修　　士　　論　　文

Efficient Performance Estimation for Mahjong Players

麻雀プレイヤーの効率的な実力推定

指導教員　　鶴岡 慶雅 教授

東京大学大学院情報理工学系研究科
電子情報学専攻

氏 名　　48-236414 大神卓也

提 出 日　　2025 年 1 月 21 日

Abstract

In games with stochastic outcomes, evaluating agent performance from limited data is challenging. Using Monte Carlo sampling results does not provide a reliable indicator due to the significant variance. The difficulty of evaluating agents is particularly prominent in mahjong, an incomplete information game with a huge state space. Traditionally, evaluating the performance of a mahjong player or AI requires a large number of games to obtain reliable statistics. For example, in the case of Suphx, a mahjong AI that outperformed humans, its performance was assessed by playing 5,760 games against human players in online mahjong, which took as long as four months. This highlights the inefficiency of conventional evaluation methods. In this study, we propose MJ-DLVAT, a Deep Learning Value Assessment Technique for Mahjong, which provides an unbiased estimate of average ranking with reduced variance. MJ-DLVAT introduces three techniques to manage the extensive game tree and board information in mahjong: splitting the game into subgames, dealing with the variance caused by drawn tiles, dealt tiles and hidden-dora, and introducing neural networks. We created a dataset using online mahjong records and trained a neural network-based value function from scratch. We evaluated MJ-DLVAT on the online mahjong records and confirmed that the average estimated rankings are unbiased estimators of average ranking. Moreover, the variance of the estimated ranking is 45.5% smaller than that of the average ranking. As a result, the number of games required to correctly evaluate a player's ability is reduced by 45.5%.

Contents

List of Figures

List of Tables

List of Algorithms

Chap.1 Introduction

1.1 Background

Artificial Intelligence(AI) agents that outperform humans have been developed for perfect information games [2, 3], as well as imperfect information games [4, 5, 6, 7, 1]. Evaluating agents in imperfect information games requires large numbers of matches because of stochastically variable outcomes. For example, Suphx [1], which outperformed humans in mahjong, played $5,760$ games against humans in online mahjong to evaluate its performance, which took as long as four months. Several studies have been conducted in poker to reduce the number of games required to evaluate human-AI match results [8, 9, 10, 11].

On the other hand, it is also important to evaluate the performance of humans against each other. In recent years, Esports has paid attention to player performance evaluation, as in soccer or other sports [12, 13, 14]. Player performance evaluation has benefits for both spectators and team managers. For spectators, analyzing and evaluating a player's abilities can increase engagement. For team managers, evaluation results can be used for player recruitment, team organization, and team tactics.

We focused on Japanese mahjong from two perspectives. First, mahjong is more complex than many other extensive-form games. If we can evaluate a player's performance in mahjong, where the average size of the information set and the history length is much larger than in poker, it could be extended to other extensive-form games. Second, mahjong has a strong social impact. M-League[1], a professional mahjong league in Japan, is watched by millions of spectators and sponsored by major corporations. Strategic thinking and team tactics are important in M-League as in Esports, so evaluating a mahjong player's ability in a small number of matches is essential.

1.2 Overview

In this study, we propose MJ-DLVAT, a method for estimating a player's performance, to replace the Monte Carlo sampling of match results in Japanese mahjong. Fig. 1.1 shows the concept of the estimation method proposed in this thesis. It calculates how lucky a player has been by analyzing the history of the match. It estimates the player's skill by subtracting luck from the obtained

[1] https://m-league.jp/

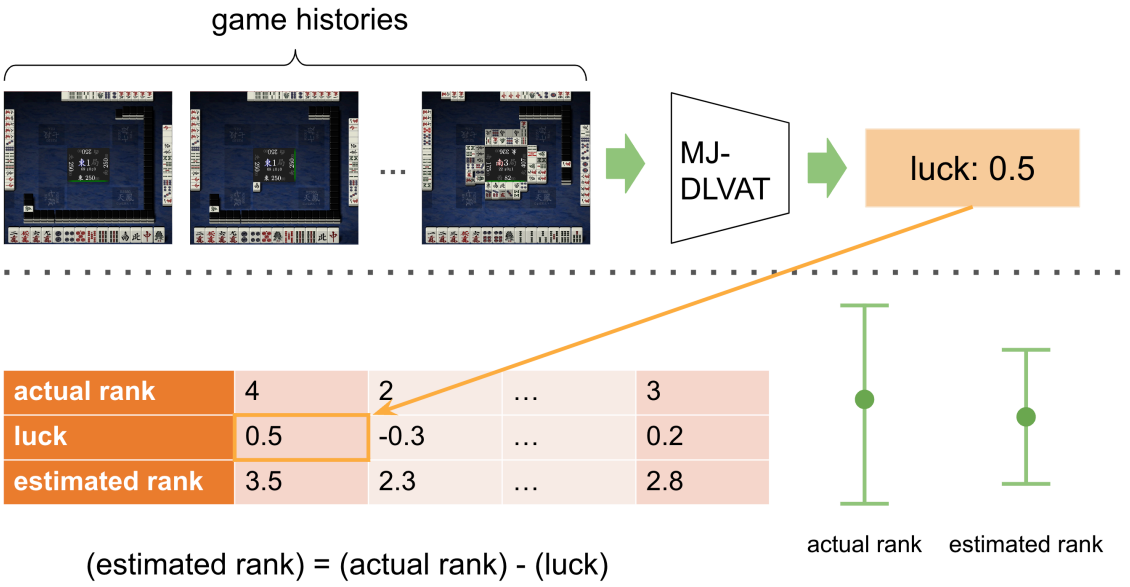


Figure 1.1: Concept of the player performance evaluation method proposed in this thesis.

ranking. This estimator has a smaller variance than the obtained ranking and is an unbiased estimator of the obtained ranking, so we can accurately assess a player's ability in fewer matches.

To achieve this, we introduce the following three methods as extensions to MIVAT [9]. First, to deal with the long action history of mahjong, we split the game into subgames, which are called rounds. We train a value function using the predicted utility for each subgame. Second, to achieve an excellent variance reduction rate, we address the variance caused by three types of random events: (1) a tile randomly drawn from the wall in each turn, (2) randomly dealt tiles at the beginning of rounds, and (3) hidden-dora which is randomly determined. We apply variance reduction techniques to each event and confirm that it is possible to reduce the variance due to all of these events. Third, to cope with the complexity of state space, we introduce a deep neural network to approximate the value function and train it from scratch by gradient descent.

We evaluated this method on the records of the online mahjong site Tenhou[2]. We show that this method reduces the variance by 45.5% while satisfying the unbiased estimation of Monte Carlo sampling. In other words, we have verified that MJ-DLVAT can reduce the number of games required to evaluate a player's performance by 45.5%.

In summary, we make the following contributions:

- We proposed a method for computing agent performance in mahjong that replaces the conventional Monte Carlo sampling of match results. Since the estimates are unbiased and variance-reduced, we have significantly reduced matches to accurately evaluate the player's

[2] `https://tenhou.net/`

performance.

- Our method learns a neural network-based value function only from the match records. We have empirically verified that this generic method, which can be applied to many other extensive-form games, can reduce variance.

1.3 Structure

This thesis is structured as follows:

- **Chapter 2: Background**
  This chapter provides an overview of Japanese Mahjong, its rules, and the challenges associated with evaluating player performance in such an incomplete information game. The mathematical formulation of extensive-form games is also introduced.

- **Chapter 3: Related Work**
  We review prior work on AI methods for imperfect information games, including variance reduction techniques, player performance evaluation methods in Mahjong, and advancements in deep learning approaches for extensive-form games.

- **Chapter 4: Proposed Method**
  This chapter introduces MJ-DLVAT, a deep learning-based value assessment technique for Mahjong. It describes the key components: splitting the game into rounds, variance reduction techniques for drawn tiles, dealt tiles, and hidden-dora, and the use of neural networks for approximating value functions.

- **Chapter 5: Experimental Setup**
  The experimental setup, including the models, dataset and evaluation metrics, is detailed. This chapter also presents the results of preliminary experiments.

- **Chapter 6: Results**
  This chapter analyzes the implications of the experimental results, including comparisons with baseline methods such as Monte Carlo sampling and error rate-based evaluations.

- **Chapter 7: Conclusion and Future Work**
  The final chapter summarizes the contributions of this thesis, discusses the impact of MJ-DLVAT on player performance evaluation in Mahjong, and outlines directions for future research, including potential applications to other extensive-form games.

Chap.2 Background

2.1 Deep Learning

Deep Learning is a machine learning framework that optimizes the parameters of a Deep Neural Network (DNN) using given data. By leveraging the high representational power of Deep Neural Networks, it enables the learning of complex relationships between inputs and outputs that traditional machine learning methods could not capture. The advancement of Deep Learning has allowed AI to achieve unprecedented capabilities across various fields. For example, AlphaGo [2] demonstrated capabilities surpassing professional players in the complex game of Go, drawing significant attention. Pluribus [15] achieved superhuman performance in multiplayer poker, showcasing the potential of AI in games with incomplete information. Stable Diffusion [16] has significantly improved traditional generative models in image generation, becoming well-known for its ability to produce high-resolution and realistic images. Furthermore, GPT-3 [17] revolutionized the field of natural language processing by enabling "few-shot learning", where diverse tasks can be performed with minimal prior information, transforming the landscape of language understanding. In the domain of speech recognition, Whisper [18] dramatically improved recognition accuracy across various environments by utilizing large-scale weakly supervised learning.

2.1.1 Deep Neural Network

A Neural Network is a model that describes and computes the relationship between inputs and outputs. Its foundational concept is an attempt to artificially replicate the computational principles of the brain, which enable intelligent behavior. In particular, a model with multiple layers, known as a Deep Neural Network, can approximate complex relationships between inputs and outputs with high accuracy by performing a large number of simple calculations.

The characteristics of a Deep Neural Network can be summarized as follows:

- High Approximation Capability: It can accurately approximate the relationship between inputs and outputs from the provided data.
- Automatic Feature Extraction: Unlike traditional machine learning models, a Deep Neural Network automatically extracts features in its intermediate layers. This eliminates the need for manual feature design, as required in conventional models.

As a result, Deep Neural Networks have become a powerful method widely utilized across various fields to solve complex problems. A Feedforward Network is one of the fundamental structures of Neural Networks, where neurons in each layer transmit information unidirectionally to the neurons in the next layer. As shown in Figure 2.1, this structure processes information sequentially, flowing from the input layer to the output layer. As a simple yet versatile example of a Deep Neural Network, the Feedforward Network finds applications in a wide range of domains.

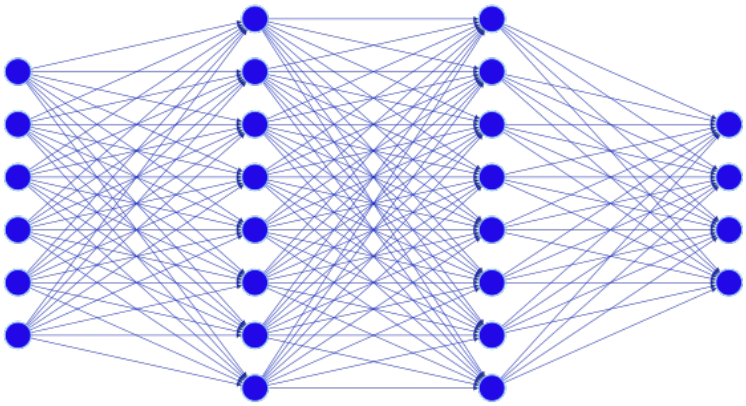


Figure 2.1: The Structure of a Feedforward Network

The basic flow of a Feedforward Network starts from the input layer, passes through several hidden layers, and finally reaches the output layer. In this process, the neurons in each layer apply a non-linear transformation to the weighted inputs and transmit the results to the next layer.

2.1.2 Loss Function

To optimize the parameters of a neural network, a loss function is introduced as a metric to evaluate how "good" the current parameters are. Given training data in the form of input and desired output pairs $((x_1, y_1), \cdots, (x_N, y_N))$, the loss function is defined as an error function $L(y, \hat{y})$ that calculates the difference between the neural network's output $\hat{y}$ and the desired output y.

The optimization of a neural network is performed by finding parameters that minimize the average error over the training data. This objective is expressed mathematically as follows:

$$w^* = \arg\min_w \sum_{i=1}^{N} L(y_i, \hat{y}_i). \tag{2.1}$$

For the error function, mean squared error or Huber loss [19] is commonly used in regression problems, while cross-entropy error is typically employed in classification problems.

2.1.3 Learning Algorithm

In the previous section, we stated that optimizing the weights of a neural network is achieved by minimizing the error function. This section describes the algorithms used to minimize the error function. Gradient descent is a widely used method for minimizing functions. Denoting the objective function as L and the parameters as $\boldsymbol{W}$, the gradient of the objective function with respect to the parameters is calculated and used for updates:

$$\boldsymbol{W} = \boldsymbol{W} - \epsilon \frac{dL}{d\boldsymbol{W}} \tag{2.2}$$

Mini-batch gradient descent is a widely used method for optimizing machine learning algorithms. It is an approach that lies between calculations using the entire dataset and those using a single randomly selected data point, offering a good balance between computational efficiency and convergence properties.

The procedure for mini-batch gradient descent is as follows: The dataset is divided into multiple small subsets (mini-batches), which are selected randomly. The size of each mini-batch is a user-defined hyperparameter that affects both computational efficiency and the stability of learning. Using the selected mini-batch B, the value of the objective function (e.g., the loss function) L is calculated. The gradient ∇L of the objective function is then computed based on all data points in the mini-batch. Using this gradient, the model parameters θ are updated as follows:

$$\boldsymbol{W} = \boldsymbol{W} - \epsilon \frac{dL}{d\boldsymbol{W}}_B \tag{2.3}$$

Here, ϵ is the learning rate, and $\frac{dL}{d\boldsymbol{W}}_B$ is the gradient of the loss with respect to the mini-batch B.

Mini-batch gradient descent offers the following advantages:

1. Improved computational efficiency: Compared to gradient descent using the entire dataset, mini-batch gradient descent significantly reduces computational costs such as memory usage. This makes model training on large datasets practical.

2. Avoidance of local minima due to randomness: By using mini-batches, randomness is introduced into the gradient calculation. This characteristic reduces the risk of optimization getting stuck in local minima, allowing for broader exploration of the search space.

Momentum SGD is a method used to stabilize the variance of gradients and improve the stability of convergence. By considering the history of past weight updates, it smooths the parameter updates. The change in parameters at time t is defined as in Equation (2.4).

$$v_t = w_t - w_{t-1} \tag{2.4}$$

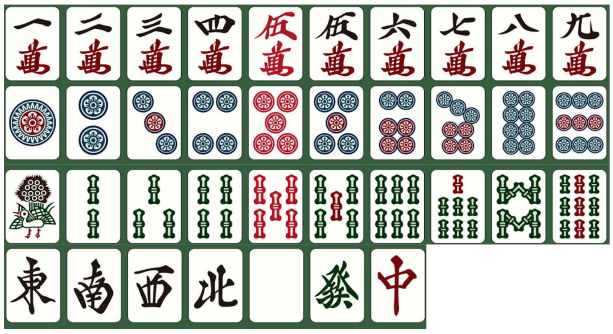


Figure 2.2: This image displays the 37 unique tiles used in Japanese Mahjong, organized by suit. The Character, Circle, and Bamboo suits are numbered from 1 to 9. The bottom row includes the Wind tiles (East, South, West, North) and the Dragon tiles (Green, Red, White).

At time $t+1$, the change in parameters in Momentum SGD is expressed as in Equation (2.5), where μ determines how much importance is placed on past update information.

$$v_{t+1} = \mu v_t - \epsilon \frac{\partial L}{\partial W} \tag{2.5}$$

This approach prevents sharp oscillations and accelerates convergence. In particular, for loss functions with surfaces resembling valleys, SGD without momentum may cause parameter updates to oscillate. Using momentum, however, suppresses oscillations and efficiently converges to the optimal solution.

2.2 Rules of Japanese mahjong

Mahjong is a traditional tile-based game played by four players. The primary objective of the game is to achieve the highest score among the four players. Each player begins the game with the same score (25,000 points), and gameplay progresses through a series of rounds. Below, we provide a detailed explanation of the game mechanics, rounds, and scoring rules.

2.2.1 Game Setup

A standard Mahjong tile set consists of 136 tiles, comprising four identical tiles for each of the 34 unique tile types. These tiles are divided into three suits (Characters, Dots, and Bamboos), along with Honor tiles (Winds and Dragons). Each player's hand is private and cannot be seen by opponents. Tiles in the wall (the remaining stack of tiles) are hidden from all players, while tiles discarded by players (known as the "river") and melded tiles are visible to everyone.

2.2.2 Gameplay Overview

At the start of each round, each player is dealt 13 tiles. During a player's turn, the following sequence of actions occurs:

- **Draw or Take a Tile**: A player can draw a random tile from the wall or, under specific conditions, claim a tile discarded by an opponent to form a meld (a combination of tiles such as a sequence or a triplet).
- **Declare Win or Discard**: After managing their hand, the player can declare a win (if their hand meets the winning criteria) or discard one tile to the river.

The round ends when one of the following conditions is met:

- A player completes a winning hand (declaring "ron" or "tsumo").
- There are no more tiles left in the wall (a draw).

2.2.3 Winning Hand and Scoring

A winning hand consists of specific combinations of tiles that meet the game's rules, such as sequences, triplets, or a pair. Players score points by completing winning hands, with points transferred from opponents. The score for a winning hand can range from 1,000 points to 48,000 points, depending on the complexity of the hand and the circumstances of the win.

2.2.4 Special Rules and Terminology

- **Riichi**: When a player is one tile away from completing a winning hand, they may declare riichi by placing a stick (1000 points) on the table. This declaration locks their hand, restricting them from making further changes except for drawing a winning tile.
- **Dora and Hidden Dora**: Bonus tiles called dora increase the score of a winning hand. If a player wins after declaring riichi, hidden dora (tiles beneath the dora indicators) are revealed, which can further boost the score.
- **Melds**: A player may form melds (e.g., sequences, triplets) by claiming discarded tiles from opponents. Melds are visible to all players and can affect strategic decisions.

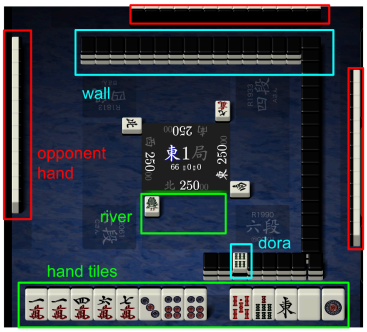


Figure 2.3: One scene of online mahjong game Tenhou with annotated regions.

2.3 Online mahjong site Tenhou

Tenhou is one of the most famous online mahjong game platforms in Japan. Each player is assigned a dan, which indicates his strength. Players accumulate points by achieving positive results in games, and upon reaching specific thresholds, they can be promoted to a higher dan. Conversely, poor performance can result in a loss of points and potentially a demotion. It has four fields with different levels of players: General, Advanced, Special, and Phoenix. Only the players with 7-11 dan are allowed to play at the Phoenix field.

Existing research on Mahjong AI [1, 20, 21] has been conducted on Tenhou because the game records of the Phoenix field are publicly available[3]. This study also uses the Phoenix field records for almost all experiments because most records other than the Phoenix field are not publicly available.

2.4 Extensive-form Game

An extensive-form game [22] is a framework for decision-making in multi-person environments with incomplete information, where not all previously taken actions are fully observable to each player, and often involves inherent randomness represented by a "chance player."

An extensive-form game is defined by

$$\left(\mathcal{N} \cup \{c\}, \mathcal{H}, \mathcal{Z}, \mathcal{A}, \{R^i\}_{i \in \mathcal{N}}, \mathcal{P}, \pi^c, \mathcal{S}\right).$$

$\mathcal{N}$ denotes the set of N players $\{1, \ldots, N\}$. c is called the chance player and is the player introduced to represent the randomness of the environment. For example, the initial dealing of cards in poker can be represented as the chance player's action determined by the policy π^c.

The history $h \in \mathcal{H}$ represents the sequence of actions of all players (including the chance player) since the beginning of the game, and $\mathcal{H}$ is the set of all possible histories. For instance, consider

[3] `https://tenhou.net/sc/raw/`

a simplified poker-like setting: at the start, Player 1 calls (action: call), then the chance player deals the initial hands (action: deal), and subsequently Player 2 places a bet (action: bet). The resulting history could be expressed as $h = (\text{p1 call}, \text{c deal}, \text{p2 bet})$.

For histories $h_1, h_2 \in \mathcal{H}$, if h_1 is a prefix of the action series of h_2 or $h_1 = h_2$, we denote $h_1 \sqsubseteq h_2$. The history obtained by the player taking action a at history h is denoted as $h \cdot a$. The terminal history set $\mathcal{Z} \subseteq \mathcal{H}$ is the set of histories where the game ends. For any non-terminal history $h \in \mathcal{H} \setminus \mathcal{Z}$, $\mathcal{A}(h)$ represents the set of possible actions.

Each player $i \in \mathcal{N}$ receives utility at the terminal history according to the utility function $R^i : \mathcal{Z} \to \mathbb{R}$. The function $\mathcal{P} : \mathcal{H} \to \mathcal{N} \cup \{c\}$ indicates which player acts at a given history.

The set $\mathcal{S}$ represents a partition of the set of histories $\mathcal{H}$ into information sets. Each element $I \in \mathcal{S}$ (called an information set) is a subset of $\mathcal{H}$ such that the player acting at these histories cannot distinguish which specific history in I has actually occurred. In other words, if $h, h' \in I$ for some $I \in \mathcal{S}$, then from that player's perspective h and h' are indistinguishable. This concept is introduced to model the incomplete information aspect of the game.

2.5 AI in Imperfect Information Games

Advancements in Artificial Intelligence (AI) have significantly improved strategic decision-making in imperfect information games. Imperfect information games refer to scenarios where players lack complete information about their opponents' actions, such as hidden cards. Techniques like Fictitious Play, Regret Minimization, and Reinforcement Learning have played crucial roles in advancing AI technologies for such games. This section provides an overview of key studies leveraging these methodologies.

The evaluation of strategies in imperfect information games is primarily conducted using Nash equilibrium. A Nash equilibrium solution represents a state where no player has an incentive to deviate from their strategy, as no unilateral deviation increases the player's payoff. This concept is expressed mathematically as follows:

$$u_i\big(s_1^*, \ldots, s_i^*, \ldots, s_n^*\big) \;\geq\; u_i\big(s_1^*, \ldots, s_i, \ldots, s_n^*\big) \quad (\forall s_i \in S_i) \tag{2.6}$$

Particularly in two-player zero-sum games, adopting a Nash equilibrium strategy ensures a guaranteed payoff regardless of the opponent's strategy, making it a robust approach for strategic decision-making.

2.5.1 Regret Minimization Framework

Regret Minimization is a fundamental approach to strategy learning in imperfect information games. It calculates the regret (2.7) for a specific action a and updates the strategy based on the

ratio of the average regret (2.8). Regret Matching [23] updates the strategy using the "regret", which measures how much more payoff could have been achieved by choosing a specific action (2.9). It has been shown that this update reduces regret at a rate of $\mathcal{O}(\frac{1}{\sqrt{T}})$. Additionally, the average strategy obtained by averaging all the strategies during the process of Regret Matching converges to a correlated equilibrium (a generalization of Nash equilibrium).

$$r^t(a) = \left(u^t(a) - \sum_{a \in A} p^t(a) u^t(a) \right) \tag{2.7}$$

$$R^t(a) = \frac{1}{T} \sum_{t=1}^{T} r^t(a) \tag{2.8}$$

$$p^t(a) = \begin{cases} \frac{R^{t-1,+}(a)}{\sum_{a' \in A} R^{t-1,+}(a')} & \text{if } \sum_{a' \in A} R^{t-1,+}(a') > 0 \\ \frac{1}{|A|} & \text{otherwise} \end{cases} \tag{2.9}$$

$$\max_{a \in A} R^t(a) \leq \frac{|u| \sqrt{|A|}}{\sqrt{T}} \tag{2.10}$$

$$|u| = \max_{t \in \{1 \ldots T\}} \max_{a, a' \in A} \left(u^t(a) - u^t\left(a'\right) \right) \tag{2.11}$$

Subsequently, Counterfactual Regret Minimization (CFR) [24] was proposed to apply regret minimization to strategy optimization in imperfect information games represented as extensive-form games, such as poker. CFR reduces the problem of minimizing overall regret, expressed in (2.13), to minimizing regret within each information set, as shown in (2.14). Here, $u_i(\sigma, I)$ represents counterfactual utility, which is the expected utility at history h weighted by the reach probability $\pi^{\sigma}_{-i}(h)$ for all histories included in the information set I (2.12). In two-player games, the average strategy was shown to converge to a Nash equilibrium. CFR requires traversing all histories to update regrets, so its effectiveness was initially validated using abstraction in limit heads-up Texas hold'em, where the number of histories is relatively small. Subsequently, several enhancements, such as CFR+ [25] and Linear CFR [26], were proposed to accelerate convergence to Nash equilibrium.

$$u_i(\sigma, I) = \frac{\sum_{h \in I, h' \in Z} \pi^{\sigma}_{-i}(h) \pi^{\sigma}\left(h, h'\right) u_i\left(h'\right)}{\pi^{\sigma}_{-i}(I)} \tag{2.12}$$

$$R_i^T = \frac{1}{T} \max_{\sigma_i^* \in \Sigma_i} \sum_{t=1}^{T} \left(u_i\left(\sigma_i^*, \sigma_{-i}^t\right) - u_i\left(\sigma^t\right) \right) \tag{2.13}$$

$$R^T_{i,\text{imm}}(I) = \frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I) \left(u_i\left(\sigma^t\big|_{I \to a}, I\right) - u_i\left(\sigma^t, I\right) \right) \tag{2.14}$$

Monte Carlo CFR [27] introduced sampling methods to update regrets by traversing only a subset of nodes in each iteration, thereby improving computational efficiency and demonstrating

its applicability to large-scale games. Subsequently, robust sampling method [28] was introduced to improve computational efficiency. Baseline function [29], which reduces the variance of Monte Carlo methods, was also proposed.

DeepCFR [30] replaced table-based algorithms with representations using deep neural networks, reducing reliance on domain-specific knowledge and increasing the applicability of CFR to large-scale games. By learning neural networks that represent the average strategy and cumulative regrets in CFR, it significantly enhanced the expressive capability of strategies. Single Deep CFR [31] improved approximation accuracy by not using neural networks to learn the average strategy.

2.5.2 Fictitious Play Framework

Fictitious Play [32] is an algorithm for finding Nash equilibrium. It estimates the opponent's strategy as the average of their past strategies, as expressed in (2.15). Based on this estimated strategy, each player updates their own strategy to the best response (2.16). By iteratively updating the strategies, the average strategies (2.17) converge to a Nash equilibrium.

$$\hat{\sigma}_{-i}^{t} = \frac{1}{t}\sum_{s=1}^{t}\sigma_{-i}^{s} \tag{2.15}$$

$$\sigma_{i}^{t+1} \in \arg\max_{\sigma_i} \mathbb{E}_{\sigma_i,\hat{\sigma}_{-i}^{t}}\left[u_i\left(\sigma_i,\sigma_{-i}\right)\right] \tag{2.16}$$

$$\bar{\sigma}_{i}^{T} = \frac{1}{T}\sum_{t=1}^{T}\sigma_{i}^{t} \tag{2.17}$$

On the other hand, Fictitious Self-Play (FSP) [33] is a machine learning framework that implements generalized weakened fictitious play [34] for extensive-form games using sample-based methods. In FSP, each player learns both an average strategy and the best response while repeatedly playing the game as follows:

The basic updates of FSP are as follows:

- Players generate game episodes via self-play and store their experiences in memory.
- Using reinforcement learning, players learn an approximate best response based on these experiences.
- Players model their average strategy based on past experiences.

The FSP updates correspond to the generalized weakened fictitious play update and are modeled as follows:

$$\Pi_{i}^{t+1} = (1-\alpha_{t+1})\Pi_{i}^{t} + \alpha_{t+1}\beta_{i}^{t+1}, \tag{2.18}$$

where Π_i^t represents player i's average strategy, β_i^{t+1} is player i's next best response, and α_{t+1} is the learning rate.

To learn the best response, reinforcement learning is applied to solve a Markov Decision Process (MDP) constructed using the opponents' average strategies Π_{-i}^t, leading to the approximate best response:

$$\beta_i^{t+1} \in \arg\max_{\beta_i} \mathbb{E}_{\beta_i, \Pi_{-i}^t}[u_i(\beta_i, \Pi_{-i})], \tag{2.19}$$

where $u_i(\beta_i, \Pi_{-i})$ denotes player i's payoff.

The average strategy is updated using supervised learning as follows:

$$\hat{\pi}_i^{t+1}(u) = \frac{\sum_{k=1}^t \rho_k(u)\beta_i^k(u)}{\sum_{k=1}^t \rho_k(u)}, \tag{2.20}$$

where $\rho_k(u)$ represents player i's weight at information state u.

FSP is well-suited for large-scale problems in extensive-form games, leveraging experience-based sampling to significantly improve computational efficiency. Experiments have demonstrated that FSP converges to approximate Nash equilibria in complex games such as imperfect-information poker [33].

Neural Fictitious Self-Play (NFSP) [35] extends the FSP framework using deep learning, enabling strategy learning in large-scale games. NFSP integrates two networks: one for reinforcement learning and another for supervised learning, to simultaneously best responses and learn players' average strategies. A key advantage of NFSP is its ability to handle games with vast state spaces using neural networks. Experiments have shown that NFSP converges to approximate Nash equilibrium in large imperfect-information games like Texas Hold'em poker [35]. Moreover, methods incorporating Monte Carlo techniques [36] and combining regret minimization with NFSP [37] have been proposed to enhance performance in large-scale and imperfect-information games.

2.5.3 Model-free Reinforcement Learning

Model-free reinforcement learning algorithms have achieved significant success in many complex imperfect-information games [5, 6, 38]. AlphaStar [5] and OpenAI Five [6] have mastered multiplayer strategic games using reinforcement learning. DeepNash [38] successfully solved Stratego, a large-scale imperfect-information game that had not been previously solved. Diplodocus [39] developed reinforcement learning policies capable of cooperating with humans. Furthermore, DouZero [40] mastered DouDizhu, a Chinese card game that emphasizes both cooperation and competition, through self-play deep reinforcement learning.

2.6 Monte Carlo Estimation of Performance

Mahjong is a four-player game, and the players' rankings are determined according to their scores at the end of the game, from 1 to 4. When a player's population average ranking is μ, the player's performance can be estimated using Monte Carlo methods from the results of the m matches played by that player. Let $\bar{x}$ be the mean of the results of m matches, and s^2 be the sample unbiased variance. If m is sufficiently large, the 95% confidence interval for the population mean μ is then obtained as

$$\bar{x} - 1.96\frac{s}{\sqrt{m}} \leq \mu \leq \bar{x} + 1.96\frac{s}{\sqrt{m}}. \tag{2.21}$$

2.7 MIVAT

MIVAT [9] is a method for variance reduction in extensive-form games. MIVAT uses data from the players' games to find an unbiased estimator of the players' utilities, such that the variance is smaller than the utilities. The estimated utility is $\hat{u}$, where the player's lucky utility l (called luck) is subtracted from the player's utility u, as follows:

$$\hat{u} = u - l. \tag{2.22}$$

If the expected value of luck is 0, the estimate $\hat{u}$ is an unbiased estimator of the utility u:

$$E[\hat{u}] = E[u - l] = E[u]. \tag{2.23}$$

The following equation shows the luck L_j of player $j \in \mathcal{N}$ in a game with a terminal history z using the value function $V_j : \mathcal{H} \to \mathbb{R}$. The luck is the sum of the differences of the value before and after the chance player's actions in the history:

$$L_j(z) = \sum_{\substack{h \text{ s.t.} \\ h \cdot a \sqsubseteq z, P(h) = c}} \Big(V_j(h \cdot a) - V_j(h) \Big) \tag{2.24}$$

$$= \sum_{\substack{h \text{ s.t.} \\ h \cdot a \sqsubseteq z, \\ P(h) = c}} \left(V_j(h \cdot a) - \sum_{a' \in \mathcal{A}(h)} \pi^c(a'|h)\, V_j(h \cdot a') \right). \tag{2.25}$$

Here, by expanding $V_j(h)$ as in Eq. (2.25), the expected value of $L_j(z)$ is 0 for any value function $V_j : \mathcal{H} \to \mathbb{R}$. The value function parameters are determined to minimize the variance of $\hat{u}$. In the MIVAT framework [9], the value function is modeled as a linear function of input features, allowing for the analytical determination of optimal parameters. In N-person zero-sum games ($N > 2$) such as poker and mahjong, the condition $\sum_{j=1}^{N} V_j(h) = 0$ must be satisfied.

To evaluate MIVAT, the method's performance was compared in terms of standard deviation of estimated utility.MIVAT was compared against DIVAT [41], which is an estimator with hand-crafted features, and traditional approaches such as Monte Carlo estimators. The evaluation involved data from both synthetic interactions between bots and real interactions with human players. For example, in two-player limit poker, MIVAT was shown to significantly reduce variance compared to DIVAT, especially when tailored to specific player populations. Additionally, MIVAT demonstrated its applicability to previously unexplored domains, such as two-player no-limit poker and six-player limit poker, by successfully reducing variance in these settings where no other estimators had been effective.

Chap.3 Related Work

3.1 Research on Mahjong

The game of Mahjong, a large-scale imperfect information game, has been used as a research platform for developing AI that solves such games [42, 20, 1, 43, 44, 45, 46]. Early research simplified the game's complexity or divided it into modular components to handle Mahjong's intricate information [42]. After demonstrating that convolutional neural networks (CNNs) can accurately imitate player strategies by processing features of Mahjong boards represented as images [20], research on Mahjong AI primarily employed deep learning models. Furthermore, as deep reinforcement learning produced AI that surpassed human performance in various games, Mahjong also saw the development of AI capable of performing at a human-equivalent level using deep reinforcement learning [1].

Research on Mahjong AI has since continued, including studies on AI for three-player Mahjong [43], efforts to improve training efficiency by reducing reward variance [44], the proposal of reinforcement learning algorithms theoretically guaranteed to converge to Nash equilibria [45], and research incorporating human domain knowledge into reinforcement learning to enhance training efficiency [46]. These advancements indicate a shift in Mahjong AI research towards reinforcement learning.

Other studies have focused on estimating hidden information about opponents [47, 48, 49, 50]. Several efforts have specifically targeted the prediction of waiting tiles—tiles opponents need to complete a winning hand [50, 47]. Research in this area includes studies using neural networks to predict waiting tiles required for an opponent ' s win [50], and studies enhancing prediction accuracy by refining input data [47]. Moreover, some attempts aim to predict opponents ' entire hands [49]. Additionally, research has been conducted on estimating the number of remaining tiles in the wall that are important for progressing one's own hand [48].

3.2 Performance estimation in mahjong

Kume et al. [51] used the error rate to estimate the performance of mahjong players.Error rate is initially proposed to evaluate the performance of backgammon players [52]. The error rate is calculated using an AI agent's state-action value function $Q(s, a)$ to the set $\mathcal{D}$ of the player's

state-action pairs (s, a), as follows:

$$e(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(s,a)\in\mathcal{D}} \left(\max_{a'\in A(s)} Q(s, a') - Q(s, a) \right). \tag{3.1}$$

Error rate indicates how much utility the player is losing per move compared to the AI. The smaller $Q(s, a)$ for action a taken by the player in state s, the larger the error rate.

They found that when estimating a player's skill level in the online mahjong site Tenhou, the accuracy of estimates based on the average rankings from 500 matches is similar to that of estimates based on around 16 matches. The state value function estimates are derived from the AI agent "Ako_Atatarashi", which has achieved the rank of Tenhou 7-dan, signifying its high proficiency in mahjong. Although they verified that error rates can be used to distinguish between novice, intermediate, and advanced users, this study aims to find estimates that can also detect differences in performance among advanced users. In addition, their study differs from ours in that the estimates are not unbiased estimators of the variable one wishes to find.

3.3 Estimating Players's skill in Poker

Duplicate Poker is a format of poker designed to minimize the element of luck and emphasize player skill. It is used in the Annual Computer Poker Competition (ACPC) [53]. The same deck order is used, and all players play under identical conditions with the same hands and community cards. The performance is evaluated by comparing the chips earned by each player based on their strategy and decision-making.

Bowling et al. [54] reduced variance by the following two methods.

- Handling Game-Terminating Actions via Importance Sampling
- Aggregation of States with Identical Public Information

The first method focuses on the scenarios where a player performs an action that effectively terminates the game earlier than it otherwise would. Specifically, the game history is extended by considering hypothetical scenarios where the player might have chosen to terminate the game earlier. For example, in poker, this could involve a player folding at an earlier stage. For each of these hypothetical scenarios, a weight which reflects the relative likelihood of the observed and hypothetical actions under the player's strategy. These weighted scenarios contribute to the final estimate, enabling the inclusion of possible early game-ending actions without requiring direct observation of those actions.

The second method reduces the number of possible states by aggregating states that are identical except for the private information of the players. Private information (e.g., the specific cards dealt to a player) is often irrelevant to the strategy evaluation from the perspective of other players

or the evaluator. For example, all game histories where the betting sequence is identical but the private cards of a player differ are grouped together. The evaluation then averages over these histories, weighting each one appropriately based on the likelihood of the private information.

Bowling et al. [54] reduced variance by the following two methods.

- Handling Game-Terminating Actions via Importance Sampling
- Aggregation of States with Identical Public Information

The first method focuses on the scenarios where a player performs an action that effectively terminates the game earlier than it otherwise would. Specifically, the game history is extended by considering hypothetical scenarios where the player might have chosen to terminate the game earlier. For example, in poker, this could involve a player folding at an earlier stage. For each of these hypothetical scenarios, a weight which reflects the relative likelihood of the observed and hypothetical actions under the player's strategy. These weighted scenarios contribute to the final estimate, enabling the inclusion of possible early game-ending actions without requiring direct observation of those actions.

The second method reduces the number of possible states by aggregating states that are identical except for the private information of the players. Private information (e.g., the specific cards dealt to a player) is often irrelevant to the strategy evaluation from the perspective of other players or the evaluator. For example, all game histories where the betting sequence is identical but the private cards of a player differ are grouped together. The evaluation then averages over these histories, weighting each one appropriately based on the likelihood of the private information.

Aggregation of States with Identical Public Information is shown to be particularly effective in reducing variance when evaluated in 2-player no-limit Texas Hold'em. Additionally, these methods are compatible with DIVAT [41], further extending their practical applicability.

The Action-Informed Value Assessment Tool (AIVAT) [10] is a method that reduces variance caused by the randomness of policies by treating the player's policy as equivalent to the chance player in MIVAT [9], assuming the player's policy is known. It also uses MIVAT [9] and the method proposed by Bowling et al. [54]. They learned the value function by solving the game using a variant of the Monte Carlo Counterfactual Regret Minimization (MCCFR) [27].

AIVAT has been widely used to evaluate performance in no-limit Texas hold'em in various studies [4, 55, 56] focusing on agents playing incomplete information games. In DeepStack [4], AIVAT reduced the number of games required for evaluation by a factor of 44 during the evaluation of results against humans. Additionally, it serves as a performance evaluation tool for OpenHoldem [57], a benchmark framework for no-limit Texas hold'em, in conjunction with duplicate poker.

Our study emphasizes analyzing the results of human-versus-human games, where the strategies are unknown. The variance caused by stochastic policies is not considered in this study. We also address the problem of mahjong, which has a larger average size of information sets and a longer game history than poker.

3.4 Prediction of outcomes in games

Existing studies focus on predicting game outcomes in Real Time Strategy (RTS) games and Multiplayer online battle arena (MOBA) games [58, 59, 60, 61, 62] such as StarCraft, DOTA2, and League of Legends. In these studies, players' performance and compatibility are estimated from the state of the game before and during the game to predict the game outcome. The StarCraft study [58] built models for each combination of opposing races and predicted the outcome by machine learning. The DOTA2 [59] study achieved high prediction accuracy by introducing deep learning.

These studies differ from ours because they aim to minimize the difference between predictions and actual game results and do not guarantee the property of unbiased estimators. However, we also use the technique for predicting game outcomes in this study to train the model (Section 4.3).

3.5 Evaluating player performance using rating systems

Numerous studies have been conducted on rating systems that quantify players' skills as numerical values for comparison. Notable rating methods include the Elo rating system [63], the Glicko system [64], which addresses variations in the reliability of evaluations, and TrueSkill [65], which enables ratings for team-based and multiplayer games involving more than three players. Additionally, TrueSkill2 [66], an improvement over TrueSkill, was proposed to accommodate new players more rapidly and incorporate statistical information about players.

Subsequent research has introduced rating systems with better incentive compatibility compared to existing methods [67] and proposed alternative approaches, such as representing skill differences between players using graph structures rather than single numerical values [68]. Efforts have also been made to enhance the interpretability and performance of skill estimation by using statistical information about players' results as features [69], as well as to estimate and utilize dynamic ratings for effective matchmaking [70], emphasizing practical applications.

This study aims to address the uncertainty inherent in incomplete information games, rather than merely focusing on the relative skill levels of the players.

Chap.4 Proposed Method

4.1 Motivation

Evaluating the performance of players and AI agents in imperfect information games remains a significant challenge. Japanese Mahjong, with its vast state space and intricate decision-making processes, exemplifies these challenges. Traditional methods, such as Monte Carlo sampling, require an impractically large number of games to achieve reliable evaluations, making them inefficient for both research and practical applications.

Several key aspects of Mahjong underscore the need for novel evaluation methods:

1. The Complexity of the Game State

   Mahjong's state space is far larger and more complex than that of many other extensive-form games, such as poker. This complexity arises from the hidden information in players' hands, the dynamic changes on the board, and the variety of potential decisions at each step. Existing methods like MIVAT [9] which uses linear value function will struggle to capture this level of complexity, leading to evaluations that are either overly simplified.

2. The Long Decision Histories

   In Mahjong, players make a series of decisions throughout each round, resulting in lengthy and intricate game histories. Unlike poker, where decision points are fewer and more discrete, Mahjong's continuous decision-making process makes it impractical to evaluate performance by exhaustively analyzing the full game history. A more efficient approach is required to break down these histories into manageable units without losing essential information.

3. Multiple Stochastic Elements

   Mahjong outcomes are heavily influenced by stochastic elements, such as the randomness of drawn tiles, initial hands, and hidden dora indicators. These elements introduce significant variance into performance evaluations.

These challenges necessitate a shift toward a more sophisticated evaluation framework. Specifically, there is a need for methods that can:

1. Capture the intricate structure of Mahjong's state space,
2. Efficiently manage the long and complex decision histories,

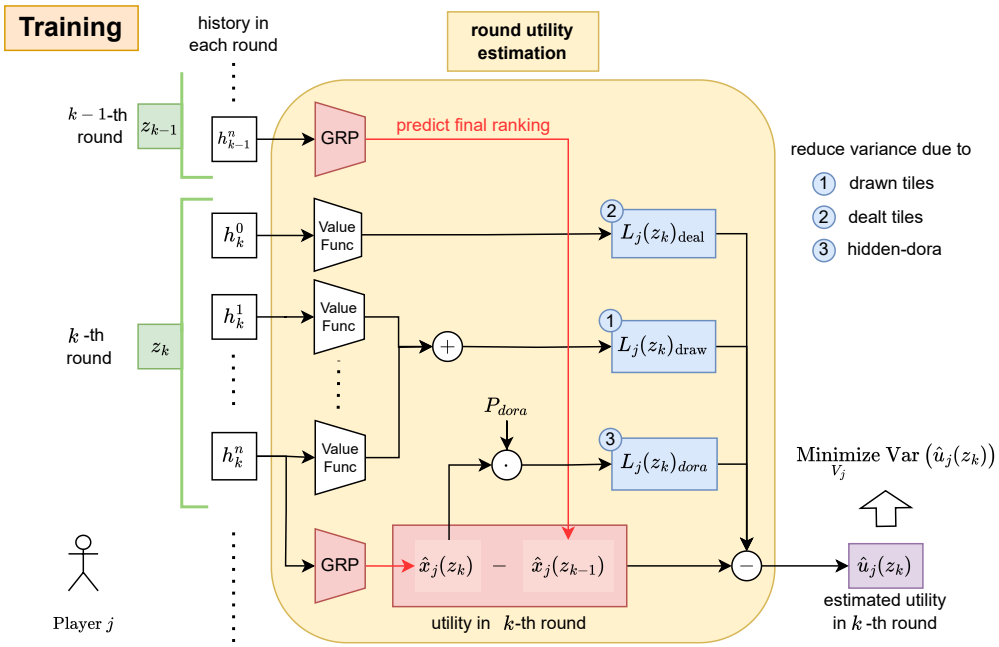


Figure 4.1: The method of training the value function for each round. We estimate the utility in the k-th round from the history to reduce the variance caused by drawn tiles, dealt tiles, and hidden-dora. Drawn tiles are calculated using the history from the 1st to the n-th record of the k-th round, while dealt tiles are calculated using the 0th record of the history. Hidden-dora is calculated based on the results of GRP and P_{dora}. We then minimize the sampled variance of these estimates.

3. Handle multiple sources of luck.

Our study aims to address these needs by introducing a framework that leverages deep learning and statistical techniques. This framework not only improves the fairness and accuracy of evaluations but also significantly reduces the number of games required to achieve reliable assessments. By focusing on the unique challenges posed by Mahjong, this research provides insights and methodologies that are broadly applicable to other imperfect information games.

4.2 Overview

Figs. 4.1 and 4.2 shows the overall picture of MJ-DLVAT. $x_j(z) \in \{1, 2, 3, 4\}$ is the actual ranking obtained by player $j \in \mathcal{N}$ in a game z, $y_j(z)$ is the unbiased and variance-reduced estimated ranking of $x_j(z)$, z_k is the split terminal history z by round. By introducing the following three methods

(in Sections 4.3 to 4.5) as extensions to MIVAT, $y_j(z)$ can be expressed as

$$y_j(z) = 2.5 + \sum_{k=1}^{K} \hat{u}_j(z_k), \tag{4.1}$$

where $\hat{u}_j(z_k)$ is the estimated utility at round k as

$$\hat{u}_j(z_k) = u_j(z_k) - L_j(z_k) \tag{4.2}$$

with the k-th round's utility $u_j(z_k)$ and luck $L_j(z_k)$.

We divide the luck $L_j(z_k)$ into values attributed to drawn tiles, dealt tiles, and hidden-dora:

$$L_j(z_k) = L_j(z_k)_{\text{draw}} + L_j(z_k)_{\text{deal}} + L_j(z_k)_{\text{dora}}. \tag{4.3}$$

The value function V_j is obtained by training

$$\underset{V_j}{\text{Minimize}}\ \text{Var}\left(\hat{u}_j(z_k)\right) \tag{4.4}$$

as the objective function.

4.3 Splitting the game into subgames

As described in Section 2.2, a game of mahjong consists of a series of subgames called rounds. The game tree of mahjong is huge, so it is difficult to learn the value function if we use the final ranking as the MIVAT's utility in Eq. (2.22). Therefore, we consider a round as the unit of the episode. We can assume that the utility of a round should be calculated based on the change in points at that round (round balance). However, it is not obvious how the round balance affects the final ranking.

Therefore, in this study, we estimate how good the results were for each round using Global Reward Prediction (GRP). In GRP, the final ranking is predicted based on the state at the end of the previous round and the end of the current round, and we set the difference between the two as the round's utility.

We predict the final ranking $\hat{x}_j(z_k) \in [1, 4]$ from the k-th round's final state using a pre-trained deep neural network, where $\hat{x}_j(z_0) = 2.5$ and $\hat{x}_j(z_K) = x_j(z)$. The utility at round k can then be defined by

$$u_j(z_k) = \hat{x}_j(z_k) - \hat{x}_j(z_{k-1}). \tag{4.5}$$

If $L_j(z_k)$ in Eq. (4.2) meets $E[L_j(z_k)] = 0$ for all $k \in \{1, \dots K\}$, then from Eqs. (4.1), (4.2) and (4.5), the following relationship between actual ranking $x_j(z)$ and estimated ranking $y_j(z)$ is

derived.

$$\begin{aligned} E\left[y_j(z)\right] &= E\left[2.5 + \sum_{k=1}^{K}\left(\hat{x}_j(z_k) - \hat{x}_j(z_{k-1}) - L_j(z_k)\right)\right] \\ &= E\left[x_j(z) - \sum_{k=1}^{K} L_j(z_k)\right] \\ &= E\left[x_j(z)\right]. \end{aligned} \tag{4.6}$$

Thus, the estimated ranks and actual ranks agree in expectation regardless of the GRP model used. Therefore, we can split the game into rounds while maintaining the properties of unbiased estimation.

4.4 Chance player's actions in mahjong

There are three types of actions by the chance player in mahjong: a player randomly draws a tile from the wall on each turn, random tiles are dealt at the beginning of the game, and the hidden-dora is randomly determined. We propose the following methods to reduce the variance for each type.

4.4.1 Reducing variance due to drawn tiles

We estimate the luck of which tile to draw by comparing the value function of the state before and after drawing a tile from the wall.

We consider each player's drawing scene as the action of chance player c_{draw}. When $P(h) = c_{\text{draw}}$, $\mathcal{A}(h)$ is the set of tile types and its size is 34. The probability of which tile to draw $\pi^{c_{\text{draw}}}(a'|h)$ can be calculated from all tiles remaining in the wall. Then, we can estimate the luck due to drawn tiles at k-th round by

$$\begin{aligned} &L_j\left(z_k\right)_{\text{draw}} \\ &= \sum_{\substack{h \ s.t. \\ h\cdot a \sqsubseteq z_k, \\ P(h)=c_{\text{draw}}}} \left(V_j\left(h\cdot a\right) - \sum_{a'\in\mathcal{A}(h)} \pi^{c_{\text{draw}}}\left(a'|h\right) V_j\left(h\cdot a'\right)\right). \end{aligned} \tag{4.7}$$

4.4.2 Reducing variance due to dealt tiles

In mahjong, each player is dealt 13 tiles at the start of the round. Considering this as the actions of the chance players, the set of actions is huge (its size is about 10^{11}). This makes it difficult

to calculate the sum over all actions when calculating each player's luck using Eq. (2.25). We, therefore, use a sampling technique to approximate the value of the state before the tiles are dealt.

Let h_k^0 be the initial history of k-th round, $\mathcal{A}^n(h) \subseteq \mathcal{A}(h)$ be the set of n sampled from the set of all actions in history h. Then, the luck of player j due to the tiles dealt is

$$L_j\left(z_k\right)_{\text{deal}} = V_j\left(h_k^0 \cdot a\right) - \frac{1}{n} \sum_{a' \in \mathcal{A}^n(h_k^0)} V_j\left(h_k^0 \cdot a'\right). \tag{4.8}$$

4.4.3 Reducing variance due to hidden-dora

In mahjong, when a player wins after declaring riichi, the hidden-dora tiles behind the dora tiles are revealed, and the player's score increases according to how many hidden-dora are in the player's hand. The hidden-dora is randomly determined at the start of the game and is hidden until the player wins. Therefore, the disclosure of hidden-dora can be interpreted as an action of the chance player. Consider the case where a winning after riichi occurs in the k-th round. Let $\hat{x}_j(z_k^i)$ be the predicted ranking resulting from the score transition if i hidden-dora tiles were applied, whereas $\hat{x}_j(z_k)$ is the one from the score transition that actually occurred. We can define the luck due to hidden-dora by

$$L_j(z_k)_{\text{dora}} = \hat{x}_j(z_k) - \sum_i \hat{x}_j(z_k^i) P_{\text{dora}}\left(i|z_k\right), \tag{4.9}$$

where $P_{\text{dora}}(i|z_k)$ denotes the probability that i hidden-dora tiles are applied in z_k, and $\sum_i P_{\text{dora}}(i|z_k) = 1$.

4.5 Value Function Approximation

4.5.1 modeling value functions using neural networks

In MIVAT [9], a linear function was used to approximate the value function, and the value function was calculated analytically by minimizing the sample variance. On the other hand, Suphx [1] and other Mahjong AIs [20, 71, 72] use Convolutional Neural Networks (CNNs) to imitate advanced players. In this study, we also use CNNs instead of linear functions to approximate the value function to handle the complex information of a mahjong board effectively.

Because input features are different between drawn tiles and dealt tiles, separate value functions are created for drawn tiles and dealt tiles. For drawn tiles, the value function outputs corresponding values for each of the 34 types of tiles. While this approach may reduce approximation accuracy due to the granularity of the output, it significantly reduces the computational complexity to a feasible level.

Additionally, as discussed in the MIVAT section (Section 2.7), for games involving N players ($N > 2$), it is necessary to impose the constraint that the sum of the luck values across all players

equals zero. To satisfy this constraint, the value function outputs four values corresponding to the four players' values. The average of these four values is then subtracted from each player's output value to ensure that the total sum is zero, maintaining consistency with the multi-player game dynamics:

$$\hat{V}_i = V_i - \frac{1}{N}\sum_{j=1}^{N} V_j, \quad \text{for } i = 1, 2, \ldots, N. \tag{4.10}$$

4.5.2 optimizing value functions

We use a stochastic gradient descent algorithm to optimize the value function. Algorithm Algorithms 1 and 2 respectively show how the value function is trained in the case of drawn tiles and dealt tiles. We calculate the unbiased variance of $\hat{u}$ for each mini-batch and use it as the loss function in our method. We set the batch size as large as possible to stabilize the computation of the loss function. The same learning process was carried out for the case of dealt tiles according to Eq. (4.8), independently of the case of drawn tiles.

Algorithm 1 Learning Value Function in MJ-DLVAT for drawn tiles

Require: subject to the following conditions

- $V_{\boldsymbol{\theta}}$: $\mathbb{R}^{(788\times34\times1)} \to \mathbb{R}^{(34\times4)}$ (Outputs estimated values corresponding to "34 chance actions $\times$ 4 players" for each state h.
  Constrained so that the sum of 4 elements for each chance action equals 0)
- N: number of iterations per epoch
- B: batch size
- z_{batch}: batch of terminal histories (size B)
- $\boldsymbol{u}_{\text{batch}} = (u_1, u_2, u_3, u_4)$: batch of utility for each of 4 players, shape $(B \times 4)$

procedure LearnValueFunctionSingleEpoch; $(V_{\boldsymbol{\theta}}, N, B)$:
  for $t = 1$ **to** N:
    $(z_{\text{batch}}, \boldsymbol{u}_{\text{batch}}) \leftarrow \text{sample}(B)$
    {Shape: $z_{\text{batch}} \in (B)$, $\boldsymbol{u}_{\text{batch}} \in (B \times 4)$ }
    $\boldsymbol{Luck} \leftarrow \mathbf{0}_{B\times4}$
    for each $h \sqsubseteq z_{\text{batch}}$:
      $$\boldsymbol{Luck} \leftarrow \boldsymbol{Luck} + \Big(V_{\boldsymbol{\theta}}(h)[\,c\,] - \sum_{c'=1}^{34} \pi^c\big(c' \mid h\big)\, V_{\boldsymbol{\theta}}(h)[\,c'\,]\Big)$$
    end for {(for each h in batch)}
    $\hat{\boldsymbol{u}} \leftarrow \boldsymbol{u}_{\text{batch}} - \boldsymbol{Luck}$
    $loss \leftarrow \text{sample\_variance}(\hat{\boldsymbol{u}})$
    $\boldsymbol{\theta} \leftarrow \text{AdamUpdate}\,(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}}\, loss)$
  end for {(for $t = 1$ to N)}
end procedure

Algorithm 2 Learning Value Function in MJ-DLVAT for dealt tiles

Require: subject to the following conditions

- $V_{\boldsymbol{\theta}}$: $\mathbb{R}^{(665\times34\times1)} \to \mathbb{R}^4$ (Outputs estimated values corresponding to 4 players" for each state h.
  Constrained so that the sum of 4 elements for each chance action equals 0)
- N: number of iterations per epoch
- M number of simulated samples for dealt tiles
- B: batch size
- h_{batch}: batch of tile-dealing histories which includes simulated samples (each history has a single dealing action)
- $\boldsymbol{u}_{\text{batch}} = (u_1, u_2, u_3, u_4)$: batch of utility for each of 4 players, shape $(B \times 4)$

procedure LearnValueFunctionSingleEpoch; $(V_{\boldsymbol{\theta}}, N, B)$:
  for $t = 1$ **to** N:
    $(h_{\text{batch}}, \boldsymbol{u}_{\text{batch}}) \leftarrow \text{sample}(B)$
    {Shape: $h_{\text{batch}} \in (B \times (M+1),)$, $\boldsymbol{u}_{\text{batch}} \in (B \times 4)$ }
    $\boldsymbol{Luck} \leftarrow \mathbf{0}_{B\times4}$
    for each $h \in h_{\text{batch}}$
      $$\boldsymbol{Luck} \leftarrow \boldsymbol{Luck} + \Big(V_{\boldsymbol{\theta}}(h[0]) - \sum_{c'=1}^{M} \pi^c\big(c' \mid h[c']\big)\, V_{\boldsymbol{\theta}}(h[c'])\Big)$$
    end for {(for each h in batch)}
    $\hat{\boldsymbol{u}} \leftarrow \boldsymbol{u}_{\text{batch}} - \boldsymbol{Luck}$
    $loss \leftarrow \text{sample\_variance}(\hat{\boldsymbol{u}})$
    $\boldsymbol{\theta} \leftarrow \text{AdamUpdate}\,(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}}\, loss)$
  end for {(for $t = 1$ to N)}
end procedure

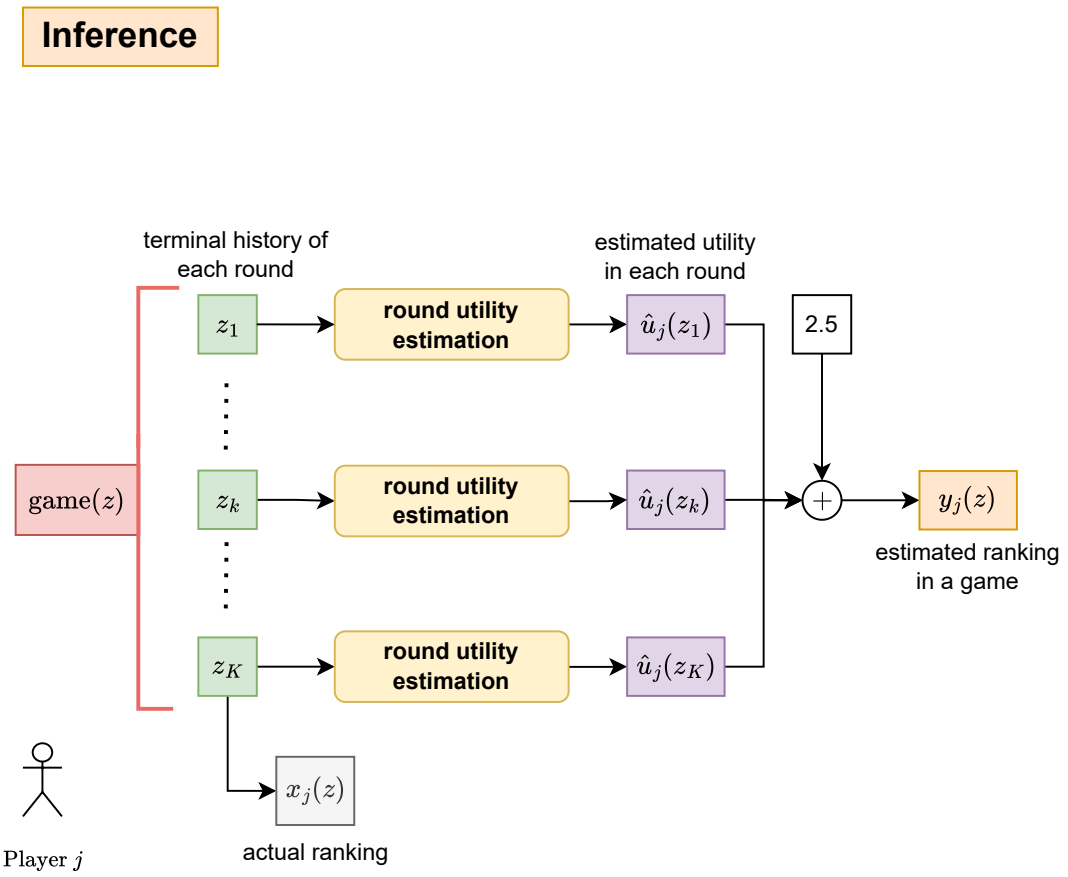


Figure 4.2: The method to infer the variance-reduced estimates for the final ranking. The process splits the game history into rounds, applying "round utility estimation"(Fig. 4.1) to calculate the estimated utility for each round. These utilities are summed to derive the "estimated ranking in a game", offering a clear assessment of the player's overall performance.

Chap.5 Experimental Setup

5.1 Global Reward Prediction Model

We utilize game data spanning from 2011 to 2014 to train and evaluate the Global Reward Prediction (GRP) model. The details of the experimental setup and the evaluation results are described below. The architecture of the model is shown in Fig. 5.1. A neural network consisting of multiple fully connected layers is employed. For training the GRP model, the state at the end of each subgame is used as input, and the model is designed to predict the rankings at the end of the match (1st place as -1.5, 2nd place as -0.5, 3rd place as 0.5, and 4th place as 1.5). The Mean Squared Error (MSE) was adopted as the loss function. The learning process aimed to minimize the mean squared difference between predicted values and actual values. The features were carefully designed to represent the scoring situations in detail. The specific features are summarized in Table 5.2. The training and validation data were prepared by randomly splitting the entire training dataset. These subsets were used for parameter updates of the model and monitoring the learning process, respectively. The model was evaluated using the MSE on the test data. Loss curve of training and validation is shown in Fig. 5.2. During the evaluation process, unseen test data were provided to the trained model, and its prediction accuracy was measured. Specifically, the MSE on the test data reached 0.7705, suggesting that the model can generalize to unseen data.

5.2 Dataset

We use the match records of the Phoenix field, the top-level field of the online mahjong site Tenhou. We utilize data from approximately $910,000$ games spanning from 2015 to 2021. Games that include players who have participated in more than $6,000$ games are used as the player-specific test dataset. This dataset includes 49 players. The remaining records are used for the training and validation sets.

These datasets are divided into rounds and labeled with predicted utilities using Global Reward Prediction as described in Section 4.3. Fig. 5.3 shows how dataset is processed. The game logs which is divided by rounds, are saved in JSON format (BoardJSON) containing the information for each state. Ground truth utility is calculated using Global Reward Prediction. list of BoardJSON

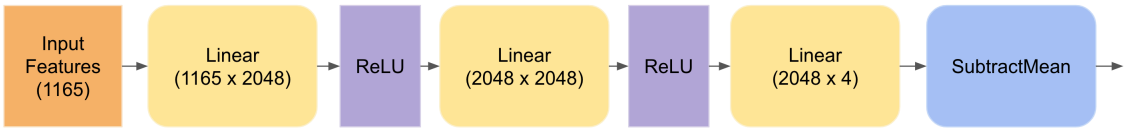


Figure 5.1: We use fully-connected Deep Neural Network with ReLU activation for Global Reward Prediction.

Table 5.1: Training settings of Global Reward Prediction Model

Parameter	Value
Optimization method	Adam [73]
Learning rate	1e-5
Dimension of Linear Layer	2048
Batch size	1024

and Ground truth utility are saved in storage. During training, Features and draw probabilities are calculated, then Luck and loss are calculated using these values.

5.3 Model

For approximating the value function, we implement a deep neural network with residual connections [74] (Fig. 5.4). The hyperparameter settings are described in Table 5.3. For input features, we follow the style of Suphx [1]. The input features of the model are listed in Table 5.4 in the case of drawn tiles. In the case of dealt tiles, some unnecessary features are removed. Following Suphx [1]'s style, each feature was represented as multiple 34x1 binary vectors (0/1) corresponding to the 34 types of tiles and arranged along the channel direction. Since the input to the value function is the history in the extensive-form game, incomplete information is also included in the input. In this study, we used the progress of the opponent's hand, the wall, the discarded tiles, and so on as input features to improve the model's ability to interpret the scene. For dealt tiles, n in Eq. (4.8) is set to 1 for training, and 5 for inference in the following experiments.

5.4 Baseline method

To validate the effectiveness of our method, we compare it with the Monte Carlo sampling [75] and error-rate-based method [51]. The Monte Carlo sampling uses the player's actual ranking in the evaluation dataset. The method using error rate requires an AI agent as a reference, but the AI agent in the previous study is not publicly available. Therefore, in this study, we

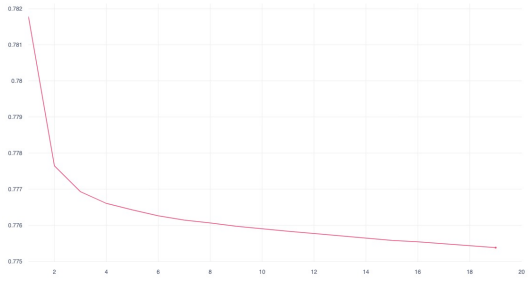
Table 5.2: Input Features of Global Reward Prediction Model

Feature Name	Dimension	Feature Detail
honba	1	Number of honba at the start of the round.
deposit	1	Number of deposit sticks accumulated.
field_wind	3	One-hot encoded representation of the current wind (East, South, West).
points	4	Normalized points of each player (divided by 100,000).
dealer	4	One-hot encoded representation of the current dealer.
tsumo	192	Binary vectors indicating whether rank changes for each player if each of 16 major scoring patterns of self-drawn wins happens.
other_ron	384	Binary vectors indicating whether rank changes for each player if each of 16 major scoring patterns in opponent discard wins happens.
direct	192	Binary vectors indicating whether rank changes for each player if each of 16 major scoring patterns in opponent discard wins by the player.
other_tsumo	384	Binary vectors indicating whether rank changes for each player if each of 16 major scoring patterns in self-drawn wins by other players.
Total	1165	

validate our method using the AI agent GOKU[4], which has been developed through supervised and reinforcement learning and has achieved Tenhou 7-dan.

As a preliminary experiment, we analyzed the error rates of players in normal, advanced, special, and phoenix fields in Tenhou. Data outside of the Phoenix Field is mostly not publicly available. To obtain game records from fields other than the Phoenix Field, we collected data from players who achieved "Tenhoui" (the highest rank in Tenhou) from here[5]. When calculating the error rate, we treat only the player's decision of which tile to discard from their hand and don't consider the meld and riichi decisions. Table 5.5 shows the results, indicating that the higher the player's ability, the lower the error rate. Although the AI agent is different from the original paper, it can be confirmed that the error rate calculation is correct.

[4] https://goku-mahjong-ai.web.app/
[5] https://tenhou.net/ranking.html



(a) Loss curve during training of Global Reward Prediction model.

(b) Loss curve during training of Global Reward Prediction model.

Figure 5.2: Loss curve of Global Reward Prediction model.

Table 5.3: Settings on learning model

Parameter	drawn tiles	dealt tiles
Optimization method	Adam [73]	Adam[73]
Learning rate	0.0001	0.001
Batch size	128	2,000

5.5 Evaluation index

We estimate the utilities using MJ-DLVAT and evaluate whether the variance is reduced compared to the Monte Carlo method and whether unbiased estimation is achieved.

5.5.1 Notation

Let n be the number of players in the test data, $\mathcal{X}_j = \{x_{i,j}\}_{i=1}^{m_j}$ be the set of rankings actually obtained by player j in the m_j matches, the set of estimated rankings be $\mathcal{Y}_j = \{y_{i,j}\}_{i=1}^{m_j}$. The total number of matches in the test data is $M = \sum_{j=1}^{n} m_j$. For $x \in \mathcal{X}_j$, let unbiased variances be $s_{x,j}^2$, sample mean be $\bar{x}_j$, population mean be $\mu_{x,j}$. The same is true of $y \in \mathcal{Y}_j$.

5.5.2 Evaluation index of variance reduction

We evaluate the variance reduction by comparing the unbiased variances. We introduce

$$\sum_{j=1}^{n} \frac{m_j}{M} \frac{s_{y,j}^2}{s_{x,j}^2} \tag{5.1}$$

as an indicator of the degree of variance reduction. It is the average of the variance reduction ratio of the player, weighted by the number of games each player played.

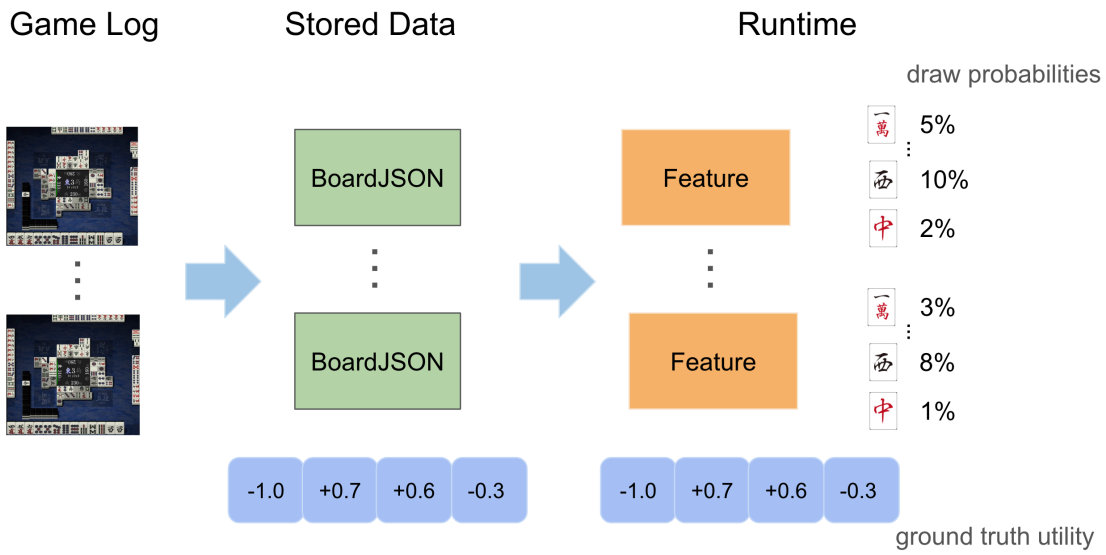


Figure 5.3: How the dataset is processed. The game logs, divided by rounds, are saved in JSON format (BoardJSON) containing the information for each state, along with the ground truth utility calculated using Global Reward Prediction. During training, Features and draw probabilities are calculated, then Luck and loss are calculated using these values.

5.5.3 Evaluation index of unbiased estimation

In evaluating unbiased estimation, we conduct a statistical hypothesis testing that the difference between the population means $\mu_{x,j}, \mu_{y,j}$ is smaller than a specific threshold $\delta > 0$. It is necessary to test for both cases $\bar{x}_j > \bar{y}_j$ and $\bar{x}_j < \bar{y}_j$ for each player j.

In the case $\bar{x}_j > \bar{y}_j$, we conduct a one-sided test at a significance level of 5% with the null hypothesis $H_0 : \mu_{x,j} - \mu_{y,j} = \delta$ and the alternative hypothesis $H_1 : \mu_{x,j} - \mu_{y,j} < \delta$. When the number of matches m_j is sufficiently large, the test statistic defined by

$$z_j = \begin{cases} \dfrac{(\bar{x}_j - \bar{y}_j) - \delta}{\sqrt{\frac{s_{x,j}^2 + s_{y,j}^2}{m_j}}} & (\bar{x}_j > \bar{y}_j) \\ \dfrac{(\bar{y}_j - \bar{x}_j) - \delta}{\sqrt{\frac{s_{x,j}^2 + s_{y,j}^2}{m_j}}} & (\bar{x}_j < \bar{y}_j) \end{cases} \tag{5.2}$$

follows a standard normal distribution. Therefore, when $z_j < -1.64$, the alternative hypothesis is adopted and $\mu_{x,j} - \mu_{y,j}$ is sufficiently small.

In the case $\bar{x}_j < \bar{y}_j$, we test with null hypothesis $H_0 : \mu_{y,j} - \mu_{x,j} = \delta$, and the alternative hypothesis $H_1 : \mu_{y,j} - \mu_{x,j} < \delta$. In the same way, when $z_j < -1.64$, the alternative hypothesis is

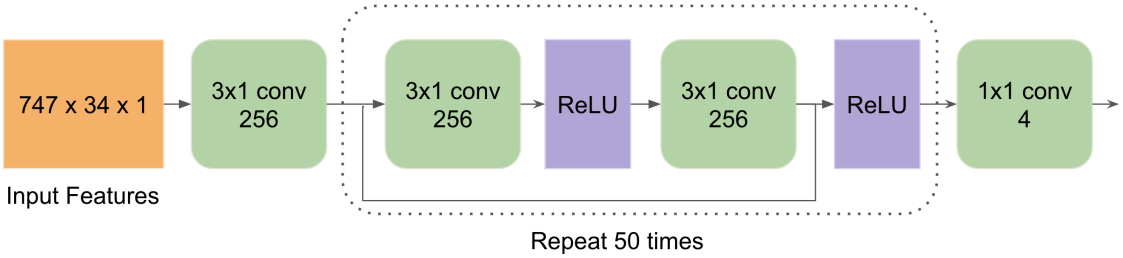


Figure 5.4: We introduce CNNs with residual connections to approximate the value function. This structure is similar to the discard model of Suphx [1].

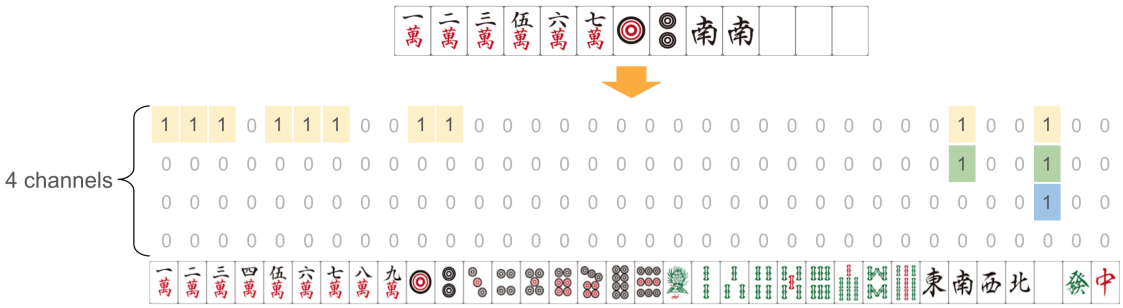


Figure 5.5: Illustration of feature representation of board information.

adopted, and $\mu_{y,j} - \mu_{x,j}$ is sufficiently small.

Table 5.4: Details of features for representing information on the board

Feature Name	Number of Channels	Feature Detail
dealer	4	One-hot encoded representation of the current dealer.
player_wind	4	Wind of player who will draw a tile at the state.
field_wind	3	One-hot encoded representation of the current wind (East, South, West).
riichi	4	Whether each player has declared riichi at the state.
action_player_id	4	Which player will draw a tile at the state.
honba	6	Number of honba at the start of the round.
deposit	6	Number of deposit sticks accumulated.
bonus_tiles	8	information about public bonus tiles.
points	80	descretized representation of points of each player.
river	84	information about discarded tiles.
hand	28	representing each player's hand tiles.
bonus_tiles_in_hand	20	Number of bonus tiles in each player's hand tiles.
shanten	28	Shanten number of each player's hand tiles
acceptable_tiles	104	The number of tiles which each player need to proceed his hand.
yaku	384	Represent tiles which is related to main yaku.
Total	747	

Table 5.5: Error rate for each level in Tenhou

Field name (level)	Error rate
Normal Field (novice)	0.896
Advanced Field (intermediate)	0.706
Special Field (advanced)	0.522
Phoenix Field (expert)	0.410

Chap.6 Results

6.1 Evaluation of Training Process of Value Function

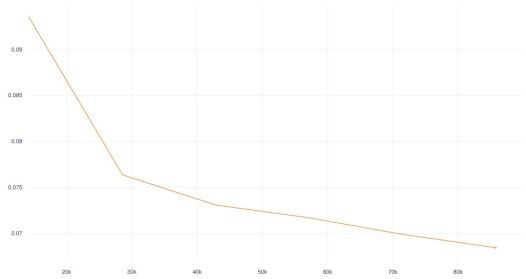
The variance of the actual ranking in the validation dataset was 0.119. After training the model, the variance of the estimate in the validation dataset decreased to 0.069, which is smaller than the variance of the actual ranking. For the case of drawn tiles, the loss values on each epoch are shown in Fig. 6.1, and the training curve is shown in Fig. 6.2. From Fig. 6.1, it can be observed that training is completed before the value function overfits the training data. This is because the amount of available training data is large relative to the computational resources available, resulting in an insufficient number of epochs. Therefore, extending the training time or accelerating the training process is expected to lead to further performance improvements. Additionally, the training curve in Fig. 6.2 is more unstable compared to typical training curves in supervised learning, suggesting that addressing training instability may contribute to performance improvements.

Table 6.1: Variance reduction results using the proposed method

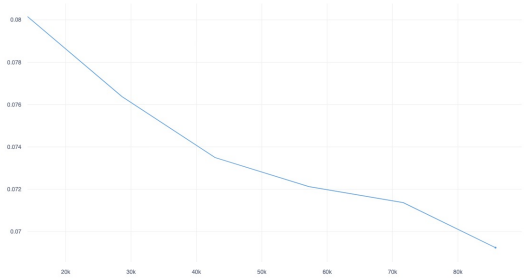
	Reduction rate of variance
Monte Carlo	—
MJ-DLVAT (all)	45.4%
MJ-DLVAT (drawn tiles)	39.9%
MJ-DLVAT (hidden-dora)	2.0%
MJ-DLVAT (dealt tiles)	3.1%

6.2 Evaluation of Variance Reduction using MJ-DLVAT

With the trained model, we estimated the ranking using the proposed method in drawn tiles, dealt tiles, and hidden-dora. Table 6.1 shows the average variance reduction percentage weighted by the number of matches for each player, according to Eq. (5.1). By estimating the ranking with MJ-DLVAT, we reduced the variance by a total of 45.5% compared to the Monte Carlo method. The breakdown shows that the variance is reduced by 39.9% due to drawn tiles, by 3.1% due to dealt tiles, and by 2.0% due to hidden-dora.



(a) Loss curve during training of value function for drawn tiles.



(b) Loss curve during validation of value function for drawn tiles.

Figure 6.1: Epoch Loss of value function for drawn tiles.

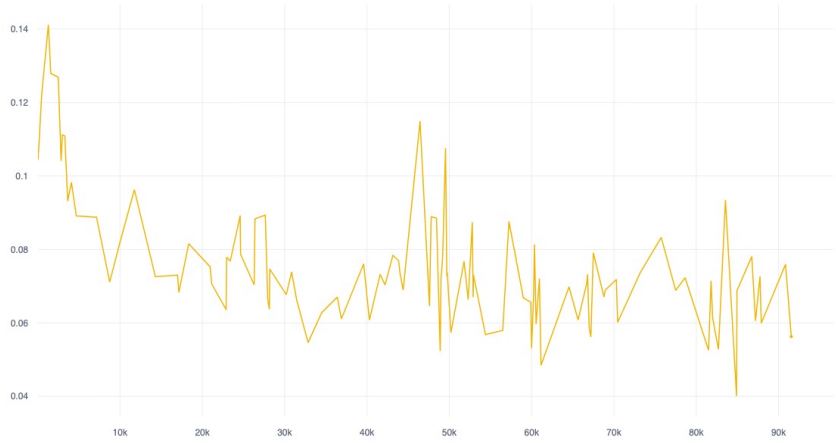


Figure 6.2: Training Curve of value function for drawn tiles. Value of loss for each minibatch during training is plotted.

As for drawn tiles and dealt tiles, the number of tiles taken by each player is 13 for dealt tiles, and 10-20 for drawn tiles in each round, which is not a significant difference. However, the percentage of variance reduced was very different. As for hidden-dora, the probability of riichi in a round is approximately 20%, and the probability of winning in that round is approximately 50%, so there are not many situations where hidden-dora is revealed. Therefore, the variance reduction ratio is considered to be small.

6.3 Evaluation of Unbiased Estimation using MJ-DLVAT

In order to correctly evaluate a player's performance, it is necessary to ensure that the estimated utility is unbiased. Although Eq. (4.6) theoretically shows that the estimated rankings and actual

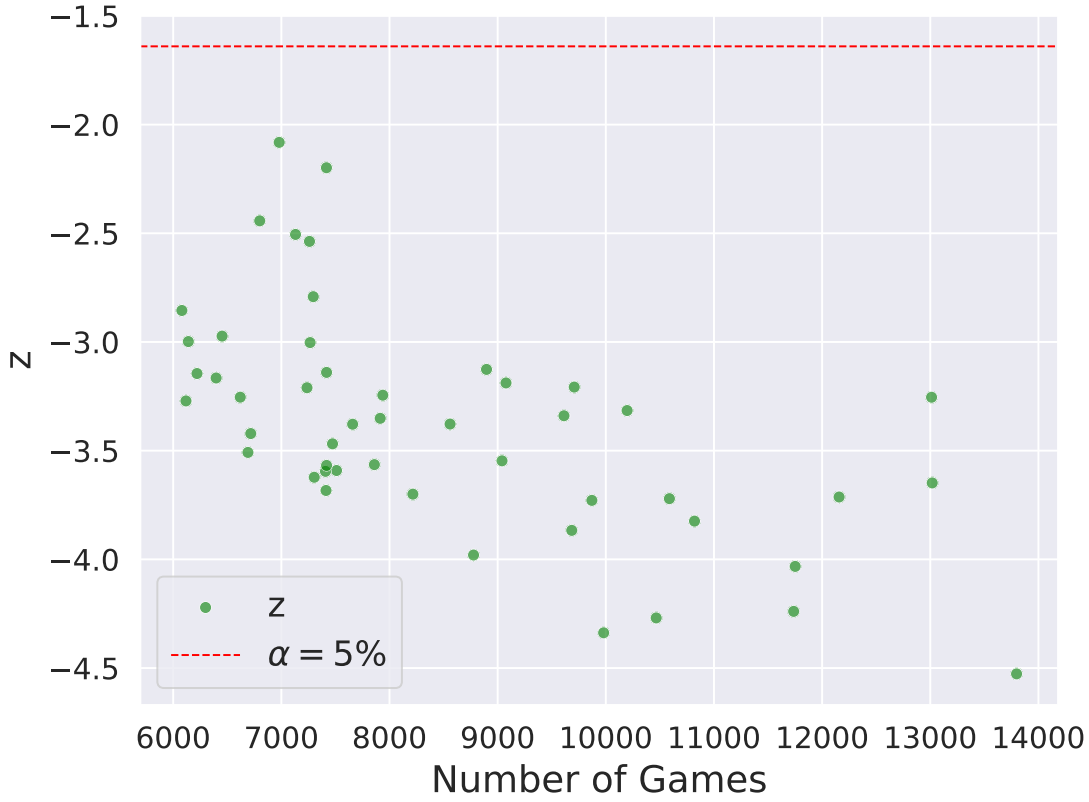


Figure 6.3: For each player, the test statistic z_j using MJ-DLVAT (all) is plotted. The estimation is unbiased for players plotted below the red line.

rankings agree in expectation, it is difficult to prove this using actual results due to the small amount of data for each players. Instead, we will demonstrate that each player is not significantly overestimated or underestimated.

We calculated the test statistics z_j based on Eq. (5.2) for each of the 102 players under evaluation. The results are plotted in Fig. 6.3, where the threshold value $\delta = 0.06$. For players plotted below the red line, the alternative hypothesis H_1 is adopted, and the estimation is unbiased within a significance level of 5%. The players who have played a sufficiently large number of matches are plotted below the red line. Therefore, we have shown that each player is not significantly overestimated or underestimated.

Fig. 6.4 shows the average value of $L_j(z)$ for each player. The values of $L_j(z)$ range between -0.03 and 0.03. This indicates that small modifications to the rank are made for players with a large number of matches.

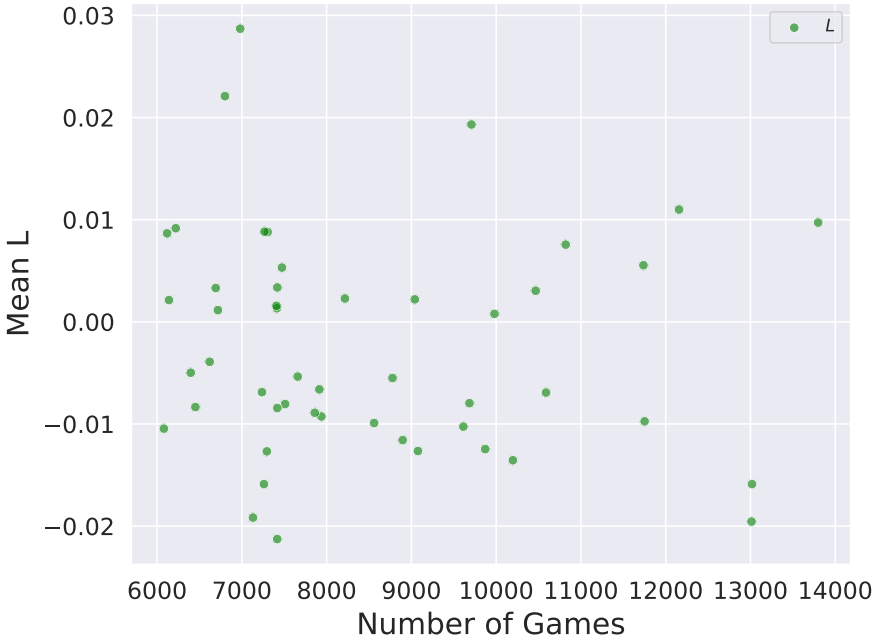
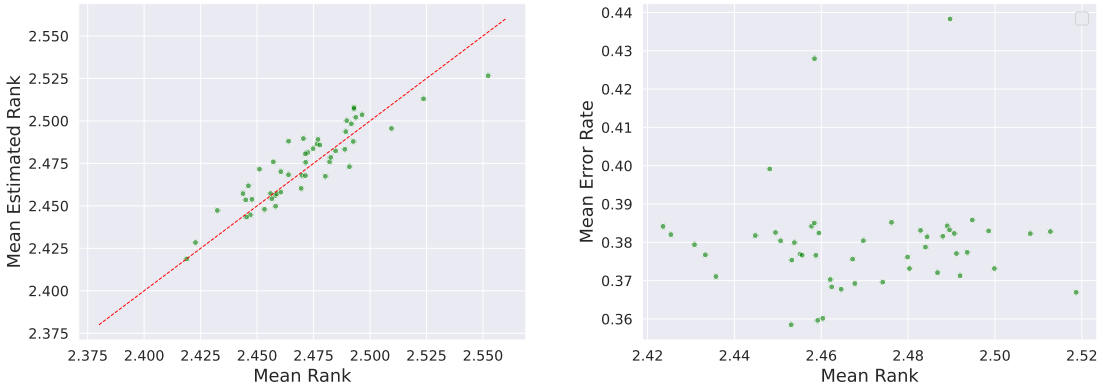


Figure 6.4: For each player, mean value of estimated luck $L_j(z)$ is plotted.

6.4 Comparing MJ-DLVAT with the Error Rate Method

We use scatter plots to compare MJ-DLVAT and the error rate method. Fig. 6.5a shows the relationship between the average ranking actually obtained and the average ranking estimated by MJ-DLVAT (all). Fig. 6.5b shows the relationship between the average ranking actually obtained and the average error rate. Comparing the two figures, the estimated ranking by MJ-DLVAT is highly correlated with the rankings actually obtained, whereas the error rate is less correlated with the rankings obtained.

As shown in Table 5.5, the error rate method successfully distinguished players' ability across different fields. However, it could not measure players' ability differences within the Phoenix field. We have two hypotheses: it does not work when the differences in players' abilities are slight, or it cannot distinguish the difference in the ability of players that is more potent than the ability of the AI agent. Using the data at hand, it is not possible to conduct any more tests related to these hypotheses.



(a) MJ-DLVAT (all)

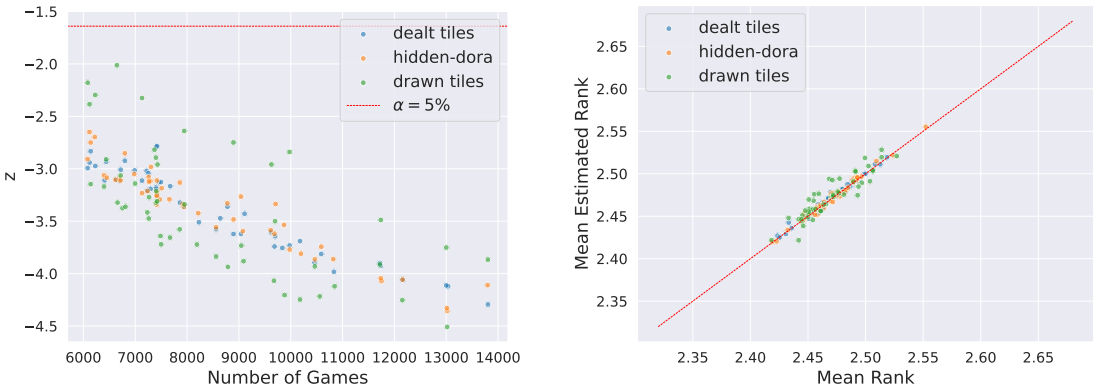
(b) Error rate

Figure 6.5: Scatter plot of estimated average value and average ranking obtained. Each point represents a player.

6.5 Detailed Results of MJ-DLVAT

Furthermore, we visualized the results of MJ-DLVAT for each type of chance player's actions. Fig. 6.6a shows the test statistics z_j in Eq. (5.2) with reduced variance due to drawn tiles, dealt tiles, and hidden-dora, respectively. There is no significant difference between each model. Fig. 6.6b shows the estimated mean rank using MJ-DLVAT for each type of chance player's actions. In all three, the correlation between the estimated utility and the actual ranking gained is large, particularly pronounced for the dealt tiles and hidden-dora.

Fig. 6.7 shows the results of the 95% confidence intervals calculated for the six players with the largest number of games in the test dataset. It can be seen that the average values of the estimated ranks and actual ranks are close. It can also be seen that the confidence intervals calculated using the estimated ranks are considerably smaller than those of the actual ranks.



(a) The unbiased estimation test statistic z_j from MJ-DLVAT.

(b) Estimated average ranking using MJ-DLVAT vs. average ranking gained.

Figure 6.6: Comparison of the result of each MJ-DLVAT model (drawn tiles, dealt tiles, hidden-dora)

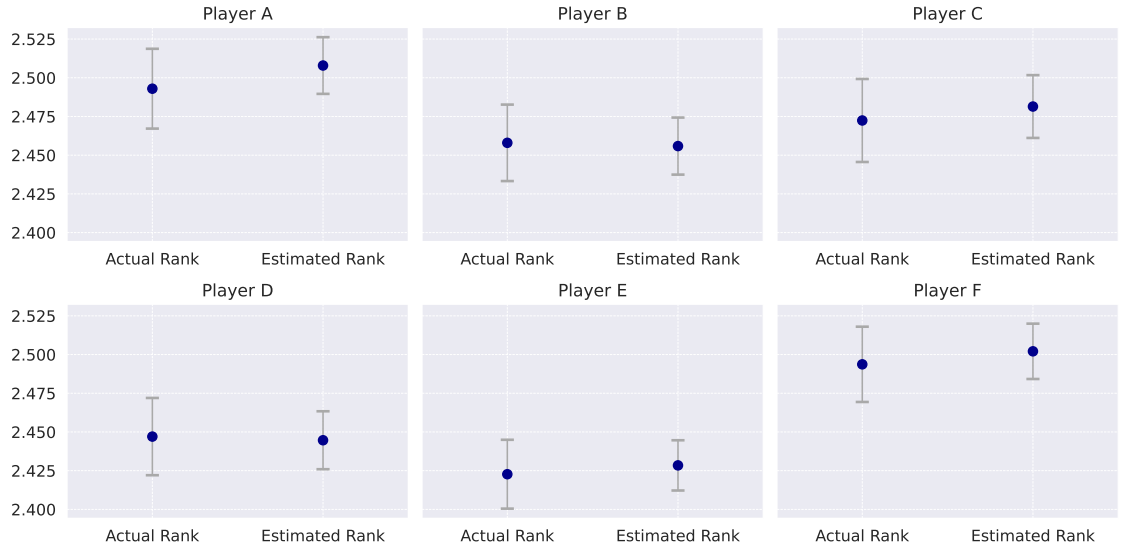


Figure 6.7: 95% confidence interval of estimated rank and actual rank for six players in the test dataset.

Chap.7 Conclusion

In this thesis, we propose a method for estimating the ability of players in mahjong. MJ-DLVAT introduces three techniques: splitting the game into rounds, reducing the variance caused by dealt tiles, drawn tiles and hidden-dora, and introducing a neural network. We showed that the proposed method is unbiased, and the value function learned using the neural network enabled a significant reduction of variance in evaluating player performance in the online mahjong site Tenhou.

Future work includes applications to other games. Existing studies that reduce variance in poker have calculated the value function with a CFR-based method, which can only be applied to games where a tree search is performed. On the other hand, in this study, we trained a deep learning model by minimizing the variance of the estimated utility and succeeded in reducing the variance. This method can be applied to many extensive-form games as long as training data can be prepared. Furthermore, although we estimated the average ranking in this study, theoretically, the proposed method can be applied to any indicator defined at the end of the game.

References

[1] Junjie Li, Sotetsu Koyamada, Qiwei Ye, Guoqing Liu, Chao Wang, Ruihan Yang, Li Zhao, Tao Qin, Tie-Yan Liu, and Hsiao-Wuen Hon. Suphx: Mastering mahjong with deep reinforcement learning. *arXiv preprint arXiv:2003.13590*, 2020.

[2] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[3] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[4] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[5] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 575(7782):350–354, 2019.

[6] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[7] Deheng Ye, Guibin Chen, Wen Zhang, Sheng Chen, Bo Yuan, Bo Liu, Jia Chen, Zhao Liu, Fuhao Qiu, Hongsheng Yu, et al. Towards playing full moba games with deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:621–632, 2020.

[8] Martin Zinkevich, Michael Bowling, Nolan Bard, Morgan Kan, and Darse Billings. Optimal unbiased estimators for evaluating agent performance. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, pages 573–579, 2006.

[9] Martha White and Michael H Bowling. Learning a value analysis tool for agent evaluation. In *IJCAI*, pages 1976–1981, 2009.

[10] Neil Burch, Martin Schmid, Matej Moravcik, Dustin Morill, and Michael Bowling. Aivat: A new variance reduction technique for agent evaluation in imperfect information games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[11] Joshua Davidson, Christopher Archibald, and Michael Bowling. Baseline: practical control variates for agent evaluation in zero-sum domains. In *International conference on Autonomous Agents and Multi-Agent Systems*, pages 1005–1012, 2013.

[12] Anthony D. Pizzo Daniel C. Funk and Bradley J. Baker. esport management: Embracing esport education and research opportunities. *Sport Management Review*, 21(1):7–13, 2018.

[13] Karol Urbaniak, Jarosław Watróbski, and Wojciech Sałabun. Identification of players ranking in e-sport. *Applied Sciences*, 10(19), 2020.

[14] Yubo Kou, Xinning Gui, and Yong Ming Kow. Ranking practices and distinction in league of legends. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, page 4–9, 2016.

[15] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[16] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10674–10685, 2022.

[17] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[18] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR, 2023.

[19] Eric R Ziegel. The elements of statistical learning, 2003.

[20] Shiqi Gao, Fuminori Okuya, Yoshihiro Kawahara, and Yoshimasa Tsuruoka. Building a computer mahjong player via deep convolutional neural networks. *arXiv preprint arXiv:1906.02146*, 2019.

[21] Naoki Mizukami and Yoshimasa Tsuruoka. Building a computer mahjong player based on monte carlo simulation and opponent models. In *IEEE Conference on Computational Intelligence and Games*, pages 275–283, 2015.

[22] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[23] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[24] Martin Zinkevich, Michael Johanson, Michael H. Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1729–1736, 2007.

[25] Oskari Tammelin. Solving large imperfect information games using CFR+. *CoRR*, abs/1407.5042, 2014.

[26] Noam Brown and Tuomas Sandholm. Solving imperfect-information games via discounted regret minimization. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 1829–1836, 2019.

[27] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. In *Advances in Neural Information Processing Systems*, volume 22, pages 1078–1086, 2009.

[28] Hui Li, Kailiang Hu, Shaohua Zhang, Yuan Qi, and Le Song. Double neural counterfactual regret minimization. In *8th International Conference on Learning Representations*, 2020.

[29] Trevor Davis, Martin Schmid, and Michael Bowling. Low-variance and zero-variance baselines for extensive-form games. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 2392–2401, 2020.

[30] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 793–802. PMLR, 2019.

[31] Eric Steinberger. Single deep counterfactual regret minimization. *CoRR*, abs/1901.07621, 2019.

[32] Julia Robinson. An iterative method of solving a game. *Annals of Mathematics*, 54(2):296–301, 1951.

[33] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 805–813, 2015.

[34] David S. Leslie and Edmund J. Collins. Generalised weakened fictitious play. *Games Econ. Behav.*, 56(2):285–298, 2006.

[35] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *CoRR*, abs/1603.01121, 2016.

[36] Li Zhang, Yuxuan Chen, Wei Wang, Ziliang Han, Shijian Li, Zhijie Pan, and Gang Pan. A monte carlo neural fictitious self-play approach to approximate nash equilibrium in imperfect-information dynamic games. *Frontiers Comput. Sci.*, 15(5):155334, 2021.

[37] Kangxin He, Haolin Wu, Zhuang Wang, and Hui Li. Finding nash equilibrium for imperfect information games via fictitious play based on local regret minimization. *Int. J. Intell. Syst.*, 37(9):6152–6167, 2022.

[38] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T. Connor, Neil Burch, Thomas Anthony, Stephen McAleer, Romuald Elie, Sarah H. Cen, Zhe Wang, Audrunas Gruslys, Aleksandra Malysheva, Mina Khan, Sherjil Ozair, Finbarr Timbers, Toby Pohlen, Tom Eccles, Mark Rowland, Marc Lanctot, Jean-Baptiste Lespiau, Bilal Piot, Shayegan Omidshafiei, Edward Lockhart, Laurent Sifre, Nathalie Beauguerlange, Remi Munos, David Silver, Satinder Singh, Demis Hassabis, and Karl Tuyls. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

[39] Anton Bakhtin, David J. Wu, Adam Lerer, Jonathan Gray, Athul Paul Jacob, Gabriele Farina, Alexander H. Miller, and Noam Brown. Mastering the game of no-press diplomacy via human-regularized reinforcement learning and planning. In *The Eleventh International Conference on Learning Representations*, 2023.

[40] Daochen Zha, Jingru Xie, Wenye Ma, Sheng Zhang, Xiangru Lian, Xia Hu, and Ji Liu. Douzero: Mastering doudizhu with self-play deep reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 12333–12344, 2021.

[41] Darse Billings and Morgan Kan. A tool for the direct assessment of poker decisions. *J. Int. Comput. Games Assoc.*, 29(3):119–142, 2006.

[42] Moyuru Kurita and Kunihito Hoki. Method for constructing artificial intelligence player with abstractions to markov decision processes in multiplayer game of mahjong. *IEEE Transactions on Games*, 13(1):99–110, 2021.

[43] Xiangyu Zhao and Sean B. Holden. Towards a competitive 3-player mahjong AI using deep reinforcement learning. In *IEEE Conference on Games, CoG 2022, Beijing, China, August 21-24, 2022*, pages 524–527. IEEE, 2022.

[44] Jinqiu Li, Shuang Wu, Haobo Fu, Qiang Fu, Enmin Zhao, and Junliang Xing. Speedup training artificial intelligence for mahjong via reward variance reduction. In *2022 IEEE Conference on Games (CoG)*, pages 345–352, 2022.

[45] Haobo Fu, Weiming Liu, Shuang Wu, Yijia Wang, Tao Yang, Kai Li, Junliang Xing, Bin Li, Bo Ma, Qiang Fu, and Wei Yang. Actor-critic policy optimization in a large-scale imperfect-information game. In *The Tenth International Conference on Learning Representations*, 2022.

[46] Xiali Li, Zhaoqi Wang, Bo Liu, and Junxue Dai. Lsac-mj: A low-resource consumption reinforcement learning model for mahjong game. *International Journal of Intelligent Systems*, 2024(1):4558614, 2024.

[47] Shinji Matsuda and Eisuke Ito. Estimating the waiting tiles of other players in mahjong. In *IPSJ SIG Technical Report*, volume 2020-MPS-130, pages 1–6, 2020.

[48] Ryuta Aoki, Hidetoshi Nagai, and Teigo Nakamura. Estimating tiles in the wall based on each player's discard tiles in mahjong. In *Proceedings of the 73rd Joint Conference of Electrical, Electronics and Information Engineers in Kyushu*, pages 233–233, 2020.

[49] Takuya Ogami, Ryoya Nara, Katsutoshi Amano, Yuki Imajuku, and Yoshimasa Tsuruoka. Opponent hand estimation in mahjong using transformer. In *Proceedings of Game Programming Workshop 2022*, pages 151–158, 2022.

[50] Hiroki Yanokuchi and Isao Shino. Estimating risk associated with discarding tiles in mahjong by neural network. In *Proceedings of the 79th National Convention of IPSJ*, volume 2017, pages 469–470, 2017.

[51] Hiroki Kume, Moyuru Kurita, and Kunihito Hoki. Estimating the performance of mahjong players using error rates. *IPSJ SIG Technical Reports*, 2019(14):1–7, 2019.

[52] Extreme gammon 2 documentation. Accessed: 2024-12-20.

[53] Nolan Bard, John Alexander Hawkin, Jonathan Rubin, and Martin Zinkevich. The annual computer poker competition. *AI Mag.*, 34(2):112, 2013.

[54] Michael Bowling, Michael Johanson, Neil Burch, and Duane Szafron. Strategy evaluation in extensive games with importance sampling. In *Proceedings of the 25th international conference on Machine learning*, volume 307, pages 72–79, 2008.

[55] Martin Schmid, Matej Moravčík, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, G Zacharias Holland, et al. Student of games: A unified learning algorithm for both perfect and imperfect information games. *Science Advances*, 9(46), 2023.

[56] Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. In *Advances in Neural Information Processing Systems 33*, 2020.

[57] Kai Li, Hang Xu, Enmin Zhao, Zhe Wu, and Junliang Xing. Openholdem: A benchmark for large-scale imperfect-information game research. *IEEE Trans. Neural Networks Learn. Syst.*, 35(10):14618–14632, 2024.

[58] Yaser Norouzzadeh Ravari, Snader Bakkes, and Pieter Spronck. Starcraft winner prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 12, pages 2–8, 2016.

[59] Kodirjon Akhmedov and Anh Huy Phan. Machine learning models for dota 2 outcomes prediction. *arXiv preprint arXiv:2106.01782*, 2021.

[60] Lijun Yu, Dawei Zhang, Xiangqun Chen, and Xing Xie. Moba-slice: A time slice based evaluation framework of relative advantage between teams in moba games. In *Workshop on Computer Games*, pages 23–40. Springer, 2019.

[61] Juan Agustín Hitar-García, Laura Morán-Fernández, and Verónica Bolón-Canedo. Machine learning methods for predicting league of legends game outcome. *IEEE Transactions on Games*, 15(2):171–181, 2023.

[62] Yin Gu, Kai Zhang, Qi Liu, Xin Lin, Zhenya Huang, and Enhong Chen. Massne: Exploring higher-order interactions with marginal effect for massive battle outcome prediction. In Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben, editors, *Proceedings of the ACM Web Conference*, pages 2710–2718, 2023.

[63] A.E. Elo. *The USCF Rating System: Its Development, Theory, and Applications.* United States Chess Federation, 1966.

[64] Mark E Glickman. The glicko system. *Boston University*, 16(8):9, 1995.

[65] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill$^{\text{tm}}$: A bayesian skill rating system. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, pages 569–576, 2006.

[66] Tom Minka, Ryan Cleven, and Yordan Zaykov. Trueskill 2: An improved bayesian skill rating system. *Technical Report*, 2018.

[67] Aram Ebtekar and Paul Liu. Elo-mmr: A rating system for massive multiplayer competitions. In Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia, editors, *The Web Conference*, pages 1772–1784, 2021.

[68] Jiasheng Wang. Graph embedding augmented skill rating system. *IEEE Trans. Games*, 15(3):460–468, 2023.

[69] Arman Dehpanah, Muheeb Faizan Ghori, Jonathan Gemmell, and Bamshad Mobasher. Player modeling using behavioral signals in competitive online games. In *International Conference on Computational Science and Computational Intelligence*, pages 569–574. IEEE, 2021.

[70] Cem Yuksel. Skill-based matchmaking for competitive two-player games. *Proc. ACM Comput. Graph. Interact. Tech.*, 7(1), 2024.

[71] Haobo Fu, Weiming Liu, Shuang Wu, Yijia Wang, Tao Yang, Kai Li, Junliang Xing, Bin Li, Bo Ma, QIANG Fu, et al. Actor-critic policy optimization in a large-scale imperfect-information game. In *International Conference on Learning Representations*, 2022.

[72] Dongqi Han, Tadashi Kozuno, Xufang Luo, Zhao-Yun Chen, Kenji Doya, Yuqing Yang, and Dongsheng Li. Variational oracle guiding for reinforcement learning. In *International Conference on Learning Representations*, 2021.

[73] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[74] S Jian, H Kaiming, R Shaoqing, and Z Xiangyu. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 770–778, 2016.

[75] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

Publications

Reviewed Papers

[1] T. Ogami, K. Amano, Y. Tsuruoka. MJ-DLVAT: A Deep Learning Value Assessment Technique for Mahjong, IEEE Conference on Games, 2024

Non-Reviewed Papers

[2] 大神卓也, 天野克敏, 奈良亮耶, 鶴岡慶雅. 分散減少法を用いた麻雀における実力推定, ゲームプログラミングワークショップ, 2024

Acknowledgement

This work was supported by the "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures (JHPCN)" and the "High Performance Computing Infrastructure (HPCI)" in Japan (Project ID: jh230044). I am grateful for their support, which made this research possible.

I would like to express my heartfelt gratitude to Professor Yoshimasa Tsuruoka, my academic advisor, for his invaluable guidance and support over the past three years since my third year as an undergraduate student. His meticulous corrections of my manuscripts and insightful advice on the direction and approach to research have been instrumental in shaping my research. I am also deeply grateful for the numerous opportunities he has provided me throughout this period. I extend my sincere appreciation to colleagues in the laboratory for their stimulating discussions. These interactions have significantly helped me refine my research. I would also like to express my gratitude to everyone who has supported me throughout this journey. Your encouragement and guidance have been a great source of motivation and inspiration. I deeply appreciate the role each of you has played in helping me complete this work.