Supervised Learning of Imperfect Information Data in the Game of Mahjong via Deep Convolutional Neural Networks

Shiqi Gao[1] Fuminori Okuya[1] Yoshihiro Kawahara[1] Yoshimasa Tsuruoka[2]

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. Most of the studies on the game of Mahjong utilize concrete game rules and knowledge to combine with their own algorithm strategies. There are also several attempts using deep learning methods these years, with the benefit of automatic feature extraction, but their results are not satisfactory so far. In this paper, we propose a new three-dimensional data model to represent the information available on the game table, and construct a well-designed convolutional network for training. We choose the accuracy of tile discarding which is also called the agreement rate as the benchmark for this study. Our accuracy on test data reaches 68.8%, which is 6.7% higher than the previous best accuracy of 62.1% reported by Mizukami *et al.* (2015), and significantly higher than that of a previous CNN model, and our proposal uses no knowledge about concrete rules of mahjong for strategy design. A strength evaluation is also made on a popular online mahjong site 'Tenhou' by dealing with action strategies in a similar way, reaching a rating of 1822 after 76 matches.

1. Introduction

Deep learning has made remarkable achievements in various industrial and academic fields such as Computer Vision (CV), Natural Language Processing (NLP) and Reinforcement Learning (RL). The most popular and successful network structures used in deep learning include Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM).

The researches on Artificial Intelligence (AI) players on perfect information games are already highly advanced and the strength of AI has reached the level that no human beings can reach even in the game of Go, which is recognized as one of the most complex perfect information games, by the AI AlphaGo series in the year of 2017. However, for imperfect information games, there is still a long way to go. It is still very difficult for AI to find the best strategy when faced with situations with imperfect information. In this paper, we choose mahjong as our research tool due to its complexity and popularity. Japanese Mahjong has a strict frame of rules for competitions, together with many high-level past game records which are called 'haifu' for the network to study. Agreement rate, the rate of the network choosing the same strategy with the real data records, is a very important benchmark for estimating the situation of the game which is also called the evaluation function. There are many traditional AI methods for the game of Mahjong, but they are mainly built by artificially extracting features and designing function blocks. By contrast, deep learning is recognized as the next generation method for game AI for its ability of automatic feature extraction and learning. Although there have been related studies in full-connected network [2] and CNN [3], they still cannot exceed the performance of traditional methods' due to their own limitations in both data structure and network training. In this paper, we introduce a new data structure with three dimensions for containing more information available on the table, and make training with CNNs. We show that our result achieves an agreement rate accuracy of 68.8%, which is much better than those of previous traditional [4], [8] and deep learning models [3], demonstrating the potential of deep learning methods on imperfect information tasks.

This paper is organized as follows. Section 2 gives basic introduction about rules and terms for the game of Japanese Mahjong. Section 3 introduces related studies and works. Section 4 provides a clear frame about our data model design and section 5 mentions corresponding network structure design for discard and stealing (which

[1] Graduate School of Information Science and Technology, The University of Tokyo
[2] Graduate School of Engineering, The University of Tokyo

is also called naki) strategies. Section 6 shows some experimental results and finally section 7 makes a conclusion.

2. Basic Rules and Terms of Japanese Mahjong

Mahjong is a tile-based imperfect information game which has been developed in China since the Qing dynasty and has been further developed throughout the world since the early 20th century. It was first brought into Japan in 1924. The game is commonly played by four players.

Japanese Mahjong, also known as Richi Mahjong, is a popular variation. Japanese Mahjong has 136 tiles with 34 types and four same tiles in each type. The 34 types are 1m (man)–9m, 1p (pin)–9p, 1s (sou)–9s, and seven honor tiles: East, South, West, North, Haku, Hatsu and Chun. The tiles are mixed and then arranged into four walls that are each two stacks high and 17 tiles wide. 26 of the stacks are used for starting hands, while 7 are used for a dead wall and the remaining 35 stacks form the playing wall [14]. Each player will have 14 tiles in hand, and the basic winning tile combination is $x(\mathrm{AAA}) + y(\mathrm{ABC}) + \mathrm{DD}$ while $x + y = 4$, where AAA represents for triplets (three same tiles), ABC represents for sequences (three number tiles with sequential numbers) and DD represents for two same tiles. One can declare a win by either picking up a tile from the wall by oneself or winning with others' discards if a legal pattern can be formed. At each round, one player can pick up a tile from the wall to form a 14-tile pattern in hand, and needs to choose one tile from hand for discarding after that. Stealing actions such as pon, chi and kan are actions one player can do with others' discards in given situations. Four players start a game of score 25000 each, and one game of Mahjong usually has four or eight parts of subgames. Compared with original mahjong games, Japanese one has several other new features, which are listed below.

2.1 Richi

Richi means declaring a ready hand and it is also a kind of yaku. A player can declare richi showing all others that he is waiting to win, just needing one more tile to form a legal hand. One can only declare richi when he does not have any stealing called. Once declaring richi, one's tiles in hand cannot be changed anymore, but he will own big chance of gaining larger scores if winning.

2.2 Yaku

Yakus are specific patterns of tiles or conditions that possess values. For Japanese Mahjong, in order for winning, at least one yaku is needed. Each yaku has a specific hand value, which devotes to hands' values exponentially. Doras in hand cannot be counted as yakus, but will devote into hand values once winning.

2.3 Dora

Hand pattern and the number of dora tiles in hand are two important factors for winning with a large score. Dora is a kind of bonus tile that adds yaku number once winning. Each time a new subgame starts, there will be one or several types of dora tiles according to dora indicators which are shown to all players. There are at most four dora indicators in each subgame, with one in initial and one type adding each time when countering a kan from any player. Normally, the succeeding tile of dora indicator is regarded as a dora. However, this is not always a linear relationship but divided into several small loops, for example the next tile of 9m is not 1p but 1m, the next tile of North is not Haku but East. Akadora is another type of specific dora. Compared with dora indicators mentioned previously, akadoras are always a regular part among doras. Each game will have three red five tiles which are counted as akadoras, one each for man type, pin type, and sou type respectively. Different from normal five number tiles, they are colored red, while having all other same functions of a normal one.

2.4 Uradora

There is an uradora indicator under each dora indicator. Uradoras are revealed and counted only when one player wins and with a richi declaration beforehand, and no one knows what the uradora indicators will be before final revealing, hence it is a potential bonus with uncertainty only for richi-declared players.

2.5 Wind

During each game, among all 34 kinds of tiles, we have seven types of honor tiles: East, South, West, North, Haku, Hatsu and Chun. For each game, there are two kinds of wind that are very important to all players: round wind and own wind. Round wind is set same to all players when one subgame starts, initialized as East and may change as game continues. Own wind, on the other hand, is different from player to player in each subgame. Not like Haku, Hatsu and Chun that any triplet of them is already recognized as one yaku, for wind tiles only round wind and player's own wind devote to yaku calculation.

2.6 Winning

There are mainly two types of winning, winning from a wall or winning from a discard. Winning from a wall is also called self-drawn or Tsumo, when the player gets his own tile picked up and then forming a legal hand. When facing a self-drawn, the other three players pay the winner some amount of score together. When the player catches a tile discarded by others and completes a legal hand, it leads to a ron, and the original owner of the discarded tile needs to pay the owner by himself entirely. The dealer, who has East as his own round wind, always pays more and gets more. Once the dealer wins, he will win a score 50% larger, no matter by self-drawn or others'

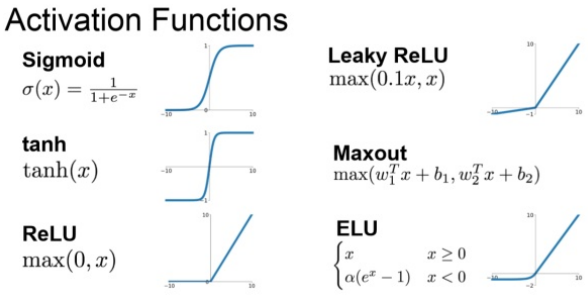


Fig. 1 Common Activation Functions [15]

discards. On the other hand, if another player win by self-drawn but not the dealer, the dealer will need to pay the score one time more than the other two normal players.

3. Related Works

3.1 Convolutional Networks

Convolutional Neural Networks (CNN) is one of the most successful structures so far in deep learning. CNN is not only widely used in CV tasks such as object recognition, object detection and Generative Adversarial Networks (GAN), but also does a quite good job in NLP and game fields. The four basic features of CNN are local connections, shared weights, pooling, and using many layers. First parts of a CNN network are usually made of convolutional and pooling layers and latter parts are mainly full-connected ones. Convolutional layers detect local conjunctions from features and pooling layers merge similar features into one [5]. Besides pooling layers, there are still some other common techniques usually utilized together with convolutional layers.

3.1.1 Activation Function

Among a neural network, the output of neurons always needs processed by activation function before transmitting to the next layer. The concept is defined by Bengio. Y. [6] in 2016 as: An activation function is a function h: R $\rightarrow$ R that is differentiable almost everywhere.

As shown in **Fig. 1**, there are quite many different activation functions. Although both Leaky ReLU and ELU are two variations of ReLU function, but till now ReLU is still most commonly used among CNN. Compared with Sigmoid, ReLU function mainly has three benefits, for it simplifies calculations among back propagation procedure, solves the problem of gradient vanishing Sigmoid will encounter then layers go deep, and causes sparsity of the network by leading some parts of the neurons toward zero.

3.1.2 Regularization

Overfitting is a common problem that can be met during training when the network considers sampling errors and goes towards high-variance, fitting well on training data but not well on test data, showing a weak generalization ability. Besides containing more data for training, L0, L1 and L2 Regularization are one common way for overfitting depression. There are also some other useful techniques, such as Noise, Model Ensemble, Batch Normalization and Dropout.

3.2 Imperfect Information Game Studies

An one on one version of Texas Hold'em, one of the most popular poker variations, has been solved by using subgame solving to reach an approximate Nash equilibrium strategy [7]. However, there are still no suitable strategies on multi-player poker solutions for the unsustainable computational cost. Game of Mahjong, theoretically, has even larger complexity than the game of Texas Hold'em. With existing techniques and skills, it is impossible to reach the Nash equilibrium for this game.

Traditional mahjong AI usually has two function blocks, the offense part and the defense part. For offense part, the tile efficiency is the main thing considered in order for faster winning and larger scores, ignoring any other players' actions. It is acted as an one-player mahjong game. For the defense part, how to avoid ron (which means defeated) by others is the task, aiming at discarding safe tiles but not attempting to get a winning. The training procedure is to achieve a suitable balance between offense part and defense part and decide output strategies in diverse situations [4], [8]. The state-of-art agreement rate accuracy for haifu data learning on test dataset before is 62.1% [4].

Deep learning has been hot these several years and is featured for its strong automatic feature extraction ability which needs no artificial extraction, together with surprising learning capability. However how to design the data structure and how to build the network is still a difficult task. Tsukiji *et al.* (2015) [2] made a 2-layer full-connected network with 1653-dimension input, receiving a test accuracy rate of 43%. In Tsukiji's new article in 2017, he designed a 5 by 34 by 5 data structure like an image to contain the tile information, 5 planes for own hand tiles and discarded tiles, and the 34 by 5 structure for each plane and achieves a test accuracy of 53.98% [3].

4. Data Structure Design

In this paper, we introduce a basic data structure named as 'plane' which is a matrix with two dimensions, 34 in height and 4 in width. A normal plane structure is shown in **Fig. 2**. We believe data in such kinds of structures is easy for training. The 34 rows representing 34 types of tiles, while the 4 columns representing the existing number of each tile type in given situations since the maximum number of each type of tile is 4.

Here we propose in all 55 planes to represent the information available in one game. The data structure is shown in **Table 1**.

For present round information, we use 15 planes to store, including one plane representing for one's own hand tiles,

four planes for four players' discarded tiles, four planes for four players' stealing tiles, one plane for dora indicators, three planes for other three players' richi information, one plane for round wind and 1 plane for own wind. One player can have at most 14 closed tiles not melded in hand, therefore the first plane will always has at most 14 ones in it. For stealing planes, one can call at most four melds in one game, with each meld containing three (pon & chi) or four (kan) tiles, hence each stealing plane will have at most 16 ones, with all other elements filled with zeros. For dora indicator plane, one game will have one indicator at the beginning, and at most four indicators during play, since once the fourth kan is declared which means the fifth dora indicator appears, it usually leads directly to the end of this round of game according to game rules. The dora indicators are usually different, but occasionally it may appear the same in one subgame. Besides, Richi information of other players is usually very important, for it is a clear declaration that some other players are already prepared to achieve a winning and having a relatively large chance of winning patterns with big scores. The player should be especially careful not to discard dangerous tiles the declaring players need to avoid ron by them. The richi planes contributes much on this task. Compared with models without richi information structure, the test accuracy raises over 1%. The richi plane here is represented in the way with some differences from previous planes. For previous ones, they are used to record tile numbers of each kind for one part of information. Obviously, this is not a suitable way for richi information representation. Therefore, here we adopt 0/1 planes to represent. An 0/1 plane is a plane filled with internal elements all zeros or all ones, or we can recognize it as a pure white/black channel in CV tasks. This works well in our situation. For round wind and own wind information, in order to cause emphasis and maintain the principle of at most four tiles for each type of tile on the table, we mark the whole corresponding row with four ones to mark wind signs.

Past actions the player made also have great influence on present choice decision strategy. After experiments, we include last four rounds' information including own hand tiles, four players' discarded tiles and four players' stealing tiles. The dora indicators may add during playing, hence last round's indicator information is needed in order to mark a last round's kan sign. For richi information, the round just after one's richi is the most threatening time for other players since it will add another one yaku if it wins during that round, so we also need last round's richi information to take care of that dangerous round. We include past dora indicator and richi information only in past 1 situation but not others, this is why we have 13 planes for past 1 situation but 9 planes each for others.

5. Neural Network Structure Design

In this section, we propose our filter design for the CNN network. Since our input is 34 by 4 in plane structure with 55 planes, we use several layers of small filters to cover the network. However, our plane structure is too narrow with only 4 in width that normal convolution filters such as 3 by 3 cannot work well here. Therefore we finally adopt three convolutional layers with 5 by 2 in filter size and 100 in filter height, as shown in **Fig. 3**. A Batch Normalization layer and a Dropout layer of rate 0.5 are added after each convolutional layer for overfitting depression. We add one full-connected layer of 300 neurons after the flatten layer. We have a final output layer at last, designed after the full-connected layer but not the same for different functions which are shown below.

	1	2	3	4
1m	0	0	0	0
2m	1	0	0	0
3m	1	1	1	0
4m	1	0	0	0
⋮				
9m				
1s	1	0	0	0
⋮				
9s	0	0	0	0
1p	0	0	0	0
⋮				
9p	1	1	1	0
東	0	0	0	0
⋮				
白	0	0	0	0
⋮				
中	1	1	1	0

Fig. 2 Data Plane Structure [9]

Table 1 Input features for neural networks

Feature	# of planes
Own hand tiles	1
Discarded tiles	4
Stealing tiles	4
Dora indicators	1
Richi players	3
Round wind	1
Own wind	1
Past 1 situation	13
Past 2 situation	9
Past 3 situation	9
Past 4 situation	9

5.1 Discard Network

In this paper, we evaluate this tile discard problem as a 34-class classification problem, since we have 34 types of tiles during the game, therefore the final layer is a softmax layer consisted of 34 neurons. A 14-class classification method has also been thought of, for one player can have

Layer (type)	Output Shape	Param #
main_input (InputLayer)	(None, 55, 34, 4)	0
Conv1 (Conv2D)	(None, 100, 30, 3)	55100
batch_normalization_20 (Batc	(None, 100, 30, 3)	12
dropout_20 (Dropout)	(None, 100, 30, 3)	0
Conv2 (Conv2D)	(None, 100, 26, 2)	100100
batch_normalization_21 (Batc	(None, 100, 26, 2)	8
dropout_21 (Dropout)	(None, 100, 26, 2)	0
Conv3 (Conv2D)	(None, 100, 22, 1)	100100
batch_normalization_22 (Batc	(None, 100, 22, 1)	4
dropout_22 (Dropout)	(None, 100, 22, 1)	0
flatten_6 (Flatten)	(None, 2200)	0
fc1 (Dense)	(None, 300)	660300
batch_normalization_23 (Batc	(None, 300)	1200
dropout_23 (Dropout)	(None, 300)	0
Softmax (Dense)	(None, 35)	10535

Total params: 927,359
Trainable params: 926,747
Non-trainable params: 612

Fig. 3 Tile Discard Network Structure

at most 14 tiles in hand, thus the prediction output can just represent the order number of the tile in an ordered hand tile. However, after experiments we find that the 34-class one attains much better results. The only potential risk for 34-class method is about illegal discard tiles since we have 34 prediction choices for the network but only at most 14 closed tiles in hand. However, as mentioned later in section 6.4, we find that all discard choices made by network are legal predictions, showing surprisingly strong learning ability of our network.

5.2 Stealing Network

Obviously, tile discarding strategy is the cornerstone for the game of Mahjong, and a good discarding strategy can always lead to good performances in games. However during real playing, stealing actions are also important factors for a game play. Here we divide the whole stealing strategy into several parts shown as below.

5.2.1 Pon

One player can declare a pon when any other player discards a type of tile that the player already exists two closed ones in his own hand. The pon strategy decision can be recognized as a binary classification problem, while one represents for making the pon action and zero represents for action canceling. However, during real training for stealing network, unbalanced dataset becomes a problem. Among all pon-able situations, only about 30% of data makes a real pon, which means only 30% of positive labels exist among dataset. Usually for normal training problems, we can use f1 score or other techniques to deal with unbalanced data in order to get good performances in all categories, however it becomes different for this task to some extent. The final goal for our task is to do a good job collaborating with discarding strategies while playing, but not just achieving a relatively good result in all categories. A small ratio of positive labels means in real situations people tend not to do that decision. Actually, from my own perspective of view, different loss functions and judgment levels just represents for different styles when playing, for instance normal training will lead to average style while f1-score-focused training may lead to a more offensive playing style. In this paper, we adopt normal classification training judgment method at last.

5.2.2 Chi

Different from pon, one player can only declare a chi with tiles discarded by the left-side player to form a legal sequence meld. Chi situations get a more severe ratio between positive and negative labels, among which only about 14% of data has positive labels. Besides, not like pon, chi has three dealing methods with one tile, for that tile can be either the smallest, largest or the middle part of the meld formed. Therefore, we recognize chi as a 4-class classification problem, with zero denoting action canceling, one, two and three denoting three types of chi actions.

5.2.3 Kan

For kan-able situations, we only have too small dataset, and the ratio of declaring a kan is too low for training. Therefore, here we take the strategy that we just declare a closed kan when that kan is an isolated one which means that tile does not own a potential chi declaration chance according to present hand pattern. All other kan chances are ignored.

5.3 Richi Network

Our player will declare a richi once he can, and announce a winning once getting that chance, for simplicity. Also, the player needs to choose a tile to discard after declaring richi. This is also a tile discard problem, so it is quite similar with normal discards, just the legal discard range gets limitations. Here we adopt the same network as the normal discard one, and determine the strategy that to choose the tile with biggest prediction possibility among legal choices for richi discarding.

6. Experimental and Simulation Results

6.1 Game records collection and training environment

Our models and networks are trained on NVIDIA Tesla K10 GPU with 32 GB memory size. We solve these strategy prediction problems as multi-classification problems with softmax layers. We adopt the supervised machine learning method for prediction model and use game records collected from the 'Houou' table at the online mahjong site 'Tenhou' in the year of 2015 as the training data. The 'Houou' table is only open for the top 0.1% mahjong play-

ers so the game records can be considered as good quality. During each game, we just follow one player's game record, and make sampling until the player's richi declaration since once richi declared, the player cannot determine one's discard tiles anymore, the discard tile is always the same as the drawn tile then.

6.2 Network training and results

6.2.1 Discard Network

We sampled 600 000 round situation data in all for network training, with 10% among it divided as the validation dataset. The final validation test accuracy arrives around 68.8%, which can been seen in **Fig. 4**. For authority, we randomly select 30 000 rounds of situations picking from the data of the year 2014 for test dataset. In order to eliminate correlation among data, while testing, we just randomly pick one situation in each game. Our proposal finally achieves an agreement rate of 68.8%, exceeding the state-of-art 62.1% accuracy result [4]. As shown in **Table 2**, not only the first choice is counted, but also the second and third choice, named as rank2 and rank3, are also compared. It can be easily seen that our proposal does a much better job on this task.

Table 2 Agreement rate for tile discard

Approach	Rank1	Rank2	Rank3
Mizukami's [4]	62.1%	82.9%	90.9%
Tsukiji's [3]	53.98%		
Ours	68.8%	93.6%	96.5%

6.2.2 Pon Network

Our validation accuracy arrives 88.2% during training as shown in **Fig. 5**. For test, We collect 30000 data from another year of 2014. Test results are shown in **Table 3**, where 'Pred' is abbreviation for 'Predicted' and 'Act' is abbreviation for 'Actual'. We achieve a whole accuracy of 88.1% on test data. We get precision of 80.1% and recall of 79.3%, leading to a f1 score of 0.793.

Table 3 Test results on pon network

Pred \ Act	True	False
True	6771	1604
False	2081	19544

6.2.3 Chi Network

Our validation accuracy arrives 90.4% during training as shown in **Fig. 6**. For test, we collect 40 000 data from year of 2014. Test results are shown in **Table 4**. As a 4-class classification problem, instead of True/False prediction, we make a prediction of 0, 1, 2 and 3. Here 0 means chi canceling, while other three numbers indicate three types of chi mentioned before. We achieve a whole accuracy of 90.4% on test data. We can also notice that, for real chi-called labels, to call which kind of chi reaches a very high accuracy of around 96.8%, which means once the player decides calling chi, he can usually do chi action type right. If we simplify it as a whether-to-chi-or-not binary classification problem, the accuracy will raise by another 0.3%. We get precision of 69.8% and recall of 55.9%, leading to a f1 score of 0.621. Considering the big size differences between positive and negative labels, this result is actually quite acceptable.

Table 4 Test results on chi network

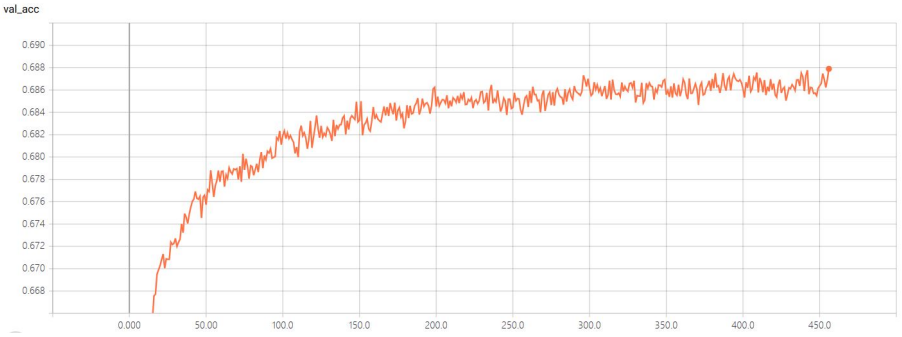
Pred \ Act	0	1	2	3
0	33216	781	808	817
1	409	854	16	7
2	535	13	1265	23
3	379	7	31	839

6.3 Evaluation on Tenhou

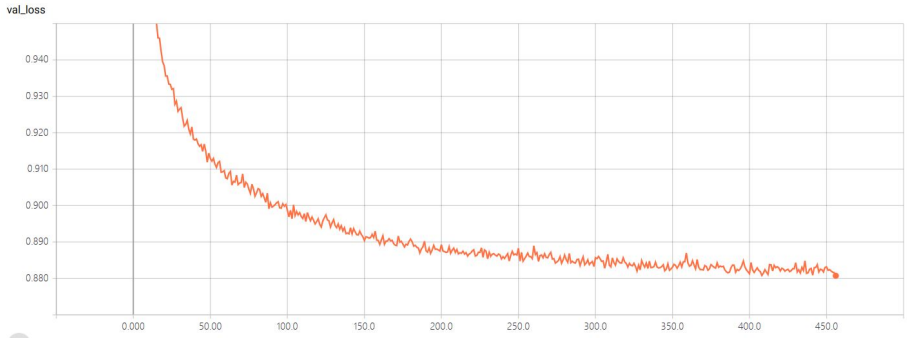
In order to see our strategy's performance, we make our program playing matches on the online mahjong site 'Tenhou' [16]. The rule is set with East-South Battle, with Akadora and with tanyao. Here we choose to play East-South Battles instead of East Battles for East Battles are usually too short that fortune takes too much ratio. We double the length of the game in order to get a more precise performance judgment. The performance of our program is shown in **Fig. 7**. We achieve a rating of 1822 after 76 East-South Battles, which is higher than that of an intermediate player.

From the figure, we can roughly conclude our program's style. The efficiency of playing is relatively high, and the program owns the ability of defense to some extent. It has a relatively high richi rate, caused by our strategy of declaring richi once able. However, this kind of simplicity can also cause problems. Since one cannot change his hand patterns after richi, and our program will immediately declare richi once it can no matter what kind of situation it is, sometimes it may lead to dangerous situations that other players own big hand patterns but the player already cannot defense anymore, causing big score losses. Besides, even with a hand pattern still can be improved easily, the program will directly declare richi once able, missing some chance of hand pattern improvement. Compared with a highly-skilled human player, our program will rely more on fortune for its high richi rate, and will have more chances of achieving a very high or low score in one game.

By going through battle logs, we also find several problems of our program. Since no knowledge about game rules is given for strategy decision, in rare cases, the program will form a hand pattern plausible but actually with no yaku. Besides, for some very unusual cases, the program obviously does not have corresponding concept, such as yaku-man patterns, due to few cases among training data. Besides, the program seems not having a strong sensitivity against one-color pattern. It is hard to say whether trying to form a one-color pattern is optimized or not in diverse situations, but the rate of forming such patterns for our program is obviously lower than normal human players.

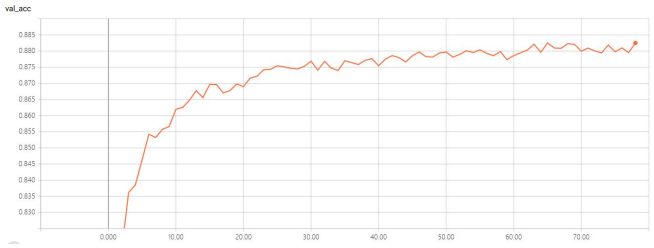


(a) Validation Accuracy

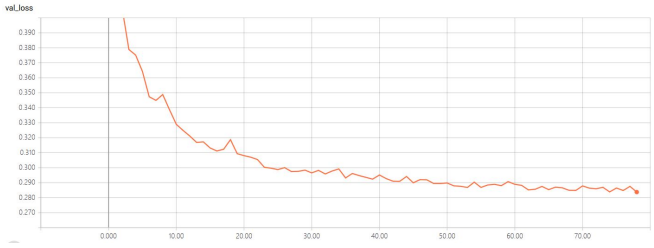


(b) Validation Loss

Fig. 4 Training curve for tile discard strategy

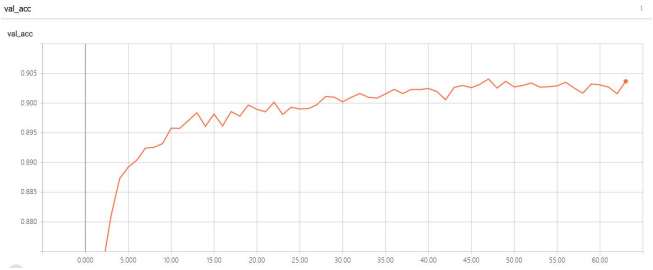


(a) Validation Accuracy

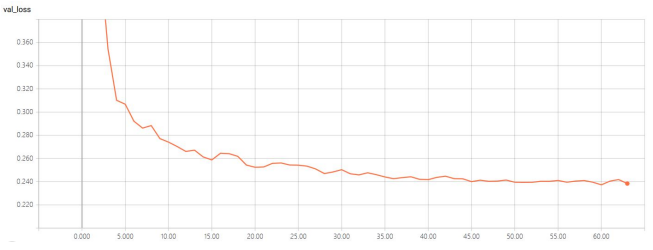


(b) Validation Loss

Fig. 5 Training curve for pon strategy



(a) Validation Accuracy



(b) Validation Loss

Fig. 6 Training curve for chi strategy

6.4 Interesting Findings

Pooling is recognized as a very efficient and useful down-sampling tool for convolutional network training especially in computer vision tasks. However during our experiment, we find the pooling layer not only not contributing to the accuracy but also making demerits instead. Usually for computer vision tasks, image recognition has property of movement invariance and rotation invariance, hence pooling usually does a great job for down-sampling, making fewer parameters and forming more robust network. However, for our task, the data plane structure is very elaborate and concise, therefore the pooling down-sample will lose too much information, causing lower accuracy.

From the test cases, we also find that our proposed data structure perfectly understands the information of hand tiles and what are the dora tiles only from the indicators. For prediction output, it is a 34-class softmax problem, however the player just has at most 14 closed tiles in hand. Within the 30 000 test cases, we find the predictions made by our neural network are all among the legal tile actions in hand, at the rate of 100%. Obviously, the network already learns the output range of legal predictions.

Fig. 7 Program Performance on Tenhou

Besides, according to the game rules, dora tile and dora indicator does not have a simple linear relationship but several loops as mentioned before. However, our network perfectly learns that, and with dora indicator plane instead of pure dora information plane after deduction, the accuracy even raises about 1%, due to the maintenance for the principle of at most four tiles each kind on the table.

7. Discussion

In this paper, we proposed a new data model designed for the game of Mahjong, achieving the best performance so far on agreement rate accuracy after training on a CNN network, with totally no concepts about game rules in advance. We simplified the prediction tasks as multi-classification problems and achieved good performances. We also made a play trial online, getting a relatively good result, with simplified framework including only discard, pon and chi networks.

There are still many points can be improved. Firstly, as an imperfect information learning task, in order to achieve better results, information should be contained as much as possible for learning. However, for present data model, there is still some information missing. Akadoras are not marked in our model yet, and we still cannot find a suitable way to contain information like round number, scores, ranks and the richi stick number into our model. Secondly, the strategy for game play is simplified in this paper, such as kan decision and richi discard choice. Last but not least, the program is not optimized toward the online mahjong site 'Tenhou'. For this site's rank system, the fourth rate takes an very important factor. Therefore, try to avoid being the last rank is the strategy one must take in order to achieve a high rating. Our program does not know any information about round number and players' scores yet, therefore having nothing to do with that. All these points mentioned can be improved in future work.

Acknowledgments This work was supported by JST ERATO Grant Number JPMJER1501, Japan.

Reference

[1] Schmidhuber. J.: Deep Learning in neural networks: An overview, Neural networks, Vol.61, pp.85–117 (2015).
[2] 築地毅, 柴原一友: ディープラーニング麻雀-オートエンコーダとドロップアウトの有効性, ゲームプログラミングワークショップ 2015 論文集, Vol.2015, pp.136–142 (2015).
[3] 築地毅, 柴原一友: CNN 麻雀-麻雀向け CNN 構成の有効性, ゲームプログラミングワークショップ 2017 論文集, Vol.2017, pp.163–170 (2017).
[4] 水上直紀, 鶴岡慶雅: 期待最終順位に基づくコンピュータ麻雀プレイヤの構築, ゲームプログラミングワークショップ 2015 論文集, Vol.2015, pp.179–186 (2015).
[5] LeCun. Y., Bengio. Y., and Hinton. G.: Deep learning, Nature, Vol.521, No.7553, p.436 (2015).
[6] Gulcehre, C., Moczulski, M., Bengio, Y., et al.: Noisy activation functions, International Conference on Machine Learning, pp.3059–3068 (2016).
[7] Brown. N. and Sandholm. T.: Safe and nested subgame solving for imperfect-information games, Advances in Neural Information Processing Systems, pp.689–699 (2017).
[8] Mizukami. N. and Tsuruoka. Y.: Building a computer Mahjong player based on Monte Carlo simulation and opponent models, Computational Intelligence and Games (CIG), 2015 IEEE Conference on, IEEE, pp.275–283 (2015).
[9] Gao. S., Okuya. F., Mizukami. N., et.al: Improved Data Structure and Deep Convolutional Network Design for Haifu Data Learning in the Game of Mahjong, 情報処理学会第 80 回全国大会, Vol.2, p.05 (2018).
[10] Silver. D., Huang. A., Maddison. C., et al: Mastering the game of Go with deep neural networks and tree search, Nature, Vol.529, No.7587, pp.484–489 (2016).
[11] Silver. D., Schrittwieser. J., Simonyan. K., et al: Mastering the game of go without human knowledge, Nature, Vol.550, No.7676, pp.354 (2017).
[12] 北川竜平, 三輪誠, 近山隆: 麻雀の牌譜からの打ち手評価関数の学習, ゲームプログラミングワークショップ 2007 論文集, Vol.2017, pp.76–83 (2017).
[13] Hiroshi. S., Tomohiro. S., Akitoshi. H., et al.: 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)
[14] Japanese Mahjong, available from ⟨https://en.wikipedia.org/wiki/Japanese_Mahjong/⟩
[15] https://Introduction to Exponential Linear Unit, available from ⟨https://medium.com/@krishnakalyan3/introduction-to-exponential-linear-unit-d3e2904b366c⟩
[16] Tenhou, available from ⟨https://tenhou.net/⟩