SpinGPT: A Large-Language-Model Approach to Playing Poker Correctly

Narada Maugin and Tristan Cazenave

LAMSADE, Université Paris Dauphine - PSL, CNRS, Paris, France

Abstract. The Counterfactual Regret Minimization (CFR) algorithm and its variants have enabled the development of pokerbots capable of beating the best human players in heads-up (1v1) cash games and competing with them in six-player formats. However, CFR's computational complexity rises exponentially with the number of players. Furthermore, in games with three or more players, following Nash equilibrium no longer guarantees a non-losing outcome. These limitations, along with others, significantly restrict the applicability of CFR to the most popular formats: tournaments. Motivated by the recent success of Large Language Models (LLM) in chess and Diplomacy, we present SpinGPT, the first LLM tailored to Spin & Go, a popular three-player online poker format. SpinGPT is trained in two stages: (1) Supervised Fine-Tuning on 320k high-stakes expert decisions; (2) Reinforcement Learning on 270k solver-generated hands. Our results show that SpinGPT matches the solver's actions in 78% of decisions (tolerant accuracy). With a simple deep-stack heuristic, it achieves 13.4 ± 12.9 BB/100 versus Slumbot in heads-up over 30,000 hands (95% CI). These results suggest that LLMs could be a new way to deal with multi-player imperfect-information games like poker.

Keywords: Imperfect-information games · Computer poker · No-limit Texas hold'em · Large language model · Supervised fine-tuning · Reinforcement learning.

1 Introduction

Poker is a family of card games, with no-limit Texas hold'em[1] being its most widespread variant. Originating in the United States in the early nineteenth century, poker has attracted tens of millions of players worldwide, with televised poker broadcasts using hole-card cameras that reveal players' cards to viewers and online platforms accelerating its spread since the early 2000s. Unlike games of complete information such as chess, poker is a game of incomplete information:

[0] Interactive research demo to play against SpinGPT : `https://spingpt.lamsade.fr/`

[1] Rules: `https://en.wikipedia.org/wiki/Texas_hold%27em`. Glossary: `https://en.wikipedia.org/wiki/Glossary_of_poker_terms`.

each player holds private cards unknown to their opponents. Moreover, bet sizes are flexible. As a result, the game tree is immense, ranging from 10^{13} nodes in heads-up (HU) limit to over 10^{160} in heads-up no-limit, and even more as the number of players increases, making an exhaustive search infeasible [13].

Accordingly, poker has for many decades served as a testbed for game theory and artificial intelligence [7,8]. Substantial progress has been made, leading to programs that compete with—and in some cases surpass—top human players in specific variants. However, most progress has concerned two-player cash games; tournament play differs fundamentally because the number of players and stack sizes change as the game progresses. Furthermore, maximizing chip expected value (cEV) is no longer optimal, as the value of chips is non-linear: doubling one's stack does not double one's chances of winning, rendering the mere accumulation of chips suboptimal[2]. As a consequence, hands played can no longer be considered independent and identically distributed and the algorithms used must be reconsidered. A popular format illustrating these difficulties is the Spin & Go, a hyper-turbo three-player mini-tournament. Although simpler than a large tournament, this format already introduces the dynamics of varying player counts and stack sizes.

2 Related Work

Before regret-minimization methods, computer poker primarily relied on knowledge-based rules, heuristic evaluation, and early equilibrium-driven abstractions. These approaches achieved competent play in constrained variants but struggled with the full game's combinatorial complexity, conceding they were "nowhere close to solving full-scale poker" [1].

A major breakthrough came with the introduction of the Counterfactual Regret Minimization (CFR) algorithm in 2007 [21]. This algorithm and its subsequent variations allow an agent to approximate an optimal strategy by learning to minimize the regret associated with suboptimal decisions. In 2015, the Cepheus program computed a near-perfect Nash equilibrium strategy for heads-up limit hold'em, thereby weakly solving this variant [2]. Two years later, in the far more complex no-limit variant, two independent programs achieved significant results. DeepStack [16] defeated professional players in an online match, and shortly after, Libratus [5] surpassed some of the world's top professionals in a live casino match. These programs relied on state-space abstractions (such as treating the ace and king of hearts similarly to the ace and king of spades pre-flop) and real-time subgame solving techniques based on improved versions of CFR.

Following these successes in one-on-one settings, researchers turned to the harder problem of multi-player poker. This step was taken in 2019 with Pluribus [6], which defeated professionals in six-player no-limit Texas hold'em, though

[2] This non-linearity is often modeled by the ICM (Independent Chip Model). It formalizes the fact that as the tournament progresses, the strategy evolves to place greater emphasis on survival and reaching the money.

one top professional finished roughly even. Although its strategy computation required substantial resources, this was the first AI to surpass human experts in a multi-player setting. To overcome the remaining computational difficulties, new approaches have emerged [3,18], but without exceeding the performance of the CFR paradigm.

Using a large language model for poker might appear counter-intuitive, because even frontier models have proven incapable of playing competently through direct prompting (zero-shot or few-shot): their pre-training corpora contain very few high-quality hand histories compared with games such as chess. This is why competitive LLM play in chess is viable, as evidenced by the tournament organized by Kaggle and Google [14]. However, fine-tuning has proved effective at closing this gap [20]. PokerGPT [11] followed this route by fine-tuning an LLM, but it was limited by its training data, which consisted exclusively of spectator data from low-stakes games (NL5). This meant that players' private cards were often unknown, and the dataset was biased by the flawed strategies of weak players. Our work directly addresses these shortcomings.

3 Methodology: A Large-Language-Model Approach

Our hypothesis is that a properly trained LLM can play poker effectively, owing to (i) broad prior knowledge (rules, basic strategy, probabilities) and (ii) facility with learning the syntax and semantics of structured representations.

Our methodology follows a two-stage training process: first, a Supervised Fine-Tuning (SFT) phase to teach the model the language and the strategy of a professional poker player, followed by a Reinforcement Learning (RL) phase using ORPO (Optimistic Model Rollouts for Pessimistic Offline Policy Optimization) algorithm [17] on solver-generated hands to refine its strategy toward game-theoretic optimality.

3.1 Model Architecture

SpinGPT is based on Llama-3.1-8B-Instruct (Meta, Llama 3.1 Community License), an 8-billion parameter open-weight language model from Meta. This model represents a trade-off between representational capacity—necessary for modeling complex poker strategies—and computational requirements, making it tractable on accessible hardware infrastructure. This model demonstrated performance comparable to or superior to that of other models of similar size [9], leading us to believe in the potential of the 8B model to achieve state-of-the-art results for our task. We also conducted tests with a smaller model, Llama-3.2-1B-Instruct.

3.2 Data and Preprocessing

We constructed a dataset comprising 320,000 poker decisions from 8,800 Spin & Go games. This data was sourced from one of this paper's authors, Narada

Maugin, and corresponds to hands he played professionally on the PartyPoker.fr platform between 2018 and 2020. The games were played at high-stakes buy-ins of €50, €100, and €250.

The raw hand history files have been anonymized. Because they contain superfluous information such as timestamps and table chat, we parsed them to retain only strategically relevant fields (stacks, positions, actions, public and private cards). Throughout, we express all stack sizes in big blinds (BB). We then transformed each decision point into a structured, text-based "instruction" prompt that the LLM can read. For instance, a typical game situation is encoded as the following instruction: `"pos:H=BTN stacks:H=29.3,BB=1.7, SB=19.0 hand:TsQs | pre:H r2,SB c,BB f | flop:4h7s6c SB b1,H c |turn: 8d SB b1,H c | river:9c SB b1 H:"`. This structured encoding contains all essential information. Indeed, in this example, the following are present: player positions (Hero denotes the focal player whose private cards and decisions are logged. Here, Hero is on the button), stack depths (29.3 BB for Hero, 1.7 BB for the big blind player, and 19.0 BB for the Small Blind player), Hero's cards (TsQs, meaning Ten and Queen of spades), and the sequence of actions on each street. The model, receiving this description as input, must provide the appropriate action as output (r6.5 in the example, meaning raise to 6.5 BB). The set of all possible actions was represented by a compact vocabulary (c for call, f for fold, x for check, a for all-in, r{amount} and b{amount} for raise and bet of a certain amount, respectively), which was integrated into the model's tokenizer. Importantly, the model is not restricted to a predefined set of actions and bet sizes; it can generate any tokens and therefore any numerical value for {amount}.

3.3 Supervised Fine-Tuning

Then, we fine-tuned the model on our poker dataset using standard supervised learning via the LLaMA-Factory framework [19]. We employed Low-Rank Adaptation (LoRA) [10] to drastically reduce training costs. This method involves adding a few trainable parameters (approximately 10 million in our case) to modulate the original weights. Our main hyperparameters were as follows: each LoRA adapter used rank $r = 8$ and scaling factor $\alpha = 16$ (no dropout, no additional bias weights). Training ran for four epochs on sequences of up to 128 tokens, starting from a learning rate of 5×10^{-5} that decayed with a cosine schedule. The entire fine-tuning took 10 hours on a single NVIDIA A100 (40 GB) in FP16 precision. We refer to this model as SpinGPT-SFT.

3.4 Offline Reinforcement Learning

While effective, the model produced by SFT (SpinGPT-SFT) merely replicates a specific human strategy from several years ago, inheriting its potential biases and weaknesses. To transcend imitation and move toward an optimal strategy, a second training phase was implemented to align the model with a Game Theory Optimal (GTO) Nash equilibrium.

To achieve this, we chose an offline RL approach. We leveraged an external, CFR-based GTO solver named InstaGTO to provide near-optimal decisions. Therefore, we used InstaGTO to create a dataset of 270,000 synthetic hands, covering a wide array of post-flop situations (1/3 are HU, 1/3 are SBvBB in 3-way, 1/6 are SBvBTN in 3-way, and 1/6 are BBvBTN in 3-way). A significant challenge with solver-generated data is its sterile, perfect nature, which differs from real-world play (e.g., it lacks pre-flop action, only covers two-player post-flop scenarios, involves specific stack depths, and contains no idiosyncratic human errors). Training exclusively on this data could lead to catastrophic forgetting of the robust patterns learned during SFT. To mitigate this, we created a mixed dataset by retaining 50,000 hands from the original human dataset.

This combined dataset of 320,000 hands was used to refine the model with the ORPO (Optimistic Model Rollouts for Pessimistic Offline Policy Optimization) algorithm [17]. ORPO is suited for this task as it combines SFT with preference alignment. Rather than simply maximizing a reward signal like Expected Value (EV), ORPO adjusts the model's policy to increase the likelihood of the solver's action (we used the solver's argmax-EV action as the positive label) relative to other suboptimal actions.

This final stage, run also with LLaMA-Factory, added 10 GPU-hours on an NVIDIA A100. It reused mainly the SFT LoRA configuration and applied QLoRA to the base model (4-bit NF4 with double quantization). We optimized with ORPO ($\beta = 0.1$) for two epochs on sequences up to 128 tokens, using a learning rate of 2×10^{-5}; the optimizer ran in BF16. The model resulting from this two-stage training pipeline is our final agent, SpinGPT.

4 Experimental Results

To evaluate our models, we use two test sets that were held out during training: a Professional test set, which contains 32,000 decisions from real hands played by the human expert, and a Solver test set, which includes 30,000 decisions generated by the InstaGTO solver across various post-flop configurations.

4.1 Evaluation Metrics

We first measure performance using two types of accuracy. Exact Accuracy is the percentage of predictions that perfectly match the ground-truth action, including both the action type and the exact amount (e.g., 'r4.6'). As a more lenient alternative, we also report Tolerant Accuracy, which considers a prediction correct if the action type matches and the bet or raise amount is within a ± 0.5 BB tolerance of the true value. For instance, if the model predicts a bet of 1.5 BB (b1.5) while the true action was a bet of 1.6 BB (b1.6), the prediction is counted as correct under this metric.

To measure how well the model predicts each poker action class (*bet*, *call*, *fold*...), we report the macro F_1 score. For every action class c, we first compute an F_1 score: the harmonic mean of precision (the fraction of predicted actions of class c that are correct) and recall (the fraction of true actions of class c that are recovered). We then take the mean of these per-class scores to obtain the macro average.

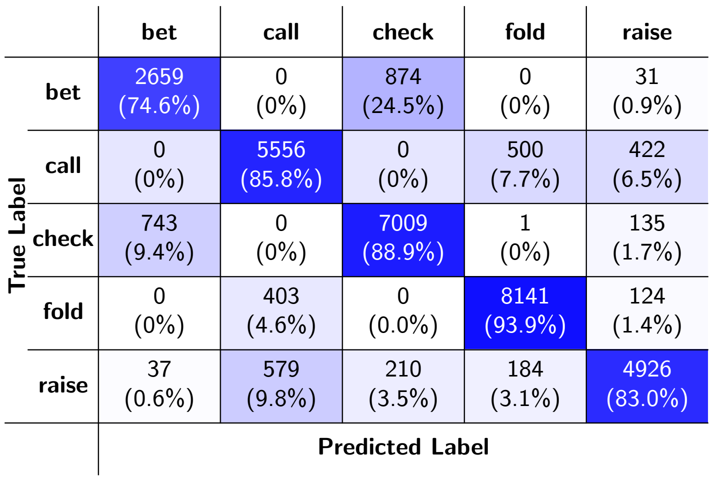
Finally, neither accuracies nor the macro F_1 score evaluates the quantitative correctness of bet sizing. For instance, if the expected move is bet 1 BB (b1) but our model gives bet 5 BB (b5), it falls into the 'bet' class but is quite far from the expected value. So, to evaluate the accuracy of bet and raise amounts, we use two other metrics. The mean absolute error (MAE), expressed in BB, and the mean absolute percentage error (MAPE), expressed as a percentage. A lower MAE and MAPE indicate better calibration of bet sizes.

4.2 Imitation performance of the SFT Model

The initial model, SpinGPT-SFT, was first evaluated on its ability to imitate the professional player. As shown in Table 1, it achieves an exact accuracy of 80%. Its macro F_1 score of 86% indicates an ability to classify all types of actions correctly. For comparison, PokerGPT, another LLM-based agent, achieved a macro F_1 score of 78% on human player data.

It is critical to note that a 100% score is theoretically unattainable. An expert's strategy is inherently stochastic; to remain unpredictable, a player employs mixed strategies, such as raising with a certain hand 70% of the time and calling 30% of the time in the exact same situation, and adapts to their opponent, playing differently against an amateur or professional player. This is compounded by noise from human errors.

Fig. 1: Confusion matrix of SpinGPT-SFT on the Professional test set.



True Label \ Predicted Label	bet	call	check	fold	raise
bet	2659 (74.6%)	0 (0%)	874 (24.5%)	0 (0%)	31 (0.9%)
call	0 (0%)	5556 (85.8%)	0 (0%)	500 (7.7%)	422 (6.5%)
check	743 (9.4%)	0 (0%)	7009 (88.9%)	1 (0%)	135 (1.7%)
fold	0 (0%)	403 (4.6%)	0 (0.0%)	8141 (93.9%)	124 (1.4%)
raise	37 (0.6%)	579 (9.8%)	210 (3.5%)	184 (3.1%)	4926 (83.0%)

The confusion matrix (Figure 1) shows high action-match, correctly identifying actions most of the time, especially when folding (94% of cases). More importantly, the model produces a very small number of illegal moves (1.4% overall). For example, in 135 instances where the correct action was check, the model predicted raise. A raise is only legal in response to a prior bet, so this prediction is impossible. This suggests the model has confused the token for a bet with the token for a raise. These types of errors, which account for a small fraction of predictions, can be easily corrected in deployment by mapping the invalid prediction to its closest legal semantic equivalent (e.g., mapping a predicted raise to a bet, or a predicted call to a check, when no prior bet exists).

Table 1: Comparison of the performance of the versions on the Professional and Solver test set.

Model	**Dataset**	**Exact Accuracy (%)**	**Tolerant Accuracy (%)**	**Macro F_1 (%)**	**MAE (BB)**	**MAPE (%)**
SpinGPT-SFT	Professional	80	84	86	0.61	19
SpinGPT	Professional	79	83	84	0.30	12
SpinGPT	Solver	72	78	77	0.93	35

4.3 Performance of the final SpinGPT model

After reinforcement learning, the final SpinGPT model shows slightly different results. On the Professional test set, its imitation accuracy drops to 79%, and its macro F_1 score to 83% (table 1). This small decrease does not indicate a regression, but rather suggests that the model is no longer simply copying human actions—it has started to adjust its behavior based on solver feedback. The MAE improved from 0.61 BB to 0.30 BB, a reduction that yields predictions within one-third of the minimum bet (1 BB).

On the Solver test set, performance is lower across all metrics: macro F_1 = 77%, exact accuracy = 72%, tolerant accuracy = 78%, MAE = 0.93 and MAPE = 0.35. This is expected, as GTO policies rely on finely balanced mixed strategies and precise bet sizes, which are difficult to understand and implement.

4.4 Gameplay Performance Evaluation

While imitation metrics confirm the model's ability to learn a strategy, they do not measure its effectiveness in actual gameplay. To assess this, we conducted two head-to-head confrontations.

Confrontation with a Benchmark Agent. We benchmarked SpinGPT-SFT against Slumbot [12], the 2018 AAAI Computer Poker Competition champion. Slumbot is designed for heads-up cash games with a constant 200 BB stack, whereas SpinGPT-SFT targets Spin & Go tournaments that begin 3-handed at 25 BB and transition to HU as stacks evolve. Despite this mismatch, Slumbot's public availability, well-documented API, and published results make it the strongest standardized reference—no high-performance bot exists for the 3-player tournament format to our knowledge.

The supervised model initially lost heavily because it shoved all-in at 200 BB far too often—an action that is correct at 25 BB and less but catastrophic at deep stacks. This failure mode reflects the absence of 200 BB examples in the training data. To obtain a fair baseline we applied a minimal heuristic: each all-in bet was replaced by a 2/3 pot bet if the agent was the first to bet, or a raise to 3x the opponent's bet otherwise. This preserves the aggressive intent of the model while avoiding losing the entire stack from time to time.

With this patch, over 30,000 hands, SpinGPT-SFT achieved a win rate of 13.4 ± 12.9 BB/100 (95% CI)[3]. Table 2 situates this result among other state-of-the-art research agents. Results from prior work are cited from the original papers; confidence interval (CI) computation methods may differ across papers.

Table 2: Comparison of win rates against the Slumbot benchmark agent.

Agent	**Win Rate** (BB/100)	**95% CI** (BB/100)	**Hands Played**	**Year**
Baby Tartanian8 [4]	3.6	± 1.2	N/A	2016
ReBeL [3]	4.5	± 1.0	N/A	2020
AlphaHoldem [18]	11.2	± 1.6	100 000	2022
PokerGPT [11]	15.8	± 4.9	10 000	2024
SpinGPT-SFT (Ours)	13.4	± 12.9	30 000	2025

Confrontation between our two agents. To quantify the improvement provided by the reinforcement learning phase, we conducted a head-to-head match between the final SpinGPT model and the SFT-only model (SpinGPT-SFT). The match was played over 10,000 hands in a duplicate format (where players switch cards after each hand to reduce variance) at a 25 BB stack depth. The final SpinGPT model achieved a win rate of 13.2 ± 7.24 BB/100 against the SFT-only version. This result shows that the reinforcement learning phase brings about a very significant improvement in game performance.

[3] The BB/100 metric is the most commonly used in poker. It represents the number of big blinds won per 100 hands on average. The basic strategy, which consists of always folding, is -75 BB/100. Two equivalent players will be at 0 BB/100, and one player is clearly winning against another starting at 6 BB/100.

5 Conclusion and Discussion

In this work, we introduced SpinGPT, an LLM-based agent for the Spin & Go poker format. We employed a two-stage training pipeline combining Supervised Fine-Tuning on professional hand histories with Reinforcement Learning using a GTO solver.

The SFT variant (SpinGPT-SFT) is trained on 320k decisions and outperformed the benchmark pokerbot Slumbot after just one heuristic all-in patch. This is notable given its deep-stack errors like limping—reasonable at ≤ 25 BB but a mistake at 200 BB. We then applied RL with ORPO to 270k solver-generated hands. The final agent, SpinGPT, achieves 78% tolerant accuracy on solver actions and clearly beats SpinGPT-SFT.

However, our work has limitations. First, SpinGPT has not yet faced strong human opposition in tournament play; this is the primary route to establish SpinGPT's true level. Future tests should instead target human opponents or bots specialized in Spin & Go (or at least heads-up with shallow stacks) to better gauge SpinGPT's actual strength. We relied on imitation, but it is difficult to know what this represents in real play. Second, the final model has not been evaluated against Slumbot because large-scale matches are costly and the benchmark format does not align. Third, as an LLM, SpinGPT can exhibit typical language-model errors and hallucinations—we observed numerical mistakes such as treating "5.11" as greater than "5.2".

Ethical and environmental considerations To reduce misuse risk, we publicly release neither the model weights nor the full training data; unrestricted release could enable terms-of-service-violating poker bots on real-money platforms. However, we provide access to these materials to researchers upon request. On the environmental side, counting both training and evaluation games, our total compute footprint is estimated at 5.2 kgCO2e using Green Algorithms [15].

Future work Future work could advance on several fronts. Immediate improvements include the use of more powerful foundation models and even higher-quality hands. In addition, a Retrieval-Augmented Generation (RAG) module could query pre-computed GTO charts on demand, ensuring optimal preflop and push-or-fold actions. To enable deeper reasoning, computationally heavier techniques such as chain-of-thought prompting could also be explored. Adaptivity is another key objective: training on data labeled by opponent skill—or letting the LLM infer tendencies from hand history—could teach the agent to adjust its style and exploit weaknesses. Beyond raw performance, explainability is another avenue. When trained on curated human data, an LLM can behave in a human-like, intuitive manner. It could therefore generate natural-language explanations of strategic choices and opponent tendencies—useful for match commentary and personalized coaching. Finally, coupling the LLM with a solver (that tackles only heads-up post-flop) supplies a GTO baseline, while the LLM governs every other spot—and even chooses to deviate from that baseline whenever exploitation seems profitable. By pursuing these directions, it may be possible for LLM-based agents to reach expert-level performance in tournaments, suggesting the potential of language models to tackle complex, imperfect-information games.

References

1. D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. *Artificial Intelligence*, 134(1):201–240, 2002.
2. M. Bowling, N. Burch, M. Johanson, and O. Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.
3. N. Brown, A. Bakhtin, A. Lerer, and Q. Gong. Combining deep reinforcement learning and search for imperfect-information games. In *Advances in Neural Information Processing Systems 34*, pages 17057–17069. Curran Associates Inc., 2020.
4. N. Brown and T. Sandholm. Baby tartanian8: Winning agent from the 2016 annual computer poker competition. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 4238–4239. AAAI Press, 2016.
5. N. Brown and T. Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.
6. N. Brown and T. Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.
7. C. Ferguson and T. Ferguson. On the borel and von neumann poker models. *Game theory and applications*, 9:17–32, 2003.
8. N. Findler. Studies in machine cognition using the game of poker. *Communications of the ACM*, 20(4):230–245, 1977.
9. A. Grattafiori, A. Dubey, A. Jauhri, S. Pandey, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
10. E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
11. C. Huang, Y. Cao, Y. Wen, T. Zhou, and Y. Zhang. Pokergpt: An end-to-end lightweight solver for multi-player texas hold'em via large language model. *arXiv preprint arXiv:2401.06781*, 2024.
12. E. Jackson. Slumbot nl: Solving large games with counterfactual regret minimization using sampling and distributed processing. In *Computer Poker and Imperfect Information*, pages 35–38. AAAI Press, 2013.
13. M. Johanson. Measuring the size of large no-limit poker games. *arXiv preprint arXiv:1302.7008*, 2013.
14. Kaggle. Chess text input leaderboard. `https://www.kaggle.com/benchmarks/kaggle/chess-text`, 2025. Benchmark page.
15. L. Lannelongue, J. Grealey, and M. Inouye. Green algorithms: Quantifying the carbon footprint of computation. *Advanced Science*, 8(12):2100707, 2021.
16. M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.
17. Y. Zhai, Y. Li, Z. Gao, X. Gong, K. Xu, D. Feng, D. Bo, and H. Wang. Optimistic model rollouts for pessimistic offline policy optimization. *arXiv preprint arXiv:2401.05899*, 2024.

18. E. Zhao, R. Yan, J. Li, K. Li, and J. Xing. Alphaholdem: High-performance artificial intelligence for heads-up no-limit poker via end-to-end reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4):4689–4697, 2022.
19. Y. Zheng, R. Zhang, J. Zhang, Y. Ye, and Z. Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410. Association for Computational Linguistics, 2024.
20. R. Zhuang, A. Gupta, R. Yang, A. Rahane, Z. Li, and G. Anumanchipalli. Pokerbench: Training large language models to become professional poker players. *arXiv preprint arXiv:2501.08328*, 2025.
21. M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 21*, pages 1729–1736. Curran Associates Inc., 2007.