Towards Human-Level Performance in Solving Double Dummy Bridge Problem

Szymon Kowalik and Jacek Mańdziuk(✉)

Faculty of Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Poland
mandziuk@mini.pw.edu.pl

Abstract. Double Dummy Bridge Problem (DDBP) is a hard classification problem that consists in estimating the number of tricks to be taken by N-S pair during a bridge game. In this paper we propose a new approach to DDBP which utilizes convolutional neural networks (CNNs) and a dedicated matrix representation of the problem, suitable for the CNN application. Following previous studies on the application of neural networks to DDBP, we take a knowledge-free approach, i.e. the CNN models are trained with no use of any expert knowledge or explicitly indicated bridge rules. As a result, two models are derived: a baseline CNN model and its ensemble refinement. The results are compared with two former neural network approaches, showing significant superiority of the CNN-based solution. Depending on the type DDBP deal, i.e. trump or notrump, our approach either outperforms or is slightly inferior to the outcomes of human bridge grandmasters solving DDBP. This state-of-the-art performance is complemented in the paper with an analysis of the internal structure (weight patterns) of the trained CNNs, which partly explains the underlying classification process.

Keywords: Deep learning · Convolutional neural network · Double dummy bridge problem · Classification

1 Introduction

The Game of Bridge. Contract bridge (or simply bridge) - one of the most popular classic card games - is an interesting Artificial Intelligence (AI) challenge. Among others, incomplete information about game state and cooperation of players in pairs can be pointed out as demanding aspects of this game. Creating a master-level bridge program is therefore a difficult task.

Formally, bridge is a trick-taking card game for four players. It begins by dealing a standard 52-card deck to the players. Each card has its value (from lowest to highest: 2, 3, ..., 10, J - *Jack*, Q - *Queen*, K - *King*, A - *Ace*) and suit (♠ - *spades*, ♡ - *hearts*, ♢ - *diamonds*, ♣ - *clubs*). The players are forming two pairs playing against each other. They are marked according to their positions

T. Mantoro et al. (Eds.): ICONIP 2021, LNCS 13111, pp. 15–27, 2021.
https://doi.org/10.1007/978-3-030-92273-3_2

at the table: N - North and S - South (first pair); E - East and W - West (second pair). Following bridge related literature, we will refer to the set of 13 cards possessed by a given player as his/her *hand*.

A bridge game begins with a bidding phase that aims at establishing a contract which is defined as the number of tricks to be taken by the partners (within a pair) assuming a given *trump* (TR) suit or its absence - the so-called *notrump* (NT) contract. After an auction the trick-taking part of game begins (aka play phase). Each *trick* consists of four cards - one played by each player. The player with the highest card takes the trick. However, any card in TR suit is considered to be higher than any card in other suits. The team that made the highest bid during auction tries to win at least as many tricks as it was declared in the contract, otherwise - it loses. More detailed bridge rules can be found in [16].

Definition and Significance of Double Dummy Bridge Problem (DDBP). The key issue during a bidding phase is to identify opponents' and partner's hands and determine the highest possible contract based on this knowledge. Experienced bridge players are able to deduce a partial distribution of cards among players based on the course of bidding. This allows to estimate the most probable variants of card locations, evaluate them and determine the optimal contract.

As proposed in [11], a similar approach can be imitated by a bridge playing program. To this end the DDBP was defined as an auxiliary problem. DDBP consists in answering the question of *"How many tricks can be taken by N-S pair assuming that location of each card is known and each player plays in an optimal way?"*. Please note that considering possible locations of certain key cards mentioned above is equivalent to solving a number of DDBP instances related to their possible distributions. This observation led to a simulation-based bidding approach that considers numerous possible cards distributions and their DDBP outcomes. Such an approach was successfully implemented in a bridge-playing program utilizing the DDBP solver [6].

The assumptions made in DDBP make the problem deterministic. However, as indicated below in Sect. 2, DDBP proven to be a demanding machine learning challenge due to its high complexity and strong sensitivity to even subtle changes in input data. For example, swapping two seemingly minor cards can result in a significant change in the DDBP outcome [18].

Contribution. The main contribution of this work is threefold. Firstly, we verify the suitability of CNNs to solving complex, combinatorial problem known as the DDBP. Secondly, we compare the efficacy of CNN DDBP solver with shallower neural network architectures considered in the past - a multilayer perceptron (MLP) and its combination with an autoencoder (MLP-AE). Thirdly, we demonstrate the state-of-the-art AI performance in solving DDBP, which in certain problem settings outperforms the results of top human bridge players, while in other settings narrows the gap between humans and AI.

2 Related Literature

Fast DDBP solver relying on the so-called partition search [5] is a core element of Ginsberg's Intelligent Bridgeplayer (GIB) program [6]. GIB was solving DDBP to find an optimal contract based on Monte Carlo (MC) simulation of various possible double dummy distributions of the key cards among four players. The program was twice winning a title of the World Computer Bridge Champion in the late 1990s [7]. The approach was further developed in [1] leading to significant reductions of the search tree and computation time.

Despite over 20 years of research, bridge-playing programs based on this idea still determine the state-of-the-art.[1] Recent research also focused on a bidding phase using deep learning techniques and double dummy analysis [20,21]. In parallel, the use the Recursive Monte Carlo method instead of MC search (which tends to stuck in local extremes [2]) was also proposed.

When it comes to DDBP itself, several neural network approaches to solving this problem were considered over the years. The key ones, from the perspective of this study, are briefly summarized below. The main rationale behind their usage is an extremely fast inference phase of neural models. Once trained, the network is capable to solve millions of deals in a fraction of time required by exact methods, e.g. partition search [5].

Multilayer Perceptron (MLP) approach to DDBP was initially proposed in [17], further developed in several subsequent papers and summarized in [18], where results of a direct comparison with human grandmasters solving DDBP task within 30 s time per instance were presented.

One of the critical factors in this research was effective representation of a hand in the input layer. Furthermore, it was noticed that TR and NT deals are significantly different, thus require separate training in order to achieve the best fit of each resulting model. Additionally, it was observed that the most effective approach in TR contracts is to assume one particular TR suit in all deals (without loss of generality - a spade suit ♠). Deals with other TR suit can be considered by simply mapping a real TR suit to ♠.

The results obtained with MLP, presented in the top row of Table 1, indicate high model performance which, however, demonstrates certain difficulties in determining the exact number of tricks. It is also apparent that the case of NT contracts is more difficult than the case of TR contracts. This discrepancy is caused by the way in which neural networks solve DDBP, i.e. basing solely on the analysis of the distribution of cards among players. Professional players, on the other hand, in addition to statistical hand analysis also benefit from mental simulation of the play phase based on their experience, which is of particular importance in NT deals [13].

[1] The 2019 World Computer Bridge Champion, Micro Bridge (http://www.osk.3web.ne.jp/~mcbridge/index.html), uses MC simulations and DDBP solver.

Autoencoder (AE-MLP). DDBP was revisited in [14] with the use of MLP combined with a shallow autoencoder. The model consists of two parts - pre-trained encoding layers that provide an efficient representation of the problem and fully connected layers responsible for further inference. Please consult [14] for the details. The efficacy of AE-MLP is similar to that of MLP (see Table 1).

Table 1. Summary of literature results in comparison with the results of human professional bridge players (bottom row) achieved in experiments described in [13]. For each combination of model and contract type three error measures are presented, from left to right referred to as (acc_2 | acc_1 | acc_0) that indicate the percentage of deals in which the error did not exceed two, one and zero tricks (exact result), respectively. This notation was proposed in [18] and is followed in this paper for the sake of comparability.

Model	NT Contracts	TR Contracts
MLP	97.34|84.31|37.80	99.88|96.48|53.11
AE-MLP	96.63|86.18|41.73	99.72|95.33|51.28
Human grandmasters	94.74|88.30|73.68	88.34|81.63|53.06

3 Proposed CNN-Based Approach to DDBP

3.1 Effective DDBP Coding

The key concept contributing to the overall efficacy of the proposed solution is innovative, CNN-plausible problem representation. It extends the idea proposed in the Poker-CNN model [19] and relies on the following assumptions [9]: (a) each card is represented by fixed matrix element; (b) spatial arrangement of matrix elements reflect certain relations between them (e.g. neighbouring elements correspond to cards of similar strength); (c) channels refer to players or suits. Initial tests showed that the first option - assigning channels to players - is more advantageous, thus it has been utilized in presented experiments.

Several DDBP encodings were proposed and tested until selecting the most effective variant, presented in Fig. 1. The encoding matrix consists of 13 columns assigned to cards' ranks and 12 rows corresponding to suits. Four channels are used, one per each player. The matrix has three parts, four rows each:

In rows 1–4 the binary coding is used. Each matrix element has a value of 1 for a hand containing a card in the corresponding rank and suit, or 0 otherwise.

Rows 5–8 are structured in a similar manner with positive values corresponding to the rank of the card to which they refer to. Each cell of this matrix fragment can thus take one of 14 values uniformly distributed in the range $\langle 0, 1\rangle$, the lowest of which - 0 - represents the absence of a given card. The stronger the card, the higher the number representing it.

Coding in rows 9–12 is derived from rows 5–8 by shifting positive values to the left, replacing zero values. So in this case, the location of the value in the column is not related to the card rank. It is only represented by a numeric value.

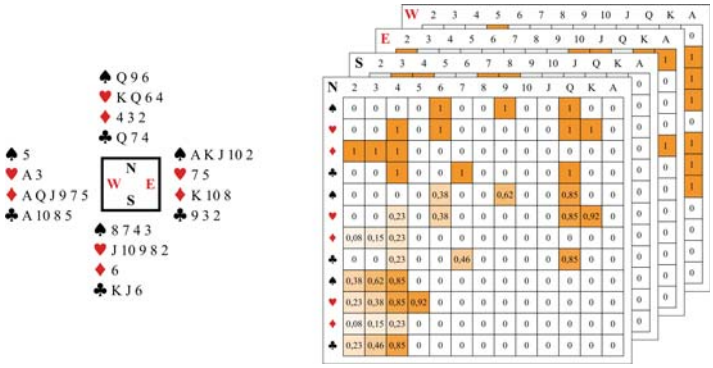

Fig. 1. Example of the DDBP encoding. It refers to the deal presented in the left part of the figure. Channels are illustrated as layers. In the upper left corner of each layer, the corresponding player is indicated.

Unlike in the previous approaches with MLP [18] and AE-MLP [14] the key aspect of proposed coding is data redundancy. Presentation of the same information in different ways facilitates more complex reasoning process (compared to previous architectures) implemented in the CNN model.

3.2 Baseline CNN-Based DDBP Solver

Our CNN model relies on AlexNet [10], but is heavily adapted to the considered problem. Figure 2 presents the model architecture and Table 2 its hyperparemeter values. The input layer is fed with matrices containing the DDBP instance, encoded in a way described in Sect. 3.1. The output layer consists of 14 neurons, each corresponding to one of the possible DDBP outcomes.

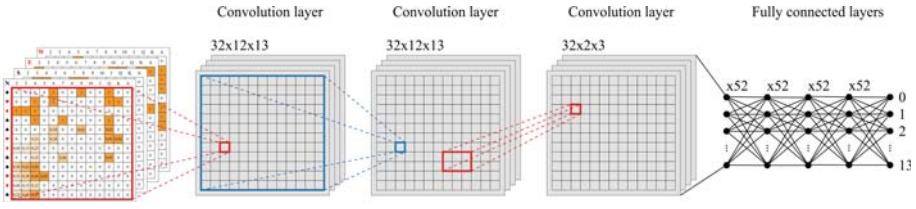

Fig. 2. Baseline CNN model architecture used in the experiments - TR contract variant.

The model is composed of 3 or 4 convolutional layers - depending on the variant (specialized for TR or NT contracts, respectively). The first two convolutional layers use filters of the same size as the input data. This approach is intended to support feature extraction with respect to the full problem knowledge (e.g. suits length determination or individual cards strength identification).

This is not a typical approach for CNNs, but in this case it turns out to be valid (as confirmed by the results). A possible explanation is attributed to the fact that the input data is also not structured in a typical way. Full knowledge of the problem seems to be much more important in this case than in image analysis - a typical CNNs domain of application - that relies on detecting patterns in chunks of the input data.

Deeper convolutional layers serve just this purpose: the filters used there have a significantly smaller size (2×3) and extract local, non-obvious features. All convolutional layers include zero-padding to keep the matrix size equal at their input and output. In Fig. 2, zero-padding is omitted for the sake of clarity of the presentation. The use of pooling layers was abandoned as they cause a reduction in the size of the problem representation, which is relatively small in terms of CNNs. Consequently, in preliminary tests we observed that the networks containing pooling layers perform visibly worse than networks without these layers. Typically for CNNs, convolutional layers are followed by fully connected layer (cf. Table 2).

Table 2. Baseline model hyperparameters values.

Hyperparameter	Value	
	NT Contracts	*TR Contracts*
Convolutional layers		
Number of layers	4	3
Filter sizes in successive layers	$(12 \times 13), (12 \times 13), (2 \times 3), (2 \times 3)$	$(12 \times 13), (12 \times 13), (2 \times 3)$
Number of filters	32	
Activation function	SELU	
Fully connected layers		
Number of layers	4 and output layer	
Number of neurons in successive layers	$52, 52, 26, 26, 14$	$52, 52, 52, 52, 14$
Activation function	ELU	Softplus

3.3 Ensemble of Classifiers

We also attempted to increase classification accuracy by using an ensemble of classifiers consisting of 10 independently trained instances of the above-described baseline CNN model. Two different types of training set construction for each component model were considered: (1) all instances of the baseline model were *trained on the full training set* (the differences between the models were caused by random weight initialization); (2) *bagging* [3].

Additionally, the influence of the voting method on the ensemble results was also investigated. Two approaches were tested: (1) *hard* (*majority*) *voting*, where

each individual model votes for the class and the ensemble result is determined by a majority vote, and (2) *soft voting* involving the summation of the probabilities assigned to each class by individual classifiers - the class with the highest sum becomes the ensemble outcome [8].

4 Experimental Evaluation

The Dataset. Using the GIB program, M.L. Ginsberg generated a library of DDBP solutions [4], hereafter referred to as the GIB library. The set consists of exact problem solutions for over 700 000 randomly generated bridge deals. For each deal, the number of tricks to be taken by N-S pair under the DDBP assumptions was calculated. It is given for all combinations of the TR suit and a player making an opening lead, which gives a total of 20 values for a single deal.

The GIB library has been used in all previous research on neural networks in DDBP referenced in this paper. For the sake of results comparability we follow the training/test set construction proposed in previous works [15,18]. Namely, deals numbered from 1 to 100 000 form a *training set* and deals numbered from 600 001 to 700 000 form a *test set*. Additionally, a *validation set* consisting of deals numbered from 500 001 to 600 000 is used.

Baseline Model Tuning. We adopted the following tuning methodology [9]. First, several hyperparameters were simultaneously tuned to discover their impact on model quality. Once a significant hyperparameter was detected, a second phase aimed at finding its optimal value proceeded. It was discovered using the best model configuration so far. This process was repeated until getting a model that no longer showed significant performance improvement by further tuning. The greatest impact on the performance of the trained model comes from the size of the filters in the convolutional layers and the number of these layers.

4.1 Results

Baseline Model. For each of the two deal types (TR/NT) 10 independent experiments were performed. The aggregated results are presented in Table 3.

Table 3. The average out-of-sample results obtained in 10 tests of the baseline model (Mean) vs the state-of-the-art results (SOTA). Higher outcomes are bolded.

	NT Contracts	TR Contracts
Mean	**98.03** | **93.17** | **57.24**	**99.89** | **97.84** | **58.42**
St. dev.	00.18 | 00.44 | 00.89	00.02 | 00.21 | 00.90
SOTA	96.63 | 86.18 | 41.73	99.88 | 96.48 | 53.11

The first observation is the superiority of the CNN model over state-of-the-art (SOTA) literature results. It is worth noting that in case of NT contracts - the

more challenging variant - the improvement is significant: 15 p.p. with respect to accuracy measure (acc_0). Actually, the CNN model yields very similar acc_0 results for NT (average 57.24%) and TR (average 58.42%) contracts. However, the difference between these variants is clearly evident when one compares the percentage of deals where the estimation error did not exceed one or two tricks (acc_1 and acc_2, resp.). This confirms higher level of complexity in the case of NT contracts, which manifests in higher errors. The proposed solution is stable - the standard deviation of accuracy (acc_0) does not exceed 1 p.p., and is naturally even lower for acc_1 and acc_2.

In summary, the baseline model is a strong estimator of the number of tricks to be taken by the N-S pair and provides a promising starting point for the ensemble approach discussed below.

Ensemble of Classifiers. Table 4 presents the results of an ensemble of classifiers built on 10 instances of the baseline model. According to the assumptions described in Sect. 3.3, four variants were considered.

Table 4. Out-of-sample results of four variants of classifier ensembles. Each ensemble is built on 10 instances of the baseline CNN model. The best results are bolded. The average results of the 10 baseline models (Mean) and the best literature results (SOTA) are outperformed by the best ensemble variant (Full/Soft). Compared to results of professional players, the Full/Soft variant is inferior only in acc_0 measure for NT deals.

Training set	Voting	NT Contracts	TR Contracts
Full	Hard	98.55 | 95.16 | 63.46	99.93 | 98.79 | 63.62
Full	Soft	**98.61** | **95.39** | 64.13	**99.94** | **98.83** | **63.83**
Bagging	Hard	98.08 | 93.68 | 58.40	99.91 | 98.27 | 60.37
Bagging	Soft	98.15 | 94.03 | 59.08	99.91 | 98.38 | 60.87
Mean		98.03 | 93.17 | 57.24	99.89 | 97.84 | 58.42
SOTA		96.63 | 86.18 | 41.73	99.88 | 96.48 | 53.11
Human grandmasters		94.74 | 88.30 | **73.68**	88.34 | 81.63 | 53.06

In all cases, the results improved significantly over a single instance of the baseline model. It is worth observing that ensembles composed of models trained on the full training set perform better than those using bagging. This implies that the size of training set is crucial for the discussed problem and using only 2/3 of the original set deteriorates model quality. Moreover, visible improvement in ensemble performance over a single model confirms the non-trivial nature of DDBP as already noted in previous research [18].

The voting method has less influence on the results, nevertheless it is possible to indicate that soft voting performs better than hard voting in both regarded cases considering all measures. Therefore, one can conclude that *even if the model*

outputs an incorrect result, it assigns a relatively high probability to the correct number of tricks to be taken by N-S pair.

Even though the ensemble performance improvement relative to a single model was certainly expected, its degree is surprisingly large. The accuracy (acc_0) increases by more than 5 p.p. (to 63%) and almost 7 p.p. (to 64%) for TR and NT contracts, resp. *In the latter case the results excelled human grandmaster scores by* 10 *p.p.* This also resulted in an outperforming the TR model by the NT one with respect to the acc_0, which is reported for the first time in the literature.

4.2 Analysis of Weight Structures of Trained Baseline CNN Models

Following previous studies, an analysis of weight structures of trained baseline models was performed in attempt to *explain the model performance*. In particular, the first convolutional layer filters, in both TR and NT models were examined after training. The goal of this process is to relate patterns found in these filters to the common bridge knowledge. An intriguing research question is whether the knowledge gained by CNN models during training will be in line with human expert knowledge. Attention was given to finding analogous patterns to those identified in previous neural network studies [12,15] as well as new, *previously unobserved ones - specific to CNN-based approach*.

As a general remark, it is worth emphasizing that the analyzed filters are applied multiple times to the matrix representing DDBP, each time with different offsets. Consequently, the structure of the filters should not be expected to explicitly correspond to the selected DDBP coding, thus the search for patterns should rather focus on the spatial relationships between the filter weights.

The contents of the exemplary filters are shown in Fig. 3. At first glance one can see that filter elements form mainly two types of patterns - *vertical stripes* and *horizontal stripes*. A *stripe* is defined as a row or column (or part thereof) that contains mostly elements with absolute values significantly higher than its surroundings and/or elements with values of the same sign. It can be deduced that vertical stripes are used to detect cards of certain ranks, while the horizontal ones have to do with suits. More detailed information on the identified patterns is highlighted below. These patterns have been identified in networks specialized in processing both TR and NT contracts.

Patterns consistent with observations from previous studies:

Vertical stripes on the right side of the filters (e.g. filter #2). Due to their location, it can be hypothesized that they are responsible for identifying cards of the highest rank, aka *honors*. Their importance (i.e. frequency of occurrence and distinctness) is greater for NT contracts than for TR ones, which is in line with the expert knowledge - having high-ranked cards is more important in NT deals [16].

A single vertical stripe in the far right column (e.g. #20). This pattern is presumably used to identify Aces - the strongest cards in a deck whose significance is widely supported by the expert knowledge.

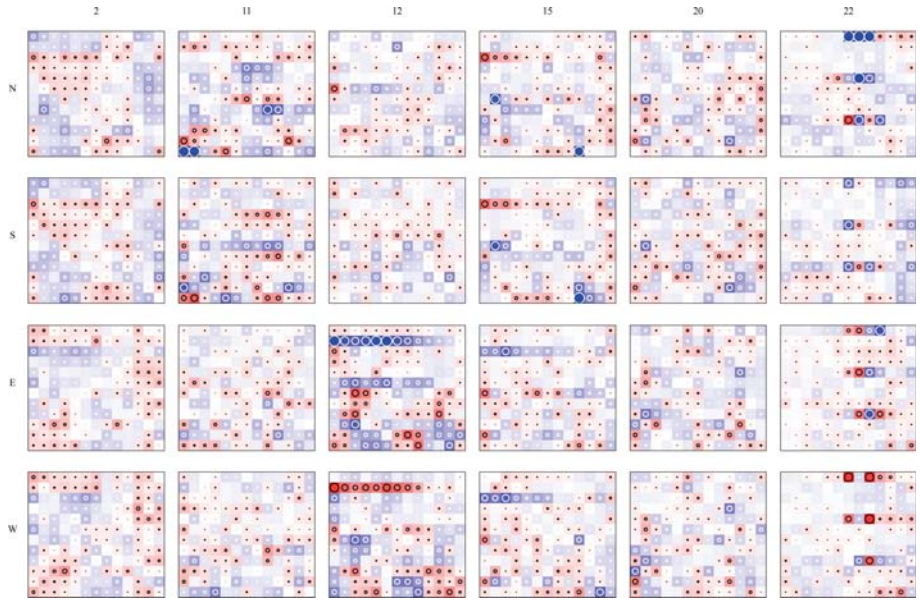

Fig. 3. Visualization of example filters in the first convolutional layer - a variant of TR contract network. The absolute value of each matrix element (filter weight) is represented by the cell color intensity and the circle radius. Positive value is indicated by red cell and black circle, negative value is marked with blue cell and white circle. (Color figure online)

Long horizontal stripes whose weights approximate the card hierarchy (e.g. #12). For TR deals, the hierarchy is reversed, which can be interpreted as detecting unfavorable patterns for a given pair of players that may have a positive effect on their opponents situation.

Long horizontal stripes with relatively low absolute values, but clearly distinguishable along the entire filter length, occurring only in the case of TR contracts (e.g. #15). They can be used to detect all cards of a particular suit in a hand, which is a crucial information for such deals.
Newly observed patterns:

Horizontal stripes repeatable every 4 rows (e.g. #22). This is in line with the structure of the input data. Correlations between various representations of the same information have been correctly detected.

Accumulation of relevant elements in the lower left corner of the filter (e.g. #11). These patterns refer to elements in the last four rows of the DDBP coding. As described in Sect. 3.1, for these elements, the location in the columns does not matter, and all non-zero values are shifted left. The presence of such pattern suggests that the network captured the high significance of the data in this part of the matrix during training.

Associating the sign of filter elements' values to players (e.g. #15). Elements of the pattern have the same sign for players within one pair and the opposite sign for players within the other pair. Hence, the network identifies the positive impact of certain card combinations on the situation of one pair and, at the same time, its negative impact for the other pair.

The above overview demonstrates that *despite knowledge-free training regime* CNNs are able to infer extensive knowledge about the game of bridge, consistent with expert experience. Most of the patterns found in previously applied neural models were also identified in our experiments. Additionally, some new patterns, arising mainly from a new way of data representation and processing, were discovered. What is more, several patterns described above can be often found within a single filter (e.g. #2). At the same time - in contrast to previous models - no filters with structures appearing to be fully random were discovered. These observations, together with excellent performance of the baseline model, suggest that deep CNNs are able to infer much more information about the DDBP than shallower MLP and AE-MLP models.

5 Conclusions and Future Work

In this paper a new approach to solving DDBP relying on CNNs was proposed. The results outperform those of the previous SOTA reference models, i.e. MLP and AE-MLP, by a significant margin. Hence, the primary conclusion of this research is that deep convolutional networks are very well suited to modeling the DDBP.

The experiments confirm the previous conclusion regarding the difference between NT and TR contracts. Despite similar accuracy (acc_0) in these two variants, it is shown that if the prediction is incorrect, the error is statistically greater for NT deals than for TR ones. What is more, a model specialized for NT contracts requires a more complex architecture in order to obtain best results, which further confirms the more challenging nature of this DDBP variant.

The internal structure analysis of the trained networks identified patterns with similar roles as those found in the MLP and AE-MLP models, albeit in the case of CNNs they are, in general, less straightforward and harder to explain. This is due to the fact that deeper networks are capable of making more complex inference, which positively influences the quality of the model but negatively impacts its interpretability.

Application of an ensemble of classifiers yielded significant improvement over the performance of a baseline model. The range of the advancement of results is yet another indication of high complexity of the considered problem.

Finally, a comparison of results presented in Table 4 reveals that proposed solution outperforms the results of top human players (bridge grandmasters) in TR deals. It is, however, still inferior in NT contracts, albeit in the exact measure (acc_0) only.

Acknowledgements. Studies were funded by BIOTECHMED-1 project granted by Warsaw University of Technology under the program Excellence Initiative: Research University (ID-UB).

References

1. Beling, P.: Partition search revisited. IEEE Trans. Comput. Intell. AI Games **9**(1), 76–87 (2017)
2. Bouzy, B., Rimbaud, A., Ventos, V.: Recursive Monte Carlo search for bridge card play. In: 2020 IEEE Conference on Games (CoG), pp. 229–236 (2020)
3. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
4. Ginsberg, M.L.: Library of double-dummy results. http://www.cirl.uoregon.edu/ginsberg/gibresearch.html
5. Ginsberg, M.L.: Partition search. In: Shrobe, H., Senator, T. (eds.) Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, vol. 2. pp. 228–233. AAAI Press, Menlo Park (1996)
6. Ginsberg, M.L.: GIB: Steps toward an expert-level bridge-playing program. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 1999), pp. 584–589 (1999)
7. Ginsberg, M.L.: GIB: imperfect information in a computationally challenging game. J. Artif. Intell. Res. **14**, 303–358 (2001)
8. Kim, J., Choi, S.: Automated machine learning for soft voting in an ensemble of tree-based classifiers. In: ICML Workshop on Automatic Machine Learning (AutoML), Stockholm, Sweden (2018)
9. Kowalik, Sz.: Deep learning in Double Dummy Bridge Problem. Master's thesis, Warsaw University of Technology, Warsaw, Poland (2021)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS 2012, vol. 1, pp. 1097–1105. Curran Associates Inc., Red Hook (2012)
11. Levy, D.N.: The million pound bridge program. In: Levy, D., Beal, D. (eds.) Heuristic Programming in Artificial Intelligence: The First Computer Olympiad, pp. 95–103. Ellis Horwood, Chichester (1989)
12. Mańdziuk, J., Mossakowski, K.: Looking inside neural networks trained to solve double-dummy bridge problems. In: 5th Game-On International Conference on Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004), Reading, UK, pp. 182–186 (2004)
13. Mańdziuk, J., Mossakowski, K.: Neural networks compete with expert human players in solving the double dummy bridge problem. In: 2009 IEEE Symposium on Computational Intelligence and Games, pp. 117–124, September 2009
14. Mańdziuk, J., Suchan, J.: Solving the double dummy bridge problem with shallow autoencoders. In: Cheng, L., Leung, A.C.S., Ozawa, S. (eds.) ICONIP 2018. LNCS, vol. 11304, pp. 268–280. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04212-7_23
15. Mańdziuk, J., Suchan, J.: Who should bid higher, NS or WE, in a given bridge deal? In: 2019 International Joint Conference on Neural Networks, pp. 1–8 (2019)
16. Manley, B., Horton, M., Greenberg-Yarbro, T., Rigal, B. (eds.): The Official Encyclopedia of Bridge, 7th edn. American Contract Bridge League Inc (2011)

17. Mossakowski, K., Mańdziuk, J.: Artificial neural networks for solving double dummy bridge problems. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 915–921. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24844-6_142
18. Mossakowski, K., Mańdziuk, J.: Learning without human expertise: a case study of the double dummy bridge problem. IEEE Trans. Neural Netw. **20**(2), 278–299 (2009)
19. Yakovenko, N., Cao, L., Raffel, C., Fan, J.: Poker-CNN: a pattern learning strategy for making draws and bets in poker games using convolutional networks. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2016), pp. 360–367 (2016)
20. Yeh, C.K., Hsieh, C.Y., Lin, H.T.: Automatic bridge bidding using deep reinforcement learning. IEEE Trans. Games **10**(4), 365–377 (2018)
21. Zhang, X., Liu, W., Yang, F.: A neural model for automatic bidding of contract bridge. In: 2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 999–1005 (2020)