ORIGINAL RESEARCH PAPER

Bloody Mahjong playing strategy based on the integration of deep learning and XGBoost

Shijing Gao1,2* | Shuqin Li1,2

1School of Computer, Beijing Information Science and Technology University, Beijing, China
2Sensing & Computational Intelligence Joint Lab, Beijing Information Science & Technology University, Beijing, China

Abstract

Bloody Mahjong is a kind of mahjong. It is very popular in China in recent years. It not only has the characteristics of mahjong’s conventional state space, huge hidden information, complicated rules, and large randomness of hand cards but also has special rules such as Change three, Hu must lack at least one suit, and Continue playing after Hu. These rules increase the difficulty of research. These special rules are used as the input of the deep learning DenseNet model. DenseNet is used to extract the Mahjong situation features. The learned features are used as the input of the classification algorithm XGBoost, and then the XGBoost algorithm is used to derive the card strategy. Experiments show that the fusion model of deep learning and XGBoost proposed in this paper has higher accuracy than the single model using only one of them in the case of high-dimensional sparse features. In the case of fewer training rounds, accuracy of the model can still reach 83%. In the games against real people, it plays like human.

INTRODUCTION

Computer games [1] is one of the most challenging research directions in the field of artificial intelligence. It is divided into complete information games and incomplete information games [2]. Algorithms for the complete information games are typically represented by Google's Alpha series AlphaGo [3], AlphaGo Zero [4], and AlphaZero [5], which were achieved outstanding achievements. The incomplete information games mainly include two models, one is CFR (Counterfactual Regret Minimization) [6,7] algorithms, such as Cepheus [8], Libratus [9], Pluribus [10] etc., another is deep reinforcement learning algorithms like AlphaStar [11], and OpenAI Five [12]. However, both models have instability, and then a more stable Fictitious Play [13] model was proposed, which verified that Fictitious Play is better than CFR in [14].

Mahjong is a four-player game. As one of the representatives of incomplete information games, most research scholars are Japanese and Chinese. Most of them research on Japanese mahjong or Chinese national standard mahjong. The research material is mainly about how to make the final decision of one's own playing behaviours according to the information of the hand tiles, the information of the played tiles and various decision-making information of other players, that is, to imitate human intelligence playing tiles. There are three main methods used in the current study of mahjong. (1) Based on machine learning and statistical methods [15, 16], according to the game rules of mahjong, train or calculate the probability of taking the maximum possible action. Such as k-gate [17] problem, constructing search tree through abstraction [18]. (2) Construct mahjong playing strategy method based on the opponent model [19], and consider the possible actions of the opponent to modify their own behaviour. (3) Methods based on deep learning. For example, reference [20] first proposed a novel competitive strategy composed of a new deep residual network [21]. Microsoft’s Suphx [22], based on deep reinforcement learning, has reached the level of top human players on Tenhou mahjong platform.

Bloody mahjong is a kind of very popular mahjong in China. Its playing method is very different from Japanese mahjong and Chinese national standard mahjong. At present, the academic research based on blood mahjong is very rare.
The online Mahjong game platform has also developed some Bloody Mahjong ‘AI programs’ to play with human when the number of players in the game is insufficient or players are temporarily in trouble. However, most of these programs are based on the rules of playing tiles, which are blunt and low in ‘intelligence’. And it decreases the player’s gaming experience. So, this paper designs an intelligent program based on the integration of deep learning and XGBoost [23], which can imitate human playing tiles, and can replace human players to continue playing and improve the game experience when hosting the platform.

2 | BLOODY MAHJONG GAME INTRODUCTION

Bloody Mahjong has four players with a total of 108 tiles. Each tile is composed of suits and numbers. The suits are divided into Bamboo, Character, and Dot, and the numbers are numbered from 1 to 9. Besides the common rules, such as Pong, Kong, and Hu, there are also some unique rules.

1. **Dingque**: It refers to that each player must choose one of the three suits mentioned above as an invalid tile before, which will not be used as a tile type of Hu combination.

2. **Change three**: After the player obtains the initial hand tiles, he needs to take out three tiles to exchange with one player. The way of exchange can be clockwise exchange, anticlockwise exchange, relative exchange and no exchange. For example, if central position is the 0th, anticlockwise exchange means 0 to 3, 3 to 1, and so on. Use D for the Dot, B for Bamboo and C for Character, the anticlockwise exchange is shown in Table 1.

3. **Hu** must lack at least one suit: After the Dingque, the player’s tiles type combination must not contain the suit of Dingque before the tile can be Hu.

There are rules for Continue playing after Hu, see the appendix for details. These special rules are not included in the Japanese Mahjong and the Chinese national standard Mahjong. These special rules greatly increase the difficulty of Bloody Mahjong.

At the beginning of the game, a player usually obtains 13 tiles as the ‘initial hand’, and then players take Change three and Dingque. After the game determines one of the four players as the dealer. Tiles are left as live wall. The dealer should have another title and discards firstly. In each round, each player first gets a title from live wall, and then discards a tile, or chooses action Pong, Kong or Hu. Due to the existence of Pong, Kong, and Hu, the player in the next round is not necessarily the next player after the current player has finished playing but may be any one of the other three players. If no one else takes Pong, Kong, or Hu, the next round is based on the clockwise next player after the last round. The game goes on until all the tiles from live wall are exhausted.

Bloody Mahjong not only has the characteristics of mahjong’s conventional state space, huge hidden information, complicated rules, and large randomness of hand cards but also has special rules such as Change three, Hu must lack at least one suit and Continue playing after Hu. This paper will conduct related design research on these issues, the purpose is to train an intelligent AI that resembles people playing tiles.

3 | DESIGN AND IMPLEMENTATION OF BLOODY MAHJONG PLAYING SYSTEM

3.1 | System framework design

Mahjong’s tile-playing action can be regarded as a classification process. The mahjong situation features are discrete. The XGBoost (eXtreme Gradient Boosting) model in machine learning has a good effect on the classification of discrete features. Therefore, this paper chose to use the XGBoost model to train the playing tiles model. However, a mahjong situation has the characteristics of strong information concealment, high-dimensional sparse features [25], huge state space, complicated rules, and so forth. It is not suitable for manually extracting features, and then use XGBoost to directly perform classification operations. Deep learning can learn and extract features well, but it requires a lot of resources. Setting the model layer too deep will cause training slowly. When the layer is too shallow, the model’s effect will be greatly discounted, and sometimes it will also produce incomprehensible behaviours. Considering, this paper decides to combine the characteristics of machine learning and deep learning, integrate the two, and study Bloody Mahjong.

The specific method is to select the deep learning DenseNet [24] model to pre-extract the features of the mahjong situation, and then reduce the dimensionality of the features. The dimensionality-reduced features are used as the input of XGBoost, and then the action of the extracted features is classified. The schematic diagram of the DenseNet and XGBoost fusion model training is shown in Figure 1, and the various parts of the model will be described in detail below.

| Player | Initial hand | Exchanged tiles | After exchanged hand |
|------|--------------|----------------|---------------------|
| 0    | 2233879 C 1222458 B | 257 C | 2339 C 136 D 1222458 B |
| 1    | 3456689 C 113556 D | 136 D | 3456689 C 155 D 135 B |
| 2    | 6 C 12347 D 1356777 B | 135 B | 2556 C 12347 D 6777 B |
| 3    | 256 C 234667 D 3468 B | 256 C | 257 C 234667 D 3468 B |

 TABLE 1 Change three
3.2 Data processing and presentation

In view of the fact that there is a lot of hidden information in Mahjong and that the randomness is relatively large, this paper considers using a large number of human player data to train a deep learning model. The purpose is to enable the model to predict some hidden information based on the experience summarized by the large amount of data.

Data used in this paper is the Bloody Mahjong log data provided by a famous online games company. The original data records the information of four players and the information of playing tiles. The original data is in the form of string and needs to be parsed into the corresponding situation data. The original data is shown in Figure 2. Contents marked in red box (the first box) in Figure 2 contain the basic information of players, the information of original tiles, Contents marked in yellow box (from left to right, from top to bottom, the second box) are the information of Change three, and Contents marked in blue box (the third box) are the information of four players’ Dingque. Fourth to tenth lines marked in green box (the fourth box) are the complete games playing data, which records the information of the player obtains tiles, discard tiles, Pong, Kong and Hu. The data is four number as a unit, for example, the first four characters in the fourth row 2D06 marked by red underline means that the player of two has played a tile of six Characters. Similar to the others, the actions M is getting tiles, D is discarding, P is Pong, G is Kong, H is Hu. The last two lines marked in white box (the last box) record the score information of the player's final win or loss after the game is over. For example, 1(1)284,468:10,000:0:274,468 means the original 284,468 points of player 1, the game loses 10,000, then the tax is 0, and the remaining score is 274,468.

First of all, the original data is cleaned to eliminate the player's failed data or incomplete data. The purpose is to reduce the impact of poor quality of data on the model during model training. According to the game situation, the model input is constructed.

3.3 DenseNet model design and implementation

3.3.1 Game situation information representation

In view of the fact that the data of Mahjong has high dimensionality and sparseness, in order to represent the situation information of Mahjong better, this paper designs to use 0 or one to indicate whether there is a certain tile, and use the columns of the array to represent the number of tiles. The purpose is to better characterize the game's situation information.

Each tile of Mahjong consists of suits and numbers. The suits are divided into Bamboos, Characters, and Dots. The
numbers are numbered from 1 to 9. This paper considers $3 \times 9 = 27$ different single tiles. Each single tile has the same four tiles, so Bloody Mahjong has a total of 108 tiles. This paper uses a $4 \times 27$ two-dimensional matrix to represent these 108 tiles. A matrix element value of 1 indicates that the tile is present, and an element value of 0 indicates that there is no tile. For example, use D for the Dot, B for Bamboo and C for Character, a player's hand tiles have 19,999 Dots, 11999 Bamboos, and 111 Characters, as shown in Table 2:

The data represent of using one-hot form is conducive to the deep learning model to learn the desired knowledge.

3.3.2 | DenseNet model input

In this paper, the special rules of Bloody Mahjong are designed as a feature plane, the purpose is to make the model better learn the knowledge of Bloody Mahjong. For example, for the rule of Change three, the information of the three tiles swapped in and the three tiles swapped out are treated as a feature plane separately. There is also information about the Dingque of different players.

Since Mahjong is incomplete information, decision-making can only be based on their own hand tiles, other three players discard tiles, above mentioned Change three tiles, whether there are other players Hu, other players Dingque information, as well as the latest action taken by last player. Therefore, according to the game records, seven kinds of features are extracted, which is used as input information of DenseNet model and as shown in Table 3. Each feature is represented by a $4 \times 27$ matrix.

| Plane | Feature description                          |
|-------|---------------------------------------------|
| 0     | Player hand characteristics                 |
| 1     | Player next to you discard characteristics  |
| 2     | Player relative to you discard characteristics |
| 3     | Player before to you discard characteristics |
| 4     | Total discard characteristics               |
| 5     | Accept, discard three tiles characteristics   |
| 6     | Other characteristics                       |

T A B L E 2  Mahjong hand representation

| 1D  | … | 9D  | 1B  | … | 9B  | 1C  | … | 9C  |
|-----|---|-----|-----|---|-----|-----|---|-----|
| 1   | 1 | 1   | 1   | 1 | 1   | 1   | 0 |     |
| 0   | 1 | 1   | 1   | 1 | 1   | 1   | 0 |     |
| 0   | 1 | 0   | 1   | 1 | 1   | 1   | 0 |     |
| 0   | 1 | 0   | 1   | 0 | 1   | 0   | 0 |     |

T A B L E 3  Features representation

| Plane | Feature description                          |
|-------|---------------------------------------------|
| 0     | Player hand characteristics                 |
| 1     | Player next to you discard characteristics  |
| 2     | Player relative to you discard characteristics |
| 3     | Player before to you discard characteristics |
| 4     | Total discard characteristics               |
| 5     | Accept, discard three tiles characteristics   |
| 6     | Other characteristics                       |

3.3.3 | DenseNet module implementation

The input of DenseNet model is the feature plane composed by the features mentioned in Table 3, the size is $7 \times 4 \times 27$ three-dimensional matrix. The output is the probability vector of 30 kinds of tile playing behaviours described in Table 3. The DenseNet model is mainly composed of Denseblock. The model in this paper is composed of three Denseblocks, each of which has the structure as Table 4.

The stride of the convolutional layer is all $1 \times 1$. The entire model also has a first features layer, plus two transition layers. The overall structure of the model is shown in Table 5.

The role of the DenseNet model in this paper is not only to obtain the extracted feature vectors for classification, but also to continuously train itself. To classify the extracted features using the XGBoost model, it can get better results.
3.4 Design and implementation of XGBoost module

After using DenseNet model to extract features in Bloody Mahjong, the features of the model are still very complex. In order to better consider the impact of these features on the player's choice of actions, this paper considers the XGBoost model that supports column sampling based on feature granularity in parallel. The model input is based on the features obtained from the output of the DenseNet model mentioned above, and then classified according to the extracted features. The final output is actions taken by the player.

The action of the player is mainly divided into four actions: discarding tiles, Pong, Kong and Hu. Pong, Kong and Hu can only happen in a specific situation, and are not relevant to a specific tile, so show whether the actions are happened or not. Discarding tiles is relevant to a specific tile, the need to specify which tile to discard, so discarding tiles has total of 27 actions. In summary, there are 30 actions that players can take. It is represented by one-hot code, use D for the Dot, B for Bamboo and C for Character, as shown in Table 6.

The main training parameters of XGBoost model are shown in Table 7.

There are two main options for Booster parameters, gbtree and gblinear. Gbtree uses a tree structure to run the data, which conforms to the feature classification extracted in this paper. Since there are a total of 30 output actions, the value of the number of categories parameter Num_class is 30. Because it is a multi-classification problem, the parameter of Objective is multi:softmax. Max_depth indicates the depth of the tree. The value is usually between 5 and 10, and the value in this paper is 8. Other parameters are set according to general requirements. At the same time set the parameter early_stopping_rounds equal to 100, which means that training will stop if there is no improvement after 100 rounds.

3.5 System implementation

When training the overall model of the system, firstly system extract the features through the DenseNet model to obtains the feature vectors of the training data set. Then the XGBoost model performs model training based on the data set constructed by these feature vectors. After training, the XGBoost

TABLE 5 DenseNet model parameters

| Layer      | Model   | In channels | Out channels | Kernel_size | Stride |
|------------|---------|-------------|--------------|-------------|--------|
| Features   | Conv2d  | 7           | 24           | $3 \times 3$ | $1 \times 1$ |
|            | BatchNorm2d | 24        | -            | -           | -      |
|            | MaxPool2d   | -          | -            | $1 \times 2$ | $1 \times 1$ |
| Dense block1 | -      | 24          | -            | -           | -      |
| Transition1| BatchNorm2d | 84        | -            | -           | -      |
|            | Conv2d     | 84          | 42           | $1 \times 1$ | $1 \times 1$ |
|            | AvgPool2d   | -          | -            | $1 \times 2$ | $2 \times 2$ |
| Dense block2 | -      | 36          | -            | -           | -      |
| Transition2| BatchNorm2d | 102       | -            | -           | -      |
|            | Conv2d     | 102         | 51           | $1 \times 1$ | $1 \times 1$ |
|            | AvgPool2d   | -          | -            | $1 \times 2$ | $2 \times 2$ |
| Dense block3 | -      | 48          | -            | -           | -      |
| Norm_final | BatchNorm2d | 111       | 111          | -           | -      |
| Flatten    | -        | -           | 333          | -           | -      |
| Linear     | Linear   | 333         | 128          | -           | -      |

TABLE 6 Player action representation

| 1 C | ... | 9 C | 1 D | ... | 9D |
|-----|-----|-----|-----|-----|-----|
| 1 B | ... | 9B  | Pong| Kong| Hu |

TABLE 7 XGBoost parameters

| Parameters           | Value       |
|----------------------|-------------|
| Booster              | Gbtree      |
| Objective            | multi:softmax |
| Num_class            | 30          |
| Gamma                | 0.1         |
| Max_depth            | 8           |
| Subsample            | 0.6         |
| Colsample_bytree     | 0.7         |
| Eta                  | 0.1         |
model is obtained, and then further training the DenseNet model, next the DenseNet model outputs the test set feature vector, and the XGBoost model obtains the test set error rate based on the feature vector. Next step, system determines whether to train the XGBoost model based on the test set error rate. Before this, first set a Best-error in pseudo code is 0.3, which is the initial error rate for the pretrained model. When the error rate of the test set is less than Best-error, Best-error is updated to the current value, and then XGBoost is further trained. The batch size [26] used during training is 128, the learning rate is 0.01, and the optimizer uses momentum [27]. The pseudo code of the training process is shown below.

Algorithm 1 The DenseNet and XGBoost fusion model training process

**Input:** data\text{original}: original data from Mahjong game; model\text{pre}: pretrained DenseNet model: 
**Output:** Test set error rate: Test\text{error}; 
BestModel\text{dense} = the DenseNet model after training; BestModel\text{XGB} = the XGBoost model after training; 
1: data\text{train}, data\text{dev}, data\text{test} = SplitData(data\text{original}); // divide the training set, validation set, and test set from the data\text{original};
2: initial DenseNet model model\text{dense};
3: features = OuputFeature(data\text{train}, model\text{pre}); // pretrained model output data features
4: training the XGBoost model model\text{XGB} with features;
5: set the maximum number of iterations epochs;
6: set the initial best error rate best\text{error};
7: initial epoch = 0;
8: repeat
9:  Train(data\text{train}, data\text{dev}, model\text{dense}); // training DenseNet model
10: features = OutputFeature(data\text{test}, model\text{dense}); // DenseNet model output test set features
11: result = Prediction(features, model\text{XGB}); // XGBoost model output test set predicted results
12: test\text{error} = ComputeError(result);
13: if test\text{error} ≤ best\text{error} then
14:  best\text{error} ← test\text{error};
15:  Save(model\text{dense});
16:  features = OutputFeature(data\text{train}, model\text{dense});
17:  training the XGBoost model model\text{XGB} with features;
18:  Save(model\text{XGB});
19: end if
20:  Save(test\text{error});
21: until (epoch > epochs)

4 | EXPERIMENTAL RESULTS AND ANALYSIS

Mahjong data comes from the online Bloody Mahjong platform of the game company. There are nearly 400,000 match data in total. Remove incompletely recorded and background-controlled data, and remove data with a score less than 4000. The lower the score is, the worse the ability of playing games is. According to data statistics, most of the data scores are over 4000, so the noise data with scores less than 4000 are removed, and the effective situation of the final extraction is about 210,000. The states that can be extracted in each game will vary according to the length of the game, but the average number of states in each game is about 10. According to the final number of games composed of these pairs of games, the ratio of training set, verification set and test set is 8:1:1, the final training set has about 1.69 million pieces of data, and the verification set has about 211,500 pieces of data and test set has about 211,500 pieces of data.

In order to verify the performance of the DenseNet and XGBoost fusion models, single DenseNet model experiments and single XGBoost model experiments conducted in this paper, and the experiments compared with the fusion model. In addition, in order to verify the consistency between the AI system and the real player's tile playing, an experiment makes to imitate the human player's games playing.

4.1 | System model performance test

Experiment 1: Performance testing of single DenseNet models

At the beginning of the experiment, a model was trained separately using a neural network, and the model parameters were described in Section 3.3.3. The training set did not remove data below 4000 score. The model training set and validation set loss and error rate are shown in Figure 3.

It can be seen from the graph that the model loss and the rate of error rate began to decrease slowly from 10 rounds. When training up to 10 rounds, the accuracy rate can reach 78.97%, and when training up to 20 rounds, it can reach 80.01%. As the number of training rounds increases, the accuracy rate of the model also increases, but the magnitude of the increase continues to decrease.

Experiment 2: Performance comparison between the single DenseNet model and the fusion model of DenseNet and XGBoost
FIGURE 3 Single DenseNet model training information

The parameter setting and training process of the fusion model are described in Section 3.2. The number of training sessions is set to 10 in the experiment. Comparing with the error rate of the single DenseNet training and the fusion model training results, the results are shown in Figure 4.

Based on the pre-training mode, at the beginning, the error rate of the fusion model is higher than that of the single neural network model, because fusion model training error includes not only the error of the neural network, but also the error of the XGBoost model. As training continues, the error rate of the fused model starts to be lower than that of single DenseNet model. Finally, on the basis of 10 rounds of training, the fusion model has an accuracy rate of about 95% on the test set, which is higher than the single DenseNet model.

Experiment 3: Performance comparison between the single XGBoost model and the fusion model in this paper

In order to compare the single XGBoost model with the fused model, on the basis of the original data, sum the matrix represented by the features in Table 2 and each plane column of the first five feature matrices, and then splice them to form a vector of 135, because the single XGBoost model does not need the input data similar to one-hot coding, and then split the sixth feature matrix in Table 2 into accepting three tiles and discarding three tiles two 27-dimensional vectors, and finally add other features. The total feature dimension is 223 dimensions. The error rate of the first 170 rounds of training set and verification set is taken, and the information of the 10th round of training XGBoost is taken for the fusion model. The results are shown in Figure 5.

Finally, the single XGBoost model stopped training when it was trained up to 1161 rounds. The accuracy of the separate XGBoost on the same test set was 76.16%, which is far lower than the fusion model. The biggest possibility is that the input data is 223 dimensions, but most of them are invalid data 0. This feature representation sparseness may be the cause of unsatisfactory training results.

FIGURE 4 Single DenseNet model and fusion model training error rate

FIGURE 5 Single XGBoost model and fusion model training error rate
4.2 Imitation the human player's tile playing test

The goal of this paper is to make a humanoid AI system that can replace humans to play cards, think like humans, and take correct actions. Therefore, the test was conducted on an online Bloody Mahjong platform. During the test, the other three parties were all people. The card AI program in this paper shows in the right.

Experiment 1: Test discarding normal tiles

Figure 6 shows a certain situation of Bloody Mahjong. The player gets six Dots and was ready to play tiles. According to the Bloody Mahjong Rule 2 Hu must lack at least one suit, and the player exactly has decided that the lack was Dots, so when the six Dots is obtained, the human choice is directly discarding the tile. The right side of Figure 6 is the AI program of this paper. It can be seen that after the information on the left is entered into the program, the program results in the 14th action, and the tile action according to Table 4 is exactly six Dots. It shows that the system plays tiles in this situation the same as human playing tiles and conforms to the human playing tile habits.

Experiment 2: Test Pong

As shown in Figure 7, the last action is one player discarding 1 Character. Because the players had more hand tiles and Pong 1 Characters, they would not break the sequence of other tiles, so Pong is a better result. On the right side of Figure 7 is the AI program of this paper. It can be seen that after inputting the information on the left into the program, the program decides to do action 28. According to Table 4, the tile playing action is just Pong. It shows that the system will choose the same action as the human do in the situation.

Experiment 3: Test Kong

In this situation, the player next current player discards a tile, and current player has three identical tiles in his hand (Figure 8). At this time, current player can choose to Pong or Kong. But according to the hand tiles, there are only three Bamboos tiles. If current player chooses Pong, current player needs to play other Bamboos tiles, so the best choice at this time is the Kong. The final prediction result of the model is also 28, corresponding to the movement of the Kong. It shows that the system can not only choose between the Kong and Pong when judging special situation, but also can accurately know the action of the Kong, which is consistent with human action.

Experiment 4: Test Hu

In this situation, the choosing action Hu will get a lot of benefits, and most people will choose Hu (Figure 9). Therefore, the final prediction result of the model is also 29, corresponding to the Hu action. It shows that the system can not only judge whether Hu's conditions have been reached, but also can accurately know that Hu's action should be taken, which is consistent with the human action.
Experiment 5: Test failed action

In this situation, the player is relative to current player discards a tile, and current player has exactly two same tiles, and the tiles will not be dismantled after being Pong (Figure 10). Therefore, it is better to choose Pong, but the model predicts that the tile will be discarded. This is because too little information is considered and the system does not have a good grasp of the Pong timing.

The fusion model in this paper first extracts 128 features from the DenseNet model, and then the XGBoost model further classifies the features based on the extracted features. According to the model obtained after 100 rounds of training, the accuracy rates under different behavioural statistics are shown in Table 8, where D means discarding ordinary tiles, P means Pong, G means Kong, and H means Hu.

Among them, the error rate of the Kong is relatively high, one is that the action of the Kong is relatively few, so the data is also relatively small, while the action of the Kong can also be subdivided into small classes such as the open Kong, the dark Kong and the supplementary Kong. The rule is more complex than other actions, so the prediction is more difficult. However, the influence of the Kong on the final win and loss of the game is not so great. In a game, the Kong is seldom seen, and in most cases, when the Kong is available, the Kong will be selected. Therefore, even if the accuracy of the Kong is not high, the rules can be used to replace it.

**FIGURE 8** Test Kong

**FIGURE 9** Test Hu

**FIGURE 10** Test failed action
In the 2020 ‘Competitive World Cup’ Chinese University Computer Games Championship & National Computer Games Tournament Mahjong Group, the model obtained after using the model in this paper for migration won the runner-up, indicating the effectiveness of the model in this paper.

| TABLE 8 Error rates of different categories in the fusion model |
|-----------------|-------|-------|-------|-------|
| Action          | D     | P     | G     | H     |
| Total           | 162,332 | 34,459 | 1403  | 13,314 |
| Error           | 7844  | 493   | 494   | 489   |
| Error_rate      | 4.83% | 1.43% | 35.21%| 3.67% |

5 | CONCLUSION

Due to the complicated game rules, Mahjong has more features and more complicated representation of the situation. No matter it is represented by a matrix or directly combined by vectors, it belongs to high-dimensional sparse data. In view of such data features, this paper uses deep learning and XGBoost fusion model to implement a system for playing Bloody Mahjong. By comparing the performance of single neural network model, single XGBoost model and fusion model, it shows that it is feasible to use neural network to extract features first, and then use XGBoost model to classify. It shows that in the face of high-dimensional sparse and discrete data features, using fusion model is a good choice. Through the experiment of imitating people to play tiles, it shows that fusion model can play Mahjong as reasonably as human.

At present, the system proposed in this paper is not ideal in predicting the effect of the Kong. Later, it will try to collect more relevant data and train the bar alone. In addition, the practical effect of this model needs to be further tested by accessing the online battle game platform.

ACKNOWLEDGEMENTS

This work is supported by key potential projects of Promoting Research Level program at Beijing Information Science and Technology University (No. 5211910927), by normal projects of General Science and Technology research program (No. KM201911232002), and by normal projects of promoting graduated education program at Beijing Information Science and Technology University.

ORCID

Sijing Gao https://orcid.org/0000-0003-0190-9171

REFERENCES

1. Wang, Y., et al.: Research and development of computer games. CAAI Trans. Intell. Syst. 11(6), 788–798 (2016)
2. Research and Application of Computer Games with Imperfect Information. South China University of Technology (2017)
3. Silver, D., et al.: Mastering the game of Go with deep neural networks and tree search. Nature. 529(7587), 484–489 (2016)
4. Silver, D., et al.: Mastering the game of Go without human knowledge. Nature. 550(7676), 354 (2017)
5. Silver, D., et al.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815 (2017)
6. Bowling, M., et al.: Computer science. Heads-up limit hold’em poker is solved. Science. 347(6218), 145 (2015)
7. Tammelin, O.: Solving large imperfect information games using CFR++. arXiv preprint arXiv:1407.5042 (2014)
8. Burch, N., Johanson, M., Bowling, M.: Solving imperfect information games using decomposition. In: Twenty-Eighth AAAI Conference on Artificial Intelligence, pp. 602–608. AAAI Press (2014)
9. Brown, N., Sandholm, T.: Libratus: the superhuman AI for no-limit poker. In: Twenty-Sixth International Joint Conference on Artificial Intelligence, pp. 5226–5228 (2017)
10. Brown, N., Sandholm, T.: Superhuman AI for multiplayer poker. Science. 356(6546), 885–890 (2019)
11. AlphaStar’s official blog: AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. (2019). https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii
12. OpenAI: OpenAI Five[DB/OL] (2018). https://blog.openai.com/openai-five/
13. Perrin, S., et al.: Fictitious play for mean field games: continuous time analysis and applications. arXiv preprint arXiv:2007.03458 (2020)
14. Ganzfried, S.: Fictitious play outperforms counterfactual regret minimization. arXiv preprint arXiv:2001.11615 (2020)
15. van Rijn, J.N.: Mahjong solitaire computing the number of unique and solvable arrangements. (2011)
16. Jumpei, K., Ryo, Y., Ayumi, S.: A Mahjong AI to change the strategy depending on the global situation. Infor. Proc. Soc. Jpn. 154–157 (2016)
17. Cheng, Y., Li, C.K., Li, S.H.: Mathematical aspect of the combinatorial game ‘Mahjong’. arXiv preprint arXiv:1707.07345 (2017)
18. Kurita, M., Hoki, K.: Method for constructing artificial intelligence player with abstraction to Markov decision processes in multiplayer game of mahjong. arXiv preprint arXiv:1904.07491 (2019)
19. Mizukami, N., Tsuruoka, Y.: Building a computer Mahjong player based on Monte Carlo simulation and opponent models. In:2015 IEEE Conference on Computational Intelligence and Games (CIG), pp. 275–283 (2015)
20. Wang, M., et al.: A novel deep residual network-based incomplete information competition strategy for four-players Mahjong games. Multimed. Tool. Appl. 78, 23443–23467 (2019). https://doi.org/10.1007/s11042-019-7682-5
21. He, K., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
22. Microsoft research Asia: Microsoft super Mahjong AI Suphx, crack the imperfect information game. (2019). https://www.msra.cn/zh-cn/news/features/mahjong-ai-suphx
23. Chen, T., et al.: Xgboost: Extreme gradient boosting. (2016)
24. Liu, K., Bellet, A., Sha, F.: Similarity learning for high-dimensional sparse data. In: Artificial Intelligence and Statistics, pp. 653–662 (2015)
25. Huang, G., et al.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)
26. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747 (2016)
27. Sutskever, I., et al.: On the importance of initialization and momentum in deep learning. In: International Conference on Machine Learning, pp. 1139–1147 (2013)
APPENDIX: BLOODY MAHJONG

GAME RULES

Bloody Mahjong is a very fun Mahjong game in Southwest China! This game is from Yichang, Hubei. It is a very popular mahjong game in China. The basic rules of playing Mahjong are the same as that of ordinary Mahjong, except that after the players draw the tiles, they exchange three tiles. Here are some terms in the game.

1. Terminology

Define 1 Pong

Pong means that other players play a tile, but there are exactly two same tiles in our hand, we can take action to Pong. For example, if player 1 has a hand of ‘2566 Characters 12,347 Dots 6777 Bamboos’, at this time, if player 0 plays ‘6 Character’, then player 1 can take Pong action.

Define 2 Kong

Kong means that we have three the same tile. If another player just plays the tile or we get the tile ourselves, you can choose the action of Kong. In another case of Kong, you can also take the action of Kong if you have Pong a tile and then you get the tile again. For example, if player 1 has a hand of ‘2566 Characters 12,347 Dots 6777 Bamboos’, at this time, if player 0 plays ‘7 Bamboo’, then player 1 can take Kong action.

Define 3 Hu

Hu refers to the combination of players’ hand into a specific tile type. The common form is A + nB, where A refers to two same tiles, B refers to three same tiles or three consecutive tiles, and the value of n can be 0 to 4. There is also a Hu rule that 14 tiles in hand are divided into seven pairs. Of course, there are other rules involved. There are many rules of Hu, many types of Hu, and the scores of Hu corresponding to different types of Hu are also different. For example, if a player current hand is ‘11,234,777 Dots 666 Bamboos’ or ‘11,224,477 Dots 556,688 Characters’, the player can choose Hu.

Define 4 Dingque

Dingque refers to that before players start playing game, each player must choose one of the three suits mentioned above as an invalid tile, which will not be used as a tile type of Hu combination. For example, if player 1 Dingque is Bamboo, when player 1 choose Hu. The player hand only can contain Dots and Characters.

Define 5 Call

Call refers to the tile type that players can form a Hu with only one tile in their hands. This tile is called Call tile. Call tile can be the tile get by ourselves, or the tile others play, and there may be more than one Call tiles. For example, if a player current hand is ‘1,134,777 Dots 666 Bamboos’, at this time, other players discard ‘2 Dot’ or ‘3 Dot’, the player could choose Hu. In this time, ‘2 Dot’ or ‘3 Dot’ called call tiles.

2. The game rules

At the beginning of the game, a player usually gets 13 tiles as the ‘initial hand’, and then determines one of the four players as the dealer. Tiles are left as live wall. The dealer should have another title and discards first. In each round, each player first gets a tile from live wall, and then discards a tile, or chooses action Pong, Kong or Hu. Due to the existence of Pong, Kong and Hu, the player in the next round is not necessarily the next player after the current player has finished playing but may be any one of the other three players. If no one else takes Pong, Kong or Hu, the next round is based on the clockwise next player after the last round. The game goes on until all the tiles from live wall are exhausted.

Rule 1 Change three

After the player obtains the initial hand, he needs to take out three tiles to exchange with others. The way of exchange can be clockwise exchange, anticlockwise exchange, relative exchange and no exchange four ways. For example, if central position is the 0th, anticlockwise exchange means 0 to 3, 3 to 1, and so on. The anticlockwise exchange is shown in Table A1.

| Player | Initial hand | Exchanged tiles | After exchanged hand |
|--------|--------------|-----------------|---------------------|
| 0      | 2235879 characters 122458 bamboos | 257 characters | 2339 characters 136 dots 1222458 bamboos |
| 1      | 3456689 characters 113556 dots | 136 dots | 3456689 characters 155 Dots 135 Bamboos |
| 2      | 6 character 12347 dots 1356777 Bamboos | 135 Bamboos | 2566 characters 12347 dots 6777 bamboos |
| 3      | 256 characters 234667 dots 3468 bamboos | 256 characters | 257 characters 234667 dots 3468 bamboos |
Rule 2 Hu must lack at least one suit

After the Dingque, the player's tiles type combination must not contain the suit of Dingque before the tile can be Hu. And when the specific tile type in the Bloody Mahjong can be Hu, player can also choose not to Hu. For example, if a player current hand is ‘11,234,777 Dots 666 Bamboos’, the player can choose Hu, but if hand is ‘11 Characters 234,777 Dots 666 Bamboos’, the player cannot choose Hu.

Rule 3 Continue playing after Hu

Players in the field after Hu, still can continue to play. Other players discard tile, if it is Call tile, can Hu again. If the tile they get is also Call tile, they can also Hu again. Players can also Kong without changing the shape of the Call. For example, after player 1 took Hu, player 0, player 2, and player 3 continue the game, and player 1 draws the tile and then discards the tile.