# Prediction of Non-competitive Bridge Bidding with Double Machine Learning Models

Yu-Yen Chu, Chang-Biau Yang and Kuo-Si Huang

***Abstract*—This paper proposes two models for predicting non-competitive bridge bids: a neural network and a neural network combined with a random forest-embedded search tree. In our combined model, either the neural network or the search tree is selected to predict each bid based on the frequency of the bidding sequence. The experimental dataset consists of 255671 non-competitive boards extracted from "just declare" mode in the BBO. We apply the 10-fold cross validation to perform the experiments. Our neural network model and combined model achieve average accuracies of 95.7% and 94.8%, respectively, on the testing dataset, demonstrating that both models effectively learn the concept of the convention card. Additionally, in the training dataset, our combined model outperforms the neural network model in categories with rare data. However, in the testing dataset, our combined model outperforms our neural network model on the precision of category grand slam, but it shows no superiority in other performance indicators of the categories with rare data. Though the proposed search tree does not help the neural network enhance the performance on categories with rare data in the testing dataset, it still shows the potential of dealing with the data imbalance problem.**

***Index Terms*—bridge bidding, contract bridge, convention card, machine learning, neural network, random forest**

## I. Introduction

Artificial intelligence (AI) has been widely used in many regions, especially in games. In 1997, Deep Blue, a computer Chess invented by IBM, beat the human professional player of Chess. Also, in 2016, AlphaGo [12], a well-known AI of Go, defeated human professional players of Go. However, in Poker, Mahjong, and bridge, the AI of these games still cannot beat human players. It is because Chess and Go are the games that have perfect information [14]. For perfect information games, the AI can use some techniques, such as brute-force methods, knowledge-based methods, and Monte-Carlo tree search [2], to analysis the game state well and find a great action. Still, in recent years, some computer bridge models were developed [1, 6, 11, 13, 15, 16], which have good results. The results may help other researchers build the AI of bridge that can beat human player.

This research work was partially supported by National Science and Technology Council of Taiwan under contract NSTC 112-2221-E-110-026-MY2.

Yu-Yen Chu is with Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan.

Chang-Biau Yang is with Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan. E-mail: cbyang@cse.nsysu.edu.tw. (Corresponding author)

Kuo-Si Huang is with Department of Business Computing, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan. E-mail: huangks@nkust.edu.tw.

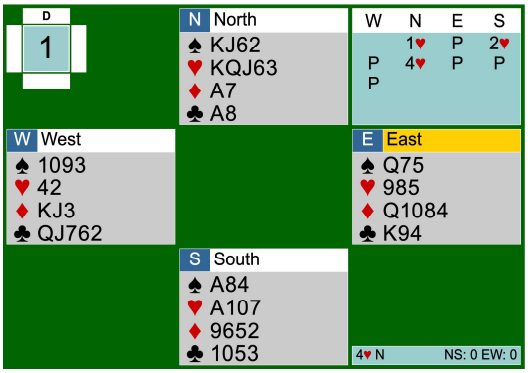


Fig. 1: An example of status for playing bridge in Bridge Base Online (BBO) [3]. (Image source: BBO, https://www.bridgebase.com/)

Bridge is a well-known competitive card game. The Bridge Base Online (BBO) [3] is a famous online bridge platform, as shown in Figure 1. There are two main stages of bridge: bidding and playing. In the playing stage of bridge, double dummy analysis (DDA) [5] finds the optimal play when all hands are visible. While DDA is inapplicable during actual play, it helps computers learn card strategies. In contrast, bidding reveals little information, making game state analysis difficult. Thus, bidding in computer bridge is more challenging than playing.

There are some previous bridge studies relevant to this paper, listed in Table I.

The experimental dataset has total 255671 boards, extracted from "just declare" in the BBO, with fixed bidding sequences, which are all non-competitive. We perform the experiments with 10-fold cross validation.

We use the neural network (NN) as our basic model to learn the concept of the convention card from the features of the hand information and the bidding sequence. Our model achieves an average accuracy of 95.7% in 10-fold cross-validation, demonstrating its effectiveness.

Inspired by the double neural networks in several papers [11, 16], to address data imbalance, we integrate a neural network with a random forest embedded search tree. Depending on bidding sequence frequency, either the neural network or the search tree is selected to predict the bid. Our combined model achieves 94.8% average accuracy in 10-fold cross-validation. While it outperforms the NN in rare data categories during training, improvements in testing are limited to grand

TABLE I: Previous bridge studies relevant to this paper.

| Year | Authors | Model and Method |
|---|---|---|
| 2015 | Ho and Lin [6] | non-competitive model with UCB algorithm |
| 2018 | Yeh *et al.* [15] | non-competitive model with deep reinforcement learning |
| 2019 | Rong *et al.* [11] | competitive model with supervised learning and reinforcement learning |
| 2022 | Zhang *et al.* [16] | competitive model with NN and reinforcement learning |
| 2023 | Jan *et al.* [7] | hand strength evaluation method with genetic algorithm |
| 2023 | Lin *et al.* [8] | non-competitive model which predicts the last bid with random forest |
| 2025 | This paper | non-competitive model with NN and random forest embedded search tree |

TABLE II: The value of each card in the hand strength evaluation methods proposed by Goren [4] and Jan *et al.* [7].

| | point count | | | | | | short suit | | | long suit | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | K | Q | J | 10 | 9 | 0 | 1 | 2 | $a$ | $b$ |
| Goren [4] | 4 | 3 | 2 | 1 | 0 | 0 | 5 | 3 | 1 | - | - |
| Jan's suit method [7] | 4 | 2.5 | 1 | 0.5 | 0 | 0 | 3.5 | 2 | 0.5 | 1.5 | 1 |
| Jan's NT method [7] | 4 | 2.5 | 1.5 | 1 | 0.5 | 0.5 | - | - | - | 0 | 0 |

slam precision. Though the search tree does not enhance overall rare category performance in testing, it shows potential for handling data imbalance.

The paper is organized as follows. Section II introduces hand strength evaluation methods and convention cards in the bridge game. Section III presents our model framework and training method. Section IV presents experimental results, evaluating model performance. Finally, Section V concludes the paper and discusses future work.

## II. Preliminaries

### A. Hand Strength Evaluation Methods

In order to choose the suitable contract level, a hand strength evaluation method is needed to represent hand strength. The most common method of hand strength evaluation is the Goren point count method [4], which considers cards A, K, Q, and J, and assigns them values 4, 3, 2, and 1, respectively. The point counted by the Goren point count method is called High Card Point (HCP).

To select an appropriate contract level, hand strength should be evaluated. The Goren point count method [4], the most common approach, assigns values of 4, 3, 2, and 1 to A, K, Q, and J, respectively, yielding the High Card Point (HCP). Additionally, 5, 3, and 1 points are added for a *void*, *singleton*, and *doubleton*, respectively.

Jan *et al.* [7] used the genetic algorithm to train several formulas for evaluating hand strength more precisely. The results are shown in Table II.

.

### B. Bidding System and Convention Card

During the bidding stage, the player and the partner aim to find a suitable contract. They exchange hand information through bids, and determine the contract by analyzing both

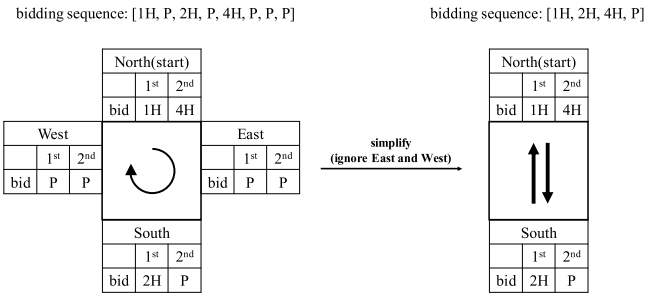


Fig. 2: An example of the simplification of the bidding sequence.

the player's own hand information and the partner's hand information. A bid can represent some hand information, and it is feasible to use the bid's suit to represent the number of cards in that suit.

*Natural* and *precision* are two of common bidding systems. The natural bidding system has more natural bids, and uses the bid's suit to represent the suit length. For example, in the natural bidding system, 1H means "12∼21 HCP, 5+ H". *Standard American Yellow Card* and *2 over 1* are two of the common natural bidding systems. The precision bidding system uses some artificial bids to more accurately divide the range of HCPs. For example, in the precision bidding system, 1C means "16+ HCP", which means a specific HCP range but the suit length of C is not guaranteed. *Precision Club* and *Polish Club* are two of the common precision bidding systems.

The convention card is the agreement of these bidding rules between teammates, usually based on one of the bidding systems, with minor changes.

## III. Our Bidding Models

Our models for non-competitive bridge bidding include a neural network and a hybrid model combining it with a random forest search tree. The hybrid model selects either the neural network or the search tree based on bidding sequence frequency.

In non-competitive bidding, opponents always bid `pass` (P), and `double` and `redouble` are ignored. The process ends when a player or his (her) bids `pass` (P) after a regular bid. For example, in Figure 1, the bidding sequence is simplified as shown in Figure 2.

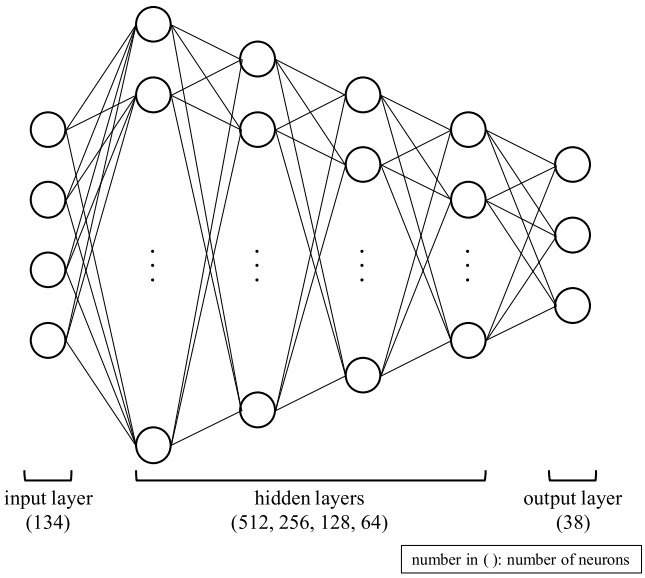


Fig. 3: The neural network structure of our model.

In order to distinguish different passes in the bidding sequence, we define two bids, *beginning pass* (BP) and *ending pass* (EP), which represent the pass at the beginning of the bidding sequence and the pass at the end of the bidding sequence, respectively.

Inspired by Rong *et al.* [11], we design a specialized state representation for the bidding process to improve model learning. This state includes hand information, vulnerability, and bidding sequence. Key hand features, such as suit length and HCP, are extracted using human bridge knowledge.

### A. Neural Network

Our model employs a fully connected neural network, as shown in Figure 3, with an input layer of 134 neurons, an output layer of 38 neurons, and four hidden layers.

The input elements of the neural network, as shown in Table III, are listed as follows.

**Card:** Each of the 52 poker cards is represented by a neuron, where a value of 1 indicates the card is in the player's hand, and 0 indicates it is not.

**Suit length:** The suit length is represented by four neurons, corresponding to the suit lengths of spades (S), hearts (H), diamonds (D), and clubs (C).

**HCP:** The HCP is represented by one neuron.

**Vulnerability:** The vulnerability is represented by one neuron, with a value of 1 for non-vulnerable and 2 for vulnerable.

**Bidding sequence:** The bidding sequence is represented by 76 neurons, with two sequences of length 38: one for the player's previous bids and one for the partner's. Each neuron has a value of 0 or 1, where 1 indicates the bid has been made, and 0 indicates otherwise. The neuron's value for "opening" is set to 0 when a regular bid is made.

TABLE III: The number of neurons in the input and output layers for a single hand.

| element | | number of neurons |
|---|---|---|
| input (134) | card | 52 |
| | suit length | 4 |
| | HCP | 1 |
| | vulnerability | 1 |
| | bidding sequence | 76 |
| output | | 38 |
| label | | 38 |

TABLE IV: The number of instances in each category. Note that a bidding sequence can contain multiple bids.

| category | instance |
|---|---|
| partial | 828763 (60.03%) |
| game | 202385 (14.66%) |
| slam | 14847 (1.08%) |
| grand slam | 800 (0.06%) |
| pass | 333837 (24.18%) |
| total | 1380632 (100%) |

The neural network output is a probability distribution over 38 bids, represented by 38 neurons. The label is a one-hot vector of length 38, where 1 indicates the bid is made, and 0 indicates it is not. Figure 4 is an example of neuron representation for the corresponding elements in the neural network.

### B. Random Forest Embedded Search Tree

Our model's search tree is built from bidding sequences, where each node represents a bid, and a path from root to leaf corresponds to a bidding sequence. Figure 5 shows the traversal path for [1H, 2H, 4H, P].

The random forest of each node is trained by the instances with corresponding bidding sequences, which can traverse in the search tree to locate the corresponding node. Figure 6 is an example of finding the random forest by corresponding bidding sequence.

## IV. Experimental Results

We run our models on Python 3.11.9. PyTorch module [9] is used to implement the neural network, and Scikit-learn module [10] is used to implement the random forests in the search tree.

### A. Performance Indicators and Parameters

In order to evaluate the performance of our models on dealing with the problem of data imbalance, we divide the dataset into five categories: partial, game, slam, grand slam, and pass. The number of instances in each category for bidding sequences are listed in Table IV. And, the number of final contracts in each category are listed in Table V. Note that "pass" cannot be a final contract.

We evaluate our models using four performance indicators: accuracy, precision, recall, and F1-score.

The loss of the neural network is calculated by the CrossEntropyLoss function in PyTorch [9], which is the average loss

| example | |
|---|---|
| hand | ♠ AQ32<br>♡ AKQ32<br>♢ 32<br>♣ K2 |
| vulnerability | non-vulnerable |
| bidding sequence | [1H, 2H] |
| output's probability distribution | 3H: 0.1<br>4H: 0.9<br>other bids: 0 |
| label | 4H |

| neuron representation | | |
|---|---|---|
| input | card (52) | [ 1 0 1 0 ... 0 1 1 ← ♠<br>1 1 1 0 ... 0 1 1 ← ♡<br>0 0 0 0 ... 0 1 1 ← ♢<br>0 1 0 0 ... 0 0 1 ] ← ♣ |
| | suit length (4) | [ 4 5 2 2 ] |
| | HCP (1) | [ 18 ] |
| | vulnerability (1) | [ 1 ] |
| | bidding sequence (76) | [ 0 0 0 0 1 0 0 0 0 0 0 0 ... 0<br>0 0 0 0 0 0 0 0 0 1 0 0 ... 0 ]<br>(1H, 2H) |
| output | probability distribution (38) | [ 0 ... 0 0 0.1 0 0 0 0 0.9 0 0 ... 0 ]<br>(3H, 4H) |
| label | one-hot vector (38) | [ 0 ... 0 0 0 0 0 0 0 1 0 0 ... 0 ]<br>(4H) |

number in ( ): number of neurons

Fig. 4: An example of neuron representation for inputs, outputs, and labels in the neural network.

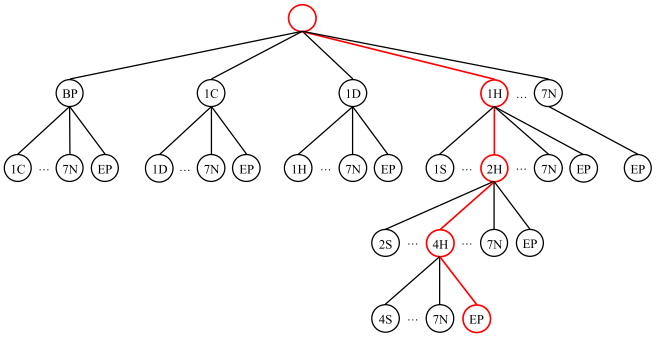


Fig. 5: An example of the path corresponding to the bidding sequence [1H, 2H, 4H, EP] in the search tree.

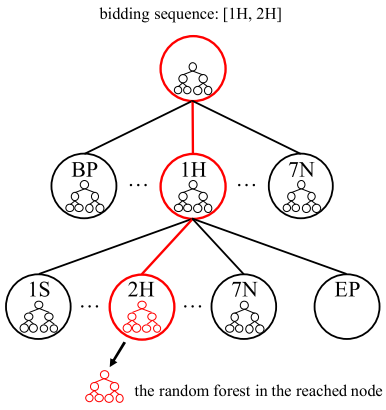


Fig. 6: An example of finding the random forest by corresponding bidding sequence.

TABLE V: The number of final contracts in each category.

| category | final contract |
|---|---|
| partial | 85848 (33.58%) |
| game | 156095 (61.05%) |
| slam | 12929 (5.06%) |
| grand slam | 799 (0.31%) |
| total | 255671 (100%) |

between the output ($x$) and the label ($y$) of the neural network, as follows.

$$\text{loss}(x, y) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{38}(\log\frac{\exp(x_{n,c})}{\sum_{i=1}^{38}\exp(x_{n,i})})y_{n,c} \quad (1)$$

In Equation 1, $N$ is the number of the outputs in $x$; $x_{n,i}$ is the value of the $i$th neuron in the $n$th output of $x$; $y_{n,i}$ is the value of the $i$th neuron in the $n$th label of $y$. The neural network executes 200 epochs, and the values of the accuracy and the loss are plotted on figures per 5 epochs.

Figure 7 shows the evaluation results for different learning rates (lr) and batch sizes. As one can see, (lr, batch_size) = $(0.001, 512)$ achieves the highest accuracy of training, but its testing loss increases after 50 epochs, indicating overfitting. Therefore, we choose the parameters (lr, batch_size) = $(0.001, 128)$ with the lowest loss of testing.

Figure 8 shows that the two hidden layer structures, (512, 256, 128, 64) and (1024, 256, 64), have similar training accuracy. However, (512, 256, 128, 64) outperforms (1024, 256, 64) in both testing accuracy and testing loss. Thus, we choose the (512, 256, 128, 64) structure.

Table VI shows that the random forest outperforms KNN in all six performance indicators for both training and testing datasets. Thus, we choose random forest as the classifier in the search tree.

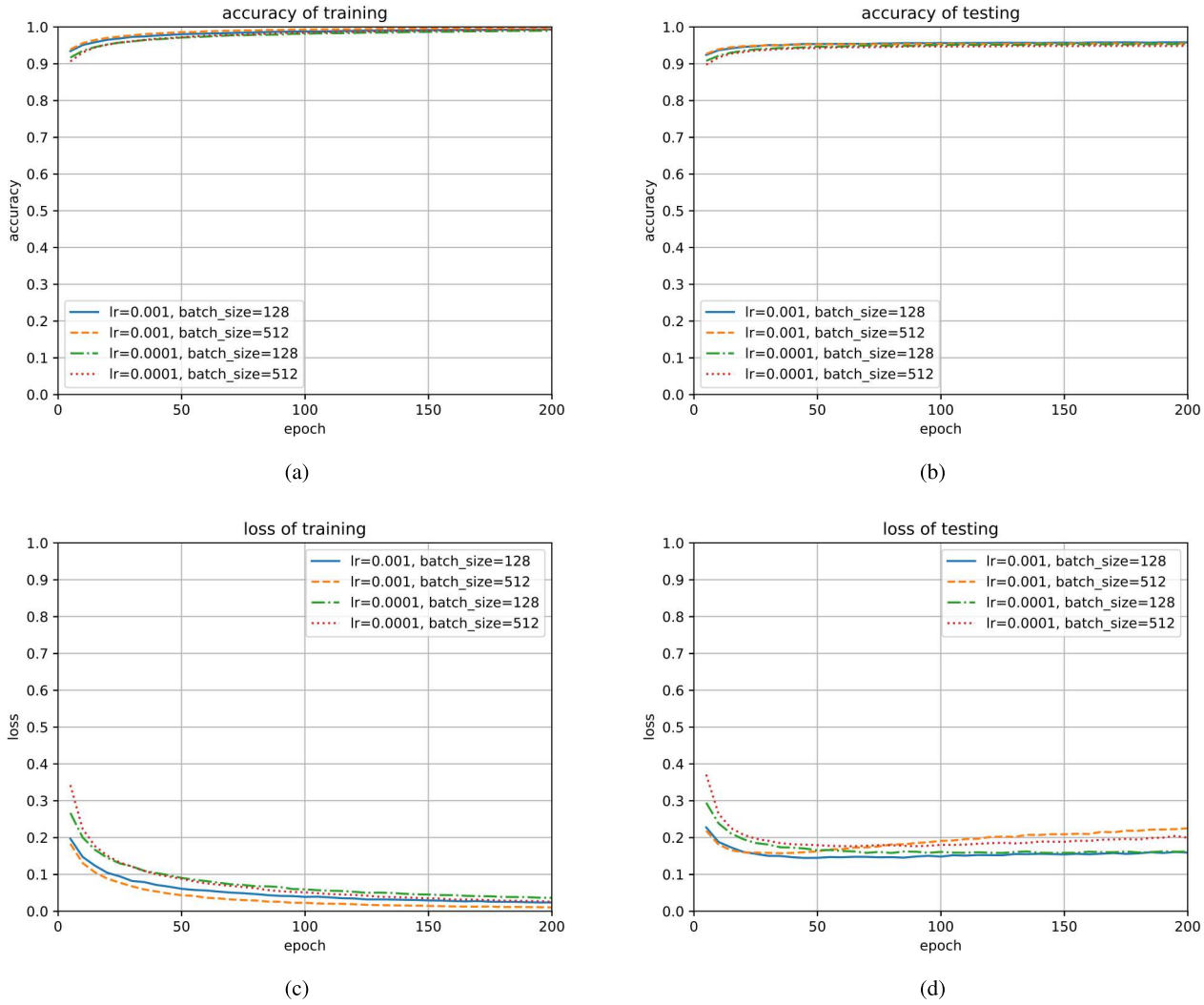


Fig. 7: Accuracy and loss of different learning rates (lr) and batch sizes. (a) Accuracy of training dataset. (b) Accuracy of testing dataset. (c) Loss of training dataset. (d) Loss of testing dataset.

TABLE VI: Different performance indicators with different machine learning classifiers.

| | | training | | testing | |
|---|---|---|---|---|---|
| | | random forest | KNN | random forest | KNN |
| indicators in the instances | accuracy | 98.89% | 96.77% | 94.79% | 94.20% |
| | $F1_3$ | 98.44% | 85.60% | 58.44% | 56.85% |
| | $F1_4$ | 97.62% | 79.92% | 35.41% | 31.82% |
| indicators in the final contracts | accuracy | 98.30% | 93.83% | 90.57% | 89.06% |
| | $F1_3$ | 98.85% | 86.93% | 67.76% | 64.83% |
| | $F1_4$ | 98.17% | 80.66% | 43.26% | 38.21% |

### *B. Performance of Our Models*

We evaluate the performance of our neural network model (NN) and our combined model (NN + tree) by using the performance indicators in both instances and final contracts. $r$ is set to $0.00025$ for the combined model. The performance indicators in the final contracts of these models are listed in Table VII.

Our neural network (NN) and combined model (NN + tree) achieve 92.1% and 90.5% accuracy in testing, respectively, indicating both predict the final contract well. While our combined model has a higher $\text{precision}_4$ (79.7% vs. 73.8%), it does not surpass the NN in other performance metrics.

## V. Conclusion

This paper proposes two non-competitive bridge bidding models: a neural network and a hybrid model combining it with a random forest search tree. Bid prediction in the hybrid model depends on bidding sequence frequency.

Future work includes refining the random forest search tree by optimizing features and parameters per node, leveraging convention card knowledge for better bid prediction. Additionally, we will extend our model to competitive bidding, which

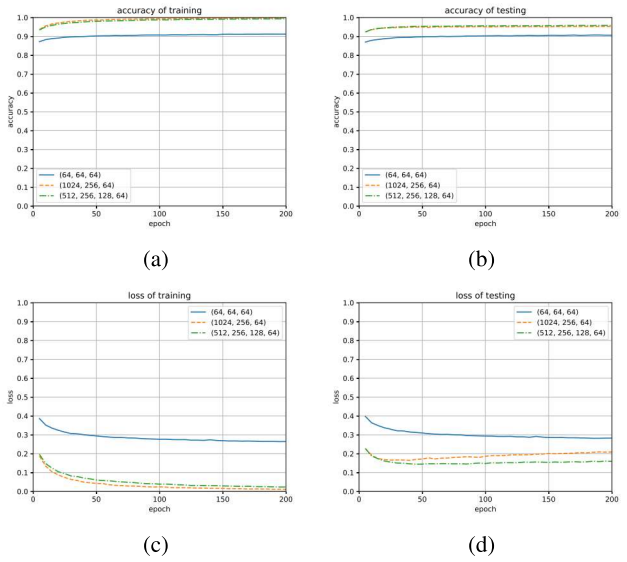


Fig. 8: Accuracy and loss of different hidden layer structures. (a) Accuracy of training dataset. (b) Accuracy of testing dataset. (c) Loss of training dataset. (d) Loss of testing dataset.

TABLE VII: The performance indicators in the final contracts of our neural network model (NN) and our combined model (NN + tree). Here, $\text{precision}_i$ means category $i$.

| | training | | | | testing | | | |
|---|---|---|---|---|---|---|---|---|
| | NN | | NN + tree | | NN | | NN + tree | |
| | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| accuracy | 0.980 | 0.000 | 0.983 | 0.000 | **0.921** | 0.000 | **0.905** | 0.000 |
| $\text{precision}_1$ | 0.971 | 0.001 | 0.971 | 0.000 | 0.915 | 0.001 | 0.904 | 0.001 |
| $\text{precision}_2$ | 0.990 | 0.000 | 0.994 | 0.000 | 0.954 | 0.001 | 0.943 | 0.000 |
| $\text{precision}_3$ | 0.981 | 0.001 | 0.997 | 0.000 | 0.850 | 0.004 | 0.847 | 0.003 |
| $\text{precision}_4$ | 0.966 | 0.004 | 0.997 | 0.001 | **0.738** | 0.029 | **0.797** | 0.021 |
| $\text{recall}_1$ | 0.987 | 0.000 | 0.986 | 0.000 | 0.965 | 0.000 | 0.960 | 0.000 |
| $\text{recall}_2$ | 0.981 | 0.000 | 0.982 | 0.000 | 0.924 | 0.000 | 0.907 | 0.000 |
| $\text{recall}_3$ | 0.936 | 0.002 | 0.980 | 0.001 | 0.624 | 0.006 | 0.560 | 0.002 |
| $\text{recall}_4$ | 0.877 | 0.008 | 0.969 | 0.002 | 0.347 | 0.015 | 0.319 | 0.012 |
| $\text{F1}_1$ | 0.979 | 0.000 | 0.979 | 0.000 | 0.939 | 0.000 | 0.931 | 0.000 |
| $\text{F1}_2$ | 0.985 | 0.000 | 0.988 | 0.000 | 0.939 | 0.000 | 0.925 | 0.000 |
| $\text{F1}_3$ | 0.958 | 0.001 | 0.988 | 0.000 | 0.720 | 0.005 | 0.674 | 0.002 |
| $\text{F1}_4$ | 0.919 | 0.004 | 0.982 | 0.001 | 0.469 | 0.015 | 0.453 | 0.013 |
| macro_precision | 0.977 | 0.001 | 0.990 | 0.000 | 0.864 | 0.007 | 0.873 | 0.005 |
| macro_recall | 0.945 | 0.002 | 0.979 | 0.001 | 0.715 | 0.004 | 0.687 | 0.003 |
| macro_F1 | 0.960 | 0.001 | 0.984 | 0.000 | 0.767 | 0.004 | 0.746 | 0.003 |
| weighted_precision | 0.983 | 0.000 | 0.987 | 0.000 | 0.935 | 0.000 | 0.925 | 0.000 |
| weighted_recall | 0.980 | 0.000 | 0.983 | 0.000 | 0.921 | 0.000 | 0.905 | 0.000 |
| weighted_F1 | 0.982 | 0.000 | 0.985 | 0.000 | 0.927 | 0.000 | 0.913 | 0.000 |

requires handling a larger bid space with more complex feature encoding.

## REFERENCES


[1] Y. Costel, "Wbridge5." http://www.wbridge5.com/.

[2] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," *Proceedings of the 5th International Conference on Computers and Games*, Turin, Italy, 2006.

[3] F. Gitelman, "Bridge Base Oline." https://www.bridgebase.com/.

[4] C. Goren, *Charles Goren Point Count Bidding in Contract Bridge*. Simon and Schuster, 1953.

[5] B. Haglund, "Double Dummy Solver." https://privat.bahnhof.se/wb758135/bridge/index.html.

[6] C.-Y. Ho and H.-T. Lin, "Contract bridge bidding by learning," *Proceedings of the 2015 Association for the Advancement of Artificial Intelligence Workshop*, Hyatt Regency in Austin, Texas, USA, 2015.

[7] J.-P. Jan, K.-S. Huang, and C.-B. Yang, "Hand strength evaluation in bridge games with the evolutionary algorithms," *Proceedings of the 40th Workshop on Combinatorial Mathematics and Computation Theory*, Taoyuan, Taiwan, 2023.

[8] Y.-Y. Lin, C.-B. Yang, and K.-S. Huang, "The prediction of the most suitable contract in non-competitive bridge bidding," *Proceedings of the 28th International Conference on Technologies and Applications of Artificial Intellegence*, Yunlin, Taiwan, 2023.

[9] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, Vol. 32, 2019.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830, 2011.

[11] J. Rong, T. Qin, and B. An, "Competitive bridge bidding with deep neural networks," *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, Montreal, Canada, 2019.

[12] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, Vol. 529, No. 7587, pp. 484–489, 2016.

[13] V. Ventos, Y. Costel, O. Teytaud, and S. T. Ventos, "Boosting a bridge artificial intelligence," *Proceedings of the 2017 IEEE 29th International Conference on Tools with Artificial Intelligence*, pp. 1280–1287, IEEE, 2017.

[14] J. Von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[15] C.-K. Yeh, C.-Y. Hsieh, and H.-T. Lin, "Automatic bridge bidding using deep reinforcement learning," *IEEE Transactions on Games*, Vol. 10, No. 4, pp. 365–377, 2018.

[16] X. Zhang, R. Lin, Y. Bo, and F. Yang, "The synergy of double neural networks for bridge bidding," *Mathematics*, Vol. 10, No. 17, p. 3187, 2022.