Wiley
International Journal of Intelligent Systems
Volume 2024, Article ID 4558614, 20 pages
https://doi.org/10.1155/2024/4558614

WILEY

Research Article

LsAc*-MJ: A Low-Resource Consumption Reinforcement Learning Model for Mahjong Game

Xiali Li[1,2] **Zhaoqi Wang**[1,2] **Bo Liu**[1,2] **and Junxue Dai**[1,2]

[1]*Key Laboratory of Ethnic Language Intelligent Analysis and Security Governance of Ministry of Education of the People's Republic of China, Beijing 100081, China*
[2]*School of Information and Engineering, Minzu University of China, Beijing 100081, China*

Correspondence should be addressed to Xiali Li; xiaer_li@163.com

Received 11 April 2024; Revised 15 May 2024; Accepted 20 June 2024

Academic Editor: Yu-an Tan

This article proposes a novel Mahjong game model, LsAc*-MJ, designed to address challenges posed by data scarcity, difficulty in leveraging contextual information, and the computational resource-intensive nature of self-play zero-shot learning. The model is applied to Japanese Mahjong for experiments. LsAc*-MJ employs long short-term memory (LSTM) neural networks, utilizing hidden nodes to store and propagate contextual historical information, thereby enhancing decision accuracy. Additionally, the paper introduces an optimized Advantage Actor-Critic (A2C) algorithm incorporating an experience replay mechanism to enhance the model's decision-making capabilities and mitigate convergence difficulties arising from strong data correlations. Furthermore, the paper presents a two-stage training approach for self-play deep reinforcement learning models guided by expert knowledge, thereby improving training efficiency. Extensive ablation experiments and performance comparisons demonstrate that, in contrast to other typical deep reinforcement learning models on the RLcard platform, the LsAc*-MJ model consumes lower computational and time resources, has higher training efficiency, faster average decision time, higher win-rate, and stronger decision-making ability.

1. Introduction

Mahjong is considered a prime example of games with imperfect information, and it is known for its vast search space. Mahjong originated in China and has since been widely circulated in East Asia, with many variants. Some notable variants include Japanese Mahjong (with 136 tiles), Chinese Official Mahjong (with 144 tiles), Sichuan Mahjong (with 108 tiles), and Popular Mahjong (also with 108 tiles). Researchers have applied deep reinforcement learning models to Mahjong, such as Suphx [1] for Japanese Mahjong and Tencent's LuckyJ [2], which have successfully defeated human expert-level players. However, the extensive computational resources required for these models present challenges for ordinary research laboratories. Suphx utilized carefully processed player logs for the supervised learning training phase, although the details of the datasets used have not been disclosed. Additionally, commonly played Mahjong variants like Chinese Official Mahjong, Sichuan Mahjong, and Popular Mahjong currently lack publicly available datasets. In addition, the intricate relationships between contextual actions in Mahjong make it difficult to fully utilize relevant information.

The Advantage Actor-Critic (A2C) algorithm is an effective approach for tackling problems in complex state spaces. It has shown impressive performance in 3D games with incomplete information [3, 4]. Mahjong, being a game with high complexity and rich information, serves as an excellent example for examining the capabilities of A2C. A2C's ability to handle high-dimensional state spaces makes it suitable for application in Mahjong, leading to improved decision-making. However, A2C encounters difficulties with convergence when dealing with highly correlated input data. In the case of Mahjong, the hand information in different rounds is strongly correlated, which poses a significant challenge for A2C. To overcome this issue, this study

incorporates an experience replay mechanism [5] into A2C. This mechanism creates a replay pool to store experiences, allowing for random sampling of small batches of data for learning. By introducing this feature, we aim to disrupt the data correlation and address the convergence challenges faced by the model in the context of Mahjong gaming.

Long short-term memory networks (LSTMs) are both general and effective at capturing long-term temporal dependencies [6]. In Mahjong, players sequentially play their tiles in a designated order, and experienced players make predictions about their action strategy based on the opponents' discards. The current and preceding actions in Mahjong significantly influence decision-making and can be considered a game with temporal characteristics. Therefore, the proposed model in this paper adopts LSTM networks, which are sensitive to temporal information, to capture and utilize the temporal patterns present in Mahjong. The hidden layer nodes of the LSTM network are employed for capturing and leveraging both the long-term reward signals and the historical tile information. This enhancement contributes to the accuracy of predicting the value associated with tile placement actions.

Mahjong has the problems of lack of high-quality datasets and difficulty in fully utilizing context-associated information. In view of these problems and the fact that it is difficult for ordinary laboratories to afford huge computational resources to carry out self-play model training from zero, this paper carries out the research work on Mahjong game models with low resource consumption and high performance. The main contributions are as follows:

(1) The LsAc*-MJ mahjong model is proposed, which consumes lower computational resources while maintaining higher decision accuracy. The model employs an improved A2C and integrates the experience replay mechanism into the A2C, resulting in lower computational resource requirements and higher decision efficiency. The suitability of this model for Mahjong, coupled with its ease of implementation in ordinary laboratories, serves to mitigate the convergence challenges linked to strong correlations among Mahjong hand information.

(2) The LSTM network is applied for the first time in the Mahjong game model. LSTM networks are applied to Mahjong game models, effectively addressing the challenge of underutilizing extensive contextual correlations in Mahjong. Hand information, coupled with action-value sequences, is input into the LSTM network. The hidden layer nodes of the network process long-term reward signals and historical tile information, enabling the model to leverage temporal relationships among historical information for optimal action selection in the current game state, thereby enhancing decision accuracy.

(3) The paper proposes a two-stage training method for knowledge-guided deep reinforcement learning models. By combining knowledge-guided training and self-play, the model learns hand information and action-value sequences based on expert knowledge during the knowledge-guided phase, enhancing the accuracy of action-value predictions. After the model's loss stabilizes, the training transitions to the self-play phase to improve action decision-making capabilities. This training method circumvents the issue of limited datasets, saves training time, and enhances decision speed.

The remainder of this article is structured as follows. Section 2 provides a brief overview of the related work in Mahjong game research. The detailed description of the proposed Mahjong game model with the improved A2C is presented in Section 3. Section 4 encompasses experiments, comparisons, and property analysis. Finally, concluding remarks and future avenues for investigation are provided in Section 5.

2. Related Work

Mahjong agents were initially constructed using knowledge-based approaches. The expert knowledge is used to set action priorities, and the search algorithms are used for decision-making. The completeness and appropriate representation of human players' knowledge as rules play a crucial role in determining the agent's proficiency. Agents such as Long Cat [7], LongCatMJ [8], MahjongDaXia [9], VeryLongCat [8], and others were developed using expert knowledge, but their decision-making abilities are limited due to the incompleteness of expert knowledge. Literature [10, 11] improved the agent's proficiency by combining opponent modeling with game tree structures. The complexity of Mahjong means that only a limited number of situations can be regularized. A large number of intuition-based decisions are difficult to describe in rules, limiting the performance of agents relying on expert knowledge in the face of variable game situations. These agents tend to prefer strategies that pursue fast listening and drawing, as well as avoiding being drawn by other players. Although compliant with the basic rules of Mahjong, the agents lack the flexibility to cope with the variable game situations in Mahjong.

As data processing power continues to increase and neural networks advance, there is an increase in the construction of data-driven Mahjong agents. Agents with data-driven approaches learn features from large amounts of game data and optimize decision models through continuous training. Literature [12, 13] employed CNN (convolutional neural networks) to construct Mahjong game models, improving discard accuracy. Literature [14] pioneered the application of residual neural networks to Mahjong in Shangrao, Jiangxi Province, China. Its game data of top-ranked master players were selected, and low-level semantic features were used to guide the model learning. The agent can learn and apply high scoring tile types is realized. Literature [15] used ResNet (residual network) to build Mahjong agents, combining them with XGBoost (eXtreme gradient boosting), low-level semantic features, and others. Literature [1, 2, 16–21] used deep reinforcement learning to construct Mahjong agents, achieving substantial

improvements in average proficiency. Suphx [1], the first Mahjong AI to surpass top human players, used handcrafted features and a complex model structure, leading to extensive computational resource requirements that ordinary laboratories cannot provide. Subsequent optimization studies focused on adjusting data structures [22], optimizing feature engineering [23], and developing distributed frameworks [23]. LuckyJ [2], developed by Tencent AI Lab, surpassed Suphx, reaching a professional ten-dan level in Japanese Mahjong and achieving victory over Chinese official Mahjong professional players. LuckyJ proposed a neural-based CFR algorithm with game tree search results as a feature, which not only has a low drawn rate but also has perfect attack and defense strategies. The main data-driven agent construction methods have been specifically analyzed, and the disadvantages are listed in detail in Table 1. It can be seen that most of these methods generally have the problems of over-reliance on high-quality datasets and high demand for computer resources. Besides, no researcher has focused on how to use the temporal relationship available in Mahjong state information. Therefore, the low resource consumption model, LsAc∗-MJ, was proposed to fully utilize the temporal sequence of mahjong state information, greatly reducing dependence on datasets while improving decision-making ability.

The A2C algorithm (Advantage Actor-Critic) is introduced by Barto et al. and is a kind of reinforcement learning algorithms. The A2C combines value functions with policy gradients through synchronous updates [24, 25]. It is an optimization of the A3C (Asynchronous Advantage Actor-Critic) algorithm [4]. This algorithm replaces multiple groups of agents with asynchronous updates with simultaneously interacting, making it simpler and more CPU-efficient. In comparison to value-based methods such as DQN [5, 26], Dueling Deep Q-Network [27], and Double Deep Q-learning [28], A2C strikes a balance between exploration and exploitation, making it particularly suitable for handling high-dimensional state space problems. Actor-Critic algorithms have demonstrated excellence in various domains, including robot control [29–31], 3D games like First-Person Shooter (FPS) [3], and Atari 2006 [4]. However, the A2C has not been applied in the context of Mahjong gaming.

LSTM (long short-term memory) is a neural network architecture designed to process sequential data [6]. Building on the foundations of recurrent neural networks (RNNs) [32], LSTM introduces memory cells to address the problem of vanishing gradients during the back-propagation process [33]. By using hidden layer nodes to capture temporal dependencies between data points, LSTM is well suited for handling and predicting data with time-series relationships. LSTM has extensive applications in natural language processing [34, 35], speech recognition [36, 37], and image processing [38]. By capturing long-term dependencies in text, speech, and images, LSTM improves prediction accuracy. However, LSTM networks have not yet been applied in the context of Mahjong.

3. Rule of Japanese Mahjong

A complete set of Japanese Mahjong contains a total of 136 tiles. These tiles are divided into two categories: numbered tiles and honour tiles. Numbered tiles consist of numbers 1 to 9 and are divided into three suits: characters, bamboos, and dots. Honour tiles include East, South, West, North, Red Dragon, White Dragon, and Green Dragon. Each tile has four copies, so a full set of Mahjong tiles comprises 136 tiles.

The main actions involved in the game include drawing tiles, chow, pong, kong, discarding tiles, and Riichi. In Mahjong, two identical tiles are generally called a pair, three identical tiles are called a triplet (e.g., 888 characters), and three sequentially consecutive tiles (e.g., 123 bamboos) are called a sequence. Typically, chow involves forming a sequence by obtaining a discarded tile from the preceding player. Pong involves forming a triplet by obtaining any discarded tile from an opponent. Kong involves obtaining any discarded tile identical to a pair after forming a pair. In the same turn, the player applying for kong receives priority in acquiring the tile, followed by the player applying for pong, and lastly, the player applying for chow. It is important to note that kong and pong actions cause the turn to pass directly to the player proposing the action, skipping other players. "Riichi" is a unique rule in Japanese Mahjong. If a player believes they have met the conditions to declare a win, they can declare "Riichi" to exchange for a high scoring reward. Declaring Riichi means the player's subsequent actions can only be winning or discarding the drawn tile; they cannot manipulate their hand tiles further.

An opening is required before the start of the game to determine the position of the dealer's starting draw. The four players draw tiles in a fixed order, each obtaining a set of 13 initial tiles. Through continuous actions, they update their hand tiles to achieve a winning hand. A winning hand in Japanese Mahjong typically consists of a pair and several sequences and triplets. Additionally, there are some special combinations such as seven pairs, fully concealed hand, full flush, thirteen orphans, nine gates, and all honours, which bring high scores. In our experiment, we focus on whether the model eventually wins or not, so scoring is simplified to only distinguish between winning, losing, or a draw.

4. LsAc*-MJ Model

4.1. Combing Experience Replay into Actor-Net and Critic-Net. The Actor-Critic series is a comprehensive algorithm that combines the advantages of value function and strategy gradient. Compared with value function-based methods such as DQN, the AC algorithm effectively balances exploration and exploitation and is more suitable for dealing with Mahjong in high-dimensional state spaces. The hand information of different rounds of Mahjong is highly correlated, and the A2C algorithm requires a more complex tuning process in its application. Given the high-dimensional state space, A2C is sometimes not as stable as the methods based on the value function alone and is prone to nonconvergence.

Table 1: Characteristics of common Mahjong agent construction methods.

Methods	Specific methods	Disadvantages
Deep learning network	CNN [12, 13] ResNet [14] DenseNet [15] MLP [22]	The level of agents relies primarily on high-quality datasets. However, the production of the dataset requires a lot of work, and the data cannot be generalized across different mahjong variants. In addition, the decision styles of the trained agents tend to be conservative because of the richness of the input information, among other reasons
Deep reinforcement learning	DQN or double DQN [17, 18] A3C with CNN [19]	The training of agents is sensitive to reward sparsity and is not easy to converge on. It is also usually necessary to combine expert knowledge in applications
Self-play	CNN and self-play [1, 16, 23] CFR and self-play [2]	Agents require complex dataset production in supervised learning. Algorithms dealing with reward sparsity and real-time optimization in self-play training are complex and have a high overhead on computational resources
Others	PCA [21]	Estimating the opponent's strategy based on their behaviors, but not taking into account the situation information

Drawing on the success of DQN [5], we introduce an experience replay mechanism for the A2C model as well. Experience replay constructs a circular experience replay pool to store the agent's experiences (i.e., state transitions, actions, rewards, and new states) while exploring the environment. During training, the algorithm samples a random batch of experiences from the pool as input, rather than just using the latest experience to update the model. Successive observations tend to be highly correlated, which can lead the model to over-rely on recent experience rather than learning a generalized strategy. With random sampling, the model can learn from different experiences, which helps improve the quality and generalization of the final strategy. Besides, experience replay improves the efficiency of data usage, by storing experiences and reusing them multiple times, the need for new data can be significantly reduced.

Aiming at the high-dimensional state space of Mahjong game, the LsAc*-MJ model is proposed on the basis of the A2C algorithm and the experience replay mechanism to improve the convergence speed and stability of the model. The architecture of the LsAc*-MJ model is shown in Figure 1.

Among them, the Actor-net and the Critic-net adopt the LSTM networks. The inputs to the network are formed by random sampling from the experience pool. The Actor-net outputs possible actions with their probability values, and the value network outputs the evaluation value for the current action. The Actor-net update function is the cross-entropy function, and the Critic-net update function is the mean-variance loss function.

Taking the s_t as input to the Actor-net, we obtain the P_{Action} (1). These probabilities consist of n possible actions, and the action a_t with the highest probability is selected and executed, transitioning to the s_{t+1}.

$$P_{\text{Action}} = [p_1, p_2, \ldots, p_n]. \tag{1}$$

The other outputs of Actor-net are a_t and $\widetilde{a}_{t+1}$, which are listed in (2) and (3).

$$a_t = \pi(\cdot \mid s_t; \theta_{\text{now}}), \tag{2}$$

$$\widetilde{a}_{t+1} = \pi(\cdot \mid s_{t+1}; \theta_{\text{now}}), \tag{3}$$

where s_t represents the game state at time t, including one's own concealed and exposed tiles, opponents' exposed tiles, and discarded tiles. The Actor-net outputs action probabilities, denoted by a_t, and a_t represents the action with the highest probability. $\widetilde{a}_{t+1}$, corresponds to the action with the highest probability in the next state after executing action a_t, θ_{now} representing the parameters of the Actor-net at time t.

The Critic-net evaluates the current state s_t, obtaining the evaluation value V_t. After inputting s_{t+1} into the Critic-net, the evaluation value V_{t+1} is obtained. The formulas for the output of the Critic-net, representing the evaluation values V_t and V_{t+1}, are as shown in (4) and (5).

$$V_t = V(s_t, a_t, \omega_{\text{now}}), \tag{4}$$

$$V_{t+1} = V\left(s_{t+1}, \widetilde{a}_{t+1}, \omega_{\text{now}}\right), \tag{5}$$

where ω_{now} represents the Critic-net parameters at time t. a_t is the action chosen by the actor based on s_t, and $\widetilde{a}_{t+1}$ is the action chosen by the Actor-net based on s_{t+1}. The update of network parameters requires calculating the TD error, utilizing the two evaluation values V_t and V_{t+1} along with the environmental reward, as shown in (6).

$$\text{TD} = R(s_t, a_t) + \gamma_1 V_{t+1} - V_t, \tag{6}$$

where γ_1 is the discount factor, representing the weight of V_{t+1}, typically ranging between 0 and 1. $R(s, a)$ is the reward value received by the environment s_t after executing action a_t. For example, $R(s_t, a_t) = 3$ indicates the model achieving victory, $R(s_t, a_t) = -1$ indicates the model experiencing failure in the game. $R(s_t, a_t) = 0$ indicates a draw in the game, where no one emerges victorious at the end.

As A2C is a synchronous updating algorithm, both θ_{now} and ω_{now} represent the Actor-net and Critic-net parameters at time t. The update formulas for θ and ω are shown in (7) and (8).

$$\theta_{\text{new}} = \theta_{\text{now}} + \alpha \nabla_\theta \log \pi_\theta(a_t \mid s_t; \theta_{\text{now}})\text{TD}, \tag{7}$$

$$\omega_{\text{new}} = \omega_{\text{now}} + \beta * \text{TD} * \nabla_\omega V_t, \tag{8}$$

where α and β are hyperparameters of the neural network. α is the learning rate of the Actor-net, and β represents the step size for updating the value function.

In the A2C model, the Q-value is computed using (9).

$$Q_t(s_t, a_t) = R(s_t, a_t) + \gamma_2 \sum_{s_{t+1}} V(s_{t+1}, a_{t+1}). \tag{9}$$

Here, γ_2 is the discount factor, taking values between 0 and 1, and is used to measure the importance of future rewards.

In each turn of Mahjong, actions are inherently correlated with historical information, resulting in input data exhibiting strong correlations that hinder the convergence of A2C. Therefore, the model incorporates an experience replay mechanism to promote convergence through random sampling. With the introduction of the experience replay mechanism, the model's Q-values become dependent on reward values and the maximum Q-value of all possible actions after the state transition. The formulation for Q-value calculation is given by (10).

$$Q_t(s_t, a_t) = R(s_t, a_t) + \lambda * \text{Max}[Q(s_{t+1}, \text{all action})], \tag{10}$$

where the parameter λ, commonly referred to as the discounting parameter, is utilized to balance current and future rewards. It takes values in the range of 0 to 1. When λ

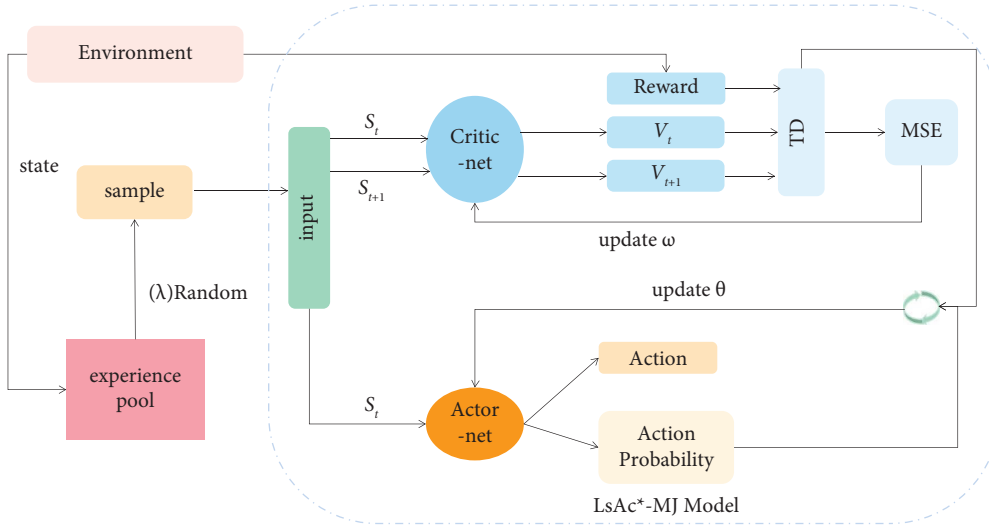

Figure 1: LsAc*-MJ model.

approaches 1, there is a greater emphasis on future rewards, while when λ approaches 0, there is a greater emphasis on current rewards.

The activation function employed is the SoftPlus, with a range of values in the interval $(0, +\infty)$, as shown in (11).

$$\text{Softpuls}(x) = \log(1 + e^x). \tag{11}$$

The pseudocode for the core components of the LsAc*-MJ model is shown in Table 2.

4.2. LSTM Networks Adopted by the Model. Mahjong is a game full of strategy and skill, in which the four players play the tiles in a prescribed order, and the temporal information embedded in the sequence of their play is crucial. First of all, the types of tiles played by the players in each round constitute the basic temporal information, which not only reflects the current strategic intention of the players but also provides clues for the opponents to infer the structure of their hands. For example, playing cards of the same suit consecutively may imply that the player is trying for a straight or a clean slate. Second, the choice and timing of actions such as chows, pongs, and kongs also constitute important timing information. By changing the dynamics of the hand through these actions, a player not only demonstrates his or her own offensive or defensive strategy but may also influence the decision-making process of other players. For example, choosing to pong a card may indicate that a player is actively building a hand, whereas ignoring the opportunity to pong a card may mean that a player is pursuing a higher-value hand or hiding his or her strength. In addition, the progression stages of the game are themselves a form of temporal information, and players' strategies and behavioral patterns can vary significantly from stage to stage. Early in the game, players may be more focused on building and accumulating hands, while near the end of the game, players are more likely to be conservative in their decision-making or to seek a quick draw. Players' behavioral patterns, including how they respond to their opponents' attacks and how they make decisions in different situations, contain a wealth of information, and if this temporal information can be appropriately utilized for decision-making, it can lead to a rapid improvement in the performance of mahjong gaming agents.

Long short-term memory (LSTM) networks are designed for processing long sequential data [6]. Through gating mechanisms and memory cells, LSTM networks provide a powerful way to process temporal information and learn temporally separated dependencies from data. Currently, LSTM has become one of the preferred techniques for dealing with sequential data and complex time series prediction problems. The proposed Actor-net and Critic-net of the model both adopt a long short-term memory (LSTM) network structure with five hidden layers, as illustrated in Figure 2. This structure includes six input channels, one fully connected layer, and an output layer.

The LSTM consists of three gate controllers: an input gate, a forgetting gate, and an output gate, which work together to decide whether information is retained, forgotten, or updated. How the three gate controllers work is shown in the following (12)–(14). The working principle of the input gate is shown in (12). The working principle of the forget gate is shown in (13). The working principle of the output gate is shown in (14). How the three gate controllers work is shown in the following (12)–(14):

$$i_t = \text{Sigmoid}(W_i \cdot [x_t, h_{t-1}] + b_i), \tag{12}$$

$$f_t = \text{Sigmoid}\left(W_f \cdot [x_t, h_{t-1}] + b_f\right), \tag{13}$$

$$o_t = \text{Sigmoid}(W_o \cdot [x_t, h_{t-1}] + b_o), \tag{14}$$

where i_t denotes the computed result of the input gate at $T = t$, f_t denotes the computed result of the forget gate at $T = t$, and o_t denotes the computed result of the output at $T = t$. W denotes the corresponding weight matrix. b is the computed bias term.

Table 2: Pseudocode of LsAc*-MJ model.

```
//LsAc-MJ model pseudocode
Initialize Actor-net (θ) and Critic-net (ω)
Initialize replay buffer D
Initialize t, α, β
while stopping criterion not met do
    environment.reset () get state s(t)
    while not done do
        a(t) ⟵ actor_net (s(t), θ)
        s(t + 1), r(t), done ⟵ environment.step (a(t))
        //store transition in replay buffer
        D.add(a(t), r(t), s(t), s(t + 1), done)
        t ⟵ t + 1
    end while
    //random sampling
Batch ⟵ D.sample(batch_size)
  //calculate discounted rewards and advantages
  Rewards ⟵ calculate_discounted_rewards(Batch.rewards)
  A ⟵ calculate_advantages(Critic-net(ω), Batch.states,
  rewards)
    //update networks parameters
    Loss ⟵ update_networks(A, Batch, θ, ω)
    θ ⟵ θ + α actor_loss_gradients
    ω ⟵ ω + β critic_loss_gradients
end while
```

In addition to the gating mechanism, the memory function of LSTM, which is represented by (15), mainly comes from its memory cell c_t represented by equation (16):

$$g_t = \left(W_g \cdot [x_t, h_{t-1}] + b_g\right), \tag{15}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t, \tag{16}$$

where g_t denotes the cellular state of the candidate at $T = t$, f_t determines the amount of signals that are continued to be memorized, and i_t determines which of the current signals will be remembered.

After the data are input into the model and stored in x_t, h_t, the hidden states of the LSTM continuously propagate between different time steps within the current layer and the same time step in the next layer. As shown in Figure 3, at time $T = t$, information progresses layer by layer from the input layer through the hidden layers, completing the horizontal transmission for the same time step. Meanwhile, at time $T = t$, the model receives information transmitted from time $T = t - 1$ and passes the processed information to time $T = t + 1$, completing the vertical transmission across different time steps. This mechanism, involving both vertical and horizontal transmission, ensures that predictions not only consider the current time step's state but also take into account the states of historical time steps, maintaining the correlation of information in the time series. This is crucial for handling historical discard information and reward information in the Mahjong, allowing for better utilization of historical information to guide current decision-making.

To ensure that the model fully understands temporal features, we vertically slice the state matrix based on information features. In addition to the player's own hand, the current discarded tiles and the opponent's exposed tiles are also taken into consideration. The feature dimensions are 6, corresponding to input_1 through input_6. Specifically, six sequences of matrices are input sequentially. The state encoding is illustrated in Figure 4.

After the information is encoded, it forms a matrix of dimensions, which is the input of the model. When inputting the state information into the network, most researchers adopt a horizontal slicing approach [12–15], which involves splitting along the 34 types of tiles. We employ the different slicing method, segmenting the input based on the respective categories. This approach not only aligns closely with human understanding of the tile information in real-world scenarios but also allows the neural network model to better capture temporal features.

4.3. Knowledge-Guided Two-Stage Training Method. In situations where ordinary laboratories cannot provide powerful computational resources, training from zero through self-play is highly sensitive to hyperparameters and tends to be slow. The common approach for many researchers is to initially use supervised learning before self-play training. However, manually decomposing and selecting game data for supervised learning is time-consuming and inefficient. Therefore, this paper proposes a novel training method that differs from supervised learning, which relies on a game dataset, and from self-play training, which starts with completely random exploration. Instead, it employs a two-stage training approach: the first stage involves knowledge-guided training, and the second stage comprises self-play training. This method saves training time, enhances the speed of model decision-making, and effectively circumvents the issue of a limited Mahjong dataset.

4.3.1. First Stage Training with Knowledge Guidance. The first stage of knowledge-guided training is illustrated in Figure 5. The four models in the game include an LsAc*-MJ model trained from zero and three knowledge-guided models. In the initial stages of training, action selection is random. After each round, the actions of the expert-guided models, along with their corresponding reward, are encoded as states and input into the LsAc*-MJ model for learning and parameter updates. By learning from the actions of the expert-guided models, excessive exploration caused by random discarding at the beginning of training is avoided. This allows the model to quickly grasp the game rules, thereby accelerating the training process.

Knowledge-guided agents make choices in decisions with the help of action prioritizations that have been set based on the knowledge accumulated by humans in order to achieve better game performance. The prioritization of these moves is derived from the general knowledge and experience of humans in the game of Mahjong. For example, as dragon tiles are not easily formed into winning tile combinations and discarding them does not result in significant point loss; players typically prioritize discarding dragon tiles. Therefore, we set discarding flower tiles as the first level and assign the highest reward value to guide the model to prioritize this

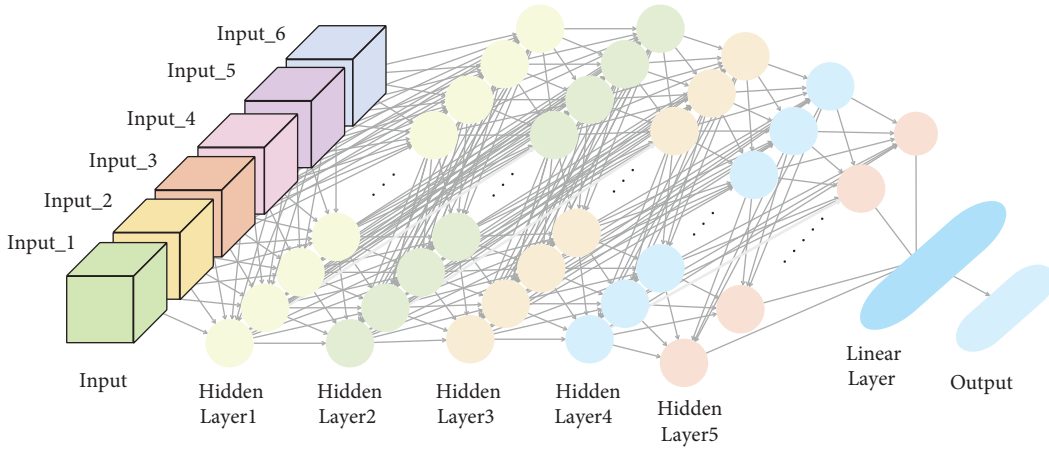

Figure 2: The network structure employed by the LsAc*-MJ model.

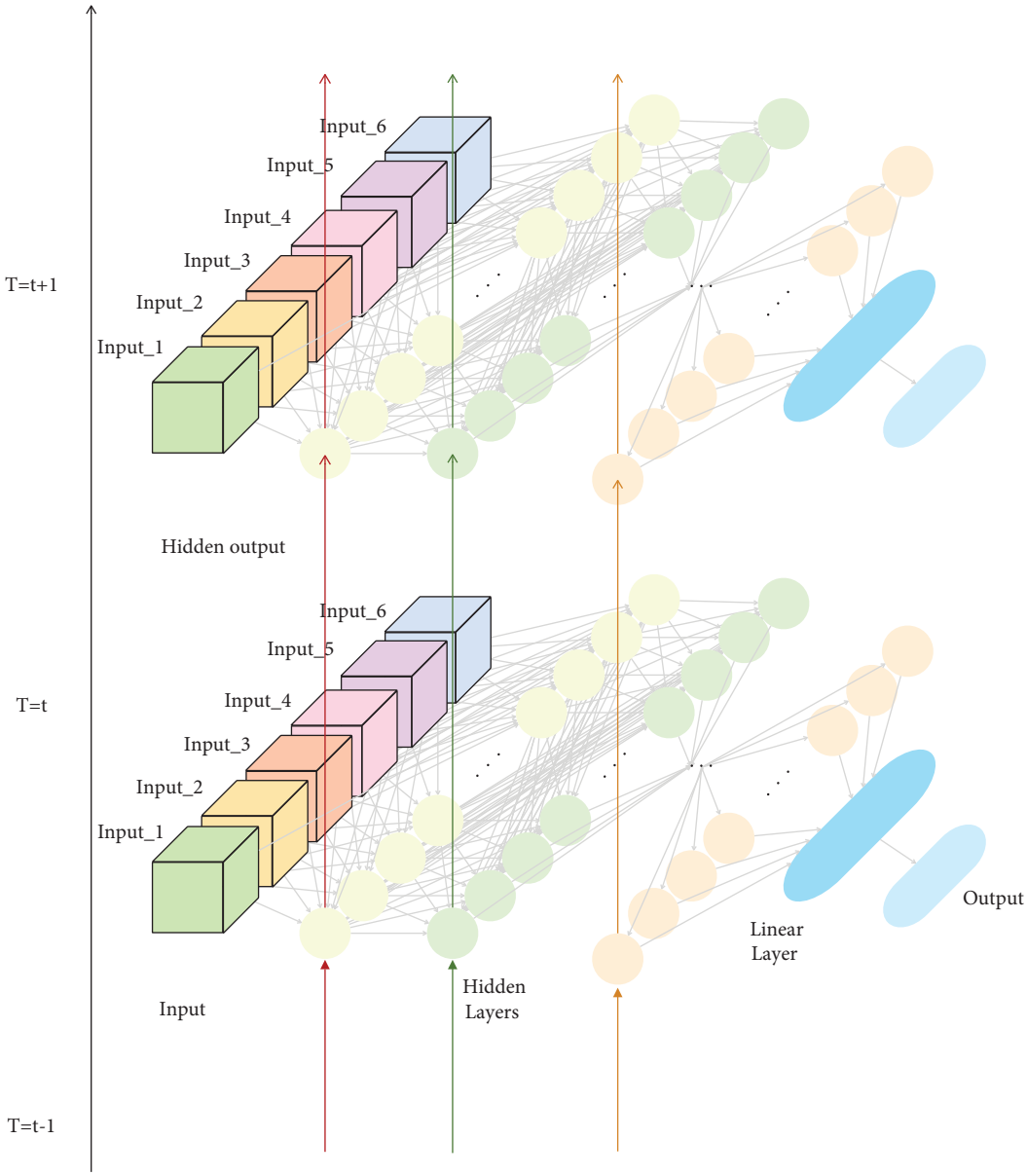

Figure 3: Information transmission in the network structure.

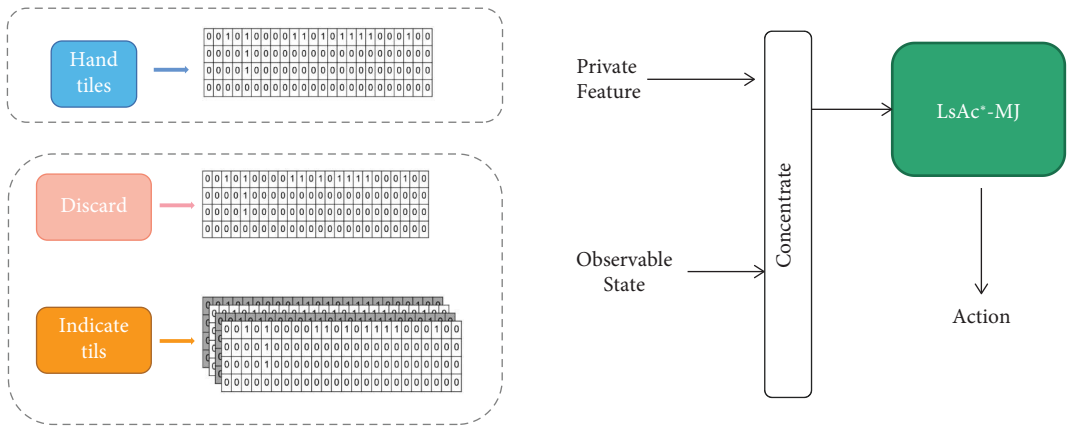

Figure 4: State encoding.

strategy during decision-making. Table 3 shows the specific prioritization of these actions and provides the brief description and examples of each level.

After observing the current state, the agent will evaluate all the actions based on the set priority of the action and its value, after which it will select the action with the highest valuation for execution. It is noteworthy that the decision-making process of the knowledge-guided models only includes strategies based on the current hand, without involving how to utilize historical information to calculate opponent information and predict reward. This design aims to avoid introducing too much knowledge that could influence the depth of exploration in reinforcement learning.

4.3.2. Second Stage Self-Play. The knowledge-guided training stage is halted once the loss function converges to stability. Then, the model's performance is continually improved through self-play. The self-play method is well suited for Mahjong and other gaming games because it allows the agent to play against itself without external input to learn updates and gradually improve its performance. The core advantage of the self-play is that it constantly seeks a balance between exploring new strategies and optimizing known ones, thus achieving self-improvement. Self-play demonstrates the potential of agents in the process of self-evolution and is also used by Suphx [1]. Therefore, we apply the self-play to the continued training of LsAc*-MJ. Without relying on human data and prior knowledge, LsAc*-MJ is able to independently explore the strategy space through self-play and find strategies that may exceed the existing human level. In addition, it also allows LsAc*-MJ to improve its generalization ability through extensive self-challenging.

In this process, the LsAc*-MJ model acts as the four players, engaging in mutual play at the same Mahjong table. Based on feedback from the game environment, the network parameters are updated, enhancing the accuracy of the model's decision-making through multiple iterations. By combining the training approach that integrates expert guidance and self-play learning, not only is training time saved, but the accuracy of decision-making is also improved. Currently, 40,000 rounds of training have been conducted on the RLCard platform, comprising 10,000 rounds of knowledge-guided training and 30,000 rounds of self-play.

5. Experiment and Analysis

5.1. Experimental Design. In this chapter, we designed some series of experiments in order to validate the level of the Mahjong game model LsAc*-MJ proposed in this paper. First, ablation experiments on the LSTM networks adopted by the LsAc*-MJ model were conducted to verify its effect on the model's gaming ability. Second, ablation experiments on the two-stage training method adopted by LsAc*-MJ were conducted to verify that it can shorten the training time. Finally, ablation experiments on the experience replay mechanism were performed to verify that the convergence of the model can be accelerated effectively. In addition, the LsAc*-MJ model was played against other reinforcement learning models including NFSP, DQN, and DuelingDQN to verify the high decision-making ability of the proposed model.

5.2. RLCard Platform. The testing platform for Mahjong AI utilizes the open-source game platform RLCard [39], developed by Texas A&M University for reinforcement learning in card games. This platform provides a testing environment for various popular card games, including Mahjong, along with some implemented algorithm instances. Researchers can integrate their own algorithms into this platform.

On the RLCard platform, Mahjong testing can automatically generate hand tiles, start games, and supervise game outcomes. We adjusted the encoding of game information returned from the platform and input it into the model for learning. The relevant parameters are set as follows: the batch size = 128, $\lambda = 0.99$, the discount factor $\gamma = 0.95$, and adopt Adam optimizer with LR-Actor = 0.0005, LR-Critic = 0.0001.

5.3. Specifications and Analysis on Four Players in the Experiments. In this work, each experiment compares only two algorithms for better results. In offline Mahjong

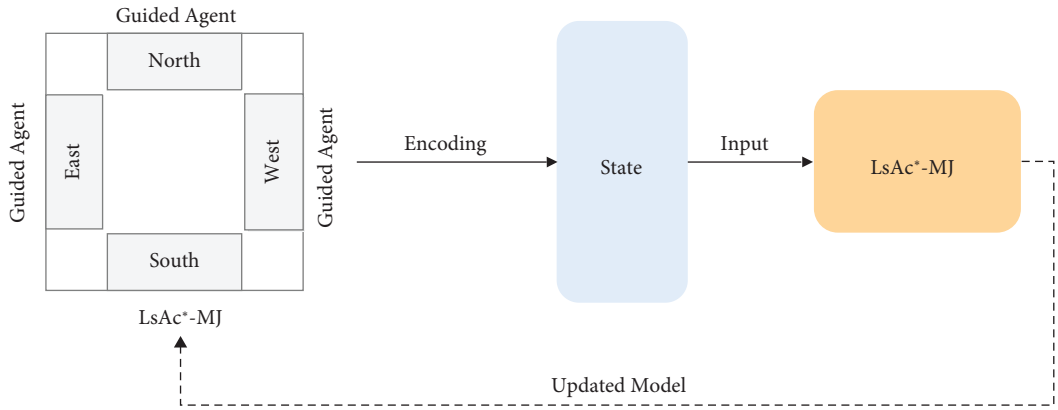

Figure 5: Knowledge-guided first stage training.

Table 3: Action priority of guided agent.

Priority	Action brief description	Example	Value
1	Discard dragons tile	Action = [27]	6
2	Discard single tile	State = [[0, 0, 1, 0, 0, ...], ...], action = [2]	3
2	Discard dead tile	When there are three nine-bots discarded action = [26]	3
2	Chow pong kong	Action = [34, 35, 36]	3
3	Other actions	Other	0
4	Divide two sequential tiles	State = [[0, 1, 1, 0, ...], ...], action = [1, 2]	−2
4	Divide sequence without middle tiles	State = [[1, 0, 1, 0, ...], ...], action = [0, 2]	−2
5	Divide pair	State = [[0, 0, 1, 0, ...], [0, 0, 1, 0, ...], ...], action = [2]	−4
5	Divide triplet	State = [[0, 0, 1, ...], [0, 0, 1, ...], [0, 0, 1, ...], ...], action = [2]	−4
5	Divide sequence	State = [[0, 1, 1, 1, 0, ...], ...], action = [2, 3, 4]	−4

games, it is easier to cheat due to more interaction with the left and right players of the table, such as the player on the left feeding the player on the right needed tiles. Therefore, in many Mahjong agents' gaming competitions, such as the Chinese Computer Gaming Championship, players from the same organization are not allowed to be adjacent to each other on the Mahjong table in order to avoid players assisting each other in cheating. In this experiment, the staggered seating setup method is also used, where we set the same algorithmic agents in nonadjacent positions on the Mahjong table. A seat rotation is performed after a number of games to ensure that the same algorithmic model participates in the experiment in every orientation.

The four directions of the mahjong table (east, west, south, and north) are represented by the numbers 1–4. Starting from the level of interaction and competition between players of different directions with other players in the game, we analyzed the effect of seating setup on the experiment. In the following, we prove the scientific validity of the experimental setup of staggered seating in this paper.

The four players are denoted as A, B, A',and B'. The seats are denoted as S_i. $O(S_i, S_j)$ denotes the total interaction degree that the model in the seat can obtain in a game, and $R(S_i, S_j)$ denotes the total competition degree that the model in the seat can obtain in a game. The interaction degree between different seats is shown in Figure 6.

For each agent, the chance of obtaining an interaction can be expressed as the following (17)–(20):

$$O(S_A) = O(S_A, S_B) + O(S_A, S_{B'}) + O(S_A, S_{A'}), \tag{17}$$

$$O(S_B) = O(S_B, S_A) + O(S_B, S_{B'}) + O(S_B, S_{A'}), \tag{18}$$

$$O(S_{A'}) = O(S_{A'}, S_B) + O(S_{A'}, S_{B'}) + O(S_{A'}, S_A), \tag{19}$$

$$O(S_{B'}) = O(S_{B'}, S_B) + O(S_{B'}, S_A) + O(S_{B'}, S_{A'}). \tag{20}$$

The interaction is the same for both parties, and thus, there is $O(S_i, S_j) = O(S_j, S_i)$, so in a game of Mahjong, there is (21)

$$O(S_A) + O(S_{A'}) = O(S_B) + O(S_{B'}). \tag{21}$$

In our experiments, A and A' use the same strategy, B and B' use the same strategy, so there are $O(S_A, S_B) = O(S_{A'}, S_{B'})$, $O(S_A, S_{B'}) = O(S_{A'}, S_B)$, $O(S_A, S_{A'}) = O(S_{A'}, S_A)$, and $O(S_B, S_{B'}) = O(S_{B'}, S_B)$. Extending this to a number of rounds shows that in a staggered setup, the interaction degree available to the agent is equal in all four directions, as in (22).

$$\frac{1}{N}\sum_{1}^{N} O(S_A) \approx \frac{1}{N}\sum_{1}^{N} O(S_B) \approx \frac{1}{N}\sum_{1}^{N} O(S_{A'}) \approx \frac{1}{N}\sum_{1}^{N} O(S_{B'}). \tag{22}$$

Similarly, it can be deduced that in the case of staggered settings, the same is true for the degree of competitiveness $R(S_i)$, as in (23).

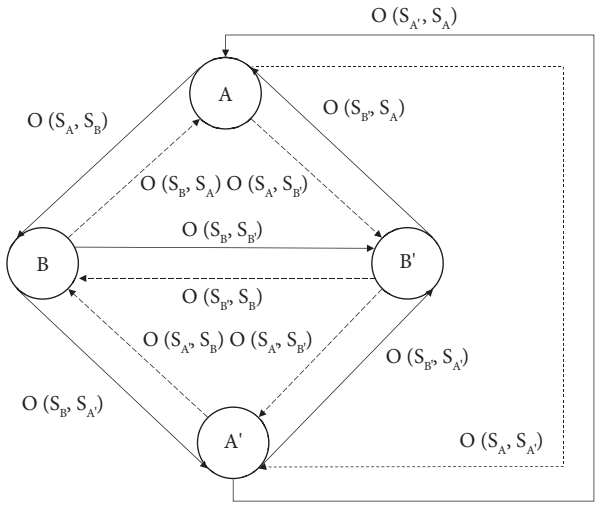

Figure 6: Topology of different seating interaction opportunities.

$$\frac{1}{N}\sum_{1}^{N} R(S_A) \approx \frac{1}{N}\sum_{1}^{N} R(S_B) \approx \frac{1}{N}\sum_{1}^{N} R(S_{A'}) \approx \frac{1}{N}\sum_{1}^{N} R(S_{B'}). \tag{23}$$

5.4. Ablation Experiments on the LsAc-MJ Model.* To verify the effectiveness of the empirical replay mechanism adopted by the LsAc*-MJ model, the LSTM network structure, and the two-stage training method on the improvement of the model convergence speed, training time, and game win-rate, this paper designs ablation experiments and trains three mahjong game models.

RL-Lstm: A reinforcement learning model without the experience replay mechanism. The other parts include the A2C algorithm, network structure, and training method consistent with the LsAc*-MJ model. It is used to compete with LsAc*-MJ to assess the impact of the experience replay mechanism on the model's competitive ability.

RL-zero: A reinforcement learning model trained using only self-play, with the other parts including the network structure, A2C algorithm, and experience replay, consistent with the LsAc*-MJ model. It is used to compete with LsAc*-MJ to evaluate the influence of expert knowledge-guided training on the model's decision accuracy and training time.

RL-cnn: A reinforcement learning model trained using the A2C model with a LeNet-5 network structure for the policy and value networks. The other parts include the A2C algorithm, experience replay, and training method consistent with the LsAc*-MJ model. It is used to compete with LsAc*-MJ to assess the impact of LSTM networks on the model's decision-making ability.

In addition, Random, a randomized card-out model provided by RLCard, was used as a common adversary, and the level of Random was kept consistent when the random seeds employed in the experiments were the same. The specific results and analysis of the ablation experiments are shown as follows.

5.4.1. Experiments and Analysis on Experience Replay

(1) Influence on Model Convergence. To assess the impact of the experience replay on model convergence, we compared the convergence of LsAc*-MJ with RL-Lstm, which does not utilize the experience replay. Both models were trained on a single GPU server. Figure 7 illustrates the convergence of the loss for the Actor-net, while Figure 8 shows the convergence of the loss for the Critic-net.

From Figure 7, it can be observed that the loss for the Actor-net of our model converge to the range of 0 to 5 after around 50,000 steps. In contrast, the RL-Lstm model, which does not utilize the experience replay, exhibits recurrent oscillations in the range of [40, 70] even after 200,000 steps.

From Figure 8, it can be observed that the loss values for the Critic-net of our model converge to the range of [0, 10] after around 50,000 steps. In contrast, the RL-Lstm model continues to exhibit concentrated oscillations in the range of [20, 50] even after 200,000 steps.

The experiments demonstrate that with the introduction of the experience replay, both the Actor-net and Critic-net exhibit a gradual reduction in loss, tending towards stability as the training epochs increase. In contrast, the model without experience replay shows significant oscillations in the loss curves without a clear convergence trend. This proves that the experience replay effectively addresses the challenge of convergence in the A2C model.

(2) Impact on Win-Round and Win-Rate. To assess the impact of the experience replay on model performance, we compared the cumulative win-round and win-rate of two models. We conducted games with the LsAc*-MJ and RL-Lstm separately against the model Random. In the first 1000 nondraw rounds, we recorded the average cumulative win-round for each model in the four positions. As shown in Figure 9, the LsAc*-MJ model performed the best in terms of cumulative win-round.

From Table 4, it is evident that our model has an average win-rate of 51.8% in positions 1–3, surpassing RL-Lstm by 1.9%. In positions 2–4, the average win-rate is 55.1%, surpassing RL-Lstm by 4.4%. Our model exhibits an average win-rate across all positions that is 3.15% higher than RL-Lstm. These results demonstrate that the Mahjong model with the experience replay achieves a higher win-rate.

The experimental outcomes validate that the experience replay weakens the correlation between Mahjong input features, enabling the model to converge rapidly and enhancing its win-rate, thereby improving decision-making capabilities.

5.4.2. Experiments and Analysis on LSTM Network Structures. To demonstrate the impact of the LSTM networks used in this paper on model performance, we

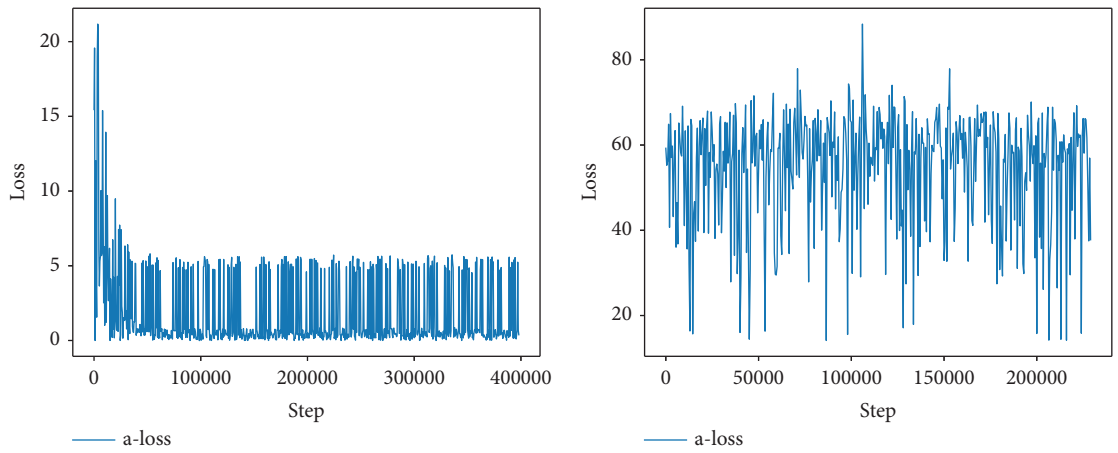

Figure 7: Actor net-loss over steps. (a) LsAc*-MJ. (b) RL-lstm.

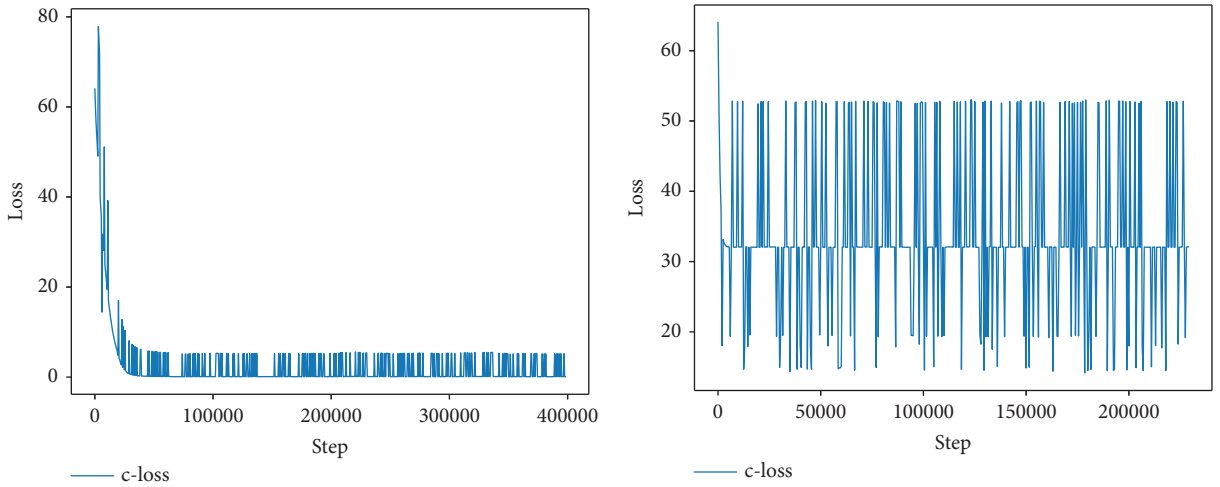

Figure 8: Critic net-loss over steps. (a) c. (b) RL-lstm.

compared the cumulative win-round and win-rate of LsAc*-MJ with RL-cnn, which employs a convolutional neural network. Each model played against the Random, and in the first 1000 nondraw rounds, we recorded the average cumulative win-round for each model in the four positions, as shown in Figure 10. LsAc*-MJ consistently achieved higher cumulative win-round per hundred rounds compared to the RL_cnn model. The average win-rate comparison between the two models is presented in Table 5, showing that LsAc*-MJ outperforms RL_cnn in each hundred rounds. In positions 1–3, LsAc*-MJ's win-rate surpasses RL_cnn by 4.4%, and in positions 2–4, it surpasses RL_cnn by 6.5%. The average win-rate across all positions is 5.45% higher for LsAc*-MJ compared to RL_cnn.

The experimental results demonstrate that the LSTM networks allow the LsAc*-MJ model to consider both the current state and the historical state in predicting tile discards. This utilization of discarded tile information and reward information in Mahjong results in higher cumulative win-round and average win-rate, enhancing the model's decision-making capabilities.

5.4.3. Experiments and Analysis of Knowledge-Guided Two-Stage Training. To demonstrate the influence of the training method advocated in this paper, we adopted the RL-zero for a comparative analysis. This model shares an identical network structure with LsAc*-MJ, incorporates the experience replay, but self-playing from an initial state. We conducted a comparison across cumulative win-round, win-rate, training time, and decision time, between LsAc*-MJ and RL-zero.

(1) Win-Round and Win-Rate Comparison. We conducted games with the LsAc*-MJ and RL-zero separately against the model Random. In the first 1000 nondraw rounds, we documented the average cumulative win-round for each model in the four positions, as illustrated in Figure 11. With

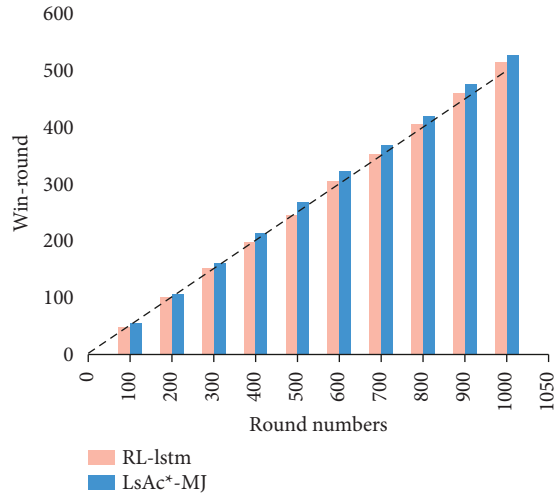

Figure 9: Win-round of RL_lstm or LsAc*-MJ (both vs. random).

Table 4: Win-rate comparison of LsAc∗-MJ or RL_lstm vs. random.

Model	Round									
	100	200	300	400	500	600	700	800	900	1000
LsAc*-MJ (1–3)	0.532	0.505	0.507	0.518	0.518	0.536	0.517	0.511	0.517	0.519
LsAc*-MJ (2–4)	0.562	0.576	0.576	0.546	0.557	0.541	0.537	0.539	0.541	0.536
RL_lstm (1–3)	0.482	0.497	0.503	0.492	0.485	0.506	0.502	0.503	0.507	0.509
RL_lstm (2–4)	0.494	0.507	0.507	0.500	0.495	0.509	0.510	0.514	0.517	0.521

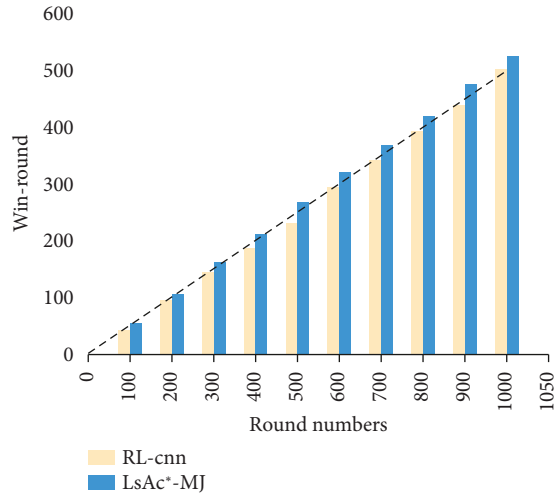

Figure 10: Win-round of LsAc*-MJ or RL_cnn (both vs. random).

Table 5: Win-rate comparison of LsAc∗-MJ or RL_cnn vs. random.

Model	Round									
	100	200	300	400	500	600	700	800	900	1000
LsAc*-MJ (1–3)	0.532	0.505	0.507	0.518	0.518	0.536	0.517	0.511	0.517	0.519
LsAc*-MJ (2–4)	0.562	0.576	0.576	0.546	0.557	0.541	0.537	0.539	0.541	0.536
RL_cnn(1–3)	0.427	0.481	0.484	0.457	0.454	0.479	0.487	0.487	0.486	0.502
RL_cnn(2–4)	0.445	0.487	0.482	0.480	0.479	0.502	0.491	0.497	0.493	0.504

an escalating number of rounds, the cumulative win-round of the knowledge-guided model LsAc*-MJ consistently exceeded those of the model RL-zero rained solely through self-play.

As shown in Table 6, the average win-rate per hundred rounds and the overall average win-rate of LsAc*-MJ consistently surpassed those of the RL_zero. The win-rate of the LsAc*-MJ exhibited an improvement of 2.35% compared to RL_zero, with a 2.4% increase in positions 1–3 and a 2.3% increase in positions 2–4. The experiments indicate that the two-stage training method with expert knowledge guidance leads to a stable enhancement in the performance of the model.

(2) Training Time Comparison. In comparison between LsAc*-MJ and RL_zero, as illustrated in Figure 12. Under the same network structure, the RL_zero required approximately 88.14 hours for convergence, while LsAc*-MJ achieved convergence in approximately 51.41 hours, resulting in a time saving of 41.7%. The experimental results indicate that the training method proposed in this paper significantly reduces the required training time.

(3) Decision Time Comparison. LsAc*-MJ and RL_zero were each pitted against the model Random in a series of 1000 game rounds, and the average decision time of per hundred rounds was recorded, as depicted in Figure 13. Across every hundred rounds, LsAc*-MJ required approximately 69.81 minutes, whereas RL_zero needed around 80.60 minutes, representing a 13.38% improvement. The two-stage training method resulted in a reduction in the average decision time, enhancing decision speed.

The above experiments demonstrate that the two-stage training method with expert knowledge guidance incurs shorter training times and yields superior training outcomes. The models trained using this approach exhibit faster decision-making speeds, which is particularly advantageous for competitive activities with stringent requirements on decision speed.

5.5. Performance Experiments on the LsAc-MJ Model.* To validate the decision-making capabilities and performance of LsAc*-MJ, experiments were conducted with three different deep reinforcement learning models: NFSP, DQN, and Dueling. A comparative analysis between the LsAc*-MJ model and the other three models was carried out based on metrics such as average win-rate, cumulative score, parameter size, and training time.

NFSP: A model trained using the NFSP algorithm (Neural Fictitious Self-Play) provided by RLCard, combining deep reinforcement learning with self-play. In RLCard, NFSP can be applied to various card games.

DQN: A model trained using the DQN algorithm (Deep *Q*-Network) provided by RLCard. The DQN algorithm utilizes a deep neural network to approximate and optimize the *Q*-value function for decision-making, and it introduces experience replay to enhance training stability and convergence speed.

Dueling: A reinforcement learning model trained using the DuelingDQN algorithm. The DuelingDQN algorithm builds upon DQN by using two networks to separately estimate state values and advantage functions, facilitating the learning and decomposition of state-action *Q*-values.

Two sets of comparisons were conducted for the experiment. First, the results of the game of LsAc*-MJ and the three reinforcement learning models against Random were compared. After that, the game results of LsAc*-MJ and the three reinforcement learning models were compared separately. The specific experimental results and analysis are shown as follows.

5.5.1. LsAc-MJ, NFSP, DQN and Dueling VS. Random.* We engaged the LsAc*-MJ, NFSP, DQN, and Dueling model in games against the Random model, respectively. In the first 1000 nondraw rounds, we calculated the average win-rate for each model in the four positions, as shown in Table 7. The average win-rate of LsAc*-MJ is 53.4%, surpassing DQN by 3.1%, Dueling by 3.5%, and outperforming NFSP by 26.0%. The experiments demonstrate that in the same gaming environment, our model achieves a higher average winning-rate.

5.5.2. LsAc-MJ VS. NFSP, DQN, and Dueling.* The experiments were divided into three groups. The first group consisted of two LsAc*-MJ players and two NFSP players, the second group had two LsAc*-MJ players and two DQN players, and the third group included two LsAc*-MJ players and two Dueling players. In the first 1000 nondraw rounds, the cumulative scores for each model in different positions are illustrated in Figures 14, 15, 16.

Figure 14 illustrates the score evolution of LsAc*-MJ in positions 1–3 and 2–4 against NFSP as the rounds progress. With increasing rounds, the score curve of the LsAc*-MJ model steadily rises, reaching a final cumulative score of 1784 points across all four positions. The average win-rate for each model in the four positions is presented in Table 8. The model achieved an average win-rate of 71.84% in positions 1–3, surpassing NFSP by 43.68%, and an average win-rate of 70.72% in positions 2–4, surpassing NFSP by 41.44%.

Figure 15 illustrates the score variation of LsAc*-MJ in positions 1–3 and 2–4 against DQN as the rounds progress. Despite fluctuations in the score curve, the LsAc*-MJ model eventually outperforms DQN with a slight advantage, achieving a final cumulative score of 128 points across all four positions.

The average win-rate for each model in the four positions was computed and is presented in Table 9. The model LsAc*-MJ achieved win-rate of 53.12% and 51.68% in positions 1–3 and 2–4, respectively, surpassing DQN by 6.25% and 3.35%.

Figure 16 illustrates the score evolution of LsAc*-MJ in positions 1–3 and 2–4 against Dueling as the rounds progress. Despite fluctuations in the score curve, our model consistently outperformed the opponent, achieving a final cumulative score of 240 points across all four positions. The

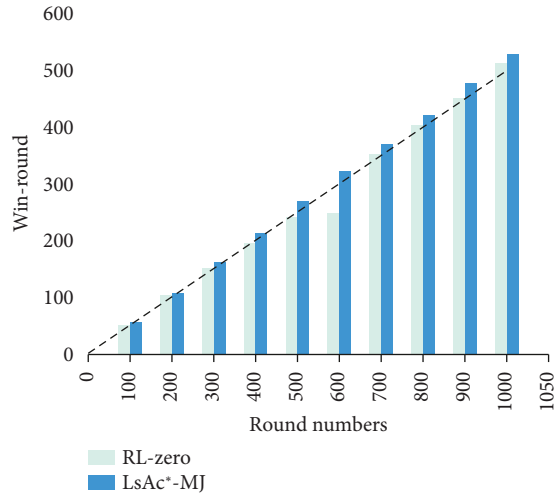

Figure 11: Win-round of LsAc*-MJ or RL_zero (both vs. random).

Table 6: Win-rate comparison of LsAc∗-MJ or RL_zero vs. random.

Model	Round									
	100	200	300	400	500	600	700	800	900	1000
LsAc*-MJ (1–3)	0.532	0.505	0.507	0.518	0.518	0.536	0.517	0.511	0.517	0.519
LsAc*-MJ (2–4)	0.562	0.576	0.576	0.546	0.557	0.541	0.537	0.539	0.541	0.536
RL_zero (1–3)	0.502	0.492	0.499	0.468	0.480	0.497	0.509	0.493	0.496	0.508
RL_zero (2–4)	0.587	0.536	0.531	0.523	0.511	0.513	0.516	0.524	0.514	0.523

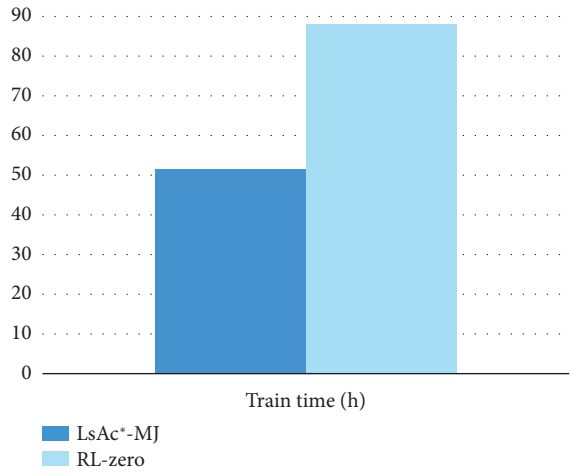

Figure 12: Trained time of RL_zero and LsAc*-MJ.

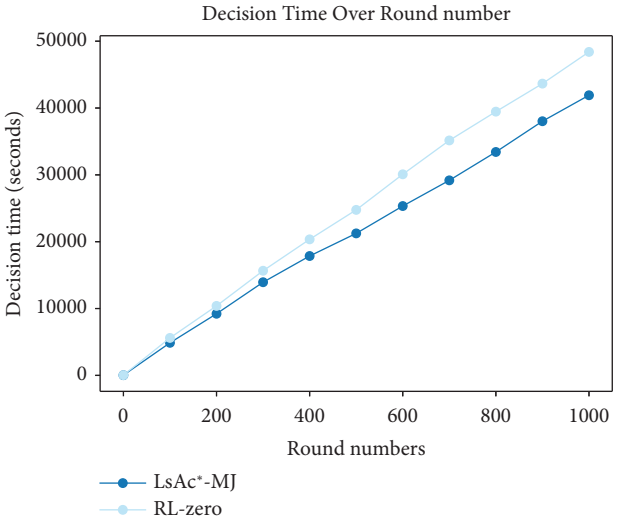

Figure 13: Decision time of RL_zero and LsAc*-MJ.

average win-rate for each model in the four positions was calculated and is presented in Table 10. In positions 1–3 and 2–4, the win-rates were 52.64% and 50.09%, surpassing Dueling by 5.58% and 1.81%.

During the experiments, we observed that both the model LsAc*-MJ and other models performed weaker in position 1 compared to positions 2 or 4. However, for the LsAc*-MJ, when placed in position 1 against the same opponent combination, the total score remained positive, surpassing the opponent's score in position 1 by over 50 points.

The experiments highlighted the significant advantages of the LsAc*-MJ in terms of average win-rate and cumulative scores. This proves that the proposed method in the paper can effectively enhance the decision-making capabilities of Mahjong models.

5.5.3. Network Parameter Size and Time Comparison. Table 11 provides a comparison of the neural network parameters and the time required for training for the LsAc*-

Table 7: Win-rate comparison of LsAc∗-MJ, NFSP, DQN, dueling vs. random.

Model	Round									
	100	200	300	400	500	600	700	800	900	1000
LsAc*-MJ	0.547	0.540	0.542	0.532	0.537	0.538	0.527	0.525	0.529	0.528
NFSP	0.301	0.269	0.273	0.270	0.251	0.265	0.275	0.277	0.281	0.283
DQN	0.505	0.491	0.507	0.505	0.502	0.516	0.505	0.499	0.499	0.500
Dueling	0.501	0.508	0.493	0.488	0.493	0.496	0.501	0.501	0.505	0.507

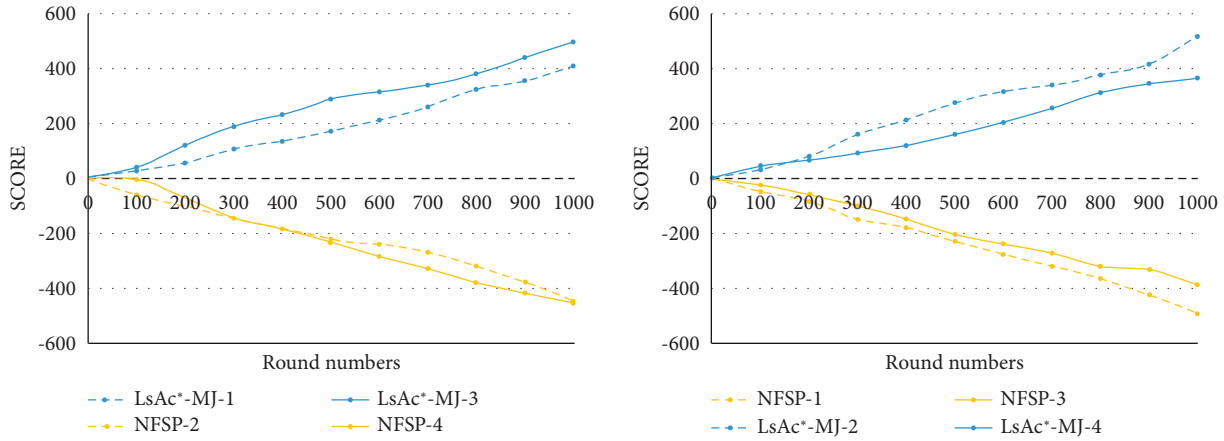

Figure 14: LsAc*-MJ vs. NFSP.

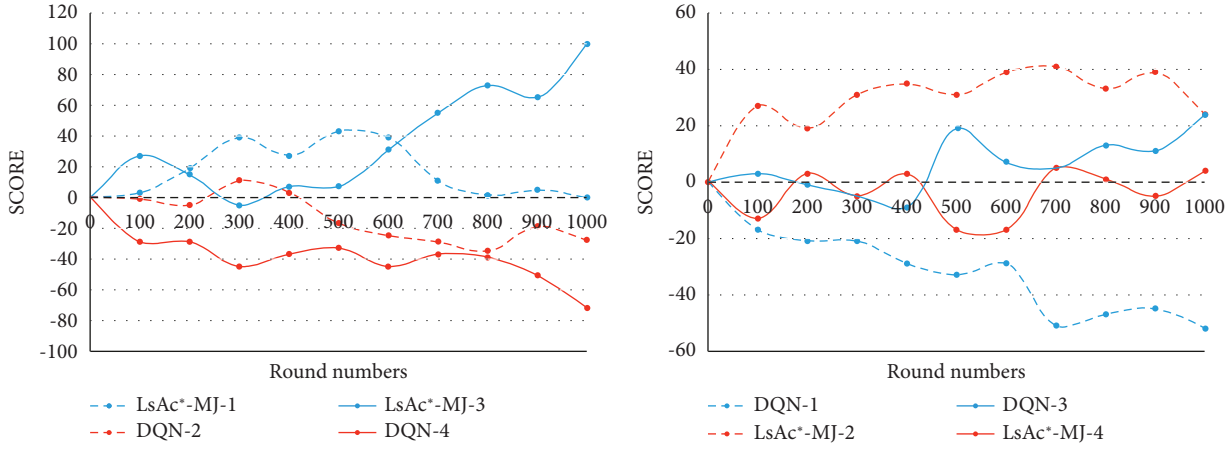

Figure 15: LsAc*-MJ vs. DQN.

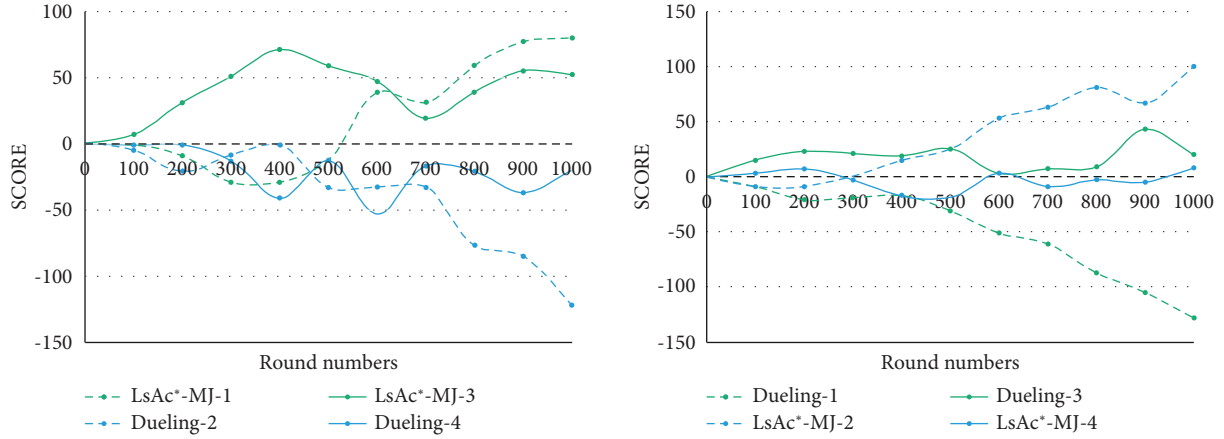

Figure 16: LsAc*-MJ vs. dueling.

Table 8: Win-rate comparison between LsAc*-MJ and NFSP.

Model	Round									
	100	200	300	400	500	600	700	800	900	1000
LsAc*-MJ (1–3)	0.665	0.718	0.745	0.729	0.729	0.719	0.714	0.719	0.721	0.726
LsAc*-MJ (2–4)	0.685	0.683	0.708	0.706	0.717	0.716	0.712	0.714	0.711	0.720
NFSP (1–3)	0.335	0.283	0.255	0.271	0.271	0.281	0.286	0.281	0.279	0.274
NFSP (2–4)	0.315	0.318	0.292	0.294	0.283	0.284	0.288	0.286	0.289	0.280

MJ, NFSP, DQN, and Dueling models over 40,000 rounds. Despite LsAc*-MJ having the largest parameter count, its scale is within the same order of magnitude as other models. Moreover, it achieves the shortest training time, saving 81.4% of time compared to the DQN that second to last ranked.

The above experiments demonstrate that our model not only enhances decision-making capabilities but also reduces training time, making it a resource-efficient and high-performance Mahjong model.

5.6. Discussion. The model proposed in the paper utilizes LSTM networks and an improved A2C algorithm, employing a two-stage training approach to achieve a Mahjong gaming model with higher training efficiency and more accurate decision-making. However, there are certain limitations in the model.

In addition to unidirectional LSTM networks, LSTM networks also have bidirectional LSTM networks. From the theoretical analysis, bidirectional LSTM may have good results in dealing with Mahjong. However, compared with bidirectional LSTM, unidirectional LSTM has higher computational efficiency and faster model convergence, which we think is more suitable for building models with low computational resource consumption. In addition, unidirectional LSTM is more consistent with human thinking logic and more interpretable than bidirectional LSTM. Therefore, we choose to use the unidirectional LSTM network to build the model.

In addition to LSTM networks, transformers are also commonly used structures at present. For models and input sequences of the same scale, LSTM usually has fewer parameters due to its relatively simple structure. Due to the existence of self-attention mechanism and feed forward neural network, the transformer requires more parameters to implement its self-attention mechanism and position encoding. Therefore, LSTM may have more advantages in limited resources. In the following work, we will fully consider the application prospects and value of transformer in Mahjong models.

At present, models that represent the top level of mahjong include Suphx developed by Microsoft and LuckyJ developed by Tencent AI Lab. We chose the model on the RLcard platform as the benchmark and did not choose the above model for the following reasons. Firstly, currently Suphx and Jueyi are both released on the Tenhou, but due to policy and network restrictions in our region, we are unable to access the Tenhou and conduct experiments with models on the platform. Secondly, reproducing Suphx based on published papers [1] poses great difficulties. The lack of publicly available datasets and significant computational resources make reproduction difficult to achieve, and the quality of reproduction cannot be guaranteed. Thirdly, papers related to Tencent's unique skills [2] have proposed new algorithms for single player Mahjong, but the method for four player mahjong has not been specifically disclosed, making it extremely difficult to reproduce. Fourthly, currently, only some papers using supervised learning [12, 13] have compared the accuracy of network prediction with Suphx. However, our model did not adopt supervised learning, so we cannot compare the network prediction accuracy with Suphx. Fifthly, the RLcard platform is an offline reinforcement learning platform designed for card games, and research results on the use of this platform in various literature studies include [17, 40, 41]. In addition to providing a Mahjong gaming environment, the RLcard platform also provides user-friendly interfaces and representative deep reinforcement learning methods such as NFSP and DQN. Therefore, we chose the RLcard platform model as the experimental benchmark.

The reward mechanism used in model training is simplistic, lacking sufficient consideration for the points that can be obtained, and lacks the setting of wind and rotation compared to reality. Unlike other card games, Mahjong can result in a draw, and not every game produces valid rewards. In experiments where different algorithm models from RLCard played against Random [17], the draw rate was above 90%. The statistics presented in the experiments focus on nondraw results.

In addition, when applying the proposed algorithm and training method to other Mahjong variants, the model should be modified for different rules and number of hands. First, the hand and action encoding is modified according to the number of tiles in a specific Mahjong variant. For example, if it is applied in popular Mahjong, the input dimension needs to be modified to $27 * 4$. Second, the division of the action values of the guiding agents during the training process needs to be modified according to the specific rules and trained again. For example, for the national standard mahjong special flower tiles that can increase the score and also participate in the draw, the rules for handling and utilizing the flower tiles need to be added. If the above modifications are not made, especially the adjustment of the bootstrapping training, the performance of the model will be lost. In August, 2023, we attempted to directly migrate the weights of Japanese Mahjong to Chinese popular Mahjong for game play and only won the second prize at the 2023 Chinese Computer Gaming Championship National Tournament.

Table 9: Win-rate comparison between LsAc*-MJ and DQN.

Model	Round									
	100	200	300	400	500	600	700	800	900	1000
LsAc*-MJ (1–3)	0.575	0.543	0.528	0.521	0.525	0.529	0.524	0.523	0.519	0.525
LsAc*-MJ (2–4)	0.535	0.528	0.522	0.524	0.507	0.509	0.516	0.511	0.509	0.507
DQN (1–3)	0.425	0.458	0.472	0.479	0.475	0.471	0.476	0.477	0.481	0.475
DQN (2–4)	0.465	0.473	0.478	0.476	0.493	0.491	0.484	0.489	0.491	0.493

Table 10: Win-rate comparison between LsAc*-MJ and dueling.

Model	Round									
	100	200	300	400	500	600	700	800	900	1000
LsAc*-MJ (1–3)	0.515	0.528	0.518	0.526	0.523	0.536	0.518	0.531	0.537	0.533
LsAc*-MJ (2–4)	0.485	0.498	0.498	0.499	0.503	0.520	0.519	0.524	0.517	0.527
Dueling (1–3)	0.485	0.473	0.482	0.474	0.477	0.464	0.482	0.469	0.463	0.467
Dueling (2–4)	0.515	0.503	0.502	0.501	0.497	0.480	0.481	0.476	0.483	0.473

Table 11: Comparison of parameters and training time of different models.

Model	Parameters	Training time (hours)
LsAc*-MJ	30.80×10^4	51.41
DQN	12.111×10^4	276.97
NFSP	6.05×10^4	355.37
Dueling	4.63×10^4	1105.21

6. Conclusions

The LsAc*-MJ model proposed in the paper uses LSTM networks to design a Mahjong model. Both the Actor-net and the Critic-net employ LSTM structures to fully use the temporal dependence in Mahjong games, leading to improved accuracy in action prediction and evaluation. The LsAc*-MJ also improves the A2C algorithm through the experience replay to weaken the correlation between Mahjong input data, thereby increasing model stability. This model adopts a two-stage training method to enhance decision accuracy and speed. Extensive experimental results demonstrate that, compared to Mahjong models constructed by other reinforcement learning algorithms, our proposed LsAc*-MJ model can rapidly train a Mahjong agent with higher cumulative scores, even with lower computational resources. In future work, we plan to gradually refine the reward mechanism to further improve the training efficiency of the model and enhance the overall performance of Mahjong models [42–45].

Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

We thank Professor Licheng Wu for his guidance on the topic selection in the early stage, and we have chosen to conduct research on Mahjong incomplete information games. This work was supported by the National Natural Science Foundation of China under Grant No. 62276285, the Key Projects of National Natural Science Foundation of China under Grant No. 62236011, and in part by the Major Project of National Social Sciences Foundation of China under Grant No. 20&ZD279.

References

[1] J. Li, S. Koyamada, Q. Ye et al., "Suphx: mastering mahjong with deep reinforcement learning," 2020, https://arxiv.org/abs/2003.13590.

[2] Tencent Ai Lab, "Decision-making AI new breakthrough, Tencent AI Lab LuckyJ top of the international mahjong platform," https://mp.weixin.qq.com/s/KF0nPfbPKJeRztZ1wujBHA.

[3] Y. Wu and Y. Tian, "Training agent for first-person shooter game with actor-critic curriculum learning," in *Proceedings of the International Conference On Learning Representations*, Toulon, France, April 2017.

[4] V. Mnih, A. P. Badia, M. Mirza et al., "Asynchronous methods for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, New York, NY, USA, June 2016.

[5] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," 2013, https://arxiv.org/abs/1312.5602.

[6] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The performance of LSTM and BiLSTM in forecasting time series," in *Proceedings of the 2019 IEEE International Conference on Big Data*, Los Angeles, CA, USA, December 2019.

[7] D. Lin and Y. Wu, "Research on artificial intelligence of Mahjong," in *The Master's Thesis of the Taiwan Jiaotong*, University, Taiwan, China, 2008.

[8] Z. Lin and Y. Wu, "Research on the artificial intelligence of mahjong," in *The Master's Thesis of the Taiwan Jiaotong*, University, Taiwan, China, 2010.

[9] C. Wu and S. Lin, "The design and implementation of mahjong program MahjongDaXia," in *The Master's Thesis of the Taiwan*, Normal University, Taiwan, China, 2016.

[10] M. Wang, H. Ren, W. Huang, T. Yan, J. Lei, and J. Wang, "An efficient AI-based method to play the Mahjong game with the knowledge and game-tree searching strategy," *ICGA Journal*, vol. 43, no. 1, pp. 2–25, 2021.

[11] D. Xu, in *Mahjong AI/Analyzer*, The Master's Thesis of the California State University Northridge, California, UK, 2015.

[12] S. Gao, F. Okuya, Y. Kawahara, and Y. Tsuruoka, "Supervised learning of imperfect information data in the game of mahjong via deep convolutional neural networks," in *Proceedings of the Information Processing Society of Japan*, Tokyo, Japan, November 2018.

[13] Y. Zheng, Y. Kazuichiro, Y. Ryouyang, and K. Hideyoshi, "Study on evaluation function design of mahjong using supervised learning," *SIG-SAI*, vol. 34, no. 5, pp. 1–9, 2019.

[14] M. Wang, T. Yan, M. Luo, and W. Huang, "A novel deep residual network-based incomplete information competition strategy for four-players Mahjong games," *Multimedia Tools and Applications*, vol. 78, no. 16, pp. 23443–23467, 2019.

[15] S. Gao and S. Li, "Bloody Mahjong playing strategy based on the integration of deep learning and XGBoost," *CAAI Transactions on Intelligence Technology*, vol. 7, no. 1, pp. 95–106, 2022.

[16] D. Han, T. Kozuno, X. Luo et al., "Variational oracle guiding for reinforcement learning," in *Proceedings Of the International Conference On Learning Representations, on Line*, Vancouver, Canada, May 2021.

[17] Y. Sun, in *Research on Expectimax Search-Based Algorithm for Incomplete Information Games*, the master's thesis of Beijing Jiaotong University, Beijing, China, 2021.

[18] J. Lei, J. Wang, T. Yan, and W. Huang, "Incomplete information game algorithm based on expectimax search and double DQN," *Computer Engineering*, vol. 47, no. 3, pp. 304–310+320, 2021.

[19] C. Zhao, B. Xiao, and L. Zha, "Incomplete information competition strategy based on improved asynchronous advantage actor critical model," in *Proceedings of the 2020 4th International Conference on Deep Learning Technologies*, Beijing, China, July 2020.

[20] D. M. Village, "NAGA: deep learning mahjong AI," https://dmv.nico/ja/articles/mahjong_ai_naga/.

[21] H. Sato, T. Shirakawa, A. Hagihara, and K. Maeda, "An analysis of play style of advanced mahjong players toward the implementation of strong AI player," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 32, no. 2, pp. 195–205, 2017.

[22] T. Truong, in *A Supervised Attention-Based Multiclass Classifier for Tile Discarding in Japanese Mahjong*, The Master's Thesis of the University of Agder, Grimstad and Kristiansand, Arendal, Agder, 2021.

[23] J. Line, Y. Xing, J. Sun et al., "Phoenix: an open-source, reproducible and interpretable mahjong agent," https://csci527-phoenix.github.io/documents.html.

[24] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," in *The Proceedings of the 31st Conference on Neural Information Processing Systems*, Long Beach, CA, USA, December 2017.

[25] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, pp. 1507–1063, 1999.

[26] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[27] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *The Proceedings of the International Conference on Machine Learning*, New York, NY, USA, June 2016.

[28] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *The Proceedings of the AAAI Conference on Artificial Intelligence*, Phoenix, AZ, USA, February 2016.

[29] B. Kim, J. Park, S. Park, and S. Kang, "Impedance learning for robotic contact tasks using natural actor-critic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics: A Publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 40, no. 2, pp. 433–443, 2010.

[30] R. Syam, K. Watanabe, and K. Izumi, "Adaptive actor-critic learning for the control of mobile robots by applying predictive models," *Soft Computing*, vol. 9, no. 11, pp. 835–845, 2005.

[31] C. Li, R. Lowe, and T. Ziemke, "Crawling posture learning in humanoid robots using a natural-actor-critic CPG architecture," in *The Proceedings Of the European Conference On Artificial Life*, Taormina, Italy, September 2013.

[32] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, Article ID 132306, 2020.

[33] G. Bellec, F. Scherr, A. Subramoney et al., "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature Communications*, vol. 11, no. 1, p. 3625, 2020.

[34] Y. Wu, M. Schuster, Z. Chen et al., "Google's neural machine translation system: bridging the gap between human and machine translation," 2016, https://arxiv.org/abs/1609.08144.

[35] M. Sundermeyer, R. Schluter, and H. Ney, "LSTM neural networks for language modeling," in *The Proceedings of the Thirteenth Annual Conference of the International Speech Communication Association*, Portland, OR, USA, September 2012.

[36] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *The Proceedings of the International Conference on Machine learning(PMLR)*, Beijing, China, June 2014.

[37] A. Graves, N. Jaitly, and A.-R. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *The Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, Europe, UK, December 2013.

[38] J. Y. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: deep networks for video classification," in *The Proceedings Of the*

IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, June 2015.

[39] D. Zha, K. Lai, S. Huang et al., "RLCard: a platform for reinforcement learning in card games," in *The Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, Yokohama, Japan, July 2020.

[40] D. Zha, J. Xie, W. Ma et al., "Douzero: mastering doudizhu with self-play deep reinforcement learning," in *The Proceedings of the International Conference on Machine Learning*, Lugano, Switzerland, July 2021.

[41] Y. Zhao, J. Zhao, X. Hu, W. Zhou, and H. Li, "Douzero+: improving doudizhu ai by opponent modeling and coach-guided learning," 2022, https://arxiv.org/abs/2204.02558.

[42] S. Li and X. Yan, "Let's play mahjong," 2019, https://arxiv.org/abs/1903.03294.

[43] L. Peng, H. Zhao, C. Gan, J. Liu, and J. Chen, "Research on winning method in computer mahjong games for the general public," *Journal of Chongqing University of Technology (Natural Science)*, vol. 35, no. 12, pp. 127–133, 2021.

[44] X. Yan, Y. Li, and S. Li, "A fast algorithm for computing the deficiency number of a mahjong hand," 2021, https://arxiv.org/abs/2108.06832.

[45] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," 2016, https://arxiv.org/abs/1603.01121.