Extended Abstract

Motivation Poker is a game where players must make decisions without knowing all information, requiring strategic thinking and deception. Current AI poker systems like DeepStack work well but cannot adapt to new opponents in real time or learn to bluff like humans do. We want to build AI agents that can learn to bluff effectively, predict opponent behavior, and improve their strategy based on feedback. Our main question is: Can AI agents learn to bluff strategically through reinforcement learning combined with opponent modeling and language-based feedback?

Method We built a training system with four main parts. First, we use curriculum learning where the AI practices against increasingly difficult opponents - starting with random players, then medium-skilled bots, and finally strong learning agents. Second, we added a transformer model that tries to predict what the opponent will do next based on their past actions. Third, we created a reward system that gives extra points when the AI successfully bluffs (bets aggressively with weak cards and wins). Fourth, we use an LLM (Mistral Medium) to analyze the AI's play style using poker statistics like VPIP and PFR, then give strategic advice that gets embedded into the AI's decision-making process.

Implementation We used the PettingZoo Texas Hold'em environment for two-player poker games. The main AI agent uses a two-layer neural network trained with Actor-Critic reinforcement learning. The opponent modeling component is a separate transformer that runs alongside the main agent and adds its predictions to the observation space. We detect bluffs by running Monte Carlo simulations to estimate hand strength - if the agent bets aggressively with weak estimated hands, we classify it as a bluff. The LLM feedback system calculates poker statistics every episode, sends them to Mistral Medium for strategic advice, and embeds the feedback using sentence transformers. We also tested adding the LLM's adherence scores as additional rewards.

Results Our experiments show clear improvements across multiple metrics. Bluff reward tuning found that intermediate reward values (around 2.0) work best, achieving 80.5% win rate and 20.9% bluff success rate against strong opponents. Opponent modeling increased win rates against strong opponents by 56.69% and improved bluff success rates significantly, though prediction accuracy varied between runs. Curriculum learning outperformed static training across all opponent levels, with the curriculum-trained agent achieving 65.2% win rate compared to 52.4% for the best static approach. LLM feedback integration led to more disciplined play, reducing aggressive metrics like AFq from 80% to 27%, and improved win rates against strong opponents by 20.65%. However, the adherence reward mechanism failed completely, eliminating bluffing behavior entirely.

Discussion The results show that our approach successfully teaches AI agents strategic deception and adaptation. The bluff reward system encourages calculated risk-taking rather than random aggression. Opponent modeling enables dynamic strategy adjustment based on predicted opponent behavior. Curriculum learning builds robust strategies that transfer well to new situations. LLM feedback helps refine play style using professional poker concepts. However, the adherence reward failure highlights challenges in combining natural language feedback with reinforcement learning - poker's randomness makes strategic advice from previous games less relevant to new situations. The wide variance in opponent modeling accuracy also suggests this component needs more investigation.

Conclusion We demonstrated that AI agents can learn sophisticated poker strategies through behavioral incentives, opponent modeling, and structured training progression. Our system produces agents capable of strategic bluffing and real-time adaptation to opponent behavior. Key findings include: optimal bluff rewards encourage effective deception, opponent modeling enables strategic adaptation, curriculum learning beats static training, and LLM feedback can guide strategic development when properly integrated. Future work should focus on more robust adherence mechanisms, improved opponent modeling consistency, and applications to other strategic domains involving deception and negotiation.

Bluffing with Precision: LLM-Guided Strategy and Opponent Modeling in Multi-Agent Poker

Sara Kothari
Department of Computer Science
Stanford University
sarako@stanford.edu

Yanny Gao
Department of Computer Science
Stanford University
rgao1218@stanford.edu

Abstract

We develop a reinforcement learning framework for training poker AI agents that can bluff strategically and adapt to opponent behavior in real-time. Our system combines four key components: curriculum learning with progressively stronger opponents (random $\rightarrow$ medium $\rightarrow$ strong), a transformer-based opponent modeling module that predicts opponent actions, a bluff-aware reward system that incentivizes strategic deception with weak hands, and natural language feedback integration using Mistral Medium to provide strategic advice based on poker statistics. Experiments in PettingZoo Texas Hold'em show significant performance improvements: bluff reward tuning achieves 80.5% win rate with 20.9% bluff success, opponent modeling increases win rates against strong opponents by 56.69%, curriculum learning outperforms static training (65.2% vs 52.4% win rate), and LLM feedback reduces overly aggressive play while improving strategic performance by 20.65% against strong opponents. While the adherence reward mechanism failed due to incompatibility between natural language feedback and RL optimization, our results demonstrate that AI agents can learn sophisticated deception strategies through structured behavioral incentives and opponent adaptation, with applications extending to broader strategic domains involving negotiation and competitive multi-agent systems.

1 Introduction

Poker is a benchmark environment for studying decision-making under uncertainty. As a canonical partially observable and imperfect-information game, it demands that players reason strategically, conceal their intentions, and bluff convincingly. These characteristics make poker an ideal testbed for developing AI agents capable of navigating ambiguity, deception, and opponent modeling. While systems like DeepStack and Pluribus have achieved superhuman performance in no-limit Texas Hold'em, they rely on offline training and domain-specific abstractions that limit real-time adaptability, behavioral nuance, and interpretability—especially in human-style deception and reactive strategy shifts.

This project investigates a central question in multi-agent reinforcement learning: *Can AI agents learn to bluff strategically and adapt to opponent behavior in real time through interactive learning, modeling, and feedback?* We aim to build poker agents that do more than win—they should bluff selectively, adapt to diverse adversaries, and respond to high-level behavioral guidance.

To address this, we design a modular training framework that integrates reinforcement learning with behavioral feedback, transformer-based opponent modeling, and language-guided reward shaping. Specifically, our contributions include:

1. **Curriculum-Based Opponent Progression:** A tiered curriculum (WeakOpponent $\rightarrow$ MediumOpponent $\rightarrow$ StrongOpponent) enables structured learning across increasing levels of adversarial difficulty, ensuring robust policy formation through progressive exposure.
2. **Transformer-Based Online Opponent Modeling:** A transformer module learns to predict opponent actions and tendencies during play, enabling belief-driven adaptation and fine-grained strategic reasoning.
3. **Bluff-Aware Reward Shaping:** We develop a Monte Carlo-based system that identifies bluff opportunities—aggressive actions with weak hands—and rewards successful bluffs, encouraging calculated deception.
4. **Natural Language Feedback Integration:** Using Mistral Medium, we generate episode-level strategic critiques based on poker metrics (e.g., VPIP, PFR, AFq). These critiques are embedded into the agent's observation space via sentence transformers. We further introduce an LLM-generated adherence score, scaled from -5 to 5, which functions as an auxiliary reward to reinforce behavior aligned with expert guidance.

Our system, deployed in the PettingZoo Texas Hold'em No-Limit environment, produces agents that bluff effectively, adjust to opponent strategies, and improve with linguistic coaching. These behaviors emerge through the interplay of deception-aware incentives, online belief modeling, and policy shaping via language.

We show that reinforcement learning agents can internalize high-level strategic patterns, including bluffing, when equipped with structured curricula and interpretable feedback. The insights extend beyond poker, offering pathways toward adaptive decision-making in complex, adversarial domains such as negotiation, auction design, and human-agent collaboration.

2 Related Work

Poker AI has advanced rapidly, with recent efforts targeting strategic reasoning, bluff detection, and opponent modeling. DeepStack (7) and AlphaHoldem (8) have demonstrated superhuman performance in heads-up no-limit Texas Hold'em, relying on counterfactual regret minimization (CFR) and end-to-end reinforcement learning, respectively. However, both systems are computationally intensive and assume static environments, limiting their adaptability to dynamic opponents and real-time feedback.

Shi et al. (9) introduce AMP3, a framework that incorporates opponent style modeling into Actor-Critic RL by classifying opponents using an offline-trained style predictor and behavior library. While AMP3 supports style conditioning, it relies on static categories and cannot adapt to nuanced deception or evolve during training. Similarly, Rupeneite (6) proposed an early opponent modeling approach that classified opponents into archetypes (e.g., tight/aggressive) using handcrafted features and offline learning. In contrast, our method employs a transformer-based opponent model that continuously predicts opponent behavior (e.g., action tendencies, bluff likelihood, hand strength) during play, enabling dynamic belief updates and policy adaptation in real time.

Wang et al. (2) apply Monte Carlo simulations to human gameplay to annotate bluffing behavior, identifying aggressive moves made with weak hands. While this offers an expert-inspired labeling pipeline, it remains observational and disconnected from agent learning. Schmid et al. (5) introduce a variational approach to learn player style embeddings from behavioral trajectories, offering insights into stylistic clustering. However, their method is designed for offline imitation and is not integrated with reinforcement learning or strategy feedback.

Recent benchmarks like PokerBench (10) reveal that even state-of-the-art LLMs struggle with consistent strategic play across game stages, highlighting the limitations of language-based agents in settings requiring adaptive reasoning and opponent modeling. BluffGPT (11) investigates the use of LLMs as improvisational bluffing agents, prompting GPT-4 to bluff or trap based on textual state descriptions. While it demonstrates emergent deceptive behavior, BluffGPT lacks reinforcement learning and treats bluffing as a static, prompt-conditioned action rather than a learned behavior. Crucially, it does not update policies through reward feedback, nor integrate strategic critiques into a training loop.

In contrast, our approach combines transformer-based online opponent modeling, deception-aware reward shaping via Monte Carlo bluff estimation, curriculum-based opponent progression, and natural language feedback from an LLM. Mistral Medium generates critiques after each episode based on poker statistics (e.g., VPIP, PFR, AFq), which are embedded into the agent's observation space using sentence transformers. We further prompt the LLM to rate the agent's behavioral alignment on a scale, which is used as an auxiliary loss to reinforce adherence to strategic feedback. This design enables policy improvement not just through rewards, but through interpretable, high-level behavioral critiques—bridging the gap between language-based insight and reinforcement-based training.

3 Method

We propose a modular reinforcement learning system for training poker agents capable of bluffing, adapting to opponent styles, and aligning with strategic feedback expressed in natural language. Our approach integrates standard policy learning with transformer-based opponent modeling, curriculum-based opponent progression, bluff-aware reward shaping, and large language model (LLM) feedback loops. The following subsections detail each component of the system.

3.1 Environment Setup

We use the `PettingZoo` implementation of Texas Hold'em No-Limit Poker as our multi-agent environment. The game features two players, each dealt private cards, followed by a series of betting rounds over shared community cards. Actions include `fold`, `check/call`, and `bet/raise`, with no fixed limit on bet size. Each episode ends when one player folds or the hand reaches showdown. The standard observations are a vector of size 54. The first 52 entries represent the union of the current player's hand and the community cards (binary values). The 53rd entry is the number of chips of player 0 (b/w 0 and 100) and the 54th entry is number of Chips of player 1. When opponent modeling is enabled, we append the predicted opponent action (a scalar), increasing the observation dimension to 55. When LLM feedback is also integrated, we append a 384-dimensional embedding of the feedback, raising the total observation dimension to 439. The standard reward structure consists of no intermediate rewards and only a reward at the end of each episode. It is equal to +(raised chips)/2 if the player won and -(raised chips)/2 if the player lost.

3.2 Policy Architecture and Optimization

The agent policy is modeled as a shared two-layer multilayer perceptron (MLP) with ReLU activations, followed by a softmax over the action space. The model outputs action probabilities for each timestep, conditioned on the current observation vector. We apply a discount factor $\gamma = 0.99$ and a learning rate of 0.01. To ensure exploration, we use entropy regularization during training. The agent is trained using Actor-Critic algorithm.

3.3 Opponent Behaviour Prediction Module

To enable dynamic adaptation to diverse strategies, the main player learning agent is equipped with a transformer-based OpponentModel module. This module predicts the opponent's next action, hand strength, and bluff tendency based on the sequence of past gameplay actions and shared game context. A separate opponent modeling module is maintained for each opponent type (Weak, Medium, Strong), and the model is trained online in parallel with the agent's policy network. Its outputs are designed to be embedded into the agent's observation space, allowing the policy to condition its decisions on inferred opponent behavior.In our current setup, however, only the predicted opponent next action (a scalar) is appended to the main player agent's observation vector, increasing its dimension by 1.

3.4 Bluff-Aware Reward Shaping

To incentivize strategic deception, we implement a custom reward function that detects and rewards successful bluffing. Bluff opportunities are identified when an agent takes aggressive actions (e.g., raise or bet) with low estimated hand strength. Hand strength is approximated using a Monte Carlo simulation of rollouts (20 samples) against randomly generated opponent hands. If such a bluff leads to a win (either by fold or at showdown), a bonus is added to the reward of the last timestep in

the episode. This delayed additional bonus is equal to +(raised chips)/2*reward_multiplier. After experimentation, we chose the reward_multiplier value to be 0.2. During the episode, additional intermediate rewards are provided if aggressive actions are taken with a low hand strength (i.e. bluff detected) irrespective of whether the agent won or not. We experiment with this intermediate reward value in our experiments.

3.5 Curriculum-Based Opponent Progression

We train agents against a progressively harder curriculum of opponents:

- **WeakOpponent**: Random policy agent. This agent takes random actions
- **MediumOpponent**: Pretrained actor-critic agent with fixed weights.
- **StrongOpponent**: Concurrently learning agent using the REINFORCE algorithm.

The agent graduates to the next opponent tier upon reaching a win rate threshold of 70% (Weak $\rightarrow$ Medium) or 65% (Medium $\rightarrow$ Strong), allowing for smooth scaling of adversarial complexity.

3.6 Natural Language Feedback Integration

After each episode, we query Mistral Medium to generate strategic critiques based on the agent's behavioral statistics. The model is prompted with before-and-after values for the following four professional poker metrics:

1. **VPIP (Voluntarily Put Money In Pot):** Percentage of hands a player decides to voluntarily invest in pre-flop.
   $\text{VPIP} = \frac{\text{Hands played pre-flop}}{\text{Total hands dealt}} \times 100$
2. **PFR (Pre-Flop Raise):** Measure of pre-flop aggressiveness—i.e., how often a player raises before the flop.
   $\text{PFR} = \frac{\text{Hands raised pre-flop}}{\text{Total hands dealt}} \times 100$
3. **AFq (Aggression Frequency):** Proportion of all actions that are aggressive.
   $\text{AFq} = \frac{\text{Raise}}{\text{Call+Fold+Raise}} \times 100$
4. **WTSD (Went To Showdown):** Frequency with which a player proceeds to showdown after seeing the flop.
   $\text{WTSD} = \frac{\text{Hands went to ShowDown}}{\text{Total hands went to Flop}} \times 100$

These statistics are passed into Mistral Mediumalongside the strategic feedback from the previous episode. The LLM then returns a natural language critique such as "tighten pre-flop range" or "reduce aggression post-flop," which is embedded using a pre-trained Sentence Transformer.The resulting feedback embedding (384-dimensional) is appended to the agent's observation space, increasing the observation vector's size accordingly. This allows the agent to interpret natural language not just as text but as actionable, structured feedback within its decision-making process.

3.7 Behavioral Adherence Scoring and Reward

To align the agent's behavior with high-level strategic feedback, we introduce a behavioral adherence reward. After each episode, Mistral Medium is prompted to assess how well the agent followed the previous feedback, based on changes in VPIP, PFR, AFq, and WTSD. The model returns an adherence score on a discrete scale from -5 (low adherence) to 5 (perfect adherence), reflecting how closely the agent's behavioral shift matched the intended advice. We experiment with two variants: (1) directly adding the adherence score (scaled by 10) to the final step reward of every episode, and (2) adding it only when the episode is lost, in order to more strongly discourage non-adherence when it results in poor outcomes.:

$$r_{\text{total}} = r_{\text{env}} + 10 * r_{\text{adherence}}$$

This simple additive mechanism encourages the agent to win not just through reward maximization, but in a manner that reflects guided strategic improvement.

4 Experiments

4.1 Experimental Design

4.1.1 Ablation Study Framework

To rigorously evaluate the contribution of each system component, we conduct comprehensive ablation studies across four key dimensions:

Opponent Modeling Evaluation: We compare agents equipped with our transformer-based opponent prediction system against baseline agents without opponent modeling capabilities. This isolates the impact of real-time opponent behavior prediction on strategic decision-making and bluffing effectiveness.

Curriculum Learning Analysis: We contrast static opponent training protocols against our adaptive three-tier curriculum progression system. Static training maintains constant opponent difficulty throughout training, while curriculum training dynamically adjusts opponent sophistication based on performance milestones.

Reward Structure Investigation: We systematically evaluate the impact of our deception-aware reward system by testing multiple intermediate (each step) bluff reward values $w_b \in \{0.0, 0.5, 1.0, 1.5, 2.0\}$. The baseline condition ($w_b = 0.0$) represents standard reinforcement learning rewards, while higher bluff reward values increasingly incentivize strategic deception. In an additional condition, we incorporate the adherence score as an auxiliary reward added at the final step of each episode, allowing us to assess whether encouraging alignment with strategic feedback improves agent learning.

Natural Language Feedback Integration: We assess the impact of Mistral Medium generated strategic critiques by comparing training with and without LLM-based feedback integration. This evaluation measures whether natural language guidance effectively shapes agent behavior and improves strategic sophistication. (Here opponent modeling predictions are also included with a bluff reward of 1 for both baseline and model with LLM feedback)

Behavioral Adherence Reward Range: In addition to experimenting with when to apply the adherence reward, we explored the effect of different output ranges for the behavioral adherence score. We tested three variants: the original discrete scale from -5 to 5, a normalized scale from 0 to 1, and a bounded positive scale from 0 to 5. These variants allowed us to investigate how the magnitude and framing of the reward signal influence the agent's ability to incorporate high-level strategic feedback into its behavior.

4.1.2 Training Protocols

Each experimental condition is trained for 10,000 episodes with performance checkpoints every 500 episodes. To account for training variance, we conduct 5 independent runs per configuration with different random seeds.

Promotion between curriculum tiers occurs when agents achieve sustained performance above threshold over 100 consecutive episodes: 70% win rate for WeakOpponent promotion and 65% for MediumOpponent promotion. These thresholds were empirically determined through preliminary experiments to ensure meaningful skill progression while maintaining training stability.

For conditions using feedback integration, adherence scores are computed at the end of each episode and added to the final step reward. This encourages behavioral alignment with strategic advice in addition to game outcome optimization.

4.2 Evaluation Metrics

Testing Procedure. Unless otherwise specified, all agents were evaluated against their respective strong opponents over 1000 test episodes/games. During testing, the strong opponent continued updating its policy online, maintaining an adaptive challenge throughout the evaluation. This setup ensured that the tested agent's performance reflects robustness against a continually learning adversary.

4.2.1 Strategic Performance Indicators

We employ multiple complementary metrics to comprehensively assess agent performance:

Win Rate Analysis: Primary performance metric measuring the percentage of games won against each opponent tier, computed over sliding windows of 100 episodes to track learning progression.

Expected Value Assessment: Mean episodic reward accumulation normalized by game length, providing insight into long-term strategic effectiveness beyond binary win/loss outcomes.

Deception Effectiveness: We quantify bluffing behavior through two key metrics: (1) *Bluff Frequency* — percentage of actions classified as bluffs using our Monte Carlo hand strength estimator, and (2) *Bluff Success Rate* — percentage of bluffs that successfully induce opponent folds or lead to episode victories.

4.2.2 Behavioral Adaptation Metrics

To assess the quality of strategic adaptation, we track:

Opponent Modeling Accuracy: Percentage of correctly predicted opponent actions, measured continuously during training to track model convergence and prediction quality.

Strategic Consistency: Variance in professional poker statistics (VPIP, PFR, AFq, WTSD) across episodes, with lower variance indicating more consistent strategic execution.

Feedback Adherence: To quantify how effectively the agent responds to strategic advice from large language models, we introduce an adherence score ranging from 0.0 (no adherence) to 5.0 (perfect adherence). This score is computed by comparing changes in key poker behavior metrics—VPIP, PFR, AFq, and WTSD—before and after receiving feedback. The score reflects both the direction and magnitude of behavioral changes, as well as their alignment with the intent of the advice. This adherence score is also used as a reward added to the last step's reward at the end of each episode.

5 Results

5.1 Bluff Reward Tuning.

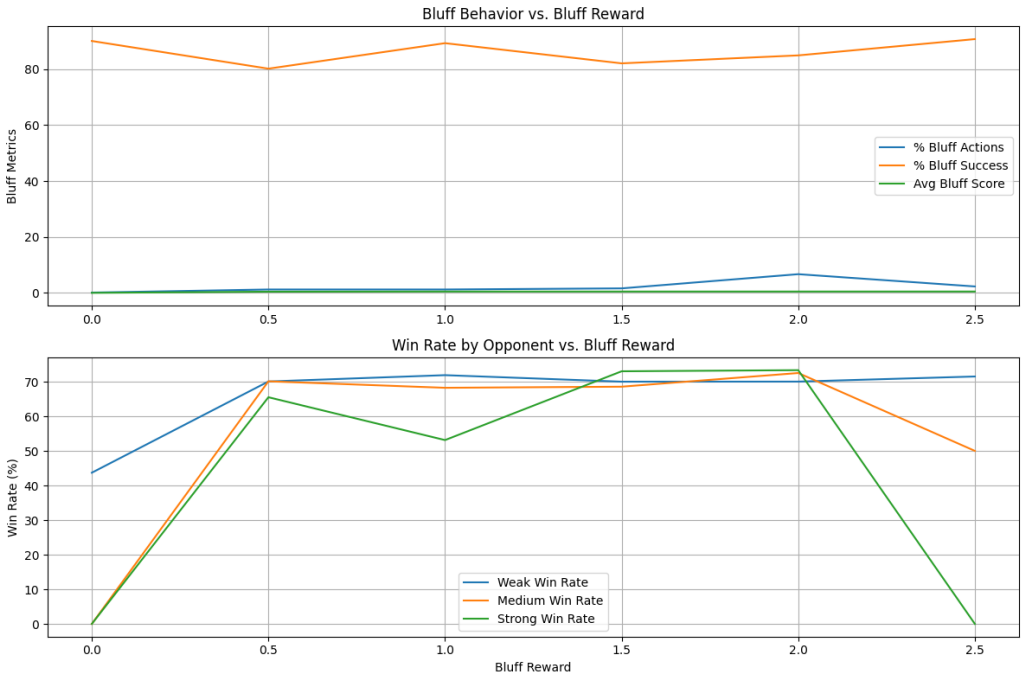

Figure 1: Strategic progression of bluff-related metrics during training

Bluff Reward	% Bluff Success	Final Win Rate
0.0	2.4%	59.1%
0.5	0.6%	63.9%
1.0	0.5%	65.2%
1.5	2.7%	78.1%
2.0	**20.9%**	**80.5%**
2.5	7.6%	63.4%

Table 1: Performance across bluff reward settings. All agents are evaluated against a fixed strong opponent trained with reward = 2.0.

As indicated in Figure 1, a bluff reward of 2.0 achieves the best trade-off, with a bluff rate of 6.68%, success rate of 84.95%, and a win rate of 73.32% against strong opponents. In contrast, removing the reward leads to nearly no bluffing and much lower win rates.

Testing Results. To validate generalization, we evaluated agents trained under varying bluff reward settings against a strong opponent initialized with weights from the best-performing training checkpoint (bluff reward = 2.0). Importantly, the opponent continues to update during evaluation, allowing for dynamic adaptation. Table 1 summarizes the test-time bluff success and win rates. We observe that a bluff reward of 2.0 yields the highest bluff success rate (20.9%) and win rate (80.5%), suggesting that this reward level encourages effective risk-taking without overcommitting.

5.2 Opponent Modeling.

Model	% Bluff	% Success	Avg Score	Weak Win	Medium Win	Strong Win
With Opponent Modeling	8.21	**77.49**	**0.475**	**71.88%**	68.23%	**53.12%**
Without Opponent Modeling	**9.30**	52.30	0.468	70.00%	**71.00%**	33.90%
Δ	-11.72%	**+48.16%**	**+1.50%**	**+2.69%**	-3.90%	**+56.69%**

Table 2: Training Performance Comparison With and Without Opponent Modeling

Model	Test Bluff Success	Test Win Rate
With Opponent Modeling	20.9%	80.5%
Without Opponent Modeling	1.30%	43.00%

Table 3: Test Performance Comparison With and Without Opponent Modeling

Table 2 shows that success improves significantly with opponent modeling (+48.16%), and win rate against strong opponents increases by 56.69%. Despite fewer bluff attempts, average bluff quality and outcome improve. By episode 9,000, average opponent action prediction accuracy across all 3 opponent types reaches 0.2218, enabling sharper decision-making. These results are observed over 50,000 training episodes, demonstrating the long-term benefits of integrating online opponent modeling into policy learning.

Testing Results. We further tested the agent incorporating opponent modeling against the strong agent over 1000 episodes (Table 3). The results demonstrate a substantial advantage in both bluff success rate (20.9%) and win rate (80.5%) compared to the baseline agent without opponent modeling, which achieved only a 1.30% bluff success rate and a 43.00% win rate. These findings indicate that opponent modeling allows the agent to adapt more effectively to strategic behavior, selectively bluffing when success is more likely.

Opponent Modeling Accuracy Progression. Opponent action prediction accuracy varied substantially across runs (5-40%), with the analyzed run in Table 2 achieving 40% final accuracy. While the causes of this variance remain unclear—potentially due to random initialization, opponent learning dynamics, or poker's inherent stochasticity—higher prediction accuracy consistently correlated with improved strategic performance. This suggests the opponent modeling signal provides valuable

information beyond simple action prediction, though the wide variance indicates this component requires further investigation.

5.3 Curriculum Learning.

Tier	% Bluff Actions	% Bluff Success	Win Rate (%)
Weak	Static: **2.21**%	76.47%	**71.88**
	Curriculum: 1.20%	**89.32%**	70.00
	Δ: -45.70%	**+16.78%**	-2.61%
Medium	Static: **2.22%**	77.93%	68.23
	Curriculum: 1.20%	**89.32%**	**70.35**
	Δ: -45.95%	**+14.58%**	**+3.11%**
Strong	Static: **8.21%**	77.49%	53.12
	Curriculum: 1.20%	**89.32%**	**72.76**
	Δ: -85.38%	**+15.27%**	**+36.94%**

Table 4: Training Performance: Bluff Rate, Success, and Win Rate: Curriculum vs Static Training

Training Regime	Test Bluff Success	Test Win Rate
Weak (Static)	0.3%	22.1%
Medium (Static)	0.1%	34.9%
Strong (Static)	**0.6%**	52.4%
Curriculum	0.5%	**65.2%**

Table 5: Test Performance for Curriculum vs Static Training Against Strong Opponent

Across weak, medium, and strong tiers, curriculum learning consistently reduces bluff frequency while significantly improving bluff success and win rate. As Table 4 has shown, over the course of just 10,000 training episodes, we observe that in the strong opponent tier, bluff rate drops by over 85%, yet win rate increases by nearly 37%, indicating more selective and effective bluffing behavior under increasing difficulty.

Testing Results. Testing the curriculum-trained agent against the strong opponent for 1000 episodes (Table 5) shows that it generalizes more robustly than all static counterparts. Despite having a slightly lower test bluff success rate than the static strong-trained agent (0.5% vs. 0.6%), the curriculum-trained agent achieves a substantially higher win rate of 65.2%, outperforming the next-best static strategy by nearly 13 percentage points. This suggests that curriculum exposure enables more adaptive and resilient strategies beyond bluffing frequency alone, particularly when facing the strongest level of adversarial play.

5.4 LLM Feedback.

Opponent	Baseline	With GPT	Change (Δ)
WeakOpponent	74.50%	71.88%	-2.62%
StrongOpponent	53.12%	73.77%	**+20.65%**

Table 6: Training Win Rate Comparison vs. Weak and Strong Opponents (Before vs. After GPT Feedback; includes Opponent Modelling and Bluff Reward of 1)

Shown by results in Table 6, LLM-generated feedback led to more disciplined play. Common advice like "tighten pre-flop range" and "reduce aggression" directly influenced decision patterns, as evidenced by the behavioral differences between Figure 2 and Figure 3. Without LLM feedback, the agent maintained highly aggressive behavior with AFq reaching 80%, PFR approaching 98%, and VPIP near 99% (Figure3). In contrast, the agent with LLM feedback showed significantly reduced aggression, with AFq decreasing from 60% to 27%, PFR dropping from 50% to 23%,

LLM Feedback	% Bluff Success	Final Win Rate
With GPT	87%	70.50%
Baseline	0.6%	65.2%

Table 7: Test time Win Rate Comparison vs.Strong Opponents (Before vs. After GPT Feedback; Baseline is Model of Bluff Reward = 1 from Table 1)

Figure 2: Poker Stats during Training With LLM feedback

and VPIP declining from initial peaks to 74% (Figure2). Despite a small performance drop against weak opponents, the model saw a large win rate improvement (+20.65%) against strong ones when combining GPT feedback, opponent modeling, and bluff reward tuning.

LLM Feedback Content Analysis. Due to the use of TensorBoard's add_text() feature—which does not persistently store all logged entries across long training runs—only a subset of feedback was retrievable for analysis. Specifically, 10 Mistral-generated feedback messages spanning various training steps were manually reviewed. All entries recommended tightening the preflop range (10/10), while half suggested increasing bluff frequency (5/10). Others advised either increasing postflop aggression (2/10) or reducing aggression frequency (2/10). Most feedback emphasized exploiting opponent tendencies such as high PFR (7/10) or high VPIP (3/10). This limited sample reflects a consistent model bias toward preflop discipline and opponent-specific exploitative strategies, though broader trends across the full training run remain inaccessible due to logging constraints.

Testing Results. Testing the agent enhanced with LLM feedback against the strong opponent over 1000 episodes (Table 7) shows a win rate improvement from 65.2% to 70.5%, demonstrating that natural language feedback effectively guides strategic adaptation. Notably, while LLM feedback during training generally discouraged aggressive behavior (as measured by reduced AFq), the final

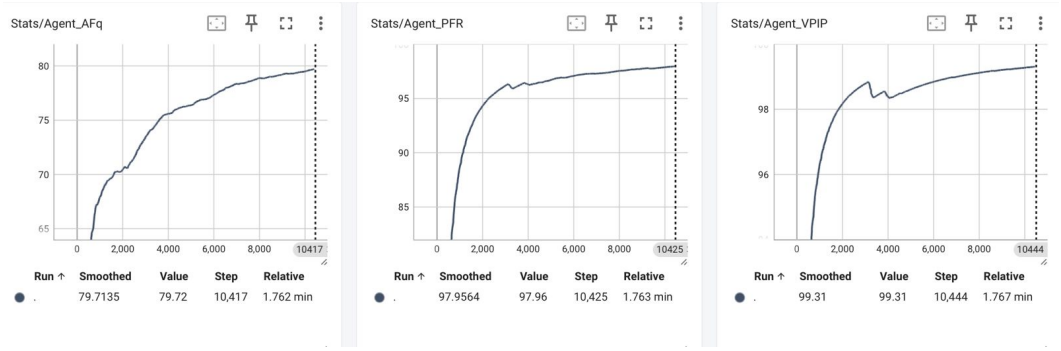

Figure 3: Poker stats during training Without LLM feedback

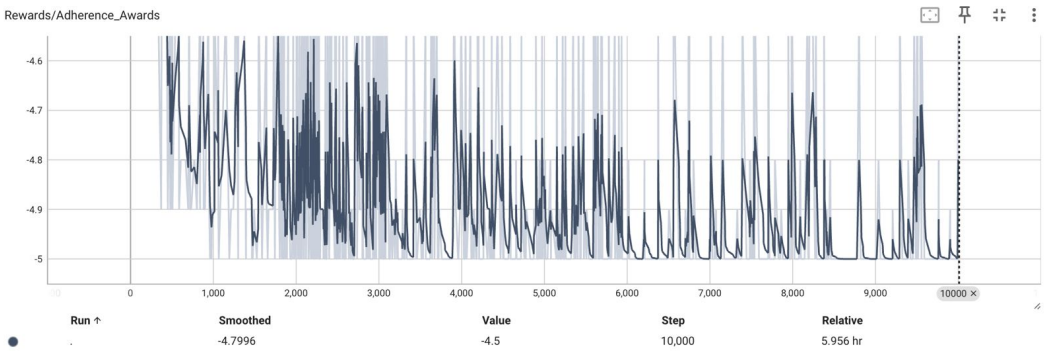

Figure 4: Adherence rewards between -5 and 5 for lost games during training

trained agent exhibits a higher bluff success rate during testing. This suggests the agent learned to bluff more selectively and effectively rather than simply bluffing more frequently—indicating that the LLM feedback helped develop more sophisticated deception strategies that prioritize quality over quantity of bluffs.

5.5 Adherence Reward.

Adherence Reward	% Bluff Success	Final Win Rate
Without Adherence Reward (From Table 7)	**87.0%**	**70.50%**
With Adherence Reward	0.00% (None attempted)	66.80%

Table 8: Testing Performance Comparison With and Without Adherence Reward

Performance Impact. Table 8 reveals that incorporating adherence rewards significantly degraded agent performance compared to the baseline without adherence rewards. The agent with adherence rewards achieved a 66.80% win rate versus 70.50% for the baseline—a decrease of 3.70 percentage points. Most notably, the adherence reward agent completely ceased bluffing behavior, attempting 0.00% bluffs compared to the baseline's 87.0% bluff success rate.

Initial Implementation Issues. Our initial approach of applying adherence rewards after every episode, regardless of outcome, proved problematic. The adherence score consistently converged to -5, indicating that the system was continuously penalizing the agent's performance. Across 2000 episodes, the LLM feedback consistently rated performance as worst-case, whether we used a -5 to 5 or 0 to 5 scale. This behavior suggests a fundamental incompatibility between LLM feedback mechanisms and the inherent stochasticity of the environment. Specifically, strategic feedback from previous episodes may not transfer effectively to new episodes due to the substantial variability between individual games.

Modified Approach and Persistent Issues. To address this issue, we modified our approach to only apply adherence rewards when the main agent lost, maintaining the -5 to 5 scale for consistency with our experimental framework. However, even with this targeted approach, adherence scores remained consistently low (-4 to -5) for lost games, and the results shown in Table 8 represent this modified implementation. The persistent negative scoring and complete elimination of bluffing behavior indicate that the adherence reward mechanism, as currently implemented, may be counterproductive to strategic poker play development. Additionally, providing positive adherence rewards following losses may have created conflicting signals that confused the learning process, as the agent received rewards for losing games when it followed strategic advice, potentially undermining the fundamental objective of winning.

6 Discussion

Strategic Enhancement Effectiveness Bluff reward tuning successfully guided AI toward sophisticated deception strategies, with moderate reward levels encouraging effective risk-taking without overcommitment. Opponent modeling proved particularly impactful, enabling dynamic adaptation and selective bluffing—prioritizing quality over quantity in deceptive actions. The agent learned genuine strategic reasoning beyond naive pattern matching.

Curriculum learning demonstrated superior generalization across all difficulty tiers compared to static training regimes. Progressive exposure to increasingly challenging opponents developed robust strategies that transferred effectively to novel competitive scenarios, particularly against the strongest adversaries.

Implementation Challenges and Limitations The adherence reward mechanism failed completely, eliminating bluffing behavior due to fundamental incompatibilities between natural language feedback and RL optimization. Poker's stochasticity made strategic advice from previous games irrelevant in new contexts, creating conflicting reward signals.

Opponent modeling accuracy varied substantially across runs with unclear causes, indicating the component requires further investigation. Technical constraints limited analysis depth—TensorBoard logging issues prevented comprehensive LLM feedback evaluation.

A significant limitation was our use of the free-tier Mistral Medium chat API rather than state-of-the-art models due to financial constraints, potentially limiting the quality and consistency of strategic feedback generation.

Broader Implications Results extend beyond poker to strategic domains requiring deception and adaptation, including negotiation and auction systems. However, the adherence reward failure highlights the complexity of human-AI collaboration in strategic contexts, where temporal scale mismatches between human reasoning and RL optimization present ongoing challenges.

7 Conclusion

This research demonstrates that strategic poker AI performance can be substantially enhanced through behavioral incentives, opponent modeling, and progressive training. Our approach developed an agent capable of sophisticated strategic reasoning and selective deception.

Key Contributions: Optimal bluff reward mechanisms encourage strategic risk-taking. Opponent modeling enables real-time adaptation for multi-agent environments. Curriculum learning proves superior to static training for developing generalizable strategic competence.

Take-Home Message Effective strategic AI requires explicit behavioral guidance and structured learning progressions, but demands careful consideration of feedback mechanism compatibility with automated optimization. Success combines complementary techniques while respecting differences between human strategic reasoning and machine learning paradigms.

Future Directions Future work should investigate sophisticated adherence reward formulations accounting for poker's stochastic nature, explore alternative opponent modeling architectures to address accuracy variance, and develop robust frameworks for integrating natural language feedback into RL systems. Extension to broader strategic interaction domains involving deception and negotiation represents promising applications.

8 Team Contributions

Sara and Yanny contributed equally to all aspects of the project, including ideation, implementation, experimentation, and writing. Sara focused on the LLM feedback integration (including adherence rewards) and opponent modeling. Yanny focused on the curriculum learning aspect. We collectively worked on the deception bluffing rewards segment.

References

[1] Yichen Shi, Yuheng Guo, Yifan Liu, and Chao Fan.
Adaptive Multi-agent Policy Learning via Opponent Style Modeling for Multi-player Poker.
Applied Intelligence, 2025.

[2] Xintong Wang, Xiaohan Zhao, Li Pan, and Xin Zhang.
AI-assisted Poker Video Labeling Based on Player Strategy Analysis.
Neural Computing and Applications, vol. 34, pp. 17903–17914, Springer, 2022.

[3] Enmin Zhao, Renye Yan, Jinqiu Li, Kai Li, and Junliang Xing.
AlphaHoldem: High-Performance Artificial Intelligence for Heads-Up No-Limit Poker via End-to-End Reinforcement Learning.
In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 4, pp. 4689–4695, 2022.

[4] Jiarui Liu, Yichuan Charlie Wu, Gerald Tesauro, Noam Brown, and Tuomas Sandholm.
GameFlowNet: Monte Carlo Sampling for Imperfect-Information Games.
In Proceedings of the 37th International Conference on Machine Learning (ICML), 2020.

[5] Michael Schmid, Malte Dell, and Mihai Lupu.
A Variational Approach for Learning from User Behavior Trajectories in Games.
In Proceedings of the IEEE Conference on Games (CoG), 2019.

[6] Annija Rupeneite.
Building Poker Agent Using Reinforcement Learning with Neural Networks.
In *Doctoral Consortium at the International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2014. SCITEPRESS.

[7] Martin Moravčík, Michael Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Christopher Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling.
DeepStack: Expert-Level Artificial Intelligence in Heads-Up No-Limit Poker.
Science, vol. 356, no. 6337, pp. 508–513, 2017.

[8] Enmin Zhao, Renye Yan, Jinqiu Li, Kai Li, and Junliang Xing.
AlphaHoldem: High-Performance Artificial Intelligence for Heads-Up No-Limit Poker via End-to-End Reinforcement Learning.
In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 4, pp. 4689–4695, 2022.

[9] Yichen Shi, Yuheng Guo, Yifan Liu, and Chao Fan.
Adaptive Multi-agent Policy Learning via Opponent Style Modeling for Multi-player Poker.
Applied Intelligence, 2025.

[10] Tongshuang Zhuang, Michael Schmid, Juho Kim, Tatsunori Hashimoto, and Percy Liang.
PokerBench: Evaluating Strategic Reasoning in LLMs via No-Limit Poker.
arXiv preprint arXiv:2402.00001, 2025.

[11] Leo Ashmore, Sayash Ghosh, Amy Yu, Rebecca Morris, Sam Bowden, Joel Z. Leibo, Adam Lerer, and Yuxuan Wang.
BluffGPT: Large Language Models as Strategic Improvisers in Poker.
arXiv preprint arXiv:2401.06781, 2024.

A Appendix

Adherence Score and LLM feedback prompts

```
def get_adherence_score_from_llm(previous_feedback, stats_before, stats_after):
    """Get adherence score from LLM (-5 to 5 scale) using Mistral"""

    evaluation_prompt = f"""
```

```
    Previous strategic advice: "{previous_feedback}"

    Agent statistics BEFORE advice:
    - VPIP (hands played): {stats_before["VPIP"]:.1f}%
    - PFR (pre-flop raises): {stats_before["PFR"]:.1f}%
    - AFq (aggression frequency): {stats_before["AFq"]:.1f}%
    - WTSD (went to showdown): {stats_before["WTSD"]:.1f}%

    Agent statistics AFTER advice:
    - VPIP: {stats_after["VPIP"]:.1f}%
    - PFR: {stats_after["PFR"]:.1f}%
    - AFq: {stats_after["AFq"]:.1f}%
    - WTSD: {stats_after["WTSD"]:.1f}%


    How well did the agent follow the strategic advice? Rate adherence from -5 to 5. :

    Consider:
    - Direction of statistical changes (did they move toward advised targets?)
    - Magnitude of changes (how much did they adjust?)
    - Overall consistency with the strategic advice given
    - negative 5 is no adherence and 5 is perfect adherence to feedback.
    - give negative rewards if the new performance is not following advice.
    Respond with only a decimal number between -5 and 5.
    """
    headers = {
        "Authorization": f"Bearer {MISTRAL_API_KEY}",
        "Content-Type": "application/json"
    }
    data = {
        "model": "mistral-medium",  # or mistral-small / mistral-large
        "messages": [
            {"role": "system", "content": "You are a helpful and critical poker coach evaluating
            {"role": "user", "content": evaluation_prompt.strip()}
        ],
        "temperature": 0.2,
        "max_tokens": 10
    }
    try:
        response = requests.post(MISTRAL_API_URL, headers=headers, json=data)
        response.raise_for_status()
        message = response.json()["choices"][0]["message"]["content"]
        score = float(message.strip())
        return max(-5, min(5.0, score))  # Clamp to [-5, 5]
    except Exception as e:
        time.sleep(60)
        print(f"Mistral adherence scoring failed: {e}")
        return 0.0  # fallback neutral

def get_gpt_feedback(player_stats, opponent_stats):
    """Get strategic feedback from Mistral"""

    def fmt(stats):
        return "\n".join([f"- {k.replace('_', ' ').title()}: {v}" for k, v in stats.items()])

    prompt = f"""
    You are a poker coach analyzing an RL agent's performance in a game against an opponent.

    Here are the agent's stats:
```

```
{fmt(player_stats)}

Here are the opponent's stats:
{fmt(opponent_stats)}

Please write a short descriptive 10-word feedback giving strategy suggestions to our playing
Be specific and technical.
Give specific suggestions (e.g., bluffing frequency, risk-taking, timing, loose/tight, aggres
"""

headers = {
    "Authorization": f"Bearer {MISTRAL_API_KEY}",
    "Content-Type": "application/json"
}

data = {
    "model": "mistral-medium",
    "messages": [
        {"role": "system", "content": "You are a helpful and critical poker coach evaluating
        {"role": "user", "content": prompt.strip()}
    ],
    "temperature": 0.7,
    "max_tokens": 50
}

try:
    response = requests.post(MISTRAL_API_URL, headers=headers, json=data)
    response.raise_for_status()
    return response.json()["choices"][0]["message"]["content"].strip()
except Exception as e:
    time.sleep(60)
    print(f"Mistral feedback generation failed: {e}")
    return ""
```