Study on Evaluation Function Design of Mahjong using Supervised Learning

Yeqin Zheng*†‡1, Soichiro Yokoyama*, Tomohisa Yamashita* and Hidenori Kawamura*

*Graduate School of information Science and Technology, Hokkaido University

Abstract

The evaluation function for an imperfect information game is always hard to define but has a significant impact on the playing strength of a program. Deep learning has made great achievements in several recent years, and already exceeded the level of top human players in perfect information games such as AlphaGo. Predicting opponents moves and hidden states is important in imperfect information games. This paper describes a model on building a Mahjong artificial intelligence with deep learning method and supervised learning theory. Four deep neural network for discarding and predicting opponents' waiting, waiting tiles and point changes are combined into one model and performs good during games. With improved feature engineering, our accuracies on validation data of these networks reach higher than Dr. Mizukami and Professor Tsuruoka's network.

Keywords—Mahjong; Supervised learning; Imperfect information game

1 Introduction

Despite the great progress made in artificial intelligence (AI) for perfect information game, we are still far from being able to humanoid performed AI that can use strategy during imperfect information games, partly because imperfect information games have too much status and some of them will exceed machines calculation. Recently, deep learning method, which aims to solve complex problems with end-to-end deep model, has been drawing increasing attention. Game AI is a very challenging task and may be able to apply to real word problems.

Monte Carlo tree search (MCTS) is a heuristic search algorithm for some kind of decision process [1, 2]. The focus is on the analysis of the most promising moves, expanding the search tree based on random sampling of the search space. With combining with other algorithms or models, MCTS performs very well in many game AIs (e.g., AlphaGo [3]).

Status analysis in perfect information game is much less studied than that in imperfect information game, as well, one-player game is much less studied than multiplayer game. Most previous studies chose perfect information game and one-player as research object. In this study, a deep model analyzes status from an imperfect information multiplayer game, mahjong, and makes decisions of discard-tile during games.

Since a research about deep reinforcement learning applied to mahjong made by Naoki Mizukami and Yoshimasa Tsuruoka [4], a deep model performs good defensive during games, improvement of aggression and game strategy will be introduced base on their research in this paper.

In the next section, we relate our work to mahjong. We then describe our proposed approach in detail in Section 3. We show the experimental results in Section 4 and conclude the paper in Section 5.

2 Related Terminology of Mahjong

The basic game has 136 tiles, including 36 characters, 36 bamboos, and 36 circles, which are the suits. These are, in turn, divided into four sets of numbers 1 to 9 in each suit. There are also 16 wind tiles and 12 dragon tiles. Many sets also include eight bonus tiles with four flowers and four seasons, but these are not needed in the basic game.

We describe that there are two states during games. One is that there are players in waiting state during games, and we called this *someone is in waiting*. Another one is that *no one is in waiting*.

Waiting/Tenpai One or more players have made winning hands and waiting for the last tiles to earn score.

N shanten After n effective tiles drawn into hands by player will make hands into winning hands and enter waiting state.

Attack route Discard a tile to make hands closer to winning hands and earn score, which may

*Graduate School of information Science and Technology, Hokkaido University

†Kita 14, Nishi 9, Kita-ku, Sapporo

‡E-mail: gyoukin_tei@complex.ist.hokudai.ac.jp

lead to a decrease in number of shanten.

Defence/fold route Discard a tile that has less danger of losing score and give up to win, which may make hands away from winning hands and lead to an increase in number of shanten.

Aggressive move Player choose a tile that may decrease the number of shanten and unsafe for current game state, also may lead to a decrease in players score because other players may have entered waiting states and waiting for this tile.

3 Model Details

In this section, we describe the feature engineering we used, how the *waiting or not*, *discard tile*, *waiting tiles* and *lose point* network learn to make decision base on the current state on table.

As we have said, our model is built with the idea from Dr. Naoki Mizukami and Pro. Yoshimasa Tsuruoka's model. We made four networks for this model and all of the model have the structure like Figure 1.

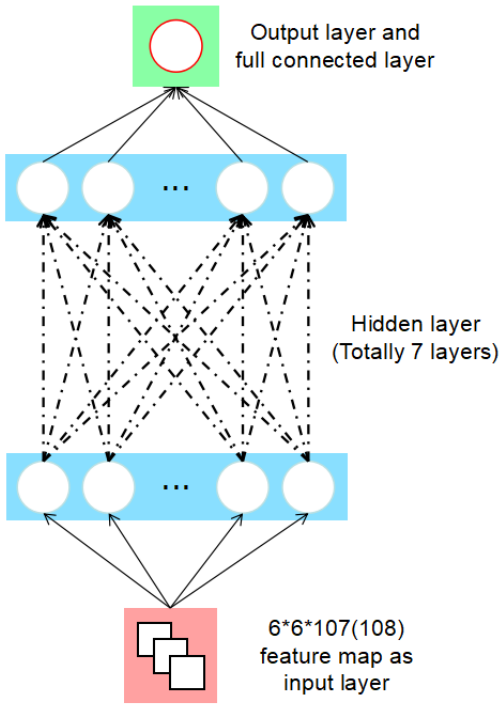

Figure 1: Structure of neural network

Although some details of the neural networks are different, they have the similar learning method.

The training data in this study is collected from a internet Mahjong site calledTenhou.net. We use games records in the highest level game "Houou table" at this site as training data. Only the top 0.1% players are allowed to play in Houou table which makes the quality of these games record guaranteed.

2p	3p	8p	9p	Tou	Nan
	4p	7p	1m	Sya	Pei
6p	5p	2m	1s	Hak	Hat
2s	3s	3m	4m	9s	Cyu
4s	5s		5m	7m	1p
6s	7s	8s	6m	8m	9m

Figure 2: Base of tile matrix

3.1 Feature engineering: State of game

Since matrix performs better for convolutional neural network (CNN), we model each non-repeating tile into a vector space, and make a 6 * 6 matrix base on the vector space. Every node in the matrix represents a tile (Figure 2). This method is more used in natural language processing [5, 6], which uses the distance between two vectors to represent the similarity or relevance of two words [7]. Also some studies use it for recommendation system [8]. These studies show that this method can improve the effectiveness and results for training network. We train the tile space with the data from tiles in final hands from over 3.2 million games.

We use Countinuous Bag-of-Words method to train the tiles space [9–11]. The result shows that the more frequently tiles appear together, the distance between their vectors will get closer. The training data set we use to train this tiles space consists of tiles in final winning hands which from over 524,000 games.

We make a feature map to describe the states during games with 7 types, tiles in hands include both close hands and melds, tiles in river, tiles discards by every player in every turns, dora tiles (in some rules), invisible tiles in walls and others close hands, tiles in own close hands which can be discarded, a tile to be discarded (only used in lose point network), totally 108 layers (Table. 1).

Features	Number of features
Hands	4
Rivers	4
Turns	96
Dora	1
Invisible tiles	1
Close hand	1
Tile to be discarded	1
Totally	108

Table 1: Feature map

3.2 Waiting or not network: prediction of waiting

We will describe how to train a model to predict whether other players are waiting or not in this subsection.

In Japanese mahjong, we can only speculate whether someone is waiting from his open hands and tiles river if he doesn't make a call of richi. So if we want to avoid losing point as much as possible, we need to make a prediction on whether someone is waiting during games, which is also an important part for our model.

We collected about 300,000 data of games states. Half of them with some players in waiting and half of them without anyone is in waiting. And we make them into 6*6*107 feature maps through feature engineering above as input to the waiting-or-not neural network.

The full connected layer for this network is designed as Table 2.

Number of convolutional kernels	Activation function
512	relu
256	relu
1	relu

Table 2: Full connected layer of waiting-or-not neural network

The final output of the waiting-or-not network is a probability about whether other players is in waiting or not.

3.3 Discard network: Prediction of discarding

In this section, we will describe a neural network about imitating professional mahjong players' movements during.

We also use a non-deep learning mahjong AI for experiment which we called it best choice algorithm. This algorithm will choose a tile which can make hands closer to winning hands during games. Also it will defence if other player make a call of richi or already open hands with over three melds, and this movement may bring hands away from winning hands. We will make a comparison between this algorithm and the discard-tile network in the training result section from several aspects.

We hope that the discard network can act as a professional mahjong player when playing, so we randomly select 3,400,000 movements from won players' hands during games. There are 100,000 data for each tile in the training data set. The training data was made into 6*6*107 feature maps through feature engineering in order to train the network.

The full connected layer of the discard network is designed as Table 3.

Number of convolutional kernels	Activation function
128	relu
34	softmax

Table 3: Full connected layer of discard network

Output of the discard network is a list of probability which are 34 tiles' probabilities of being discarded base on current game states.

3.4 Waiting tiles network: prediction of tiles be waited

In other to avoid losing point during games, player should have the ability of speculating others' state. If there is someone being speculated as in waiting, discarding a safe tile is a better for defense. We will describe a network to predict the dangerous for 34 tiles in this section, which we wish the model can performs good defensive during playing.

Training data for waiting-tiles network is collected from games record which with some players are in waiting state. There are 136,000 data for training this network and 4,000 data for each tile are included. The feature engineering will generate 6*6*107 feature maps for these data as input for the neural network. Then we use a python library named "Mahjong" to calculate the waiting tiles of these hands as output.

The full connected layer of the discard network is designed as Table 4.

Number of convolutional kernels	Activation function
128	relu
34	softmax

Table 4: Full connected layer of waiting-tiles network

The waiting-tiles network performs an output with a list of 34 probabilities about how dangerous 34 tiles are bases on the current games state.

3.5 Lose point network: prediction of point changing

Keeping losing the lest point is helpful to avoid falling to the last in a game. Professional Mahjong players also need to calculate the point changes base on others' river and their open hands during games in order to avoid from the unfavorable situation. So we describe a neural network which can predict the losing points base on the current games states.

We collected training data from games records when players discard a tile which lead to a decrease in points. Point changes during games are affected by the following conditions:

1) Which player is in dealer turn.
2) How much times a round continues.

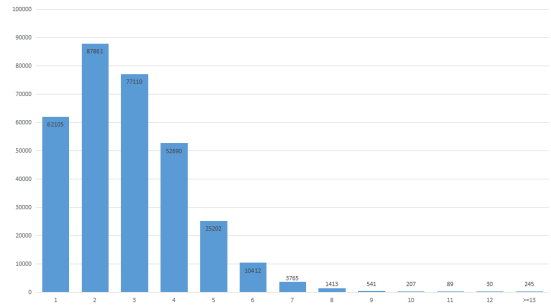

Figure 3: Records of each number of han

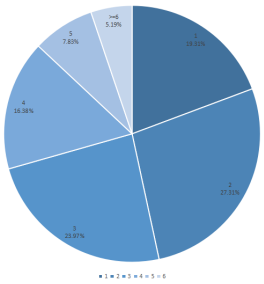

Figure 4: Records of each number of han

3) The yaku of hands.
4) The number of fu in hands.
5) The number of han in hands.

Because of the effects above, we decide to let the lose-point network predict the number of han rather than the point directly. We collected about 322,000 games records and calculate the frequency of each han that appeared in these records. The recorded statistics is shown in the Figure 3. And the frequency graph (Figure 4) shows that the number of han which greater or equals to 6 only takes 5.19% of the games records, which leads to a problem called unbalanced category for training data. This problem will make a classifier invalid. So we decide to divided the result into 6 classes, which includes 1 to 5 han and greater or equal to 6 han.

After defining the classifier, we randomly selected 16,500 data from each class and use them to generate 6*6*108 feature maps through feature engineering. The feature maps for the lose-point network have one more layer, which to record a tile may to be discarded, than other networks so that we can calculate lose point expect for every tile in hands. And we describe the full connected layer for this neural network as Table 5.

Number of convolutional kernels	Activation function
128	relu
6	softmax

Table 5: Full connected layer of lose-point network

The output of this neural network is a list that consists of 6 probabilities about how many han in other's hand if he wins this round.

3.6 Structure of model: make a decision of movement

In this section, we will combine the neural networks we described above into a model. And we will explain how this model make a decision of movement when playing.

We divided the movements during game into two types, attack mode and defense mode. When in the attack route, player makes hands closer to a winning hands and the number of shanten may become less and less until waiting when the number of shanten is 0. On the other hands, when in the defense route, player discards tiles to keep safe from losing point, which may lead to an increase in the number of shanten because the tiles discarded may be necessary for a winning hands.

The process from start to end in a round is shown in Figure 5. When a round start, the system will make a win check first and make a call of win if have a winning hands automatically. If haven't won, the model will make a decision of which tile to be discarded. After that, a richi check will be made by the system. If the left tiles, which is being waited, left over 3 pieces, our model will make a call of richi automatically. We calculate this threshold with data from games states. These data choose from the states when someone is waiting but his hand has no yaku and only richi can be made to earn points, and he chooses not to make a call of richi. We calculate the left tiles he is waiting for at that situation and get a average which is 2.61, so we choose 3 as a threshold to decide whether to make a call of richi or not.

The details of model making a decision is shown in Figure 6. Outputs from neural networks is also shown in Table 6.

Neural network	Output
Waiting-or-not network	WR
Discard network	DR
Waiting-tiles network	WTR
Lose-point network	LP

Table 6: Neural network and output

After the win check, we first use the output from the waiting-or-not network, which means the probability of whether others are waiting or not, to decide whether to choose a attack mode movement or a defense mode movement. A threshold will be used to help to make this decision. We collect the data of games states when there is player in waiting but no one make a call of richi. Then we use the waiting-or-not network to calculate the probability for these games states and calculate the average which is 0.245. If the output from waiting-or-not

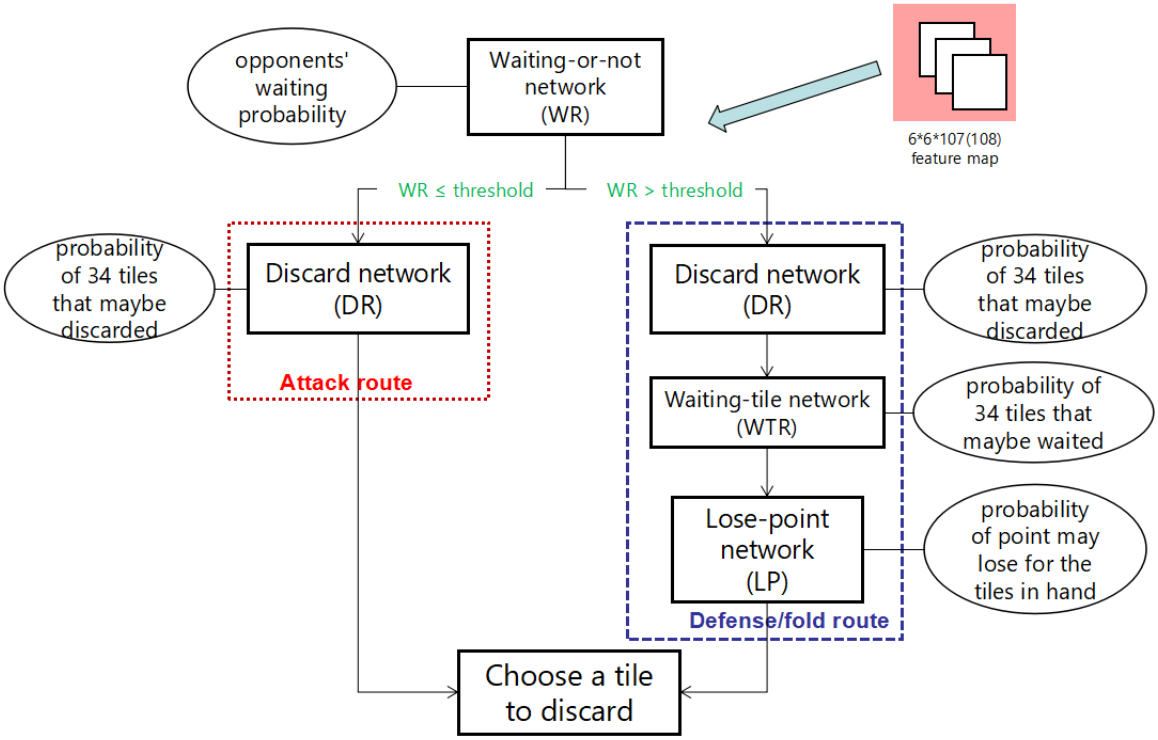

Figure 6: Model details

network larger than this threshold, the model will choose the defense mode route, otherwise, the model will choose the the attack mode route.

Then the output from the discard network, which include 34 probabilities about 34 tiles may to be discarded as a professional Mahjong player, will be calculated. Although the output include the probabilities of 34 tiles, we only choose the tiles which in hands for the next action. If the attack mode route is chosen, the output of model will be the maximum probability from the chosen output.

If the defense mode route is chosen, discard network will also performs an output. Then the waiting-tile network and the lose-point network perform the outputs and choose the output which in hands. Then we choose the minimum of the less point may lost (lose point expectation, LPE) as the output of model with the following method:

$$LPE_i = WR * DR_i * WTR_i * LP_i \quad , \qquad (1)$$

where i represent the tiles' ID in hand.

We also make experiments with another calculation which is

$$LPE_i = WR * DR_i * LP_i \quad , \qquad (2)$$

where i represent the tiles' ID in hand. But the model use this calculation shows less aggressive during games, we will show a comparison between them in the section 5.

The total process of model making a decision can be summarized as following:

$$\begin{cases} Tile_j \ and \ DR_j = Max[DR_1, DR_2, ...DR_i], \\ \quad if \quad WR \leq threshold \\ Tile_j \ and \ LPE_j = Min[LPE_1, LPE_2, ...LPE_i], \\ \quad if \quad WR > threshold \end{cases} \qquad (3)$$

where i and j represent the tiles' ID in hand.

During the games, system will send a send a win check if someone play a tile which you are waiting. Our model will return true if it receives this message.

Also, a meld check will be receives if we can make a meld during rounds. In this situation, our model will use the methods from the best choice algorithm we mentioned above to decide whether to call a meld or not. The algorithm makes a call decision with the following judgement methods:

1) If there are 2 same wind tiles, which is the wind of player's round, in hands and the last wind tile has just been discarded, it will make a call of wind meld.

2) If there are 2 same dragon tiles in hands and the last same dragon tile has just been discarded, it will make a call of dragon wind meld.

3) Calculate the difference between hands and the yaku which can be composed of open hands. If the total predicted yaku greater than a threshold, it will make a call of meld. If in turn of own dealer, the

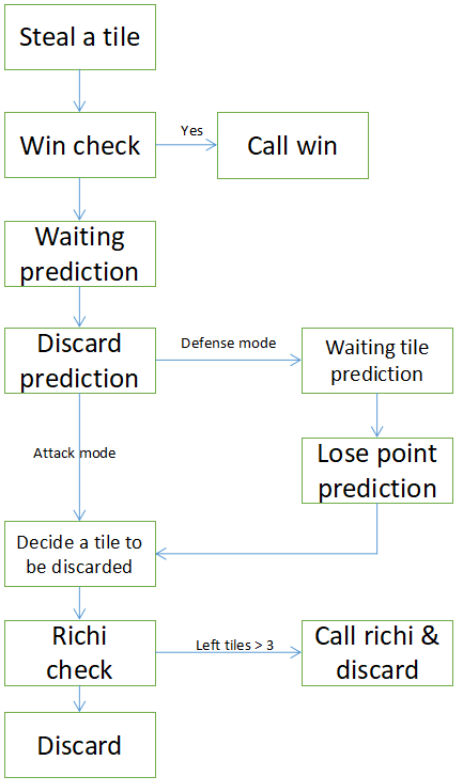

Figure 5: Process during a round

threshold will be 3. And in others' dealer turns, the threshold will be 4.

4 Training networks and results

We will show the training process and final accuracy of each neural network in this section. All the neural networks use cross validation method during training and we set 70% of the data set to training data and the left 30% to validation data.

4.1 Waiting or not network

The final output of waiting-or-not network is a probability about whether other players is waiting and we use mean square error to record the training process (Figure 7). The horizontal axis is epochs and the vertical axis is mean square error of validation data.

Finally it turns out with the accuracy of 82.7% in predicting other players' state.

4.2 Discard network

The process of training discard network is shown in Figure 8. The horizontal axis is epochs and the vertical axis is accuracy of validation data. The final accuracy after training is 88.4%.

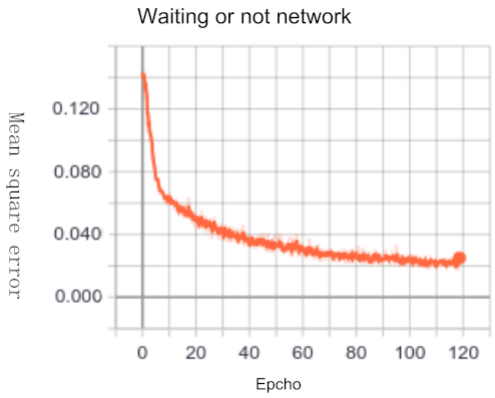

Figure 7: Training process of waiting-or-not network

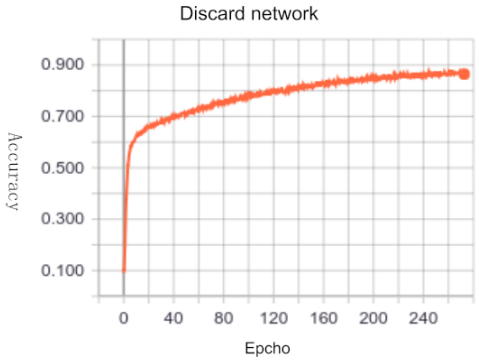

Figure 8: Training process of discard network

4.3 Waiting tiles network

The graph for the training process of waiting-tile network is shown as following (Figure 9). The horizontal axis is epochs and the vertical axis is accuracy of validation data set. The waiting-tiles network shows a accuracy of 40.2% in predicting the danger tiles.

4.4 Lose point network

Lose point network show a good prediction for point changes which is 88.7%. The process of neural training is shown in Figure 10. The horizontal axis is epochs and the vertical axis is accuracy of validation data set.

5 Experiment

As we described above, we designed several model for our experiment and make a comparison between them. We will describe the performance of these model and their strengths and weaknesses.

There are three AIs involved in our experiment. The first one is a non-deep learning Mahjong AI. We

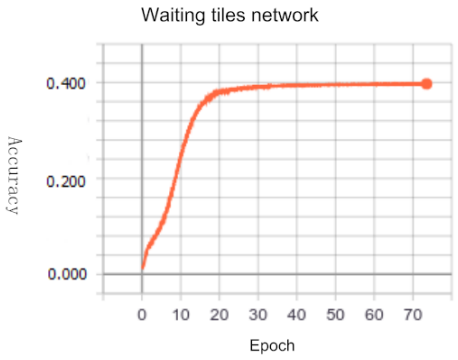

Figure 9: Training process of waiting-tiles network

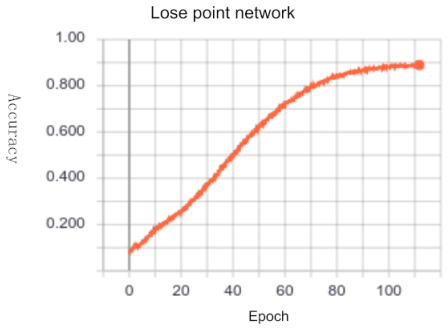

Figure 10: Training process of lose-point network

called this AI as best choice algorithm (BCA) because it will discard a tile only focus on own hands if it don't want to defense. This AI also have defense mode and it will be turned to only when someone else make a call of richi or open hands with over 3 melds. In the defense mode, it will discard a tile that opponent, who may in waiting, has discarded. If lack of these tiles, it will turn to attack mode again and choose a tile discarded to closer to winning hands.

The second one is designed base on the first one. We use the first AI as attack mode. Then we combine waiting-or-not network, waiting-tiles network and lose-point network into defense mode for this AI.

And the their one is our model which when have described its details in section 3. And then, we evaluated these three AIs on tenhou.net. Due to the limitation of level and rank, we did the evaluation on the "Ippan" table and "Joukyuu" table. And we will compare their performance in games. Each game lasts about 40 minutes.

5.1 Evaluation on "Ippan" table

60 games played by each AI on "Ippan" table at tenhou.net and the statistical data is shown in Table 7 and the comparison in Figure11. The opponents' average level is 1.5. Players in every level can participate the games, which make it possible to play with level 0 (sinzin) and level 20 (10 dan). During our experiments, the highest level player we met is level 17 (8 dan) and 53% of the opponents are level 0.

	First	Second	Third	Fourth
BCA	16	18	15	11
BCA + defense mode	10	17	27	6
Our model	13	16	20	11

Table 7: Games record on "Ippan" table

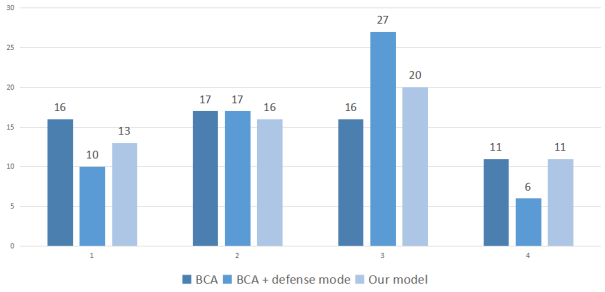

Figure 11: Comparison of games records on "Ippan" table

We calculate the win rate in the following Table 8. At the same time, we use a value, which is named attack when others may in waiting (AOW), to describe the aggressive of an AI during games. We statistic movements which decrease the number of shanten but the waiting-or-not network predict someone was in waiting. In general, if the output of waiting-or-not network shows that someone is in waiting, a defense movement doesn't likely to cause a decrease in the number of shanten but may lead to an increase. The calculation of AOW is:

$$AOW = \frac{Attack\,while\,WR \geq threshold}{Game\,state\,while\,WR \geq threshold} \quad (4)$$

we use the data, which discarded a tile and decreased the number of shanten, for this calculation.

We also calculate the human average with data from tenhou.net, and the human's data includes data from level 0 players to level 14 players because 98% of players we met on "Ippan" table are in this level range.

During the experiment, we found that on"Ippan" tables, most players choose to attack especially quick attack with making calls of melds. It's very normal to see a player open hands with three melds. But the training data we used to train our network is from expert player and it's very unusual to see that in expert's games. Tenhou.net publics a melds rate which shows the melds rate in low level (from level 0 to level 12) players is over 30.00% while it's only 25.30% for expert.

	1st	2nd	3rd	4th	Win rate	Feed rate	AOW
BCA	27%	30%	25%	18%	24%	11%	14%
BCA + defense mode	17%	28%	45%	10%	18%	8%	0%
Our model	22%	27%	33%	18%	20%	9%	8%
Human's average	20%	23%	27%	30%	20%	19%	-

Table 8: Comparison between AIs and human average

	1st	2nd	3rd	4th	Win rate	Feed rate	AOW
BCA	19%	23%	30%	28%	16%	18%	12%
BCA + defense mode	22%	28%	33%	17%	17%	8%	1%
Our model	24%	29%	27%	20%	21%	11%	7%
Human's average	25%	25%	25%	25%	23%	15%	17%

Table 9: Comparison between AIs and human average

With the situation above, the games states are very easy to be predicted that someone is in waiting. The AIs with a waiting-or-not prediction network showed low win rates because they were in defense mode always. But in the meantime, they showed a good performance of defensive. Almost half of human's average in feed rate. We said that players on "Ippan" table like to make calls of melds, and this also means it's difficult for them in defensive. The feed rate of low level players is about 19.40% while it's only 13.70% for experts. But we can see that the price of high defense is low aggressive which due to a lot of 3rd in records.

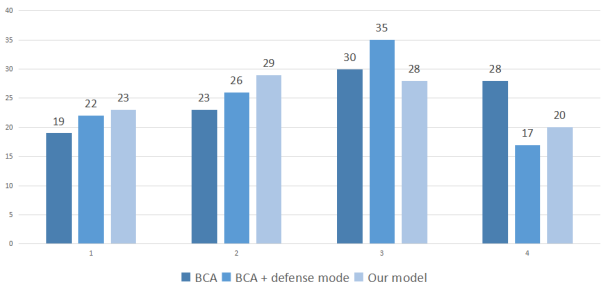

Figure 12: Comparison of games records on "Joukyuu" table

5.2 Evaluation on "Joukyuu" table

We product 100 games for each AI on "Joukyuu" table. The results records is shown in Table 10. The average level of opponents during experiment on "Joukyuu" table is 11.7. Only players who level higher than level 9 (1 kyuu) can participate in games on "Joukyuu" table. During our experiments, the highest level player we met is level 17 (6 dan). And we also make a comparison on these games record shown in Figure 12.

	First	Second	Third	Fourth
BCA	19	23	30	28
BCA + defense mode	22	28	33	17
Our model	24	29	27	20

Table 10: Games records on "Joukyuu" table

The win rate is shown in Table 9 following. The human's average is collect from tenhou.net with player level range from level 9 to level 17. Due to the tenhou.net doesn't public the games records on "Joukyuu" table, we use the games records on "houou" table to calculate the AOW as following. And we only met 3 times during these 300 games that a player get off-line during playing.

The melds rate for players in this level range is about 27.03%, which means games on "Joukyuu" table will be slower than games on "Ippan" table. The win rate and feed rate in table shows that human player in this level range can make a balance between attack and defense. In this level range, AIs with deep model performs more like a human player than non-deep learning AI in attack which we can get from the top rate and win rate. The reaaon that BCA's top rate and win rate become very low on "Joukyuu" table we believe it's the discard method. We statistical the games states of BCA's palying and discard network's playing in Table 11 and use our waiting-or-not network and waiting-tiles network to make prediction for them. The table shows that its's easier to be speculate the non-deep learning AI's state and what tiles it's waiting for. This result also means that the opponents of BCA algorithm will easy to defend and it's hard to get a tile from others' hands which it's waiting for. We think this is because the discard method of BCA lacks of strategic.

Then we make a comparison on AIs with deep model. We can find that AI without discard network shows a low feed rate and last rate, and it

Discard method	Waiting	Waiting rate	Accuracy(waiting-or-not network)	Accuracy(waiting-tiles network)
BCA	438	53.94%	91.32%	57.53%
Discard network	411	49.58%	83.43%	39.90%

Table 11: Comparison between BCA and discard network

meas this AI performs a good defense during game with human players. But at the same time ,it shows a low top rate and win rate, also with a high 3rd rate, which means this AI choose to fold in most of time. Although it may not lose point directly to others, it's difficult for it to earn point without attack. We think this AI lacks of aggressive during playing.

6 Conclusion

In this paper, we have proposed a model to imitate expert players on imperfect information games with supervised learning theory and deep learning method. Our model performs a human-like movement during games with good defensive during these games on tenhou.net although they haven't get higher top rate than human average.

In the future work, we plan to add other network to the model to judge when to make a call of melds, when to make a call of richi and the impact of global scores. Because human players may be more aggressive when his score lags behind his opponent and more inclined to defend when he leads. It will do help to make a more human-like model.

References

[1] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1): 1–43, 2012.

[2] Max Magnuson. Monte carlo tree search and its applications. *Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal*, 2(2):4, 2015.

[3] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

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

[5] Phil Blunsom, Edward Grefenstette, and Karl Moritz Hermann. New directions in vector space models of meaning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational , 2014.

[6] Xingyuan Peng, Dengfeng Ke, Zhenbiao Chen, and Bo Xu. Automated chinese essay scoring using vector space models. In *Universal Communication Symposium (IUCS), 2010 4th International*, pages 149–153. IEEE, 2010.

[7] Asif Ekbal, Sriparna Saha, and Gaurav Choudhary. Plagiarism detection in text using vector space model. In *Hybrid Intelligent Systems (HIS), 2012 12th International Conference on*, pages 366–371. IEEE, 2012.

[8] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. Deep reinforcement learning based recommendation with explicit user-item interactions modeling. *arXiv preprint arXiv:1810.12027*, 2018.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[11] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.