization across opponents. When agent strategies do form a cycle, the same infoset can be labeled with conflicting target actions in different matchups (each strategy in the cycle is countered by another), so the policy is pulled in opposite directions and the gradients become noisy; excluding cycles removes such contradictory targets while keeping the diverse, opponent-dependent responses that exploitation requires. Section 4.3 ablates this choice by comparing mixtures with and without such cycles.

Instantiation in poker. In Kuhn poker, the mixture combines three matchups: Random vs. GTO, which exposes the agent to non-strategic noise; GTO vs. GTO, which anchors it near equilibrium; and Bluff vs. Counter-Bluff, which teaches it to punish over-bluffing. In Leduc Hold'em, Tight vs. Loose replaces the bluffing matchup. These agent strategies counter their opponents without forming a counter cycle, so the mixture satisfies the principle; Appendix A.2 lists sample counts.

3.3.2 Opponent-Aware Policy Optimization

We optimize the exploiter with GRPO, which replaces PPO's learned critic with group-normalized advantages. Given a prompt q (an infoset) and G responses $\{o_i\}_{i=1}^G$ sampled from $\pi_{\theta_{\text{old}}}$ with rewards

R_i, the advantage normalizes each reward within its group,

$$\hat{A}_i = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^G)}{\text{std}(\{R_j\}_{j=1}^G)}, \tag{2}$$

which estimates relative quality with low variance and no value network; the objective is

$$J_{\text{GRPO}}(\theta) = \mathbb{E}\Big[\frac{1}{G}\sum_{i=1}^{G}\Big(\min\big(r_i(\theta)\hat{A}_i,\ \text{clip}(r_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\big) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\Big)\Big], \tag{3}$$

where $r_i(\theta) = \pi_\theta(o_i \mid q)/\pi_{\theta_{\text{old}}}(o_i \mid q)$, ϵ is the clipping range, and β weights the KL penalty to a frozen reference policy π_{ref}, which keeps the updated policy close to the base LLM.

Opponent-aware reasoning reward. The reward has three components. The *format reward* $R_{\text{format}} \in \{0, 1\}$ is 1 only if the output strictly follows `<think>...</think><answer> [ACTION] </answer>`, where `<think>` holds the reasoning and `<answer>` the action. The *tag reward* $R_{\text{tag}} \in [0, 1]$ adds 0.25 for each of the four tags that appears exactly once (Han et al., 2025). The *decision reward* $R_{\text{decision}} \in \{0, 1\}$ is 1 if `[ACTION]` matches the ground-truth action. The total reward $R_i = R_{\text{decision}} + \alpha_{\text{tag}} R_{\text{tag}} + \alpha_{\text{format}} R_{\text{format}}$ rewards correct decisions and structured reasoning.

4 Experiments

We evaluate GOE-LLM on Kuhn poker and Leduc Hold'em against rule-based and LLM baselines, ablate its data mixture, profiler, and model size, and test whether the exploitation skill learned on poker transfers zero-shot to five games from GTBench (Duan et al., 2024) (Section 4.4).

4.1 Experimental Setup

Environments and opponents. We use two classic two-player zero-sum poker games, Kuhn poker and Leduc Hold'em (Southey et al., 2005), implemented in TextArena (Guertler et al., 2025); both are multi-round with actions check, bet, call, raise, and fold. We build 24 opponent strategies for Kuhn poker (21 exploitable, 3 GTO) and 14 for Leduc Hold'em from six predefined types each (Table 1); types marked † in Tables 2 and 3 are unseen during training. The Kuhn GTO strategy follows Kuhn (2016), and the Leduc GTO strategy is computed with CFR (Zinkevich et al., 2007); all strategies and training data are in Appendices A.1 and A.2.

Table 1: Number of opponent strategies per type used for evaluation. † marks types unseen during training; all types but GTO are exploitable.

Kuhn poker		**Leduc Hold'em**	
Type	#	Type	#
Random	1	Random	1
GTO	3	GTO	1
Bluff	8	Tight	3
Value†	4	Loose†	3
Passive†	4	Passive†	3
Aggressive†	4	Aggressive†	3
Total	**24**	**Total**	**14**

Baselines. We compare with rule-based agents: **Random** (uniform over legal actions), **GTO** (Kuhn, 2016; Zinkevich et al., 2007) (a precomputed equilibrium strategy), and **BR***, an idealized best response with full knowledge of the hidden opponent that serves as an upper bound. LLM baselines are **LLM** (Yao et al., 2023b), which acts on the current observation without opponent-modeling (OM) information; **LLM+OM**, which also receives the profiler's OM description but is not fine-tuned; and **GTO-LLM** (Zhuang et al., 2025), fine-tuned on pure GTO data. By default, LLM-based methods use Qwen2.5-3B and share the prompt in Appendix A.3; ablations also use Qwen2.5-1.5B and 7B. We set $\alpha_{\text{tag}} = \alpha_{\text{format}} = 0.1$ following Han et al. (2025); the remaining hyperparameters and the training environment are in Appendices A.5 and A.6.

Evaluation. Each agent plays 3000 hands per opponent strategy in Kuhn poker and 600 in Leduc Hold'em, and we report the average win chips per hand (Kuhn, 2016; Zhuang et al., 2025).

Exploitation rate. Raw returns in Kuhn poker are small, since the game value is only $-1/18$ for P0 and $+1/18$ for P1, so gaps between methods are hard to read. We thus report the exploitation rate,

Table 2: Performance comparison of GOE-LLM and baseline methods in Kuhn poker. The left block shows results when the agent plays as P0 (first player), the right block as P1 (second player). Each entry reports the average return per hand. BR* denotes an idealized best-response agent that plays the optimal counter-strategy. Except for Random, each column averages over all strategies of that type; † marks types unseen during training. **Bold** and underlined entries denote the best and second-best results in each column, excluding the oracle BR*, which serves as an upper bound.

Method	Agent as P0 (first player)							Agent as P1 (second player)						
	Rand.	GTO	Bluff	Value†	Pass.†	Agg.†	**Avg.**	Rand.	GTO	Bluff	Value†	Pass.†	Agg.†	**Avg.**
Baselines														
Random	0.111	-0.171	-0.221	-0.303	-0.146	-0.136	-0.145	-0.150	-0.106	-0.279	-0.125	-0.020	-0.143	-0.137
GTO	0.133	**-0.051**	-0.064	-0.052	-0.065	-0.050	-0.025	0.167	**0.063**	0.060	0.046	0.053	0.070	0.076
LLM	**0.317**	-0.146	-0.154	-0.250	-0.148	-0.081	-0.077	0.054	-0.109	-0.166	-0.211	-0.063	0.060	-0.073
LLM+OM	0.222	-0.160	-0.178	-0.188	-0.086	-0.212	-0.100	-0.085	-0.143	-0.219	-0.091	-0.035	-0.112	-0.114
GTO-LLM	0.010	-0.055	-0.035	**0.145**	**0.043**	-0.117	-0.001	0.074	0.052	0.021	**0.125**	0.019	-0.163	0.021
GOE-LLM	0.124	-0.054	**-0.028**	0.073	-0.015	**0.024**	**0.021**	**0.212**	0.058	**0.219**	0.115	**0.093**	**0.101**	**0.133**
BR*	0.509	-0.063	0.095	0.147	0.081	0.084	0.142	0.415	0.052	0.243	0.126	0.125	0.127	0.181

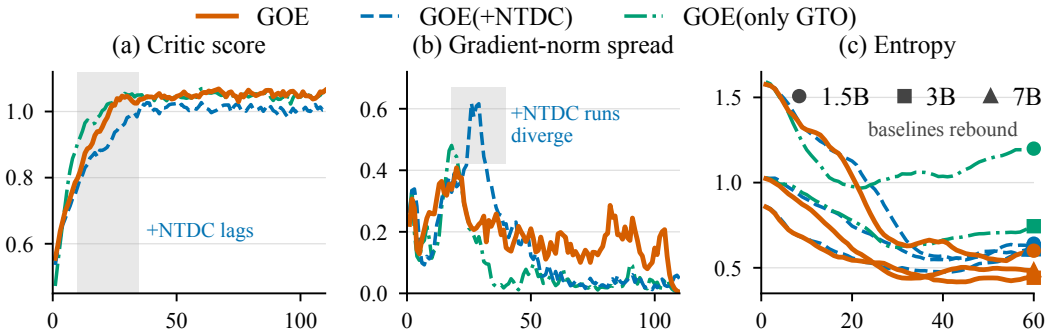


Figure 3: Training dynamics of the three data mixtures: removing non-transitive dominance cycles keeps GOE training stable. (a) Critic score: GOE(+NTDC) rises more slowly and plateaus lower. (b) Spread (max − min) of the actor gradient norm across runs: GOE(+NTDC) runs diverge around steps 20–35 (shaded box). (c) Policy entropy of each run over the first 60 steps, markers give the model size: GOE's entropy decreases and settles, whereas GOE(+NTDC) and GOE(only GTO) start to rebound. The x-axis is the training step; curves are EMA-smoothed; (a) shows the run mean.

the share of the surplus available from an opponent that an agent captures beyond equilibrium play:

$$\mathrm{ER}(\pi, o) = \frac{\bar{R}(\pi, o) - \bar{R}(\pi_{\mathrm{GTO}}, o)}{\bar{R}(\mathrm{BR}^*, o) - \bar{R}(\pi_{\mathrm{GTO}}, o)} \times 100\%, \tag{4}$$

where $\bar{R}(\pi, o)$ is the return per hand of agent π against opponent o averaged over both seats, which cancels the game value. $\mathrm{ER} = 0\%$ matches the GTO strategy, $\mathrm{ER} = 100\%$ matches the idealized best response, and $\mathrm{ER} < 0$ means the agent earns less than simply playing GTO.

4.2 Main Results

Tables 2 and 3 report results in Kuhn poker and Leduc Hold'em. Against both seen and unseen opponents, GOE-LLM wins more chips per hand than the game value on average over both seats.

GOE-LLM is a strong opponent exploiter. In Kuhn poker, GOE-LLM achieves the best average return among all non-oracle methods, 0.021 chips per hand as P0 and 0.133 as P1, closest to the idealized BR* agent (0.142 and 0.181). In Leduc Hold'em it is again best, with 0.083 as P0 and 0.034 as P1. The per-opponent results in Leduc Hold'em show where these gains come from: as P0, GOE-LLM is the only method with positive returns against all three Loose and all three Aggressive strategies, both unseen during training (e.g., 0.277 vs. −1.208 for LLM against Loose2), while LLM+OM earns more against Tight and Passive opponents but loses heavily to Loose and Aggressive

Table 3: Performance comparison of GOE-LLM with baseline methods in Leduc Hold'em. Each entry reports the average return per hand; numbers 1–3 index the strategies of each type, and † marks types unseen during training. **Bold** denotes the best result in each column for each role.

Method	Role	Random	GTO	Tight			Loose†			Passive†			Aggressive†			Avg.
				1	2	3	1	2	3	1	2	3	1	2	3	
Baselines																
LLM	P0	0.263	-0.063	0.092	-0.054	0.529	-0.975	-1.208	-1.850	**0.433**	-0.375	-0.408	-1.113	-1.429	-1.917	-0.577
	P1	-0.463	**0.121**	0.379	-0.063	0.242	-0.425	-0.292	-1.083	-0.529	-0.242	-0.525	-0.683	-0.754	-0.488	-0.343
LLM+OM	P0	0.007	-0.618	**0.727**	**0.815**	**0.792**	-1.760	-2.108	-2.438	0.362	**0.108**	**0.337**	-2.423	-2.413	-2.440	-0.790
	P1	-0.285	0.115	**0.577**	**0.917**	**0.767**	-1.358	-1.012	-1.013	**0.533**	**0.582**	**0.248**	-1.362	-1.495	-1.197	-0.285
GOE-LLM	P0	**0.268**	**0.042**	-0.098	-0.105	-0.120	**0.118**	**0.277**	**0.082**	0.088	-0.167	0.062	**0.185**	**0.190**	**0.347**	**0.083**
	P1	**0.203**	0.082	-0.063	-0.093	0.175	**0.143**	**-0.035**	**0.287**	0.043	0.093	0.060	**-0.128**	**-0.265**	**-0.027**	**0.034**

ones (below -1.7 chips per hand). GOE-LLM thus adapts its strategy to exploit different suboptimal opponents instead of committing to a single fixed style across all opponents.

Training on mixed data promotes generalization. GTO-LLM, trained on pure GTO data, plays close to equilibrium against the GTO opponent (-0.055 as P0 and 0.052 as P1, near the game value of Kuhn poker), but degrades sharply against Random and Aggressive opponents, suggesting that it memorizes the equilibrium strategy rather than learning to play. GOE-LLM instead exceeds the equilibrium value against every opponent type in both positions, including unseen types.

GOE-LLM captures the most exploitable surplus. Table 4 makes the gap in Kuhn poker explicit. GOE-LLM captures 37.9% of the surplus that BR* extracts over GTO play, while every other learned method has a negative average rate (GTO-LLM -11.4%, LLM -73.9%, LLM+OM -97.4%), i.e., earns less than simply playing GTO. GOE-LLM is also the only learned method with a positive rate against every opponent type, including the unseen Value (69.5%), Passive (41.3%), and Aggressive (55.0%) types. GTO-LLM nearly matches BR* against Value opponents (98.9%) but drops to -157.1% against Aggressive ones, so its gains do not come from adapting to the opponent.

Table 4: Exploitation rate (%, Eq. 4) in Kuhn poker: 0% matches GTO, 100% matches BR*, and negative values earn less than GTO. The GTO type is omitted as it leaves no surplus. **Bold**/underline: best/second best, excluding reference rows.

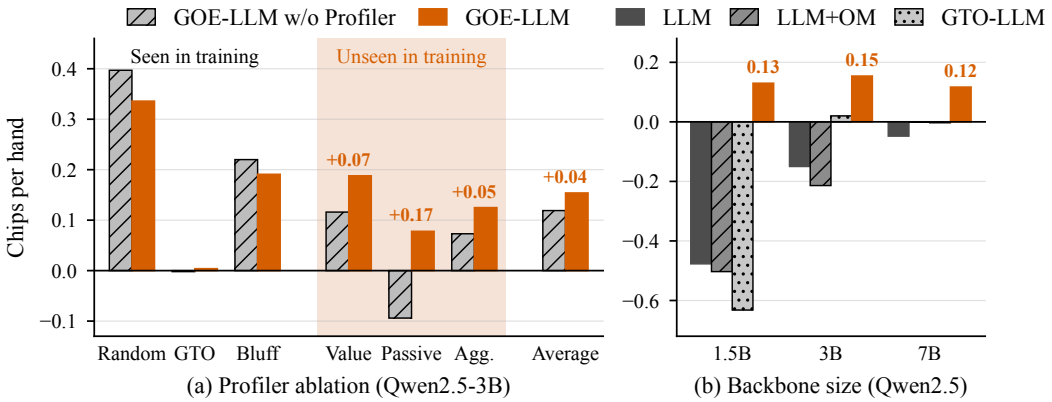
Method	Rand.	Bluff	Value†	Pass.†	Agg.†	**Avg.**
GTO	0.0	0.0	0.0	0.0	0.0	0.0
BR*	100.0	100.0	100.0	100.0	100.0	100.0
Random	-54.3	-145.0	-151.3	-70.6	-156.5	-122.4
LLM	**11.4**	-92.4	-163.1	-91.3	-21.5	-73.9
LLM+OM	-26.1	-114.9	-97.8	-50.0	-180.1	-97.4
GTO-LLM	-34.6	-2.9	**98.9**	33.9	-157.1	-11.4
GOE-LLM	5.8	**57.0**	69.5	**41.3**	**55.0**	**37.9**

Vanilla LLMs are weak opponent exploiters. Without fine-tuning, the LLM agent averages -0.077 and -0.073 chips per hand in Kuhn poker, and adding OM information does not help (LLM+OM: -0.100 and -0.114); the same holds in Leduc Hold'em (LLM: -0.577, -0.343; LLM+OM: -0.790, -0.285). Base LLMs thus cannot turn opponent information into effective exploitation on their own, which motivates fine-tuning on a mixture of best-response data.

4.3 Ablation Studies

Data mixture strategies. We compare GOE(only GTO), trained only on equilibrium data; GOE, which mixes best responses under the Mixture-of-Best-Responses Principle; and GOE(+NTDC), which adds data with non-transitive dominance cycles (Figure 3; full curves in Appendix B.2). GOE(+NTDC) learns more slowly, with a critic score that catches up only after about 40 steps and plateaus lower, and its runs diverge sharply in gradient norm around steps 20–35, confirming that cycles destabilize optimization. Without cycles, GOE converges in about 25 steps, while GOE(only GTO) converges fastest given its single target. GOE's entropy decreases and settles, whereas the entropy of the other two mixtures starts to rebound, so GOE keeps a stable yet non-degenerate policy.

Contribution of the MLP profiler. GOE-LLM (w/o Profiler) withholds the opponent-type description. In Figure 4(a), returns summed over both seats (P0 + P1) stay comparable against seen opponents (Random, GTO, Bluff; slightly higher against Random, 0.397 vs. 0.336) but drop against unseen types: from 0.188 to 0.116 (Value), 0.078 to -0.094 (Passive), and 0.125 to 0.073 (Aggressive). The



(a) Profiler ablation (Qwen2.5-3B) (b) Backbone size (Qwen2.5)

Figure 4: The profiler helps most against unseen opponent types, and GOE-LLM is the only method with a positive return at every model size. Returns are chips per hand summed over both seats (P0 + P1). (a) Return of GOE-LLM with and without the profiler in Kuhn poker (Qwen2.5-3B); the shaded region marks opponent types unseen during training, and orange numbers give the gain from the profiler. (b) Return of four methods per backbone size; orange-bar labels give GOE-LLM's return.

profiler identifies opponent types with over 96% accuracy, and its descriptions guide the exploiter's adaptation; per-type accuracy and position-specific results are in Appendices A.4 and B.1.

LLM sizes. Figure 4(b) compares Qwen2.5-1.5B, 3B, and 7B. LLM-based baselines generally improve with model size, and GTO-LLM stabilizes around 0, as expected of equilibrium play. GOE-LLM achieves positive returns at every size (0.130, 0.154, and 0.117 chips per hand for 1.5B, 3B, and 7B, summed over both seats, P0 + P1). Its advantage is largest at 1.5B, where all baselines lose chips and GTO-LLM fails to learn a usable equilibrium strategy (-0.632), indicating that best-response mixtures are also easier for small models to learn than equilibrium data alone.

4.4 Generalization of GOE-LLM

We test whether the exploitation skill learned on Kuhn poker transfers zero-shot beyond poker, using five GTBench games (Duan et al., 2024): Tic-Tac-Toe, Connect-4, and Breakthrough (perfect information), and Liar's Dice and Prisoner's Dilemma (imperfect information). We expect transfer mainly to the latter, where good play requires reasoning about the opponent's hidden dice or unseen move. As GTBench has no strategic opponents, all games are played against its Random agent.

Setup. All methods use the GTBench prompt, with "You are playing against a random opponent." appended to the system prompt, replacing the profiler's output. **Base** uses the original Qwen2.5-Instruct weights (1.5B, 3B, 7B) and **GOE-LLM** our Kuhn-poker checkpoints of the same sizes, both with one sample per move; **SC** (self-consistency) (Wang et al., 2023) uses these weights and majority-votes over $K \in \{3, 7, 11, 15\}$ chain-of-thought samples, at $K\times$ the cost. Each setting is played for $\geq$1,000 games. We report the win rate, counting aborts due to illegal LLM moves as losses, and test each setting against Base with a two-proportion z-test, Holm-corrected over the 15 settings.

Results. Figure 5 shows the change of GOE-LLM over Base per setting; Figure 6 compares it with SC by game type, and Figure 10 (Appendix C) gives SC per setting.

(1) GOE-LLM transfers to imperfect-information games. With one sample per move, GOE-LLM raises the win rate by 18.0 pp on average, significantly in 5 of the 6 settings and never significantly worse. The gain is largest in Prisoner's Dilemma, significant at every size (+41.5, +17.0, +35.5 pp), with the average win rate rising from 42.7% to 74.0%; in Liar's Dice it rises from 57.4% to 62.0%, significantly at 3B and 7B. SC gains only about 3 pp in this group at every K and is significantly worse than Base in one setting at $K = 15$ (7B Liar's Dice, -8.5 pp). At 7B, GOE-LLM has the highest win rate of all methods, including SC with $K = 15$, in both games (Figure 11).

(2) SC, not GOE-LLM, helps in perfect-information games (Figure 6). Here GOE-LLM changes the win rate by only +0.9 pp on average, with its only two significant regressions in Tic-Tac-Toe (1.5B and 3B). SC improves with more samples, from +4.7 pp at $K = 3$ to +11.2 pp at $K = 15$

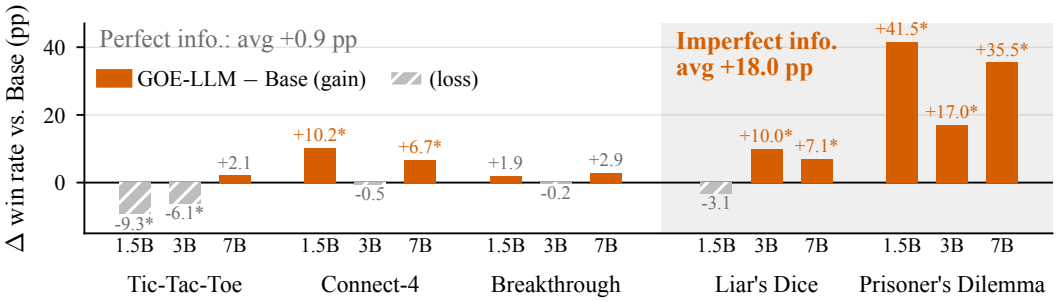


Figure 5: GOE-LLM trained on Kuhn poker transfers zero-shot to imperfect-information games (shaded, +18.0 pp on average) and stays close to Base in perfect-information games (+0.9 pp). Bars show the win-rate change (pp) of GOE-LLM over Base per game and model size; hatched bars are losses. Asterisks mark significance (two-proportion z-test, Holm-corrected over the 15 settings).

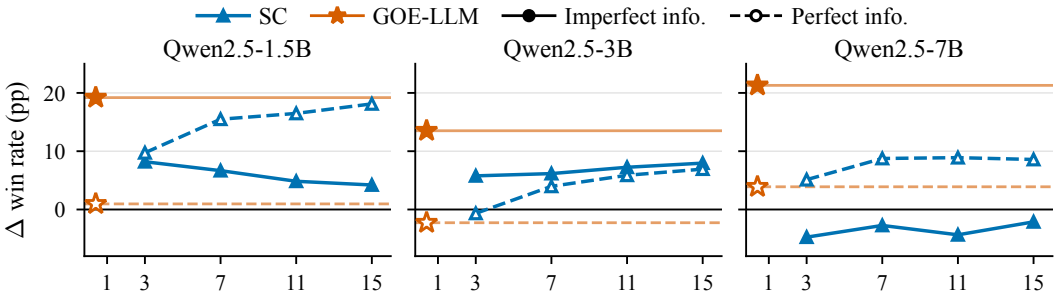


Figure 6: Self-consistency (SC) helps mainly in perfect-information games, whereas GOE-LLM with one sample per move helps in imperfect-information games, at every model size. Each panel is one Qwen2.5 size; the x-axis is the number of samples per move K for SC, and GOE-LLM ($K = 1$) is shown as a star with a horizontal reference line. The y-axis is the win-rate change over Base (pp), averaged over the imperfect-information games (solid, filled: Liar's Dice and Prisoner's Dilemma) or the perfect-information games (dashed, hollow: Tic-Tac-Toe, Connect-4, and Breakthrough).

(largest in Connect-4, 49.7% to 76.1%), consistent with errors in these games being mostly tactical. Over all five games, GOE-LLM with one sample matches SC with $K = 15$ (54.8% vs. 55.1%) at only $1/15$ of the inference cost (Figure 12), without any extra sampling at test time.

(3) GOE-LLM's gains are strategic, not rule following. Since aborted games count as losses, we split each change into a legality and a strategy term via win rate = completion rate × win rate among completed games (Figure 13). In imperfect-information games, 17.6 of GOE-LLM's 18.0 pp gain comes from strategy (all of it in Prisoner's Dilemma, where games always complete), whereas 2.5 of SC's 3.3 pp comes from fewer illegal moves. One explanation is that the Kuhn training mixture includes a Random opponent, so the exploiter has learned to act on the description of a random opponent and to exploit its lack of strategy (in Prisoner's Dilemma, for instance, the best response to a random partner is to defect), whereas a board game rewards accurate look-ahead, which exploitation training does not teach. The exploitation learned on Kuhn poker thus carries over to unseen imperfect-information games, and SC and GOE-LLM are complementary across game types.

5 Conclusion

We presented GOE-LLM, a framework for opponent exploitation by LLM agents in two-player zero-sum imperfect-information games. A lightweight MLP profiler turns an opponent's recent behavior into a language description, and an LLM exploiter trained on a cycle-free mixture of best responses adapts to it without collapsing to a single equilibrium strategy. In Kuhn poker and Leduc Hold'em, GOE-LLM exploits diverse opponents, including types unseen during training, and ablations confirm the roles of the cycle-free mixture and the profiler. Trained only on Kuhn poker, the exploiter also transfers zero-shot to imperfect-information games in GTBench, complementing self-consistency by improving strategic rather than tactical play. These results point toward LLM agents that exploit opponents in larger and more complex imperfect-information games beyond poker.

AI USE STATEMENT

In this work, we used generative AI tools for translation. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims or proofs, propose or refine hypotheses, design research methodology or experiments, implement methods, clean or reformat datasets, perform qualitative or thematic data analysis, or interpret results. Additionally, we used generative AI tools to polish the writing of the paper to improve readability. We have reviewed all AI-assisted text and verified that it faithfully reflects our methods and results. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies opponent exploitation by LLM agents in small, standard two-player zero-sum poker games (Kuhn poker and Leduc Hold'em) and in five GTBench games. All opponents are rule-based programs or random agents, and all training data are synthetically generated from matches between such programs; no human subjects were involved, no new human data were collected, and no personally identifiable information was processed. All base models (Qwen2.5-Instruct) and environments (TextArena and GTBench) are publicly available. The main foreseeable misuse is deploying exploitation agents against human players, for example in real-money online poker, where automated play typically violates platform rules and could cause financial harm; our study is confined to toy-scale games against scripted opponents and is not directly applicable to such settings, and we do not endorse such use. More broadly, agents that profile an opponent and adapt to its weaknesses could be misused to manipulate people in negotiation or other competitive interactions, so deployments beyond games should disclose automated play and follow the rules of the setting. We have adhered to the ICLR Code of Ethics throughout the preparation and submission of this work.

REPRODUCIBILITY STATEMENT

We describe all components needed to reproduce our results. The opponent strategies and their parameters are given in Appendix A.1 and Table 1, and the construction and sizes of the training mixtures are given in Appendix A.2. The complete prompts for Kuhn poker and Leduc Hold'em are in Appendix A.3, the MLP profiler's features, training, and choice of history length k are in Appendix A.4, and the GRPO hyperparameters and training environment are in Appendices A.5 and A.6. The reward is fully specified in Section 3.3, and the evaluation protocol (hands per opponent, metrics, and the exploitation rate in Eq. 4) in Section 4. The GTBench setup, including prompts, number of games, and the Holm-corrected two-proportion z-tests, is described in Section 4.4 and Appendix C. The Kuhn poker and Leduc Hold'em environments are adapted from TextArena (`https://github.com/LeonGuertler/TextArena`), and our CFR implementation for the Leduc GTO strategy follows the open-source `pycfr` (`https://github.com/tansey/pycfr`). All experiments use publicly available Qwen2.5-Instruct checkpoints (1.5B, 3B, and 7B) from HuggingFace, fine-tuned with `verl` (`https://github.com/volcengine/verl`) and evaluated with `vllm` (`https://github.com/vllm-project/vllm`). In the supplementary material, we provide the code for data generation, the datasets, and the training and evaluation scripts, as well as detailed experimental results, to facilitate verification and replication of our findings.

REFERENCES

David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In *International Conference on Machine Learning*, pp. 434–443. PMLR, 2019.

Ariyan Bighashdel, Yongzhao Wang, Stephen McAleer, Rahul Savani, and Frans A. Oliehoek. Policy space response oracles: A survey. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 7951–7961. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/880. URL `https://doi.org/10.24963/ijcai.2024/880`. Survey Track.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Sijia Cui, Shuai Xu, Aiyao He, Yanna Wang, and Bo Xu. Empowering llms with parameterized skills for adversarial long-horizon planning, 2025. URL https://arxiv.org/abs/2509.13127.

Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. Gtbench: Uncovering the strategic reasoning capabilities of llms via game-theoretic evaluations. *Advances in Neural Information Processing Systems*, 37:28219–28253, 2024.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. Chessgpt: Bridging policy learning and language modeling. *Advances in Neural Information Processing Systems*, 36:7216–7262, 2023.

Sam Ganzfried and Tuomas Sandholm. Safe opponent exploitation. *ACM Transactions on Economics and Computation (TEAC)*, 3(2):1–28, 2015.

Zhenyu Guan, Xiangyu Kong, Fangwei Zhong, and Yizhou Wang. Richelieu: Self-evolving llm-based agents for ai diplomacy. *Advances in Neural Information Processing Systems*, 37:123471–123497, 2024.

Leon Guertler, Bobby Cheng, Simon Yu, Bo Liu, Leshem Choshen, and Cheston Tan. Textarena. *arXiv preprint arXiv:2504.11442*, 2025.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. In *IJCAI*, 2024.

Peixuan Han, Zijia Liu, and Jiaxuan You. Tomap: Training opponent-aware llm persuaders with theory of mind. *arXiv preprint arXiv:2505.22961*, 2025.

Bret Hoehn, Finnegan Southey, Robert C Holte, and Valeriy Bulitko. Effective short-term opponent exploitation in simplified poker. In *AAAI*, volume 5, pp. 783–788, 2005.

Sihao Hu, Tiansheng Huang, Gaowen Liu, Ramana Rao Kompella, Fatih Ilhan, Selim Furkan Tekin, Yichang Xu, Zachary Yahn, and Ling Liu. A survey on large language model-based game agents. *arXiv preprint arXiv:2404.02039*, 2024.

Chenghao Huang, Yanbo Cao, Yinlong Wen, Tao Zhou, and Yanru Zhang. Pokergpt: An end-to-end lightweight solver for multi-player texas hold'em via large language model. *arXiv preprint arXiv:2401.06781*, 2024.

Jen-tse Huang, Eric John Li, Man Ho Lam, Tian Liang, Wenxuan Wang, Youliang Yuan, Wenxiang Jiao, Xing Wang, Zhaopeng Tu, and Michael R. Lyu. Competing large language models in multi-agent gaming environments. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Ruozi Huang, Xipeng Wu, Hongsheng Yu, Zhong Fan, Haobo Fu, Qiang Fu, and Wei Yang. A robust and opponent-aware league training method for starcraft ii. *Advances in Neural Information Processing Systems*, 36:47554–47574, 2023.

Seth Karten, Andy Luu Nguyen, and Chi Jin. Pokéchamp: an expert-level minimax language agent. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=SnZ7SKykHh.

Harold W Kuhn. A simplified two-person poker. *Contributions to the Theory of Games*, 1:97–103, 2016.

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.

Pengdeng Li, Shuxin Li, Chang Yang, Xinrun Wang, Xiao Huang, Hau Chan, and Bo An. Self-adaptive psro: Towards an automatic population-based game solver. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 139–147. International Joint Conferences on Artificial Intelligence Organization, 8 2024a. doi: 10.24963/ijcai.2024/16. URL https://doi.org/10.24963/ijcai.2024/16. Main Track.

Shuxin Li, Chang Yang, Youzhi Zhang, Pengdeng Li, Xinrun Wang, Xiao Huang, Hau Chan, and Bo An. In-context exploiter for extensive-form games. *arXiv preprint arXiv:2408.05575*, 2024b.

Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. Avalonbench: Evaluating LLMs playing the game of avalon. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023. URL https://openreview.net/forum?id=ltUrSryS0K.

Jonathan Light, Min Cai, Weiqin Chen, Guanzhi Wang, Xiusi Chen, Wei Cheng, Yisong Yue, and Ziniu Hu. Strategist: Self-improvement of LLM decision making via bi-level tree search. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=gfI9v7AbFg.

Mingyang Liu, Chengjie Wu, Qihan Liu, Yansen Jing, Jun Yang, Pingzhong Tang, and Chongjie Zhang. Safe opponent-exploitation subgame refinement. *Advances in Neural Information Processing Systems*, 35:27610–27622, 2022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Chengshuai Shi, Kun Yang, Jing Yang, and Cong Shen. Transformers as game players: Provable in-context game-playing capabilities of pre-trained models. *Advances in Neural Information Processing Systems*, 37:132001–132049, 2024.

Max Olan Smith, Thomas Anthony, and Michael P Wellman. Learning to play against any mixture of opponents. *Frontiers in Artificial Intelligence*, 6:804682, 2023.

Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: opponent modelling in poker. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pp. 550–558, 2005.

Richard L Tenney and Caxton C Foster. Non-transitive dominance. *Mathematics Magazine*, 49(3): 115–120, 1976.

Haolin Wang, Xueyan Li, Yazhe Niu, Shuai Hu, and Hongsheng Li. Empowering LLMs in decision games through algorithmic data synthesis. In *Will Synthetic Data Finally Solve the Data Access Problem?*, 2025. URL https://openreview.net/forum?id=1RIHEJWN1L.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Kaixuan Xu, Jiajun Chai, Sicheng Li, Yuqian Fu, Yuanheng Zhu, and Dongbin Zhao. DipLLM: Fine-tuning LLM for strategic decision-making in diplomacy. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=hfPaOxDWfI.

Shuai Xu, Sijia Cui, Yanna Wang, Bo Xu, and Qi Wang. Strategy-augmented planning for large language models via opponent exploitation. *arXiv preprint arXiv:2505.08459*, 2025b.

Jian Yao, Weiming Liu, Haobo Fu, Yaodong Yang, Stephen McAleer, Qiang Fu, and Wei Yang. Policy space diversity for non-transitive games. *Advances in Neural Information Processing Systems*, 36: 67771–67793, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023b.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5348–5375, 2024a.

Yadong Zhang, Shaoguang Mao, Tao Ge, Xun Wang, Yan Xia, Wenshan Wu, Ting Song, Man Lan, and Furu Wei. LLM as a mastermind: A survey of strategic reasoning with large language models. In *First Conference on Language Modeling*, 2024b. URL https://openreview.net/forum?id=iMqJsQ4evS.

Yiming Zhang, Athul Paul Jacob, Vivian Lai, Daniel Fried, and Daphne Ippolito. Human-aligned chess with a bit of search. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=bc2H72hGxB.

Richard Zhuang, Akshat Gupta, Richard Yang, Aniket Rahane, Zhengyu Li, and Gopala Anumanchipalli. Pokerbench: Training large language models to become professional poker players. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26175–26182, 2025.

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

APPENDIX

A IMPLEMENTATION DETAILS

A.1 STRATEGIES

Kuhn Poker Agents.

In our implementation, we build a series of rule-based strategy agents based on the game structure of Kuhn Poker. The core idea is to use predefined probability tables conditioned on card strength, player position, and action history to decide whether to bet, call, fold, or check, thereby simulating players with different styles. Each type of agent is controlled by a small set of parameters: GtoAgent uses a parameter α to adjust the bluffing frequency with weak cards, where $\alpha = 1/3$ approximates the equilibrium strategy, smaller values represent more conservative play, and larger values indicate more aggressive play; BluffAgent extends α to higher ranges and allows an additional parameter to control bluffing tendencies after the opponent checks, thereby modeling over-bluffing strategies; ValueAgent and PassiveAgent parameterize the degree of value betting and passive folding, respectively, to represent solid or conservative styles; while AggressiveAgent uses its parameter to determine bluffing frequency with weak cards, modeling highly aggressive play. We also provide natural language opponent descriptions for prompt conditioning. Overall, the strategies exhibit a typical non-transitive countering relationship: Aggressive (bluff-heavy) strategies can exploit Passive strategies, Passive strategies can counter Value strategies, and Value strategies counter Aggressive ones.

Leduc Hold'em Agents.

In the case of Leduc Hold'em, we similarly design a set of rule-based opponent strategies to capture diverse and contrasting playing styles. The baseline LeducGTOAgent follows approximate equilibrium strategies derived from PyCFR, balancing value betting and bluffing across different rounds. On top of this baseline, we introduce parameterized variants: LeducTightAgent emphasizes conservative play by drastically reducing betting or bluffing frequencies with weaker hands, focusing only on strong holdings; LeducLooseAgent represents the opposite style, frequently entering pots and bluffing with weak hands, thereby creating exploitable over-aggression; LeducAggressiveAgent systematically increases betting and raising frequencies regardless of card strength, aiming to apply constant pressure; while LeducPassiveAgent reduces proactive betting and relies mainly on checking and calling, reflecting a cautious and defensive style. We also include a LeducRandomAgent as a baseline without strategic structure. Each agent is controlled by simple parameters (e.g., tightness, looseness, aggression, passiveness) that adjust the relative weights of betting, calling, folding, and checking. For clarity in evaluation, we additionally provide natural language descriptions of these strategies to guide prompt conditioning. Overall, these strategies represent interpretable stylized opponents in Leduc Hold'em, where tight strategies can be exploited by bluffing, loose strategies can be punished by calling more often, aggressive styles are countered by defensive value play, and passive styles are vulnerable to sustained pressure and frequent betting and raising.

A.2 DATASET DETAILS

Kuhn Poker Agent Dataset

In the Kuhn Poker task, we constructed a balanced and diverse dataset by generating match trajectories from three complementary opponent pairings: RandomAgent vs. GTOAgent to introduce non-strategic noise against equilibrium play, GTOAgent vs. GTOAgent to capture distributions close to Nash equilibrium, and BluffAgent vs. Counter-BluffAgent to model systematic over-bluffing and its corresponding counter-strategies. These matchups generated 32k, 64k, and 32k raw samples respectively, from which we selected a final 64k training set with an 8:2 train–validation split (51.2k : 12.8k). Importantly, GOE-LLM and the GTO-LLM baseline were both trained under identical hyperparameter settings, including a batch size of 512. Because checkpoints were saved every 100 steps, one full training run corresponds to processing 51.2k samples, ensuring that both models are evaluated after being exposed to exactly the same amount of data. GOE-LLM constructs a 64k training corpus combining diverse opponent styles, whereas the baseline GTO-LLM dataset (96k total samples) is exclusively generated from GTOAgent vs. GTOAgent interactions.

Leduc Hold'em Agent Dataset

For Leduc Hold'em, we followed a comparable data-generation protocol while expanding the behavioral coverage of strategic deviations. Specifically, the dataset combines samples from RandomAgent vs. GTOAgent (24k), GTOAgent vs. GTOAgent (32k), and TightAgent vs. LooseAgent (24k) matchups, jointly modeling equilibrium play, non-strategic noise, and classical tight–loose exploitation dynamics. From these raw 80k trajectories, we constructed an 80k training dataset with an 8:2 train–validation split (64k : 16k). This ensures that the training distribution does not inadvertently encode cyclical or anti-correlated behavioral patterns, but instead reflects a broad yet stable mixture of strategically interpretable interactions. Consistent with the Kuhn Poker setting, both GOE-LLM and the GTO-LLM baseline were trained with identical optimization hyperparameters, such that each reported checkpoint corresponds to the same effective exposure—64k samples—with the baseline GTO-LLM relying entirely on GTOAgent vs. GTOAgent equilibrium-only data during construction.

Table 5: Composition of training data for Kuhn Poker and Leduc Hold'em. The table reports the generated samples from each matchup and the final selected training set size.

Game	Matchup	Generated Samples	Training Samples(Train:Val)
Kuhn Poker	Random vs. GTO	32,000	64,000(51,200:12,800)
	GTO vs. GTO	64,000	
	Bluff vs. Counter-Bluff	32,000	
Leduc Hold'em	Random vs. GTO	24,000	80,000(64,000:16,000)
	GTO vs. GTO	32,000	
	Tight vs. Loose	24,000	

A.3 Prompt Design

Prompt 1: Kuhn Poker Agent

You are an expert Kuhn Poker player, tasked with making optimal decisions in a two-player simplified poker environment. Your objective is to maximize expected value while reasoning about hidden information and opponent behavior.

Game Rules
Kuhn Poker is a fundamental imperfect-information game with the following properties:
- Deck: 3 cards {J, Q, K}, ranked J $<$ Q $<$ K.
- Initial setup: Each player antes 1 chip and is dealt 1 private card (cards are dealt without replacement).
- Showdown: If both players remain without folding, the player with the higher card wins the pot.

Action Rules
At each decision point, the following actions may be taken (subject to the betting history):
- [Check]: Pass without betting (only if no bet is on the table).
- [Bet]: Add 1 chip to the pot (only if no bet is on the table).
- [Call]: Match an opponent's bet by adding 1 chip to the pot (only if a bet has been made).
- [Fold]: Concede the pot to the opponent (only if a bet has been made).

Decision Context (State Representation)
The state will be described in natural language, including:
- Your role: Player {player_id} ({first} to act this round).
- Private information: Your card = {card}.
- Action history: {history}.
- Legal actions available: {available_actions}.

Output Format
Your response must explicitly include reasoning and action selection:

```
<think> Your thoughts and reasoning </think>
<answer> [ACTION] </answer>
```

- The <think> field should describe your strategic reasoning (e.g., hand strength, bluffing, opponent modeling).
- The <answer> field must contain exactly one action from {available_actions}.

Important Notes
1. You must always provide a reasoning process in the <think> field.
2. Your final choice in <answer> must be strictly one of the available actions.
3. Decisions should consider both exploitative opportunities and minimization of exploitability.

Prompt 2: Leduc Hold'em Agent

You are an expert Leduc Hold'em player, tasked with making optimal strategic decisions in a two-player fixed-limit poker environment. Your goal is to maximize expected utility through well-reasoned betting actions, given private information (your card), public information (community card, betting history), and the opponent's behavior.

Game Rules
Leduc Hold'em is a simplified poker variant with perfect recall and incomplete information:
- Deck: 6 cards (two each of J, Q, K).
- Initial setup: Each player antes 1 chip and receives 1 private card.
- Round 1 (Pre-flop): A betting round begins. Bet size is 2 chips. Maximum of 2 raises per round.
- Round 2 (Post-flop): One public card is revealed. A second betting round begins with fixed bet size of 4 chips. Maximum of 2 raises per round.
- Showdown: If both players remain, the winner is determined by hand strength .
- Pot Limitations: Betting is fixed-limit, ensuring bounded strategy space.

Action Rules
At each decision point, you may choose from the following legal actions (subject to game constraints):
- [Check]: Pass without adding chips (only if no outstanding bet).
- [Bet]: Initiate betting (2 chips pre-flop, 4 chips post-flop; only if no outstanding bet).
- [Raise]: Increase the current bet by the fixed size (2 or 4 chips), only if fewer than 2 raises have occurred this round.
- [Call]: Match the opponent's current bet (only if a bet exists).
- [Fold]: Concede the pot immediately (only if a bet exists).

Decision Context (State Representation)
The current decision state is described in natural language and includes:
- Your role: Player {player_id} ({first} to act this round).
- Private information: Your hole card = {card}.
- Public information: {board_card}.
- Action history: {history}.
- Legal actions available: {available_actions}.

Output Format
Your response must explicitly include both reasoning and action selection:

```
<think> Detailed reasoning about hand strength, betting history,
opponent modeling, and risk/reward tradeoffs. </think>
<answer> [ACTION] </answer>
```

- The `<think>` field should explain why you select a given action.
- The `<answer>` field must contain exactly one action from {available_actions}.

Important Notes
1. Always provide reasoning in the `<think>` section before deciding.
2. Your decision should balance value extraction, bluffing opportunities, and minimization of exploitability.
3. The final output must conform strictly to the format above.
4. You must never output actions not listed in {available_actions}.

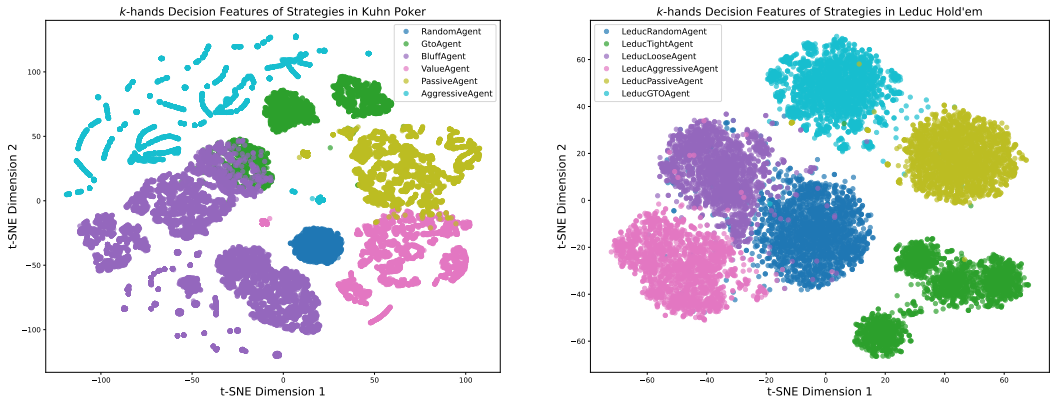
A.4 MLP Profiler Details

The MLP Opponent Profiler is designed to enhance the generalization capability of the LLM Exploiter against unseen opponents. Specifically, we maintain a history of the opponent's behavior trees over the last k games. We extract features from these decision trees to form a fixed-length vector representation. This vector is then fed into a pre-trained MLP classifier that maps it to a discrete opponent type space. Finally, we translate the identified opponent type into a language-based description, which is provided as auxiliary information to the LLM Exploiter. This enables the LLM to adjust its strategy based on the classified opponent type, without having to infer it from raw action histories.

The value of k is a critical hyperparameter that balances the trade-off between responsiveness and stability in opponent modeling. A smaller k allows the profiler to quickly adapt to recent changes in the opponent's strategy, while a larger k provides a more stable and comprehensive view of the opponent's behavior over time but reacts more slowly to sudden strategy shifts.

We visualize the decision feature vectors for different values of k using t-SNE in Figure 7. In Kuhn Poker, when k is small, the distribution of decision vectors is more scattered, making it difficult to distinguish between different opponent types. However, when $k = 10$, the decision vectors form distinct clusters, with clearer boundaries between different opponent types. This clustering effect facilitates the training and generalization of the MLP classifier. For the more complex Leduc Hold'em poker, which has a larger strategy space, a longer history k is needed to capture the opponent's

behavior patterns. We find that when $k = 50$, the decision vectors also form distinct clusters in the t-SNE visualization, aiding in distinguishing between different opponent types.



(a) Kuhn Poker with $k = 10$

(b) Leduc Hold'em poker with $k = 50$

Figure 7: More complex games require longer history k to capture opponent behavior patterns.

Table 6: Confusion matrix across six strategy categories. Most classes show high classification accuracy, with Value(s), Passive(s), and Aggressive(s) achieving near-perfect performance.

True \ Pred	Random	GTO(s)	Bluff(s)	Value(s)	Passive(s)	Aggressive(s)	Accuracy
Random	600	0	0	0	0	0	1.0000
GTO(s)	0	1730	69	0	1	0	0.9611
Bluff(s)	0	61	4739	0	0	0	0.9873
Value(s)	0	0	0	2392	8	0	0.9967
Passive(s)	0	4	0	3	2393	0	0.9971
Aggressive(s)	0	0	0	0	0	2400	1.0000

A.5 Training Parameters

Opponent Profiler.

Table 7: Training parameters for the Opponent Profiler.

Parameter	**Value**
Hidden Layer Sizes	(128, 64)
Activation Function	ReLU
Solver	Adam
L2 Regularization Coefficient (α)	1e-4
Batch Size	64
Learning Rate	1e-3
Maximum Iterations	200
Early Stopping	Enabled
Patience (No Change Tolerance)	10 iterations
Validation Fraction	0.1

LLM Exploiter.

Table 8: Training parameters employed in the LLM Exploiter experiment.

Parameter	Value
Algorithm Advantage Estimator	grpo
Training Batch Size	512
Maximum Prompt Length	1024
Maximum Response Length	512
Overlong Prompt Filtering	False
Truncation Strategy	error
Learning Rate	1e-6
PPO Mini-batch Size	128
PPO Micro-batch Size per GPU	16
KL Divergence Loss	Enabled
KL Loss Coefficient	0.001
KL Loss Type	low_var_kl
Entropy Coefficient	0
Gradient Checkpointing	Enabled
KL in Reward Computation	Disabled
Critic Warmup Steps	0
Rollout Sample Size	5

A.6 Training Environment Details

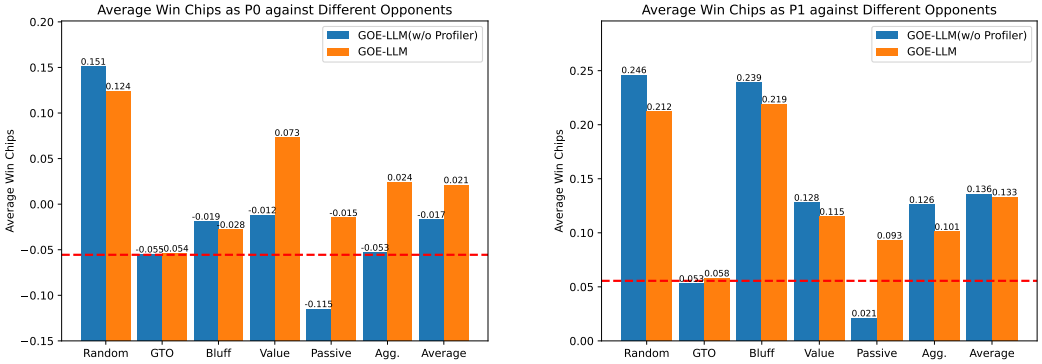
Table 9: Integrated environment configuration for VERL-based GRPO training of the LLM exploiter (hardware, software, memory, distributed training, inference, and data preprocessing).

Category	Parameter	Value
Hardware	GPUs per Node	2
	Number of Nodes	1
	Tensor Model Parallel Size	2
	GPU Memory Utilization	0.9
Software	Training Framework	VERL
	Inference Engine	vLLM
	Logging System	Console + Weights & Biases
Memory	Gradient Checkpointing	Enabled
	Parameter Offload (Actor)	False
	Parameter Offload (Reference)	True
	Optimizer Offload	False
	Remove Padding	True
Distributed	FSDP Parameter Offload	False
	FSDP Optimizer Offload	False
	Micro-batch Size per GPU	16
	Log-Prob Micro-batch Size	32
	Multi-GPU Communication	NCCL
Inference	Inference Backend	vLLM
	Rollout Samples	5
	Memory Management	Dynamic
	Batch Processing	Parallel
Data Preprocessing	Data Loading	Parquet Reader
	Sequence Truncation	Error on Overflow
	Prompt Filtering	Disabled
	Batch Processing	Dynamic Batching

B Ablation Details

B.1 MLP Profiler.

As shown in Figure 8, we can see that GOE-LLM(w/o Profiler) and GOE-LLM perform similarly against seen opponents (e.g., Random, GTO, Bluff), but show clear differences against unseen opponents (Value, Passive, Aggressive). With the MLP Profiler, GOE-LLM generally achieves higher win rates against these unseen opponents, with the most notable improvements observed when P0 plays against Value and Passive opponents. On average, GOE-LLM enhances adaptability to unseen opponents while maintaining its performance against seen ones, demonstrating that the MLP Profiler effectively improves the model's generalization and robustness to unseen opponent types.



(a) Player 0 (P0)

(b) Player 1 (P1)

Figure 8: Critic score curves for LLM exploiters trained with and without the MLP-based opponent profiler, across different player positions (P0 and P1); higher scores are better.

B.2 Mixture of Training Data.

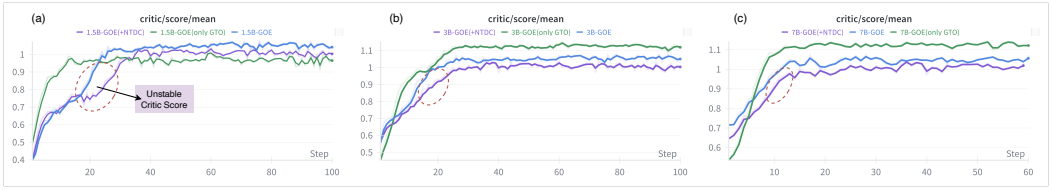


Figure 9: Critic score curves for LLM exploiters trained on different mixture datasets, across different model sizes (1.5B, 3B, 7B), complementing the averaged curves in Figure 3 of the main text.

As shown in Figure 9, the critic score curves across different model sizes (1.5B, 3B, 7B) consistently demonstrate clear differences among the three mixture strategies. For the smallest model (1.5B), GOE-LLM achieves the balance between stability and performance, with faster improvement and higher final scores compared to GOE-LLM(only GTO), while GOE-LLM(+NTDC) suffers from severe fluctuations and degraded performance due to the instability introduced by non-transitive dominance cycles. At the 3B scale, GOE-LLM remains the most effective, outperforming the other two baselines; although the negative impact of NTDC is partially mitigated relative to 1.5B, the corresponding curve still exhibits noticeable instability. At the largest scale (7B), GOE-LLM continues to deliver the highest and most stable critic scores, but the performance gap with GOE-LLM(+NTDC) narrows, suggesting that larger models have stronger robustness to complex, cyclic opponent data. In contrast, only GTO remains consistently inferior across all model sizes, highlighting the importance of diversified yet principled data mixtures for effective exploitation learning.

C Additional Results on GTBench

This section provides additional views of the GTBench experiments in Section 4.4. All settings, methods, and significance tests follow the main text unless stated otherwise.

Comparison with self-consistency. Complementing Figure 6 in the main text, Figure 10 reports both methods for every setting. GOE-LLM gains in imperfect-information games with a single sample per move, whereas SC gains in perfect-information games and grows with K.

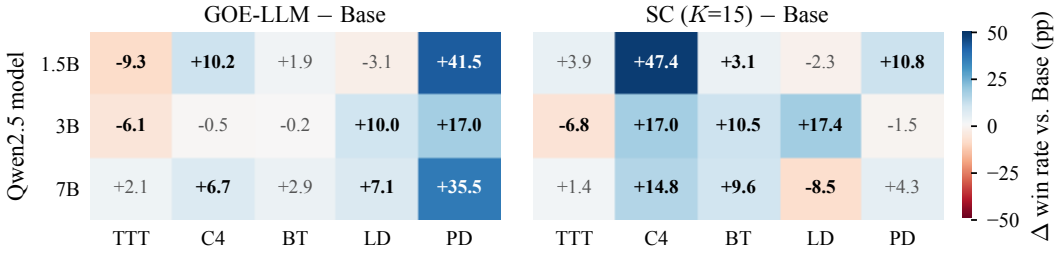


Figure 10: Win-rate change over Base (pp) for each model size and game. Bold entries are significant (two-proportion z-test, Holm-corrected). TTT: Tic-Tac-Toe, C4: Connect-4, BT: Breakthrough (perfect information); LD: Liar's Dice, PD: Prisoner's Dilemma (imperfect information). GOE-LLM gains concentrate in LD and PD, while SC (K=15) gains concentrate in C4 and BT.

Effect of model size. Figure 11 shows the win rate of each method as a function of model size. In Prisoner's Dilemma, GOE-LLM is above Base and SC at every size. In Liar's Dice, GOE-LLM is above Base at 3B and 7B and reaches the highest win rate at 7B. In perfect-information games, GOE-LLM tracks Base, while SC is above both in Connect-4 and Breakthrough at 3B and 7B.

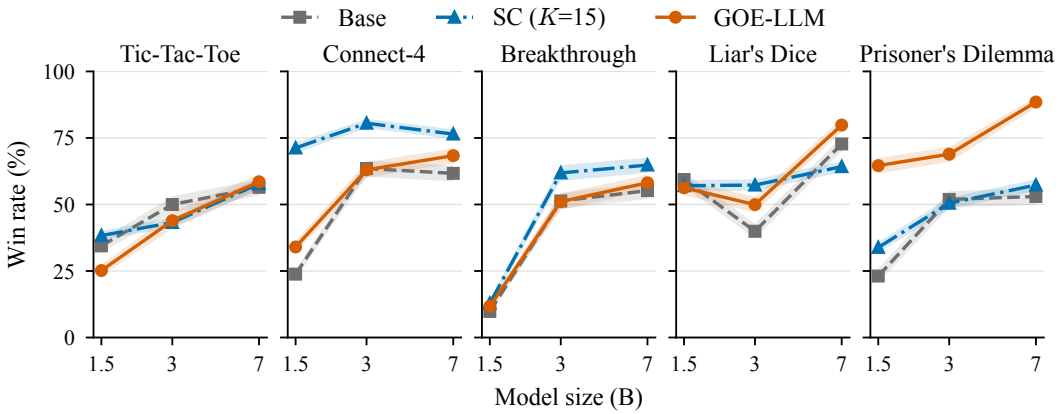


Figure 11: Win rate vs. model size on each GTBench game. Shaded bands are 95% Wilson intervals.

Inference budget. Figure 12 compares GOE-LLM and Base, both with one sample per move, against SC with $K \in \{3, 7, 11, 15\}$ samples. In Liar's Dice and Prisoner's Dilemma, GOE-LLM is above SC at every K. Averaged over the five games, GOE-LLM (54.8%) matches SC with $K = 15$ (55.1%) at $1/15$ of the inference cost, using one sample per move instead of fifteen.

Legality vs. strategy. Figure 13 splits each win-rate change into a legality term (fewer aborted games) and a strategy term (higher win rate among completed games). GOE-LLM's gains in Liar's Dice and Prisoner's Dilemma are almost entirely strategic. SC's gains in imperfect-information games, and its largest gain overall (1.5B Connect-4), come mostly from fewer illegal moves.

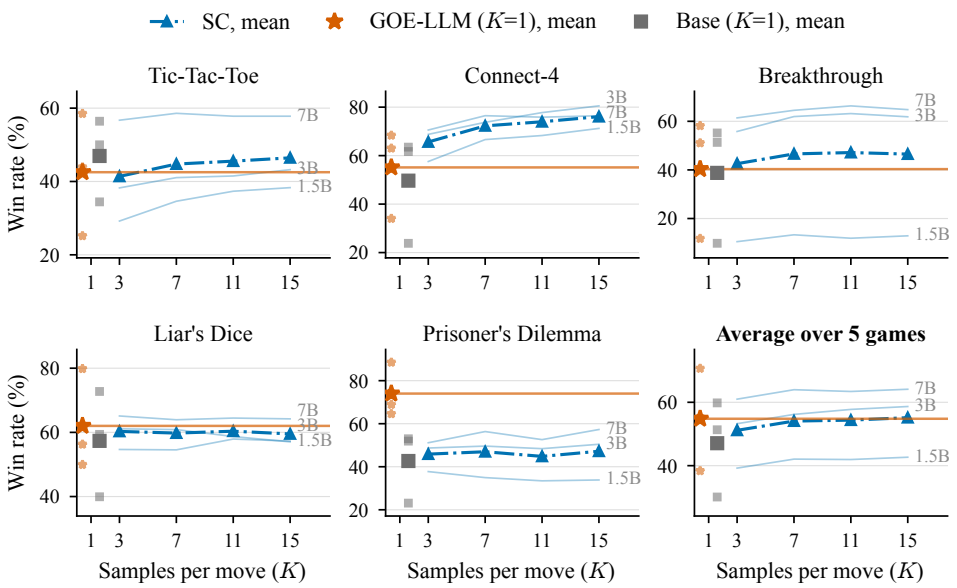


Figure 12: Win rate vs. samples per move K on each GTBench game. Thick markers are means over the three model sizes; thin lines and small markers are individual model sizes.

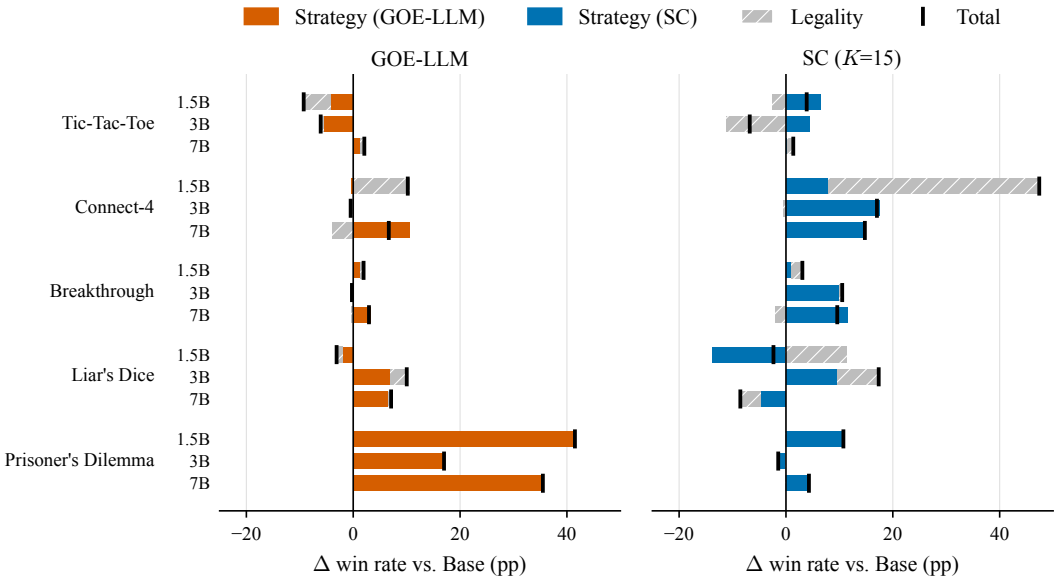


Figure 13: Decomposition of the win-rate change over Base into legality and strategy terms, using win rate = completion rate × win rate among completed games. Black ticks mark the total change.

Game outcomes. Figure 14 shows the share of wins, draws, losses, and aborted games for each method, averaged over the five games. At every model size, GOE-LLM wins more often than Base; at 7B it also wins more often than SC with $K = 15$, with $15\times$ fewer samples.

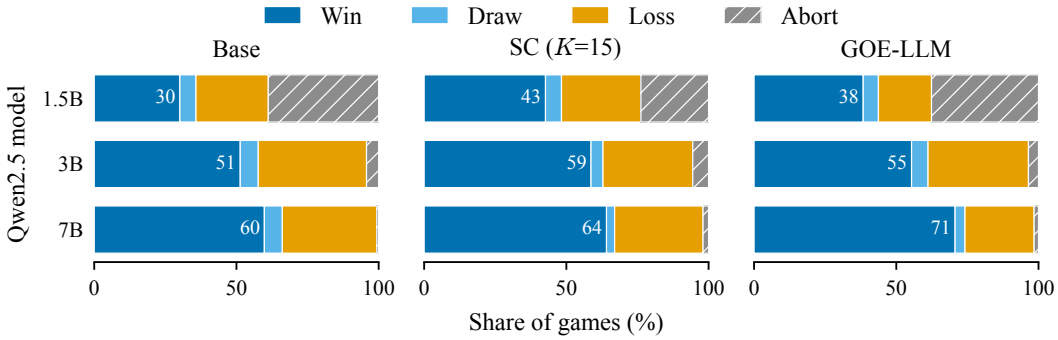


Figure 14: Outcome shares averaged over the five GTBench games. Abort denotes games ended by an illegal LLM move, counted as losses in the win rate, as in the main text.