An Ablation Study of PPO-KL vs PPO-Clip on No Limit Texas Hold'em

Akhilesh Kasturi, Sebastian Thomas

Northeastern University, Khoury College of Computer Sciences

April 18, 2025

Abstract

Reinforcement learning (RL) is the process of learning how to act in an environment by mapping situations to actions in order to maximize a numerical reward signal. Rather than being explicitly told which actions to take, an RL agent must discover effective strategies through trial and error. Deep Reinforcement Learning (DRL) has revolutionized the field of game-playing AI and has become an exceptionally popular approach for developing agents that can master complex games. Beginning with DeepMind's groundbreaking Deep Q-Network (DQN) in 2013, which successfully learned to play Atari 2600 games directly from pixel inputs and achieved human-level performance across 49 games. [1] In this project, we conduct an ablation study comparing two variants of the Proximal Policy Optimization (PPO) algorithm—PPO-Clip and PPO-KL—within the context of No-Limit Texas Hold'em using the RLCard environment.

Keywords— Reinforcement Learning, Proximal Policy Optimization, RLCard

Introduction

Reinforcement Learning (RL) has shown great promise in imperfect information games like poker, where decision-making under uncertainty is paramount. In this project, we perform an ablation study comparing two variants of the Proximal Policy Optimization (PPO) algorithm: PPO-Clip and PPO-KL, within the RLCard environment for No-Limit Texas Hold'em. Our objective is to understand how each constraint mechanism influences the agent's strategy, learning behavior, and performance.

We develop custom enhancements to the environment, including an enriched state representation, sophisticated reward shaping, and a self-play training mechanism. Our experiments reveal striking differences between the two PPO variants in how they explore, converge, and generalize within the poker domain.

Game Overview

No Limit Texas Hold'em is one of the most popular game of poker played in casinos. It is played for its simple rules, deep strategic richness, and excitement created through the "no limit" betting system, where players can bet all or any of their chips at any moment.[2]

Game Structure and Rules

Basic Gameplay

- Each player is dealt two private cards (*hole cards*).
- Five community cards are dealt face up in the center of the table in three stages: the flop (three cards), the turn (one card), and the river (one card).
- Players aim to make the best possible five-card hand using any combination of their hole cards and the community cards.

Blinds and Dealer

- The game uses "blinds" to initiate action: the two players to the left of the dealer post forced bets called the small blind and big blind.
- The dealer position rotates clockwise after each hand, ensuring fairness.

Betting Rounds

There are four betting rounds:

1. **Preflop**: After hole cards are dealt, betting starts with the player to the left of the big blind.
2. **Flop**: Three community cards are dealt, followed by a betting round.
3. **Turn**: A fourth community card is dealt, followed by another betting round.
4. **River**: The fifth and final community card is dealt, followed by the last betting round.

Betting Actions

On each betting round, players can:

- **Fold**: Surrender their hand.
- **Call**: Match the current bet.
- **Raise**: Increase the bet.
- **Check**: Pass the action (if no bet has been made in the current round).

Hand Rankings

From strongest to weakest:

1. **Royal Flush** - Ace, King, Queen, Jack, and Ten of the same suit
2. **Straight Flush** - Five consecutive cards of the same suit
3. **Four of a Kind** - Four cards of the same rank
4. **Full House** - Three of a kind plus a pair
5. **Flush** - Five cards of the same suit, not consecutive
6. **Straight** - Five consecutive cards of mixed suits
7. **Three of a Kind** - Three cards of the same rank
8. **Two Pair** - Two cards of one rank and two cards of another rank
9. **One Pair** - Two cards of the same rank
10. **High Card** - Has no pairs, three-of-a-kind, four-of-a-kind, flushes, straights, or any other recognized poker hand.

Players use their best five-card combination from their two hole cards and the five community cards. They may use both, one, or none of their hole cards.

No Limit Betting Structure

- **No Limit** means there is no maximum bet—players can wager any amount of their chips at any time, up to "all-in."
- The minimum opening raise must be at least twice the big blind. Subsequent raises must be at least as large as the previous raise in that round.
- *Example:* If the big blind is 10, the minimum raise is to 20. If someone raises to 50, the next raise must be at least 90 (50 + the difference between 20 and 50).

Methodology

PPO is inspired by the same question as TRPO: how can we make the most significant change to a policy given the data we presently have, without accidentally causing performance collapse? While TRPO attempts to handle this problem using a complex second-order method, PPO is a collection of first-order methods that employ a few additional techniques to keep new policies near to old. PPO approaches are substantially easier to implement, and they appear to perform at least as well as TRPO. [3]

PPO has two primary variants: PPO-KL and PPO-Clip.

1. **PPO-KL:** PPO-Penalty, like TRPO, approximates a KL-constrained update, but penalizes the KL-divergence in the objective function rather than making it a hard constraint, and automatically adjusts the penalty coefficient during training to ensure that it is scaled appropriately.
2. **PPO-Clip:** PPO-Clip doesn't have a KL-divergence term in the objective and doesn't have a constraint at all. Instead relies on specialized clipping in the objective function to remove incentives for the new policy to get far from the old policy.

PPO with Clipped Objective

We maintain two policy networks. The first one is the current policy that we want to refine. [4]

$$\pi_\theta(a_t|s_t)$$

The second is the policy that we last used to collect samples.

$$\pi_{\theta_k}(a_t|s_t)$$

With the idea of importance sampling, we can evaluate a new policy with samples collected from an older policy. This improves sample efficiency.

$$\max_\theta \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \hat{A}_t \right]$$

But as we refine the current policy, the difference between the current and the old policy is getting larger. The variance of the estimation will increase and we will make bad decisions because of the inaccuracy. So, say for every 4 iterations, we synchronize the second network with the refined policy again.

$$\pi_{\theta_{k+1}}(a_t|s_t) \longleftarrow \pi_\theta(a_t|s_t)$$

With clipped objective, we compute a ratio between the new policy and the old policy:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$$

This ratio measures how different between two policies. We construct a new objective function to clip the estimated advantage function if the new policy is far away from the old policy. Our new objective function becomes:

$$L^{\text{CLIP}}_{\theta_k}(\theta) = \mathbb{E}_t \left[\sum_{t=0}^{T} \min \left(r_t(\theta)\hat{A}^k_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^k_t \right) \right]$$

If the probability ratio between the new policy and the old policy falls outside the range $(1-\epsilon)$ and $(1+\epsilon)$, the advantage function will be clipped.[5]

PPO with KL Divergence

In order to prevent large policy updates, PPO penalizes for the expectation (as seen in PPO-Clip) as follows:

$$\max_{\theta} \mathbb{E}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}A_t - \beta \cdot \text{penalty}\right]$$

where β is the coefficient for the weight of penalty.

In PPO-Penalty, KL-divergence is used for this penalty term.

$$\text{penalty} := \text{KL}(P(\cdot|\pi_{\theta_k}(s))||P(\cdot|\pi_\theta(s)))$$

KL-divergence (Kullback-Leibler divergence): We assume that both $P(x)$ and $Q(x)$ are stochastic distributions. KL-divergence $\text{KL}(P||Q)$ is then defined as follows:

$$\text{KL}(P||Q) := -\int P(x) \ln \frac{Q(x)}{P(x)} dx$$

By this definition, $\text{KL}(P||Q)$ will be always positive or zero, and zero if and only if both distributions are same. This means that $\text{KL}(P||Q)$ indicates how far between these distributions, P and Q. If Q is so far from P, $\text{KL}(P||Q)$ will become largely positive.

We can define a ratio between the new policy and the old policy:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$$

In order to penalize for large update between $P(\cdot|\pi_{\theta_k}(s))$ and $P(\cdot|\pi_\theta(s))$, we look for the optimal parameters θ, such as:

$$\max_{\theta} \mathbb{E}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}A_t - \beta \cdot \text{KL}(P(\cdot|\pi_{\theta_k}(s))||P(\cdot|\pi_\theta(s)))\right]$$

Even if the first term $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}A_t$ increases, the new θ might be rejected when the difference between $P(\cdot|\pi_{\theta_k}(s))$ and $P(\cdot|\pi_\theta(s))$ is too large.[6]

Environment Setup

We use the RLCard [7] library to simulate two-player No-Limit Texas Hold'em games. Our wrapper class around the RLCard environment provides:

- Custom state encoding
- Reward shaping
- Self-play support

State Vector Components

Our enhanced state vector includes:

- Base Observation Vector: The default RLCard observation.
- Hand Strength Estimation: A float (0-1) derived from card features using a heuristic function.
- Game Stage Encoding: One-hot encoded round indicator (preflop, flop, turn, river).
- Player Position: One-hot encoding for SB, BB, etc.
- Normalized Pot and Stack: Values normalized to 20,000 (pot) and 10,000 (stack).

This provides a comprehensive representation of both card information and betting context, essential for strategic decision-making.

Hand Strength Estimation

Our `_estimate_hand_strength` function evaluates private and public cards using:

- Bonuses for high cards (T, J, Q, K, A)
- Bonus for pocket pairs (+0.3)
- Bonus for suited cards (+0.1)
- Bonuses for matching public cards
- Base strength starts at 0.3 and can reach 1.0

This serves as a useful heuristic for guiding early learning.

Reward Shaping

Implemented in the environment's `step()` function, key components include:

- Base Reward: Final payoffs from RLCard
- Win Bonus: +20% for winning

Action-Specific Bonuses:

- +0.01 for not folding
- Rewards for appropriate raises (scaled by pot size)
- Bonuses/penalties for all-in, calls, and checks based on hand strength

This encourages rational poker behavior: aggressive play with strong hands, pot control with weak ones, and discourages reckless calls.

Self-Play Implementation

Our self-play system includes:

- Opponent Pool: Starts with RandomAgent
- Snapshotting: Every 500 episodes, a copy of the agent is added to the pool

- Random Sampling: Each training episode samples a random opponent
- Environment Configuration: Opponents are passed to `env.set_agents()`

Strategic advantages include:

- Curriculum learning (facing stronger versions of self)
- Exposure to diverse strategies
- Robust policy development

Experiment

Our project employs a neural network design specifically suited to address the particular challenges of No-Limit Texas Hold'em. The architecture builds upon the general actor-critic paradigm with several key improvements aimed at capturing the complex patterns and strategic elements inherent in poker play.

Parameter	Value
GAE Lambda	0.95
Value Coefficient	0.5
Entropy Coefficient	0.05
Clip Ratio (PPO-Clip)	0.2
KL Target (PPO-KL)	0.01

Table 1: PPO Hyperparameters

Parameter roles:

- **GAE Lambda:** Balances bias-variance for long-term reward credit
- **Value Coefficient:** Weighs value loss vs policy improvement
- **Entropy Coefficient:** Encourages exploration; decays over time
- **Clip Ratio:** The ratio between the probability of an action taken by a new policy and the probability of that same action taken by the old policy.
- **KL Target:** Computed by finding the expected difference in log probabilities between the two policy distributions and add a penalty to the loss function to prevent large policy updates.

Training Details

- Trained for 25,000 episodes
- Trained agents against a mix of RandomAgent and past selves
- Evaluation performed every 50 episodes
- Metrics: average reward, win rate, best reward during evaluation

[1] *Average reward calculated over the last 100 episodes of training.

Results

The following section presents a comparative analysis of key performance metrics between two reinforcement learning algorithms implemented for No-Limit Texas Hold'em, PPO-Clip vs PPO-KL.

Metric	PPO-Clip	PPO-KL
Average reward*	884.9515	98.03
Final win rate	0.3874	0.9177
Best evaluation reward	5041.7968	2355.66
Best win rate	0.3888	0.9160

Table 2: PPO-CLIP vs PPO-KL

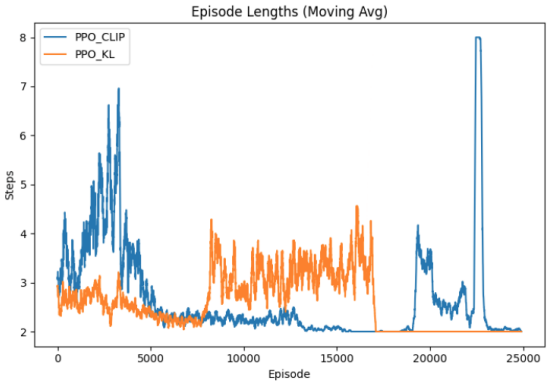

Figure 1: KL vs Clip: Episode Lengths

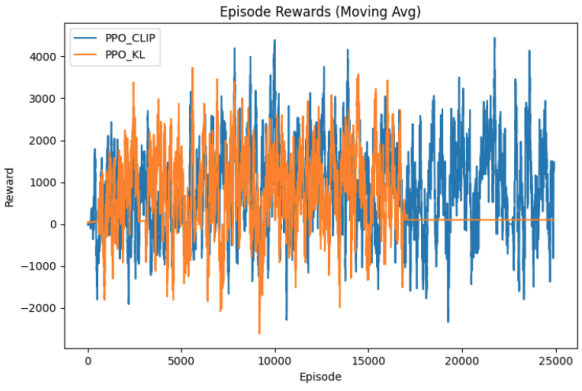

Figure 2: KL vs Clip: Episode Rewards

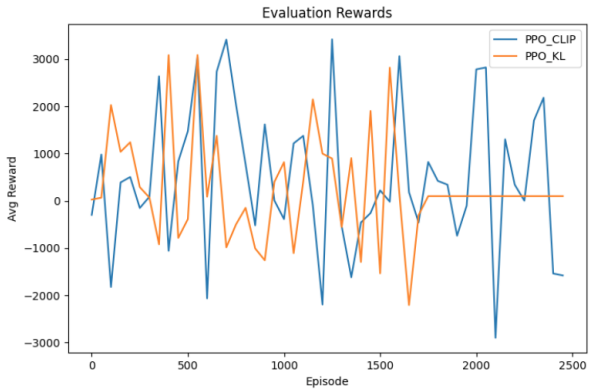

Figure 3: KL vs Clip: Evaluation Rewards

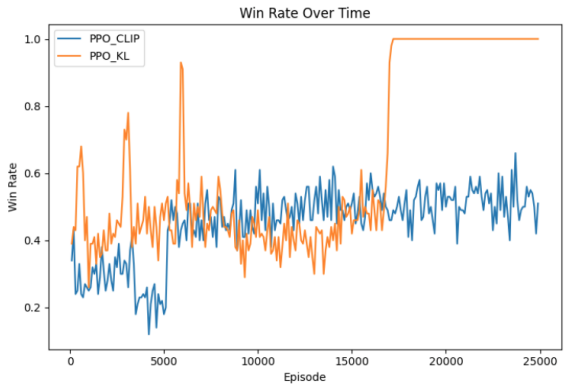

Figure 4: KL vs Clip: Win Rate

Interpretation

From the training curves in Figures 1, 2, and 3, we observe a striking behavioral divergence between PPO-KL and PPO-Clip. The PPO-KL agent exhibits a sharp decline in episode length, episode reward, and evaluation reward—converging near zero—while its win rate simultaneously spikes to nearly **100%**. This apparent contradiction can be explained by the nature of the strategy the agent adopts.

PPO-KL appears to have discovered a highly aggressive policy, consistently going all-in early in the hand. This forces the opponent (typically a RandomAgent or a past version of the agent itself) to fold, ending the episode quickly and without significant pot buildup. As a result, the agent accumulates wins but receives minimal chip rewards, leading to short episodes and low average returns. The KL constraint plays a critical role here: by penalizing large deviations from the existing policy, it encourages the agent to maintain this exploitative strategy once it is found. The algorithm essentially "locks in" the agent's behavior, suppressing further exploration and preserving what it has identified as an optimal policy within the current opponent pool.

In contrast, the PPO-Clip agent demonstrates continued exploration throughout training. The clipping mechanism allows for controlled yet flexible policy updates, enabling the agent to experiment with a broader range of strategies. This results in more variable episode lengths and reward outcomes, but also higher peaks in evaluation reward. PPO-Clip does not converge as tightly as PPO-KL, but it exhibits richer strategic behavior and greater potential for discovering high-reward trajectories.

The final quantitative results support these observations. **PPO-Clip** achieves an average reward of **884.95** over the last 100 episodes and a final win rate of **38.74%**, with a best evaluation reward of **5041.80**. On the other hand, **PPO-KL** secures a much higher final win rate of **91.77%** but only an average reward of **98.03**, and a best evaluation reward of **2355.67**. These results indicate that PPO-Clip trades consistency for strategic diversity and higher reward potential, whereas PPO-KL prioritizes stability and exploitability in a narrow, high-confidence policy regime.

Strategically, PPO-KL functions as a low-variance, high-confidence exploiter of its opponent pool. Its tendency to repeat an early all-in strategy illustrates the danger of overfitting to a predictable or passive opponent, particularly when the learning algorithm discourages policy change. PPO-Clip, while less consistent, offers a more robust foundation for generalization in dynamic multi-agent environments, as it continues to probe the strategy space throughout training. This analysis exemplifies the classic *exploration–exploitation trade-off* in reinforcement learning, with each PPO variant embodying a distinct philosophy toward policy optimization under uncertainty.

Future Directions

While this study focuses on a two-player version of No-Limit Texas Hold'em, extending the framework to support **multi-player games** (e.g., 5 or 6 players) is a critical next step. Real-world poker involves more complex interactions, including positional strategy, multi-way pots, and greater variance in opponent behavior. Training agents in a multi-agent setting will require scalable self-play strategies and more robust opponent modeling.

A key observation from this study is that the PPO-KL agent often converges to an **aggressive all-in strategy**, likely leveraging frequent bluffing to force folds and secure easy wins. While effective in a limited opponent pool, such a strategy may be brittle against stronger or more adaptive agents. Therefore, a valuable extension would involve designing mechanisms for detecting and responding to **bluffs**.

To simulate bluffing and bluff-catching behavior, several techniques can be explored:

- **Opponent Modeling**: Train an auxiliary model that predicts the likelihood of an opponent bluffing based on observed actions and betting patterns. This model could inform the agent's decision to call or fold in uncertain situations.
- **Bayesian Policy Inference**: Use probabilistic reasoning to estimate opponent strategy types

(e.g., aggressive, passive, balanced) and adapt accordingly. [8]

- **Meta-Learning**: Allow agents to rapidly adapt to novel opponents, enabling detection of unusual behaviors such as excessive aggression or inconsistency that might indicate bluffing.

Additionally, incorporating **exploitability analysis** or testing agents against known strong policies (such as Counterfactual Regret Minimization-based [9] bots) could better assess whether discovered strategies are truly robust or simply overfit to the training pool.

Ultimately, enabling agents not only to bluff but also to detect and adapt to bluffs is crucial for advancing towards human-level poker performance.

Conclusion

This ablation study highlights how different PPO constraint mechanisms influence strategy development in complex, imperfect-information games like poker. PPO-KL's conservative updates lead to consistent exploitation of learned strategies, while PPO-Clip supports broader exploration and higher-reward opportunities. Our custom enhancements—including a rich state vector, shaped rewards, and dynamic self-play—were essential in enabling these agents to learn meaningful strategies.

Future work could explore combining both PPO variants, leveraging PPO-Clip's exploration early in training and PPO-KL's stability later on. This project offers valuable insight into how RL agents evolve distinct styles based on algorithmic design and environmental guidance, and provides a strong foundation for building more robust poker-playing agents.

GitHub Repository

The complete source code for this project is available on GitHub at the following link: `https://github.com/CS-5180/No-Limit-Texas-Hold-em`

References

[1] N. D. L. Fuente and D. A. V. Guerra, *A comparative study of deep reinforcement learning models: Dqn vs ppo vs a2c*, 2024. arXiv: `2407.14151 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2407.14151`.

[2] OddsShark, *No limit Texas Hold'em poker game*, `https://www.oddsshark.com/poker/games/no-limit-texas-holdem`, Accessed: April 2025, n.d.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. arXiv: `1707.06347 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1707.06347`.

[4] J. Hui, *RL — proximal policy optimization (PPO) explained*, `https://jonathan-hui.medium.com/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12`, Accessed: April 2025, 2018.

[5] OpenAI, *Proximal policy optimization (PPO)*, `https://spinningup.openai.com/en/latest/algorithms/ppo.html`, Accessed: April 2025. Revision: 038665d6, 2018.

[6] LovelyBuggies, *From PG to PPO*, `https://lovelybuggies.github.io/archive/posts/From_PG_2_PPO.pdf`, Online Document, Accessed: April 2025, 2025.

[7] DATA Lab, *Rlcard.games.nolimitholdem — RLCard documentation*, `https://rlcard.org/rlcard.games.nolimitholdem.html`, Accessed: April 2025, n.d.

[8] BetMGM, *Online casinos with progressive jackpot slots*, `https://casino.betmgm.com/en/blog/online-casinos-with-progressive-jackpot-slots/`, Accessed: April 2025, 2023.

[9] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Advances in Neural Information Processing Systems*, vol. 20, MIT Press, 2008.