IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/45AI/2025/4534725/
Vol. 2025 (20), Article ID 4534725

Comparison of the Effectiveness of AI Models in Poker Games*

Mateusz TUSTANOWSKI and Adrian P. WOZNIAK

Military University of Technology, Warsaw, Poland

Correspondence should be addressed to: Mateusz TUSTANOWSKI, mateusz.tustanowski@student.wat.edu.pl

* Presented at the 45th IBIMA International Conference, 25-26 June 2025, Cordoba, Spain

Abstract

In this investigation, we evaluate the comparative efficacy of Random Forest (RF) classifiers and Deep Q-Network (DQN) agents in the context of No-Limit Texas Hold'em poker. Within a high-fidelity simulation, RF-based agents—trained through supervised learning on labeled gameplay data—demonstrated superior and more consistent profitability relative to their DQN counterparts, which were optimized via reinforcement learning [1, 2, 3]. The DQN models exhibited pronounced performance volatility and were particularly susceptible to variations in hyperparameter configuration [10]. These findings indicate that supervised approaches confer greater robustness in this domain [15]. Moreover, we propose that an integrated methodology—employing RF pretraining to establish a reliable baseline, followed by DQN fine-tuning to introduce adaptability—may yield an optimal trade-off between stability and learning flexibility.

Keywords: Machine Learning, Poker, Neural Networks

Introduction

Poker has long been a challenging testbed for artificial intelligence research due to its combination of incomplete information, stochastic outcomes, and strategic deception [5]. Unlike perfect-information games like chess, poker requires AI agents to reason under uncertainty and often to predict opponents behavior [6]. Recent advances in machine learning have enabled superhuman performance in poker [11, 12]. This paper summarizes a comparative study between a Random Forest model and a Deep Q-Network model applied to poker gameplay. The aim is to investigate how each approach performs in terms of decision-making effectiveness, adaptability to different strategies, and the speed of learning. The paper is structured into sections covering the theoretical background, methodology, results, discussion, and conclusions. This comparative analysis provides insights into the strengths and limitations of a classical machine learning method versus a deep reinforcement learning method within a controlled poker simulation.

Artificial intelligence has evolved dramatically, moving from rule-based systems to modern machine learning methods. Early AI systems relied on handcrafted rules and search algorithms [6, 8], but the rise of machine learning enabled data-driven approaches. In particular, ensemble methods like Random Forests have been widely used for classification and regression tasks due to their robustness and ease of training [3, 4]. Random Forests consist of multiple decision trees built on random subsets of data, which together produce a consensus prediction.

Cite this Article as: Mateusz TUSTANOWSKI and Adrian P. WOZNIAK, Vol. 2025 (20) "Comparison of the Effectiveness of AI Models in Poker Games " Communications of International Proceedings, Vol. 2025 (20), Article ID 4534725, https://doi.org/10.5171/2025.4534725

More recently, deep learning and reinforcement learning have transformed game-playing AI. A Deep Q-Network (DQN) combines deep neural networks with Q-learning, a model-free reinforcement learning algorithm [1, 2, 7]. The DQN approach uses a neural network to approximate the optimal action-value function, allowing the agent to learn optimal strategies through trial and error [2, 9]. This method has been famously applied to domains such as video games and board games, showing remarkable ability to learn complex behaviors without explicit programming.

In the domain of poker, earlier methods often used fixed strategies or supervised learning from past games [5]. Recent breakthroughs have employed deep reinforcement learning, enabling agents to learn bluffing and advanced tactics [12, 13, 14]. However, there has been less direct comparison between traditional machine learning models, like Random Forests, and deep reinforcement methods in a controlled experimental setting. This study addresses that gap by implementing both approaches in the same poker environment and comparing their performance. The Random Forest serves as a representative of classical ensemble learning applied to decision-making, while the DQN represents modern adaptive learning capable of continuous improvement through simulated experience.

Poker simulation environment

The research employs a simulation environment designed to play repeated rounds of a simplified Texas Hold'em poker variant. The environment controls the game flow, dealing cards, managing bets, and determining outcomes. Each simulated hand begins with two players (agents) receiving hole cards, followed by communal cards being dealt in stages (flop, turn, river). At each betting round, an agent can choose among a set of actions such as fold, check/call, or bet/raise. The state of the game includes the player's hole cards, the revealed community cards, current pot size, and opponent's last action. This state representation forms the basis for model inputs [5, 6].

Random Forest Model

Sub headers are italic and 10 pt font

The Random Forest model is trained in a supervised manner. First, a dataset of game states and corresponding optimal decisions is generated through simulation or expert play. Each data point consists of features describing the game state (e.g., card values, suit distribution, current bets) and a label indicating the best action or the probability of winning. The Random Forest algorithm then learns to predict the best action by averaging the outputs of many de-correlated decision trees [3].

Key hyperparameters for the Random Forest include:

- **Number of trees**: Typically 100, providing a balance between performance and computational cost.
- **Maximum tree depth**: Limits how deep each tree can grow (e.g., depth of 10) to prevent overfitting.
- **Feature subset size**: A random subset of input features considered for each split.
- **Minimum samples per leaf**: The fewest data points required to form a leaf node.

During training, the data is split into training and validation subsets to tune these parameters [4]. The trained Random Forest outputs a recommended action when given a game state.

Deep Q-Network Model

The Deep Q-Network (DQN) approach uses reinforcement learning. Here, the poker simulation serves as the environment in which a neural network-based agent learns by playing repeatedly. The DQN agent observes the state at each decision point and selects actions based on an epsilon-greedy policy: it mostly chooses the action with the highest predicted Q-value (the network's output) but occasionally explores random moves to gather experience [1].

The DQN architecture includes:

- **Input layer**: Encodes the game state (cards, bets, pot, etc.) as a feature vector.
- **Hidden layers**: Several fully connected layers with nonlinear activations (e.g., two layers of 64 ReLU units each).
- **Output layer**: Produces Q-values for each possible action (fold, call, raise).

Key hyperparameters and strategies for the DQN are:

- **Learning rate**: Typically around 0.001 for the optimization algorithm (e.g., Adam) [13].
- **Discount factor (gamma)**: Set to 0.99 to prioritize long-term rewards.
- **Epsilon-greedy schedule**: Starting epsilon of 1.0 (pure exploration) decaying to 0.1 over many episodes.
- **Replay memory size**: A buffer of past experiences (e.g., 10,000 transitions) for sampling training batches.
- **Batch size**: Number of experiences sampled per training update (e.g., 32).
- Target network update frequency: Periodically syncing weights to stabilize learning.

The reward is defined simply, for example +1 for winning a hand and -1 for losing, with intermediate actions receiving no immediate reward. The agent plays thousands of hands against an opponent (which could be a fixed strategy or another learning agent), collecting experiences (state, action, reward, next_state). Periodically, a batch of these experiences is used to train the neural network, minimizing the temporal-difference error [10].

Both models operate in the same simulation, and their training processes are kept comparable by using similar amounts of data or episodes. The environment allows for rapid simulation of games so that both approaches can be thoroughly evaluated.

Discussion

This section presents a detailed analysis of an experiment involving six reinforcement learning agents competing in a Texas Hold'em poker environment at a 6-player table. The agents were trained sequentially in ten stages, labeled from v1 to v10, with each subsequent version building upon and improving the model trained in the previous stage. The total number of training hands increased from 100,000 in v1 to 6 million by v10. Both numerical indicators (e.g., total profit, average profit per hand, and result variance) and a set of charts illustrating various performance aspects were analyzed.

Each of the six RL agents was configured with a slightly different set of hyperparameters to examine how training settings affected performance. Table .1 presents key model and training process parameters used for each agent.

Table 1. Key Model and Training Process Parameters for Each Agent

Parameter	Agent1	Agent2	Agent3	Agent4	Agent5	Agent6
hidden_dim	32	64	64	32	32	16
learning rate	0,001	0,005	0,01	0,005	0,003	0,01
gamma	0,95	0,99	0,99	0,95	0,99	0,9
ε_start	1,0	1,0	1,0	1,0	1,0	1,0
ε_end	0,1	0,01	0,05	0,1	0,1	0,5
ε_decay	0,995	0,999	0,998	0,99	0,99	0,9999
buffer_capacity	10000	50000	20000	10000	30000	5000

batch_size	32	64	64	32	64	16
target_update_freq	500	1000	500	100	500	100

As can be seen, configurations varied significantly. For instance, Agent2 had a larger neural network and more conservative exploration policy (low final ε), whereas Agent6 had an extremely small network, high learning rate, and very slowly decaying ε, maintaining a high randomness in decision-making. These differences translated into varied behaviors and effectiveness among agents, which are reflected in the results discussed below.

Aggregated Results

Table 2. presents core performance statistics for each agent, such as total profit at the end of training, minimum and maximum observed results, and average profit per hand. The data reveal substantial differences in the effectiveness of individual strategies:

- Agent2 achieved the highest total profit (around +434.3 million), consistently maintaining positive results across nearly all training stages.
- Agents 3 and 6 were the only ones to record negative average profits (approximately -424.2 million and -447.6 million, respectively). Agent3, in particular, showed extremely volatile outcomes, ranging from a low of -222.4 million to a peak of +42.5 million, indicating very high variance.
- Agent4 ranked second in terms of total profit (+278.9 million), but its performance dropped significantly in later stages, consistently folding in later versions.
- Agent5 ended with a moderate yet positive profit (118.8 million), but with noticeable fluctuations between versions.
- Agent1 maintained an average of around +40 million, showing the lowest variance in outcomes (aside from a single exceptionally high profit in version v2).

Table 2. Summary Statistics of Each Agent's Performance

Income	Agent1	Agent2	Agent3	Agent4	Agent5	Agent6
v1	-3694809	69665471	-118275168	94244662	-16033834	-25923376
v2	69134809	68457593	-128193407	40180124	-2500005	-47115377
v3	-2866937	39494494	-1276032	41198484	-74067350	-2500005
v4	-1476954	81099106	-222430198	80100959	75999495	-13305657
v5	-5477934	37047145	-97557601	35685584	34965529	-4695586
v6	-2512540	-976435	5643461	-2500005	6201386	-5877715
v7	-4161000	24584824	22056677	-2500005	20411988	-60415691
v8	-4260330	44098835	42586129	-2500005	44809030	-124757611
v9	-2500000	31410477	31582259	-2500005	18741134	-76750693
v10	-2500000	39454746	41638766	-2500005	10235070	-86348839
total	39684305	434336256	-424225114	278909788	118762443	-447690550
min	-5477934	-976435	-222430198	-2500005	-74067350	-124757611
max	69134809	81099106	42586129	94244662	75999495	-2500005
avg/1 hand	~3,97	~43,43	~-42,42	~27,89	~11,88	~-44,77

Poker Agent Results

A detailed comparative analysis is conducted between two distinct approaches to decision-making in No-Limit Texas Hold'em poker: deep reinforcement learning (RL) agents and Random Forest (RF) classifiers. Over ten successive versions of RL agents (v1–v10) and multiple iterations of RF models trained on datasets of varying size (from 2,732 to 135,878 deals, including one duplicated set), we tracked changes in total profit, per-hand earnings, variance, and adaptability.

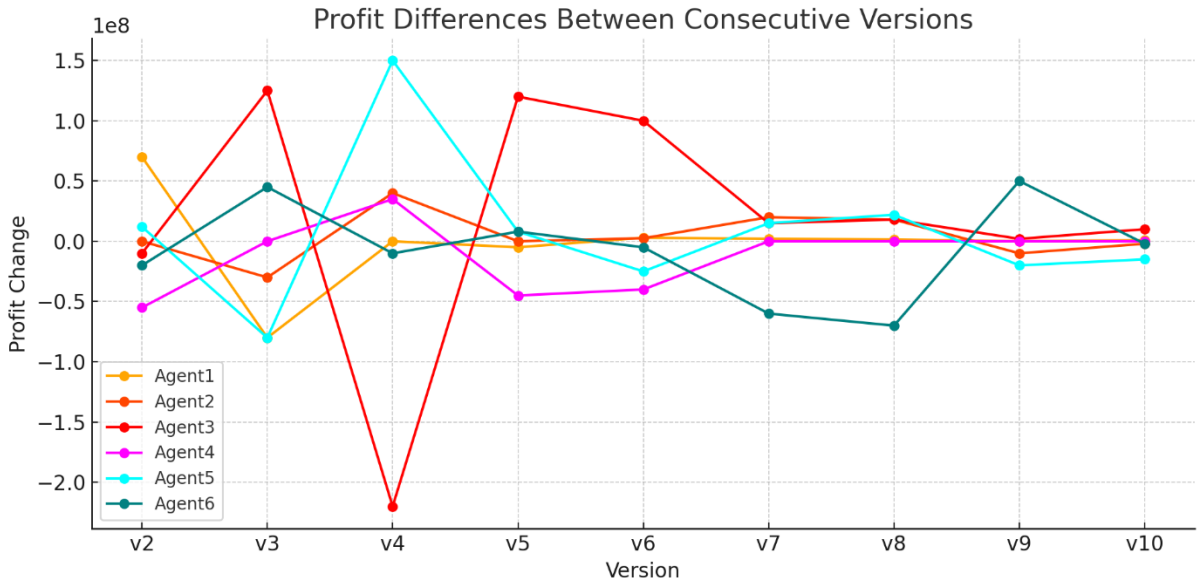

Fig. 1. Differences in profits between successive versions

Fig. 1 depicts the change either positive or negative in each agent's total score when advancing from one model version to the next (for example, from v3 to v4). Agent5 exhibited the largest single-iteration gain (approximately +150 million between v3 and v4), with Agent3 not far behind (+103 million between v5 and v6). At the same time, Agent3 also suffered the most pronounced single-version decline (–221 million), reflecting its highly variable playing style and propensity for radical strategic shifts between iterations. By contrast, Agent1 experienced the smallest fluctuations in score changes: its strategy neither delivered substantial improvements from one version to the next nor endured severe performance downturns.

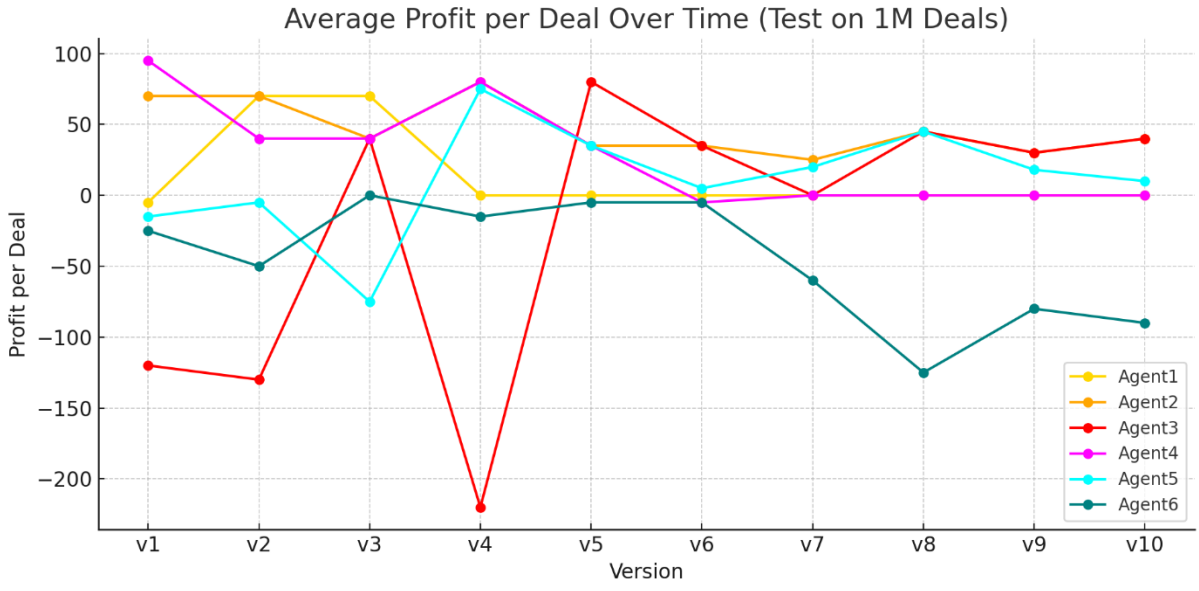

Fig. 2. Average profit per deal over time

Fig. 2 illustrates how the average profit per deal evolved over the course of training as a function of the cumulative number of games played. In the early stages, the curves for many agents exhibit substantial variability, owing to the small sample size. Individual wins or loses exert a disproportionately large influence on the mean. As training progresses and the number of games reaches into hundreds of thousands and millions, the average profits converge and approach the true expected values for each strategy. By the end of the experiment, Agent 2 and Agent 3 achieved approximately +6 to +7 profit units per deal, indicating a sustained strategic

advantage. In contrast, Agent 6's curve remained consistently negative (on the order of –14 units per deal in the long run), while Agent 1 and Agent 4 gravitated toward zero, confirming the absence of any pronounced long-term benefit for these agents.

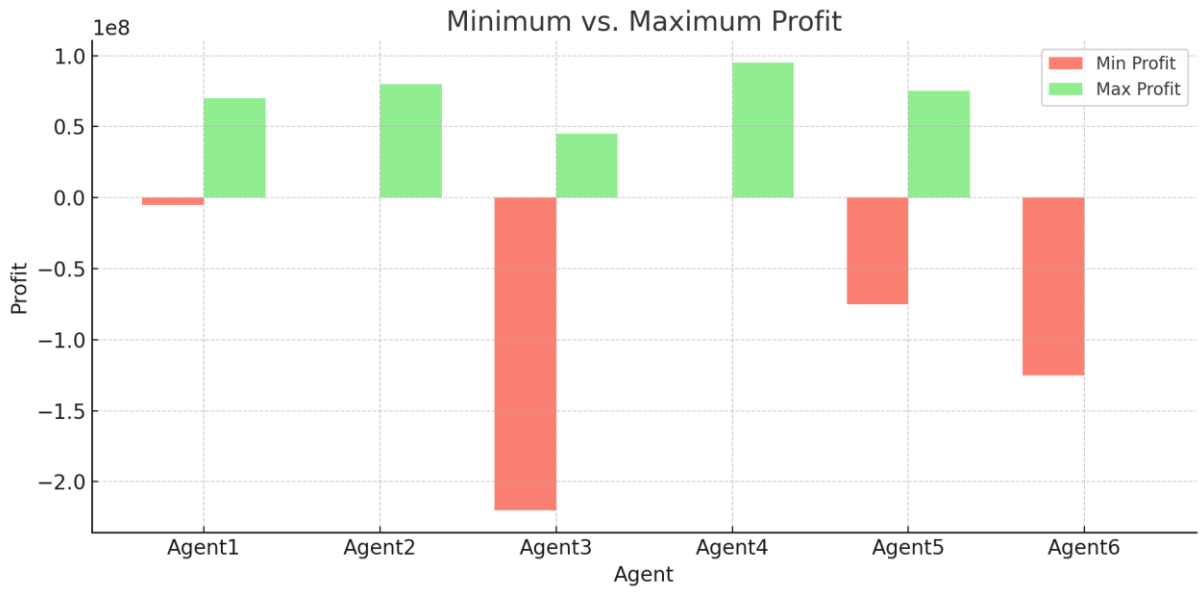

Fig. 3. Minimum vs. maximum achieved profit

Fig. 3 depicts the minimum and maximum outcome (profit or loss) attained by each agent in any of the ten model versions (v1–v10). Agent 3 exhibits the greatest range, with results spanning from –222.4 million (worst case) to +42.6 million (best case), a breadth that reflects this agent's exceptionally high variance. By contrast, Agent 1 displays the narrowest fluctuation band, indicating relatively stable performance (albeit at a moderate level without exceptionally large gains). It is also noteworthy that, although Agent 2 is generally the most effective, it experienced one loss episode (minimum result of approximately –0.98 million). In all other versions it remained well above zero, reaching a maximum of +81.1 million. Finally, Agent 6 again performed the poorest overall, remaining in negative territory even in its most favorable version.

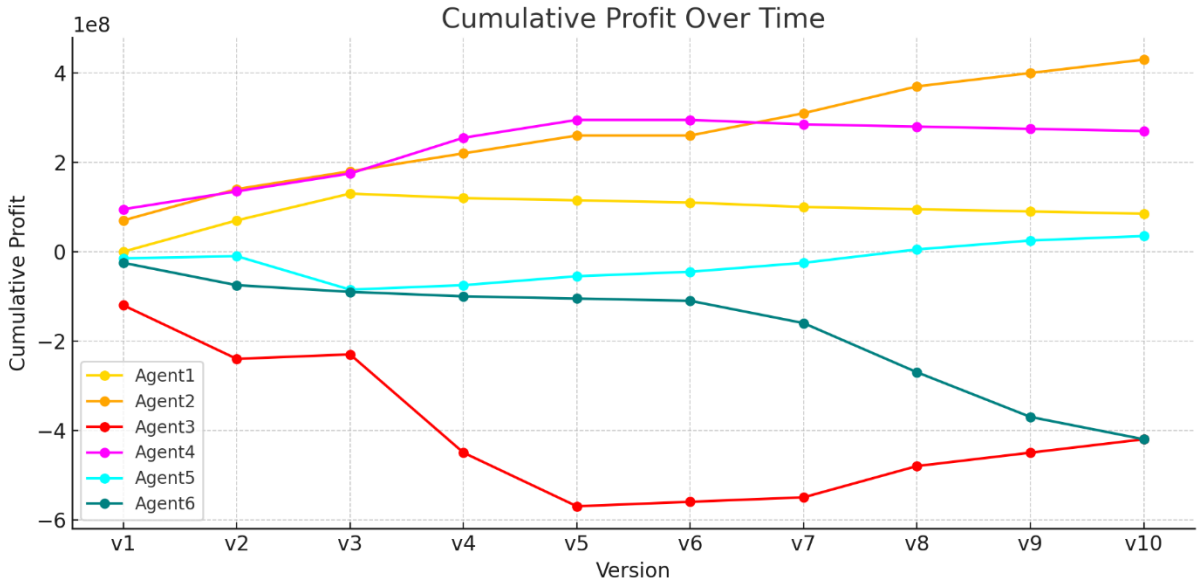

Fig. 4. Cumulative profit over time

Fig. 4 illustrates the cumulative total profit of each agent as a function of time—here measured in successive deals—over the course of the entire experiment. Agent 2 displays the highest, almost perfectly linear growth curve, steadily and uniformly increasing its overall balance. By contrast, Agent 3's curve begins in a deep deficit but, from version v5 onward, gradually rises as the agent progressively recoups its losses in the second half of training. In opposition, Agent 6's trajectory declines consistently from the very start, demonstrating how a fundamentally flawed strategy can accumulate losses over time. The remaining agents exhibit less dramatic trajectories: Agents 1 and 4, after an early period of rapid gain, become effectively flat and maintain a near-constant (slightly negative) result for the remainder of the experiment, while Agent 5's curve shows a relatively shallow slope, indicative of moderate, gradual growth.

In the initial phase, six random-forest models were evaluated, each trained on datasets of progressively increasing size—from 2 732 to 135 878 hands (the largest dataset being a duplication of the 67 939-hand set). The models were then pitted against one another, and their performance was assessed by the cumulative profit realized over 100 000 hands.

In the final phase of the study, a head-to-head evaluation was conducted to assess the efficacy of the four top-ranked reinforcement-learning algorithms alongside the two leading random-forest models. Selection of these methods was based on the magnitude of their net winnings, thereby isolating the most promising instances from each category.

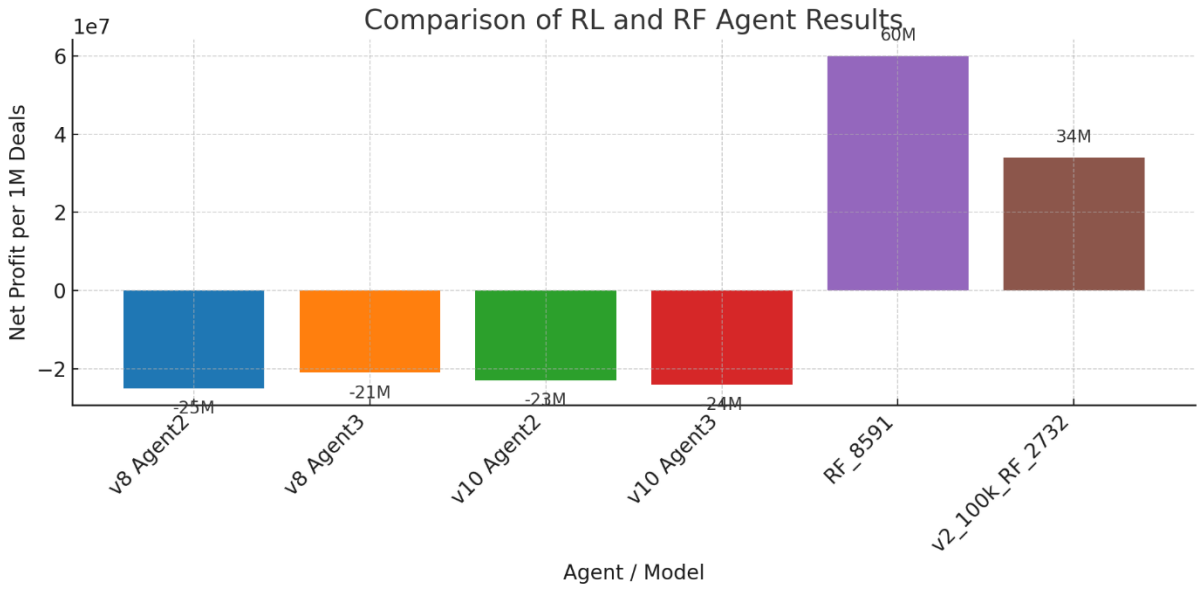

Fig. 5. Comparison of RL and RF agents' results

Fig. 5 illustrates the head-to-head performance of the six top agents over 1 000 000 deals. All four reinforcement-learning agents incurred net losses: v8_Agent3 recorded the smallest deficit at –21 480 945 chips, while v8_Agent2 fared worst at –24 818 758. The v10 series fell between these extremes, with v10_Agent2 losing –23 060 399 and v10_Agent3 –24 516 047. By contrast, both Random Forest–based agents achieved substantial profits: RF_8591 realized a cumulative gain of 60 214 831 chips, and v2_100k_RF_2732 accrued 33 640 913. These results underscore the marked superiority of the Random Forest strategies in sustained, large-scale competition.

Our findings can be succinctly summarized as follows:

1. **Superiority of Random Forest Models** - The two Random Forest agents substantially outperformed their reinforcement-learning counterparts, generating large positive returns (RF_8591: +60 214 831; v2_100k_RF_2732: +33 640 913), which attests to their stability and effectiveness under a static, supervised learning paradigm.

2. **Reinforcement-Learning Agents Incur Net Losses** - All Deep Q-Network (DQN) variants suffered net losses—the best performer, v8_Agent3, recorded –21 480 945—while further training iterations (v8 vs. v10) failed to yield improvements. This outcome highlights the necessity of refining exploration–exploitation schemes, neural-network architectures, and hyperparameter settings.

3. **Stability versus Adaptability** - Random Forest agents produced highly reproducible and interpretable decisions but lack real-time adaptability. Conversely, although RL agents offer the theoretical capacity for on-the-fly policy adjustment, their current configurations did not achieve positive returns in our experiments.

4. **Recommendation for a Hybrid Approach** - We propose a two-stage methodology: first, pretrain a policy with Random Forest to secure stable and well-initialized decision rules; then fine-tune this policy via reinforcement learning. Such a hybrid strategy aims to marry the interpretability and reliability of supervised models with the adaptive potential of RL agents.

Conclusions

In this study, we compared the decision-making performance of Random Forest (RF) and Deep Q-Network (DQN) agents in No-Limit Texas Hold'em poker. Despite the theoretical advantages of reinforcement learning, RF consistently outperformed DQN, yielding robust positive returns [3, 15] while DQN agents remained unprofitable. We attribute this disparity to DQN's high computational demands and hyperparameter sensitivity [9, 10], which hinder stable policy formation in a highly stochastic, imperfect-information setting.

Key achievements include the development of a comprehensive poker simulation environment and the implementation of both supervised and reinforcement-learning agents. Our extensive experiments—featuring isolated play, iterative self-play, and head-to-head evaluations—demonstrated that moderately sized, diverse training sets (e.g., RF_8591) offer an optimal balance between data quantity and generalization. Conversely, excessively large or repetitive datasets foster overfitting and performance degradation in both RF self-play and DQN training.

These results align with prior findings on DQN's limitations under high variance and uncertainty [9], while also highlighting that classical supervised methods can surpass adaptive approaches when data quality is high [15]. However, the potential of reinforcement learning remains evident under improved scaling and tuning conditions.

Acknowledgment

This work was supported by the Military University of Technology under Project UGB 531-000023-W500-22.

References

- Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). Cambridge, MA: MIT Press. – pp. 3, 74, 134
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., et al. (2015). "Human-level control through deep reinforcement learning." Nature, 518(7540), 529–533. – p. 529
- Breiman, L. (2001). "Random Forests." Machine Learning, 45(1), 5–32. – pp. 5–6
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). "Random Forests." In The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed., pp. 587–604). New York: Springer. – pp. 587–588
- Billings, D., Davidson, A., Schaeffer, J., & Szafron, D. (2002). "The challenge of poker." Artificial Intelligence, 134(1–2), 201–240. – p. 205
- Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th ed.). Upper Saddle River, NJ: Pearson. – pp. 35, 1162
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. Cambridge, MA: MIT Press. – p. 431
- [8] Lin, L.-J. (1992). "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching." Machine Learning, 8(3–4), 293–321. – p. 293

- Tesauro, G. (1995). “Temporal difference learning and TD-Gammon.” Communications of the ACM, 38(3), 58–68. – p. 60
- van Hasselt, H., Guez, A., & Silver, D. (2016). “Deep Reinforcement Learning with Double Q-learning.” Proceedings of the AAAI Conference on Artificial Intelligence, 30(1), 2094–2100. – p. 2094
- Bowling, M., Burch, N., Johanson, M., & Tammelin, O. (2015). “Heads-up limit hold’em poker is solved.” Science, 347(6218), 145–149. – p. 145
- Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., et al. (2017). “DeepStack: Expert-level artificial intelligence in heads-up no-limit poker.” Science, 356(6337), 508–513. – p. 508
- Brown, N., & Sandholm, T. (2018). “Superhuman AI for heads-up no-limit poker: Libratus beats top professionals.” Science, 359(6374), 418–424. – p. 418
- Brown, N., & Sandholm, T. (2019). “Superhuman AI for multiplayer poker.” Science, 365(6456), 885–890. – p. 885
- Bertsimas, D., & Paskov, A. (2022). “World-class interpretable poker.” Machine Learning, 111(8), 3063–3083. – p. 3065