Reinforcement Learning-based AI for a Card Game Yu-Gi-Oh!

Koki Nakagawa s1300164 Supervised by Prof. Maxim Mozgovoy

Abstract

This study applies reinforcement learning to Yu-Gi-Oh!, a complex imperfect-information trading card game. An AI environment was constructed using EDOPro and WindBot, and a simplified Neural Fictitious Self-Play (NFSP) framework combining supervised learning and reinforcement learning was implemented.

Game states were represented as 32-dimensional feature vectors, and actions were modeled hierarchically at phase and card levels. Reinforcement learning was conducted using offline linear Q-learning with a mixing parameter η controlling policy selection.

Although no significant improvement in win rate was observed due to limited training data, behavioral analysis showed that increasing η led to more activation actions and greater use of reinforcement-learning-based decisions. These results indicate that even a simplified NFSP framework can influence strategic behavior in a complex trading card game environment.

1 Introduction

1.1 Background

In recent years, advancements in artificial intelligence technology have significantly advanced AI research in the gaming field. In perfect information games like Go and Shogi, systems like AlphaGo [1] and AlphaZero [2], utilizing self-play reinforcement learning, have achieved performance surpassing humans, demonstrating the effectiveness of reinforcement learning.

On the other hand, trading card games (TCGs), treated as games of imperfect information, present significant challenges for constructing robust decision-making methods; including reinforcement learning using simple search or rule-based approaches. This is due to factors such as the non-disclosure of hand information, probabilistic elements, complex card effects, and an enormous number of action choices.

1.2 Related Work

Among AI research on games of incomplete information, poker AI research has made particularly significant progress. AI systems like DeepStack [3] and Libratus [4] achieve high performance by combining reinforcement learning and game theory techniques, representing a landmark success in decision-making under incomplete information.

One learning method proposed in this field is Neural Fictitious Self-Play (NFSP) [5]. NFSP enables stable strategy learning in self-play environments by combining best-response learning via supervised learning. Its effectiveness has been reported in poker environments.

Regarding AI research targeting TCGs, studies using Monte Carlo Tree Search (MCTS) have been conducted on Magic: The Gathering (MTG) [6]. That study addresses challenges such as handling the enormous state space resulting from card combinations and dealing with probabilistic elements.

However, to the best of my knowledge, no reinforcement learning research targeting Yu-Gi-Oh! [7] a particularly popular TCG in Japan, has been found.

Unlike MTG, Yu-Gi-Oh! does not employ a conventional mana resource system and instead includes unique mechanics such as tribute summoning, complex effect chains, and highly flexible action sequences, These characteristics suggest that approaches designed for other TCGs may not be directly applicable, and specialized methods may be required for effective AI decision-making.

Furthermore, Yu-Gi-Oh! has an official digital platform, Master Duel [8], which indicates the potential for future AI-based gameplay. This highlights the practical relevance of developing AI techniques for this game.

1.3 Target / Environment

This study focuses on Yu-Gi-Oh! and constructs AI agents using EDOPro [9], an open-source duel simulator, and WindBot [10], its AI execution environment.

Since WindBot already implements card effect processing and match progression, this research can focus on designing and evaluating the learning

algorithms for the decision-making component rather than developing the game engine itself.

1.4 Purpose

The purpose of this study is to establish a foundation for treating Yu-Gi-Oh! a popular trading card game in Japan, as a subject for AI research. Furthermore, it aims to implement a simplified version of Neural Fictitious Self-Play (NFSP) within this environment and experimentally verify its effectiveness.

1.5 Contributions

The main contributions of this study are as follows:

1. Construction of an experimental AI environment for Yu-GI-Oh! using EDOPro and WindBot.
2. Implementation of a simplified Neural Fictitious Self-Play (NFSP) framework adapted for a trading card game environment.
3. Experimental evaluation of the effectiveness of the proposed approach in improving decision-making performance.

2 Method

2.1 System Overview

This study constructs a learning environment based on EDOPro, an open-source Yu-Gi-Oh! duel simulator, and WindBot, its built-in AI execution framework.

EDOPro handles the games rules, card effects, and duel progression, while WindBot is responsible for decision-making. The proposed learning module is implemented inside WindBot and replaces part of its original rule-based logic with machine-learning-based policies.

At each decision step, the current duel state is converted into a fixed-length feature vector and passed to the policy module. The selected action is then executed in the simulator, and the transition is recorded in a log file. These logs are later used for supervised learning and reinforcement learning.

This design allows the study to focus on learning algorithms.

Figure 1 illustrates the overall architecture of the proposed system.

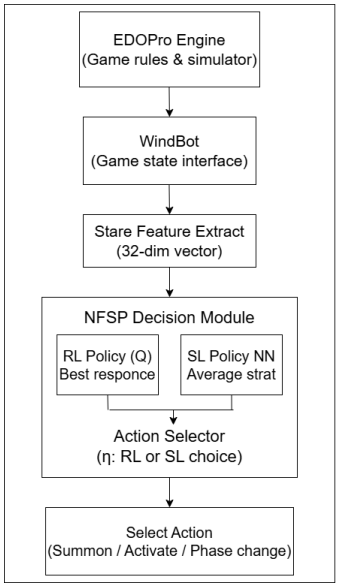

Figure 1: Overall architecture of the proposed system integrating EDOPro, WindBot and the NFSP-based decision module.

2.2 State Representation

To enable machine learning, the duel state is represented as a numerical feature vector.

The state is encoded as a 32-dimensional vector. Including information such as life points, hand size, field status, and game phase indicators. Each element is normalized to maintain numerical stability during learning. The dimension was chosen to balance representational capacity and computational efficiency.

This simplified representation is designed to capture essential strategic information while keeping the model computationally lightweight.

2.3 Action Representation

The action space in Yu-Gi-Oh! is extremely large due to complex card effects and numerous decision branches. To handle this, actions are represented using a two-stage hierarchical structure.

1. Head decision (phase-level action)

The agent first selects a high-level action:

- Summon.
- Activate.
- MonsterSet.
- SpellSet.
- Repos.
- ToBattlePhase.
- ToMainPhase2.
- ToEndPhase.

This determines the type of operation executed in the current phase.

2. Activate decision (card-level action)

If the Head decision is Activate, a second policy selects the specific card effect to activate.

Each candidate action is encoded as:

Activate | card_id | description

These keys uniquely identify activation options and are mapped to integer IDs for learning.

2.4 Supervised Learning Policies

To initialize a stable policy, we prepare models for the head policy (decision-level policy) network and the activate network and train a supervised learning model from recorded duel logs.

Both networks take the 32-dimentional state vector as input, select the highest-scoring action among legal actions, and output an action score. These supervised policies provide stable baseline performance and guidance for NFSP training.

2.5 Reinforcement Learning Agents

To improve decision-making quality, we introduce a reinforcement learning agent based on Q-learning [11] using function approximation.

We implemented two independent reinforcement learning agents: the reinforcement learning head Each agent observes the state vector s and selects an action a. Q-values are updated using temporal difference learning.

Rewards are computed from duel outcomes and intermediate state transitions (including life point differences).

Offline training is performed using recorded duel trajectories.

2.6 Neural Fictitious Self-Play Integration

This study adopts a simplified Neural Pseudo-Self Play (NFSP) framework.

NFSP integrates two learning components: approximating the optimal response for reinforcement learning (RL) and approximating the average strategy for supervised learning (SL) [5].

This simplified design allows practical experimentation while maintaining the core concept of NFSP.

2.7 Data Collection and Processing

Training data is collected by running matches using the baseline WindBot agent. These logs are then post-processed to generate an SL dataset, containing state−action pairs and legal action information for supervised learning, and an RL dataset, consisting of state transitions and duel outcomes for reinforcement learning.

2.8 Offline Reinforcement Learning

Reinforcement learning is performed using offline Q-learning.

The normalized RL datasets are processed sequentially, and each transition is used to update the Q-function parameters using TD (0) updates, a standard temporal-difference learning method in reinforcement learning [12].

Only Head actions are trained in this stage, and the resulting parameters are saved as a binary model file. The trained model is then loaded by the NFSP agent during gameplay.

This offline training approach allows reinforcement learning without requiring expensive online self-play during training.

2.9 Training Procedure and Evaluation Setup

The overall training and evaluation process consists of the following steps:

- Normalize logs to create SL and RL datasets.
- Train supervised learning models from SL datasets
- Train RL Head model using offline Q-learning
- Integrate SL and RL models into the NFSP framework
- Run matches with NFSP enabled and vary η
- Measure win rate changes under different η values.

Figure 2 shows the data collection and training pipeline.

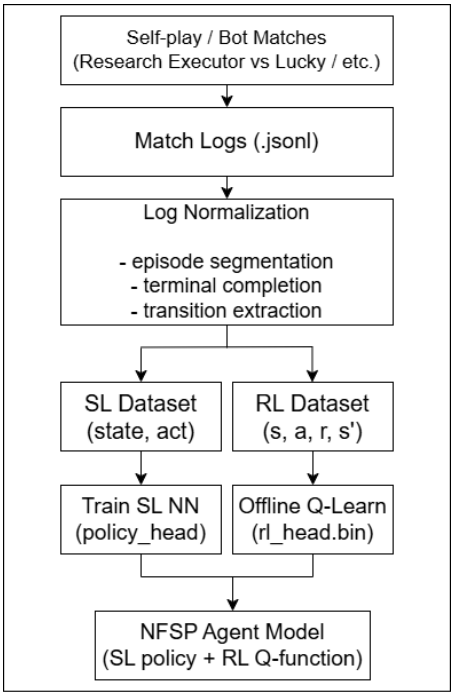

Figure 2: Data collection and training pipeline for supervised learning and offline reinforcement learning.

3 Experiments

3.1 Experimental Setup

To evaluate the effectiveness of the proposed NFSP-based agent, experiments were conducted under three different mixing parameters:

$\eta=0$, $\eta=0.1$, $\eta= 0.3$

For each setting, 50 matches were played against the same baseline WindBot agent using a fixed deck configuration.

All matches were recorded in JSONL format, including:

- State vectors (32-dimensional)
- Selected actions
- Legal action sets
- Policy source information (RL or SL)
- Duel outcomes

The supervised policies (Head and Activate) were trained from 100 baseline matches (NFSP disabled).

3.2 Action Distribution Analysis

To analyze the impact of NFSP on agent behavior, we measured the frequency of each selected head action type under each η setting, the usage ratio of RL-based and SL-based decisions, and the decision frequency at the activation level.

This analysis allows evaluation of behavioral changes even statistical win-rate significance is difficult to establish due to limited sample size.

4 Results

4.1 Win Rate Trends

Due to the limited number of matches and high variance in duel outcomes, win rate differences across η values were not statistically significant. Therefore, interpretation based solely on the win rate was inconclusive.

4.2 Changes in Action Distribution

Table 1 illustrates action type distribution under different η values. It shows that the proportion of activation actions increases as η increases:

$\eta= 0 \rightarrow 28.9\%$, $\eta= 0.1 \rightarrow 28.5\%$, $\eta= 0.3 \rightarrow 36.6\%$

This indicates that reinforcement learning encouraged more activate card usage compared to the purely supervised policy.

Meanwhile, as shown in Table 2, random head selections decreased as η increased. This suggests that the trained policy is actively influencing decision-making.

Table 3 shows that the frequency of RL usage similarly increased with η for determining activation levels. This confirms that reinforcement learning influences decision-making processes at both the phase level and card level.

On the other hand, the frequency of phase-ending actions remained relatively stable across settings. This suggests that some strategic actions are still dominated by baseline tendencies.

4.3 Interpretation

Although win rate improvements were not statistically conclusive, the consistent increase in Activate actions and RL policy usage demonstrates that the proposed NFSP framework successfully modifies agent behavior.

These findings indicate that even limited training data, reinforcement learning integration can influence strategic tendencies in a complex trading card game environment.

Table 1: Action Type Distribution under Different η Values

Action Type	$\eta= 0$	$\eta= 0.1$	$\eta= 0.3$
Activate	28.90%	28.54%	36.63%
GoToEndPhase	26.36%	27.65%	26.40%
GoToBattlePhase	11.56%	10.84%	9.41%
SpellSet	10.75%	12.17%	10.23
Summon	6.59%	5.75%	6.60%
GoToMainPhase2	6.47%	6.42%	4.13%
Repos	5.32%	4.65%	3.63%
MonsterSet	4.05%	3.98%	2.97%

Table 2: Head Policy Source Distributions

Policy	$\eta= 0$	$\eta= 0.1$	$\eta= 0.3$
Head_rendom	64.21%	52.20%	49.35%
SL_head	23.67%	24.73%	27.61%
RL_head	12.12%	23.08%	23.04%

Table 3: Activate Policy Source Distributions

Policy	$\eta= 0$	$\eta= 0.1$	$\eta= 0.3$
SL_act_masked_max	73.08%	68.29%	65.79%
RL_act	12.82%	24.39%	26.32%
SL_act	14.10%	7.32%	7.89%

5 Discussion

5.1 Effect of simplified NFSP Integration

The experimental results indicate that integrating NFSP into the Yu-Gi-Oh! AI environment successfully influenced the agent's decision-making behavior.

Although the win rate differences across η values were not statistically significant, behavioral analysis revealed meaningful changes. In particular, the increase in Activate actions and RL-based selections suggests that reinforcement learning contributed to more proactive gameplay decisions.

This demonstrates that the proposed framework can modify strategic tendencies even when trained on a limited amount of gameplay data.

5.2 Influence of the Mixing Parameter η

The parameter η controls the probability of selecting the reinforcement learning policy instead of the supervised policy.

As η increased from 0 to 0.3, the proportion of RL-based decisions increased correspondingly, and the agent showed a higher tendency to use card effects.

However, when η was set close to 1.0, the agent frequently failed to perform meaningful actions and often skipped phases without activating cards. This behavior suggests that the RL policy alone had not yet learned sufficiently stable strategies, and that the supervised policy plays an important role in stabilizing decision-making.

This observation is consistent with the theoretical design of NFSP, where the supervised component approximates the average policy and prevents unstable exploration from dominating behavior.

6 Future Work

Future research directions include:

- Large-scale automated self-play to increase training data volume.
- Full implementation of NFSP with neural network function approximation.
- Comparison with alternative methods such as MCTS or deep reinforcement learning.
- Extension to more complex decks with advanced card interactions.
- Improvement of state representation to capture richer strategic information.

These extensions may lead to stronger performance and deeper insights into AI decision-making in trading car games.

7 Conclusion

This research applied an NFSP-based learning framework to a complex imperfect-information trading card game.

Due to limited training matches, statistically significant win-rate improvements were not observed.

However, we have identified certain consistent behavioral changes, such as increased Activate action frequency and increased usage of RL-based decisions.

These results demonstrate that the proposed framework successfully influences agent strategy patterns.

This work provides an initial empirical step toward applying NFSP to complex trading card game environments.

References

[1] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," Nature, vol. 529, no. 7587, pp. 484-489, Jan. 2016.

[2] D. Silver et al., "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," arXiv:1712.01815, 2017.

[3] M. Moravcik et al., "DeepStack: Expert-level artificial intelligence in heads-up no-limit poker," Science, vol. 356, no. 6337, pp. 508-513, March. 2017.

[4] N. Brown and T. Sandholm, "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals," Science, vol. 359, no. 6374, pp. 418-424, Dec. 2017.

[5] J. Heinrich and D. Silver, "Deep Reinforcement Learning from Self-Play in Imperfect-Information Games," arXiv:1603.01121, 2016.

[6] P. Cowling, C. Ward, E. Powley, "Ensemble Determinization in Monte Carlo Tree Search for the Imperfect Information Card Game Magic: The Gathering," IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, no. 4, pp. 241-257, Dec. 2012

[7] Konami Digital Entertainment, "Yu-Gi-Oh! Official Website." [Online]. Available: https://www.yugioh-card.com/ [Accessed: Feb. 2026]

[8] Konami Digital Entertainment, "Yu-Gi-Oh! Master Duel." [Online]. Available: https://www.konami.com/yugioh/masterduel/ [Accessed: Feb. 2026]

[9] Project Ignis: EDOPro [Online]. Available: https://github.com/edo9300/edopro[Accessed: Feb 2026]
[10] WindBot Ignite [Online]. Available https://github.com/ProjectIgnis/windbot/ [Accessed: Feb. 2026]
[13]
[11] C.J.C.H. Watkins and P. Dayan, Q-learning, Machine Learning, vol. 8, pp. 279-292, 1992.
[12] R.S. Sutton and A.G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge: The MIT Press, 2014.