Learning Without Human Expertise: A Case Study of the Double Dummy Bridge Problem

Krzysztof Mossakowski and Jacek Mańdziuk, *Member, IEEE*

***Abstract*—Artificial neural networks, trained only on sample deals, without presentation of any human knowledge or even rules of the game, are used to estimate the number of tricks to be taken by one pair of bridge players in the so-called double dummy bridge problem (DDBP). Four representations of a deal in the input layer were tested leading to significant differences in achieved results. In order to test networks' abilities to extract knowledge from sample deals, experiments with additional inputs representing estimators of hand's strength used by humans were also performed. The superior network trained solely on sample deals outperformed all other architectures, including those using explicit human knowledge of the game of bridge. Considering the suit contracts, this network, in a sample of 100 000 testing deals, output a perfect answer in 53.11% of the cases and only in 3.52% of them was mistaken by more than one trick. The respective figures for *notrump* contracts were equal to 37.80% and 16.36%. The above results were compared with the ones obtained by 24 professional human bridge players—members of The Polish Bridge Union—on test sets of sizes between 27 and 864 deals per player (depending on player's time availability). In case of suit contracts, the perfect answer was obtained in 53.06% of the testing deals for ten upper-classified players and in 48.66% of them, for the remaining 14 participants of the experiment. For the *notrump* contracts, the respective figures were equal to 73.68% and 60.78%. Except for checking the ability of neural networks in solving the DDBP, the other goal of this research was to analyze connection weights in trained networks in a quest for weights' patterns that are explainable by experienced human bridge players. Quite surprisingly, several such patterns were discovered (e.g., preference for groups of honors, drawing special attention to *Aces*, favoring cards from a trump suit, gradual importance of cards in one suit—from *two* to the *Ace*, etc.). Both the numerical figures and weight patterns are stable and repeatable in a sample of neural architectures (differing only by randomly chosen initial weights). In summary, the piece of research described in this paper provides a detailed comparison between various data representations of the DDBP solved by neural networks. On a more general note, this approach can be extended to a certain class of binary classification problems.**

***Index Terms*—Double dummy bridge problem (DDBP), example-based learning, feedforward neural networks, game of bridge, knowledge-free approach, knowledge representation.**

I. Introduction

THE game of bridge is one of the best known card games. There are many interesting aspects of this game and one of them is the estimation of hand's strength. The so-called double dummy bridge problem (DDBP) considered in this paper consists in answering the question about the number of tricks to be taken by a given pair of players on condition that all four hands are revealed and all players play optimally (see Section II-B for more details).

In this paper, the description and the results of experiments with using artificial neural networks as DDBP estimators are presented. In some of the experiments, human methods of estimating hand's strength are also used to check if this additional input data based on the human knowledge of the game could improve the quality of the results.

In spite of very promising numerical outcomes, the other goal was to check how the information about the problem being solved is internally represented by the networks in their weights spaces. It turned out that for some types of problem representations in the input space, it was possible to find interesting, repeatable patterns of connection weights, e.g., the highlights of the *Aces* (the top cards in a deal) or the preference for the trump suit cards or the preference for the suit honors (the top cards in a suit), etc. Most of these patterns can be justified by human knowledge of the game of bridge.

It is worth to underline that the focus of this paper is on verification of neural networks' abilities to learn the evaluation function for the DDBP rather than perfect solving of this problem. Actually, the DDBP can be effectively solved using sophisticated exhaustive search methods without applying the computational intelligence (CI) techniques [1].

On the other hand, due to numerous nuances of the bidding and playing phases in bridge, the DDBP is, in our opinion, an interesting and challenging problem for knowledge-free, example-based learning methods. The so-far attempts to solve the DDBP with the use of neural nets, based exclusively on raw data [2], [3], were unsuccessful. The authors of the above cited papers stated that

> Neural networks in their purest form take the raw input data, and learn to construct appropriate outputs without doing anything more than recalculating the weights of the connections between their nodes. However, it is in practice considerably more efficient to perform a certain amount of pre-processing on the input, so as to construct values representing features which humans consider important in the domain.

These precomputed features concerned, for example, "*specific high cards or total suit lengths*." On the contrary to the above statement, our claim is that with appropriate choice of neural architecture, and in particular the input representation of a deal, it

Manuscript received August 09, 2007; revised August 11, 2008; accepted August 24, 2008. First published January 13, 2009; current version published February 06, 2009.

The authors are with the Faculty of Mathematics and Information Science, Warsaw University of Technology, 00–661 Warsaw, Poland (e-mail: mossakow@mini.pw.edu.pl; mandziuk@mini.pw.edu.pl).

Digital Object Identifier 10.1109/TNN.2008.2005526

is possible to achieve a high score in solving the DDBP without the need of any domain-related preprocessing of the raw data.

The final motivation for writing this paper were comparative results accomplished by professional bridge players in solving DDBP under restrictive time constraints, considering two variants of the problem—the classical one (all four hands are revealed) and the more difficult, though more "realistic" one (two hands of the pair being scored are revealed and the remaining two are hidden—only the *sum* of the opponents' hands is available as a result of subtracting both known hands from the whole deal). Humans visibly outperform neural networks in the classical variant and the *notrump* contracts, but in the remaining three cases (a classical variant and the suit contracts or partly covered variant and *notrump* or suit contracts) the neural networks are very competitive to humans.

In summary, the goal of this paper is twofold. First, it presents a successful application of neural networks to nontrivial classification problem of practical importance. Second, the in-depth exploration of internal networks' structures, and in particular, analysis of how domain-specific knowledge is represented, may hopefully be of more general applicability and interest, extending beyond the bridge community.

Initial results were published in our previous works devoted, respectively, to *notrump* [4], [5] and suit [6] contracts. The experiments described in the above cited papers were restricted to simple neural architectures, namely, the 52 and 104 codings—described in Section V. Preliminary results concerning more elaborate networks were presented in [7].

This paper summarizes previously published accomplishments and presents the latest experimental results. In particular, the efficacy of neural networks is compared here with the results attained by professional human bridge players (including Grand Masters and International Masters). Also, a comparison between *notrump* and suit contracts is presented and possible explanations of better performance accomplished by neural networks in the latter case are proposed. Additionally, a comprehensive overview of the artificial intelligence (AI) applications in bridge domain is presented.

The reminder of this paper is organized as follows. Section II provides a short description of the game of bridge followed by a definition of the DDBP and characteristics of the source data used in the experiments. Section III gives a look at hitherto AI and CI research activities in the domain of computer bridge, starting with a brief chronological presentation of the main accomplishments, followed by a more detailed discussion on selected issues. In particular, previous efforts of applying neural nets in bridge are also presented. In Section IV, details concerning various ways of representing deals as inputs for artificial neural networks used in the paper with some illustrative figures can be found. Section V describes various architectures of neural networks used in the experiments. The schemes of deals selection are described in Section VI. Also, the first conclusions concerning application of neural nets to *notrump* and suit contracts can be found there. Section VII contains analysis of trained networks, emphasizing the existence of some repeatable, characteristic patterns found in their weights. These patterns represent various practical aspects of the game of bridge (e.g., the relative importance of cards within one suit) and can be intuitively explained by human players. In Section VIII, human methods used to estimate strength of a hand are described. Comparison between results achieved by training on sample deals *with* and *without* these estimators (i.e., with and without adding explicit human knowledge about the game) is presented. The observations put forward in Sections VII and VIII are considered the main contribution of the paper. The most interesting results are recapitulated and assessed in Section IX. In particular, neural networks' efficiency is compared here with the results accomplished by ten internationally recognizable bridge players (ranked as Grand Masters, International Masters or Masters, playing in the First or the Second Polish Bridge League), and 14 other professional bridge players (members of the lower ranked bridge teams). Six sample deals illustrating strong and weak points of the trained networks are presented and discussed in Section X. The last section summarizes the main conclusions.

II. Problem Definition

A. The Game of Bridge

Contract bridge, usually known simply as bridge, is a trick-taking card game.

There are four players in two fixed partnerships (pairs). Partners sit facing each other. It is traditional to refer to the players according to their position at the table as *North (N)*, *East (E)*, *South (S)*, and *West (W)*, so *N* and *S* are partners playing against *E* and *W*.

A standard 52-card pack is used. The cards in each suit rank from highest to lowest: *Ace (A)*, *King (K)*, *Queen (Q)*, *Jack (J)*, 10, 9, 8, 7, 6, 5, 4, 3, 2. The dealer deals out all the cards one at a time so that each player receives 13 of them.

Next an auction to decide who will be the declarer takes place. A bid specifies a number of tricks and a trump suit (or that there will be no trumps). The side which bids highest will try to win at least that number of tricks bid, with the specified suit as trumps.

There are five possible trump suits: *spades* (♠), *hearts* (♡), *diamonds* (♢), *clubs* (♣), and "*notrump*," which is the term for contracts played without a trump.

After three consecutive passes, the last bid becomes the contract. The team who made the final bid will now try to make the contract. The first player of this team who mentioned the denomination (suit or *notrump*) of the contract becomes the declarer. The declarer's partner is known as the dummy.

The player to the left of the declarer leads to the first trick. Immediately after this opening lead, the dummy's cards are exposed.

The play proceeds clockwise. Each player must, if possible, play a card of the suit led. A player with no card of the suit led may play any card. A trick consists of four cards, and is won by the highest trump in it, or if no trumps were played, by the highest card of the suit led. The winner of a trick leads to the next.

The aim of the declarer is to take at least the number of tricks announced during the bidding phase. The players of the opposite pair try to prevent him from doing it. Details of scoring depend on the variant of the game. The most popular ones are: the rubber bridge and the duplicate bridge. For more details about the game, please refer, for example, to [8] or [9].

B. The Double Dummy Bridge Problem

The DDBP is not a variant of the game of bridge. It is rather an auxiliary problem closely connected with the bidding phase of the game. More specifically, the problem consists in answering the following question: "How many tricks are to be taken by one pair of players assuming perfect play of all four sides, with all four hands being revealed?"

There is an important difference between solving DDBP and the real bridge playing, since in the latter, the exact placement of most of the cards is unknown. Consequently, in a real play, the player has to calculate probabilities of cards' distributions and choose a strategy with the highest expected outcome. In DDBP, there is no hidden data and the best strategy can be pointed out.

Estimating hand's strength is a crucial aspect of the bidding phase of the game of bridge, since the contract bridge is a game with incomplete information and during the bidding phase each player can see only his/her cards and has to make several assumptions about placement of other cards. This incompleteness of information forces considering many variants of a deal (cards distributions). The player should take into account all these variants and quickly estimate the expected number of tricks to be taken in each case.

In actual play among professional players, the location of crucial cards can be partly inferred from the bidding phase. Considering the above, it is worth to note that assuming any particular variant of cards' location is equivalent to the case of having all four hands revealed, *ergo* the DDBP. This observation, first discussed in [10], led to the following idea: use the assumption of particular cards' placements combined with Monte Carlo simulations and a fast DDBP solver in order to estimate the most probable (the most effective) contract [10]. This idea was used by Ginsberg in his world champion playing program [11].

C. The GIB Library

The data used in solving the DDBP was taken from the GIB Library [12], created by the Ginsberg's Intelligent Bridgeplayer [11]—the above mentioned computer bridge champion in 1998 and 1999.

The GIB Library includes 717 102 deals and for each of them provides the numbers of tricks to be taken by the *NS* pair for each combination of the trump suit (including *notrump* contracts) and the hand that makes the opening lead. Together, there are 20 numbers for each deal (five trump suits by four sides). All these numbers were calculated by the GIB program under the assumption of a perfect play of all players.

In most experiments reported in this paper, 100 000 deals from the library (with numbers from 1 to 100 000) were used for training and another 100 000 ones (numbered from 600 001 to 700 000) were used for testing. Training and testing set sizes were chosen after some preliminary tests that confirmed no further improvement in case of using bigger sets.

D. Representation of Results

All results presented in this paper consist of three numbers $(A|\ B|C)$ representing the fractions in percent of test deals for which the prediction error did not exceed two tricks (A), one trick (B), and zero tricks (C). The most interesting is obviously the last number, i.e., the level of perfect answers, however, it should be emphasized that there are deals for which it is very difficult (even for experienced human players) to point out the correct number of tricks. A small difference in the cards' location, e.g., exchange of two plain cards, can change the result by one or more tricks. Under these circumstances, the two remaining figures (B and C) are also worth focusing on.

III. Previous Works

This section presents an overview of AI and CI papers in computer bridge domain. First, various accomplishments are mentioned very briefly in chronological order. Next, several achievements related to particular aspects of the game are discussed in more detail in the four subsections devoted, respectively, to the bidding phase, the play phase, the DDBP, and application of neural nets in the game of bridge.

Contract bridge has not attracted high attention of AI and CI researchers. For some reason, other games, such as chess or go or even poker, have much longer lists of publications. It is a bit surprising since there are many interesting nuances of the game of bridge from the AI point of view, e.g., imperfect information, cooperation of players in pairs, two completely different phases of the game—the bidding and the play (both should be played excellent to gain the best possible result).

The oldest work that appeared in the literature comes from 1962: Carley wrote a Master's thesis [13] on a computer program playing bridge. In the next year, Berlekamp [14], also in a Master's thesis, successfully solved *notrump* instances of the DDBP. In 1969, Napjus [15], [16] used the machine learning approach to write a bridge program that learned to play any combination of declarer/dummy cards in a single suit. In the beginning of 1970s, Wasserman [17] created a program for bidding. A few years later, Stanier wrote a program for both bidding and playing, which used the knowledge from the bidding phase in the play phase [18]. He also tried to create a strategy for the play stage using information gathered during bidding [19]. The idea of using the bidding phase to help playing was continued by Quinlan [20], who tried to locate missing high cards. In 1983, Throop [21] recapitulated the research in the field of computer bridge programs.

In 1983, Lindelof created a computer-oriented bidding system called COBRA [22]. In the mid 1980s, Berlin [23] invented tactics to solve subproblems of the card combinations in individual suit. In the late 1980s, Wheen [24] created a program solving the DDBP using the $a-b$ minimax algorithm, and MacLeod [25] extended Stanier's idea of taking advantage of the information acquired during the bidding phase. In 1989, Levy [10] gave a recipe for a computer bridge program that would be able to defeat the best human bridge players. The idea was quite simple: use the COBRA bidding system and a fast DDBP solver for the play phase with limiting the problem of imperfect information using the Monte Carlo algorithm.

It may be that the Levy's paper was so influential or the reward of 1 000 000 pounds sterling offered by a world bridge champion Mahmood Zia for a computer bridge program that could beat him was very encouraging, nevertheless, the number of papers devoted to computer bridge started to increase rapidly.

In [26]–[29], Frank created the FINESSE system able to suggest a strategy of play (however, only for a few last tricks,

not the whole deal). He also investigated the Monte Carlo algorithm used by other researchers [30], and proposed other heuristic algorithms for games with imperfect information [31]–[34]. Later he focused his attention on the problem of explaining the strategy found by a computer program using natural language (in English) [35]–[37].

Smith and Nau analyzed the game-tree search algorithms for games with imperfect information and created the Tignum system using the hierarchical task-network planning, which was able to generate significantly reduced game trees [38]–[40]. Their ideas were successfully incorporated into Bridge Baron [41], [42], the best computer bridge program in the mid 1990s.

Ginsberg [43] followed the ideas presented by Levy and used the Monte Carlo algorithm with a very fast DDBP solver in a computer bridge program. He invented the partition search algorithm that allowed to substantially restrict the game tree [44] and created the GIB computer bridge program that was the computer bridge champion in the late 1990s [45], [11].

A. The Bidding Phase

In the first bridge program mentioned in the literature [13], there were only four bidding rules with 13 cases in total. The next attempt to create a computer bridge program [17] allowed using some common patterns of bidding, such as take out double and *Ace* asking. A rule-based bidding system was also created in [46], where the aim was to achieve the beginner's level.

Also, COBRA [22], the first bidding system created with computer's assistance, relied on rules. Tests of this system showed that using it visibly improved the effectiveness of bidding. On the other hand, COBRA is more complicated than "human-type" systems, and for this reason was not used by other researchers [47].

Some of the researchers fixed their attention on the opening bid as a problem, which can be learned from examples. Various methods were used to allow to learn the optimal opening bid: abductive explanation-based learning [48], artificial neural networks [49], [50], probabilistic neural networks with evolutionary programming-based clustering technique [51], or rough-fuzzy set theory [52].

A very interesting issue of the bidding phase is cooperation of players in a pair (*N* with *S* and *W* with *E*). In [53], each player is modeled as an independent, active agent that takes part in the communication process. Also, other researchers represented the bidding phase as cooperation of two agents [54], [55] or cooperation of agents in competing pairs [56]. In [47], an agent-based algorithm was proposed, which was able to achieve, after appropriate learning, a bidding ability close to that of a human expert.

B. The Play Phase

The play phase seems to be much less interesting for the AI researchers than the bidding phase. Smith and Nau were probably the only researchers who created a successful AI approach [38], [40], [41] to the whole play phase.

Other researchers limited their interest to the play of one suit [16], [23], [27], [36], [57]–[59]. In most cases, these AI approaches tried to imitate human strategy of the play by using some "tactics" [27], [60] or "thematic actions" [58].

Frank and Basin [35], [37] created a system that was able to find a strategy of a play, and additionally, a "human" explanation of it. Unfortunately, they could not break some complexity problems and their system worked for a single suit only (cards from other suits were ignored).

C. The Double Dummy Bridge Problem

The DDBP also attracted some interest of researchers. In 1963, Berlekamp [14] invented a heuristic nonexhaustive theorem proving technique for solving DDBP in case of *notrump* contracts. Later other techniques were used, e.g., the $a - b$ minimax algorithm [24] and the hash table [1].

An interesting system identifying complex positions in DDBP was also developed in [61].

D. Applications of Neural Networks

Among other techniques and research tools, artificial neural networks were used for solving bridge problems [62], especially for the bidding phase [63], [64], [49].

Gambäck and Rayner [2], [3] used neural networks as estimators for the number of tricks to be taken in the DDBP. They used two networks, one for *notrump* contracts and one for suit contracts. Each network took as input the sets of cards from the four hands, and as output a set of 14 real numbers, representing the estimated probabilities that 0–13 tricks will be collected by the playing pair. The authors concluded that the networks in their purest form (using only raw input data, without any preprocessing of the input or adding human knowledge in any form) were not able to successfully serve as estimators of the number of tricks. Adding some input data that humans consider important in the domain (called "precomputed feature points" by the authors) improved the results. Unfortunately, there are no numerical results presented in [2] and [3], which does not allow for making any direct comparisons.

On the contrary to the above cited articles, the results of experiments described in this paper suggest that with appropriate representation of a deal, neural networks can be very effective in solving the DDBP relying solely on example-based training and are capable of extracting necessary information from raw data, without any human intervention or the use of human knowledge about the game of bridge.

IV. Representation of a Deal

The following two sections are devoted to the description of the neural network architectures and the learning schemes and presentation of initial results. Particular emphasis is put on the possible ways of problem representation in the input layer, which—due to its relevance—is separately discussed in this section. Presentation of network architectures and the results follow in the next section.

The way a deal was represented turned out to be crucial for the quality of the achieved results. During the experiments, the following four ways of coding a deal were invented and tested.

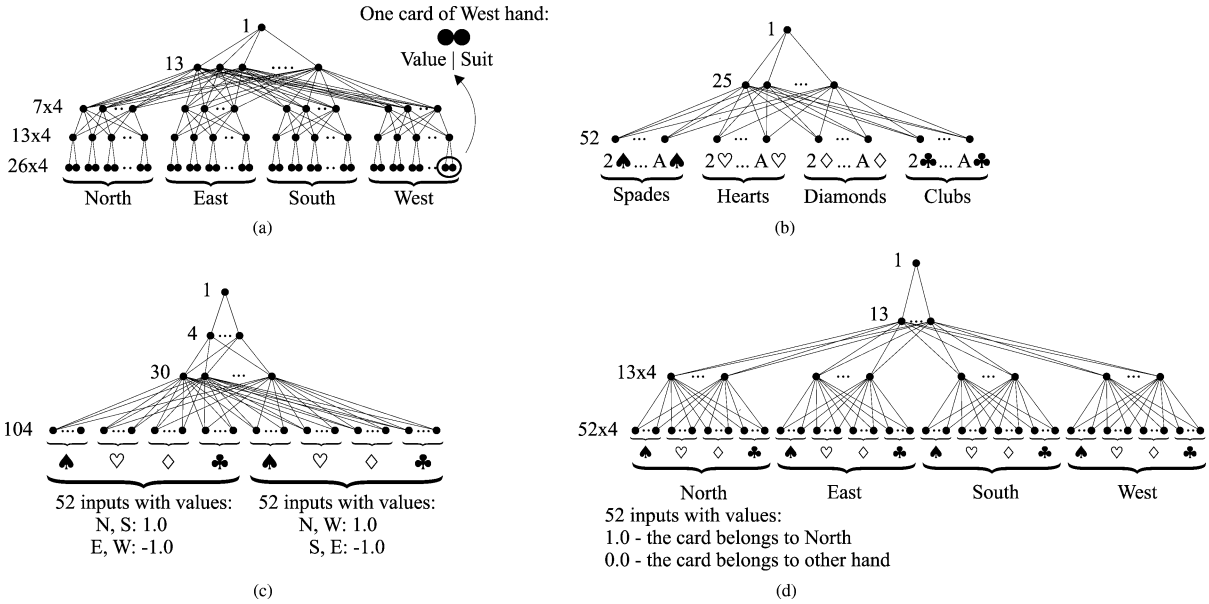


Fig. 1. Example network architectures with different deal representations proposed in Section IV. For 52 and 104 representations, a standard feedforward, fully connected architectures are used. In the networks utilizing 52×4 or 26×4 representations, the first hidden layer (for 52×4 case) or the first two hidden layers (for 26×4 one) are connected selectively. They are responsible for collecting information about individual hands from "scattered" and redundant input. (a) (26×4) representation. (b) 52 representation. (c) 104 representation. (d) (52×4) representation.

A. (26×4)

In the first way of deal representation, 104 input values were used, grouped in 52 pairs. Each pair represented one card. The first value in a given pair determined the rank of a card (*A, K, Q,* etc.) and the second one represented the suit of a card (♠, ♡, ♢, or ♣). Hence, 26 input neurons (13 pairs) were necessary to fully describe the content of one hand [see Fig. 1(a)].

A few schemes of transforming card's rank and suit into real numbers suitable as input values for the network were tested. Finally, the rank of a card was transformed using a uniform linear transformation to the range $[0.1, 0.9]$, with biggest values for *Aces* (0.9) and *Kings* (0.83) and smallest for *three spots* (0.17) and *two spots* (0.1). Some other ranges, e.g., $[0, 1]$ or $[0.2, 0.8]$, were also tested, but no significant difference in results was noticed. A suit of the card was also coded as a real number, usually by the following mapping: 0.3 for ♠, 0.5 for ♡, 0.7 for ♢, and 0.9 for ♣.

B. 52

In the second way of coding a deal, a different idea was utilized. Each input neuron was assigned to a particular card from a deal and a value presented to this neuron determined the hand to which the respective card (assigned to this input) belonged.

There were 52 input values, each representing one card from a deck. Positions of cards in the input layer were fixed, i.e., from the leftmost input neuron to the rightmost one the following cards were represented: $2♠, 3♠, \ldots, K♠, A♠, 2♡, \ldots, A♡, 2♢, \ldots, A♢, 2♣, \ldots, A♣$ [see Fig. 1(b)].

A value presented to the input neuron denoted the hand to which a given card belonged, i.e., 1.0 for *North*, 0.8 for *South*, -1.0 for *West*, and -0.8 for *East*. Interestingly, as came out from further experiments, using the same input value (-1.0) for both *West* and *East* hands improved the results. Moreover, hiding the information about exact cards' assignment in the *NS* pair, i.e., using the input value equal to 1.0 for both *North* and *South* hands, yielded another slight improvement (cf., the first three rows of Table IV).

C. 104

The third proposed way of coding a deal was a straightforward extension of the 52 representation to the 104 one. The first 52 input values represented assignments to pairs exactly in the same way as in the 52 representation (value 1.0 represented *NS* and -1.0 represented *WE*), and the remaining 52 ones pointed out the hand (value 1.0 for *N* and *W* and -1.0 for *S* and *E*). In both groups, the positions of cards were fixed according to the same order [see Fig. 1(c)].

D. (52×4)

The last tested way of coding a deal arose from the results obtained by the networks that used human estimators of hand's strength (discussed in Section VIII). These results suggested that it was difficult for the networks using the above presented ways of coding a deal to extract information about the lengths of suits on hands. On the other hand, this information is crucial, especially for suit contracts, so there was a need to invent another representation of a deal in which the lengths of suits on hands would be perceptible for neural networks, although still basing it on *raw data*.

In this deal coding, 208 input neurons were divided into four groups, one group per hand, respectively, for *N*, *E*, *S*, and *W* players. Four input neurons (one per hand) were assigned to each card from a deck. The neuron representing a hand to which this card actually belonged received input value equal to 1.0. The other three neurons (representing the remaining hands) were assigned input values equal to 0.0. This way, a

hand to which the card was assigned in a deal was explicitly pointed out.

In this representation, one suit on one hand was represented by 13 input neurons. The number of input values equal to 1.0 determined the length of this suit on the hand, so the networks using this representation of a deal had a chance to find shortnesses (which are very important in bridge), especially voids (no cards in a suit) and singletons (one card in a suit).

E. The Trump Suit and the Opening Lead

When talking about the deal representation in the input layer, despite card locations, there are two more issues, which need to be addressed. The first one is the way of pointing out the network which suit is a trump in the experiments where the training and testing sets do not consist only of deals with one specific trump suit (see Section VI-B for details). For the $52, 104$, and (52×4) representations, a trump suit was pointed out by increasing the absolute value of the input neurons assigned to the cards of the trump suit and decreasing it for the cards of all other suits (usually input values for other suits were divided by 2). This method could not be applied to the (26×4) representation, because the suit was represented here by the value presented to one of two input neurons representing the card. Therefore, in this case, one additional neuron was added to the input layer and the value presented to it indicated the trump suit. Not surprisingly, the networks had problems with perceiving this additional information (only one out of 105 input neurons in total was used to represent this information), and the results obtained by the networks trained on deals with various trump suits were worse than the ones obtained by analogous network trained only on deals with one, arbitrarily chosen trump suit. This is why the majority of the results for suit contracts presented in this paper are restricted to one arbitrarily chosen suit (*spades*). Hence, no distinction between the trump suit and the remaining suits in the input layer is necessary.[1]

The other important issue is a way of representing the hand making the opening lead. In some deals, the number of tricks varies when a hand making the opening lead is changed, even if the positions of all cards are fixed (see Section VI-D for details). In the (26×4) representation, four additional input neurons were used. Each of them was assigned to one hand and had a nonzero value if that hand was to make the opening lead. In the remaining representations [$52, 104$, and (52×4)], there was no need for additional input neurons. In these cases, the problem was solved by fixing the positions of input neurons representing a hand making the opening lead, i.e., each deal was presented in the training set twice (e.g., once with the opening lead from the *North* hand and once from the *South*). In the latter case, the deal was "rotated" to assure that always the same input neurons were assigned to a hand that made the opening lead.

F. Comparison of Representations

The best results obtained by the four types of networks for *spades* contracts are presented in Table I. This comparison emphasizes superiority of the (52×4) representation. The result, $99.80|95.54|50.91$, means that the network answered perfectly in over 50% of deals and was wrong by more than one trick only in 4.46% of deals and by more than two tricks in 0.2% of deals. The worst results were obtained by the network using the (26×4) representation. The efficiency of the two remaining ways of coding a deal was comparable to each other.

TABLE I
COMPARISON OF THE BEST RESULTS OBTAINED WITH VARIOUS APPROACHES TO CODING A DEAL FOR *SPADES* CONTRACTS WITH THE OPENING LEAD FROM THE WEST HAND

The Network	Results
(26x4)-(13x4)-(7x4)-13-1	97.67 | 84.24 | 36.82
52-25-1	98.77 | 88.00 | 40.13
104-30-4-1	98.61 | 87.17 | 39.21
(52x4)-13x4-13-1	99.80 | 95.54 | 50.91

V. ARTIFICIAL NEURAL NETWORKS

Due to their ability to generalize knowledge acquired from the training data, artificial neural networks are potentially very well suited to the task considered. It is worth emphasizing that in the majority of the experiments described in this paper, the training data contained only sample deals—represented according to one of the four schemes discussed in the previous section. Only in the experiments in which human estimators of hands' strength were used (presented in Section VIII), additional inputs representing human knowledge of the game were used. Nevertheless, in either case, the networks were not aware of the bridge rules and were trained on raw data in the example-based manner.

Another important feature of artificial neural networks, when considered as estimation tools to be used during the bidding phase of the game of bridge, is their speed. Calculating the expected number of tricks to be taken using a trained network is extremely fast, and can be efficiently carried out for many potential variants of a deal, which need to be considered due to partly hidden information.

In all the experiments, training and testing were performed with the JNNS's [65] assistance.[2]

In most cases, logistic (unipolar sigmoid) activation function was used in all neurons. Only when negative values were presented in the input layer, the hyperbolic tangent (bipolar sigmoid) activation was applied.

All networks were trained using the resilient backpropagation (Rprop) algorithm [66], with the following choice of method's parameters: initial and maximum values of an update-value factor were equal to 0.1 and 50.0, respectively, and weight decay parameter was equal to 0.0001.

A. The Input Layer

The size of the input layer was determined by the chosen way of coding a deal as discussed in the previous section.

B. Hidden Layers

Avoiding the presentation of the human knowledge of the game of bridge in any form was one of the underlying assumptions in all the experiments (except for the ones with using

[1]The above restriction is applied without loss of generality of results, since it is always possible to exchange the positions between the actual trump suit cards and the *spades* suit cards in the input representation.

[2]The Java Neural Network Simulator (JNNS) is a freely available successor to the Stuttgart Neural Network Simulator (SNNS).

TABLE II
COMPARISON OF AN INFLUENCE OF THE NUMBER OF OUTPUT NEURONS ON RESULTS FOR *SPADES* CONTRACTS WITH OPENING LEAD FROM THE WEST HAND

The Network	Number of Outputs	Results
(26x4)-(13x4)-(13x4)-26-13-1	1	96.93 | 80.98 | 33.99
(26x4)-(13x4)-(13x4)-26-14	14	97.35 | 83.06 | 36.02
52-25-1	1	98.77 | 88.00 | 40.13
52-25-1 (proportional)	1	98.66 | 87.41 | 39.98
52-25-14	14	98.05 | 85.69 | 38.66
104-30-4-1	1	98.61 | 87.17 | 39.21
104-30-14	14	97.18 | 82.58 | 35.87
(52x4)-(26x4)-26-13-1	1	99.80 | 95.54 | 50.91
(52x4)-(26x4)-26-14	14	99.02 | 89.78 | 42.05

TABLE III
COMPARISON OF AN INFLUENCE OF THE NUMBER OF HIDDEN NEURONS AND CONNECTIONS FOR *SPADES* CONTRACTS WITH THE OPENING LEAD FROM THE WEST HAND

The Network	Number of Neurons	Number of Connections	Results
(26x4)-(13x4)-(7x4)--13-1	198	845	97.67 | 84.24 | 36.82
(26x4)-(13x4)-(13x4)--26-13-1	248	2483	96.93 | 80.98 | 33.99
52-25-1	78	1325	98.77 | 88.00 | 40.13
52-26-13-6-1	98	1774	98.76 | 87.96 | 40.20
104-30-4-1	139	3244	98.61 | 87.17 | 39.21
(52x4)-(8x4)-8-1	249	1928	99.67 | 94.03 | 47.74
(52x4)-(13x4)-13-1	274	3393	99.78 | 95.00 | 50.03
(52x4)-(26x4)-26-13-1	352	8463	99.80 | 95.54 | 50.91

human estimators of hands' strength, described in Section VIII). This premise was also applied to networks' architectures, especially the topology of connections between the hidden neurons.

The networks used in this research can be divided into two groups. The first group (with 52 or 104 ways of coding a deal) contains fully connected networks, without any dedicated groups of hidden neurons. The second group contains networks with (26×4) or (52×4) representations, where subnetworks responsible for individual hands can be pointed out [see Fig. 1(a) and (d)]. Such structure does not violate the above assumption, since it does not provide any human knowledge of the game, but only defines a deal, i.e., the assignment of cards to hands.

It should be emphasized that there was no direct input information about suits. It means that all the basic information, obvious for human bridge players, had to be autonomously discovered during the training process. This included the existence of four suits, the power of a trump suit, the influence of a rank of a card (the *Ace* is the highest and the *two* is the lowest), the cooperation of players in one pair, etc.

C. The Output Layer

Two ways of transforming the networks' outputs into the number of tricks were applied. In the first case, one output neuron was used, and in order to get the result, decision boundaries were defined (within the range $[0.1, 0.9]$) denoting particular numbers of tricks. For all presented results, these decision boundaries were defined *a priori* and target ranges for all possible numbers of tricks (from 0 to 13) were of pairwise equal length.

A simple experiment with changing decision boundaries was also performed. The idea was to check whether making ranges in the output proportional to the number of deals with specific number of expected tricks instead of using pairwise equal ranges would improve the results. The network $52 - 25 - 1$ using such special output ranges achieved slightly worse result than the same network with pairwise equal output ranges (see rows three and four of Table II).

The other way of transforming the networks' outputs into the number of tricks relied on using 14 output neurons. Each of the output neurons represented one target number of tricks. In the training phase, exactly one out of 14 output values was set to a nonzero value (usually 1.0). In the testing phase, the output neuron with the highest value defined the final prediction (classification).

Although the latter approach seems to be more suitable, in most of the experiments, the networks using 14 output neurons achieved worse results than the corresponding networks having only one output neuron (see Table II).

D. Networks' Sizes and the Learning Speed

Table III contains the comparison of the numbers of neurons and connections in tested architectures. It is interesting to compare the results of networks using the same deal representation but differing by the number of hidden layers, hidden neurons, and connections.

Two networks using the (26×4) representation differ significantly by the number of connections. The bigger network $[(26 \times 4) - (13 \times 4) - (13 \times 4) - 26 - 13 - 1]$ obtained worse results than the smaller one $[(26{\times}4){-}(13{\times}4){-}(7{\times}4){-}13{-}1]$. The reason was overfitting to the training data. In both cases, the same training and testing sets were used. For the smaller network, the results achieved for both sets were comparable. For the bigger one, the results achieved for the training set (not presented) were significantly better than the ones for the testing set.

No overfitting was observed for the remaining networks presented in Table III. Results achieved by networks $52 - 25 - 1, 52 - 26 - 13 - 6 - 1, (52 \times 4) - (13 \times 4) - 13 - 1$, and $(52 \times 4) - (13 \times 4) - 26 - 13 - 1$ are very close to each other in the respective pairs.

Advantage of the $(52{\times}4)$ representation is indisputable. Even a rather small network $(52 \times 4) - (8 \times 4) - 8 - 1$ with less than 2000 connections achieved much better result than any other network with another input coding.

It is interesting to compare the number of iterations required by different types of network architectures in the training phase. The networks using the (26×4) representation required over 100 000 epochs, the ones using the 52 or 104 representations required only about 1000 epochs, and the networks using the (52×4) coding required between 10 000 and 20 000 epochs.

VI. DEALS SELECTION AND RESULTS

Training and testing data were taken from the GIB Library, which contains deals with precalculated numbers of tricks to be taken by one pair of players for each combination of a trump suit and a hand that makes the opening lead (see Section II-C

TABLE IV
COMPARISON OF RESULTS OBTAINED BY THE $52 - 25 - 1$ NETWORK FOR *NOTRUMP* AND SUIT CONTRACTS (INPUT VALUES IN PARENTHESES)

Description	Results
notrump (N: 1.0, S: 0.8, W: -1.0, E: -0.8)	95.81 | 79.95 | 34.02
notrump (N: 1.0, S: 0.8, W: -1.0, E: -1.0)	95.97 | 80.46 | 34.35
notrump (NS: 1.0, WE: -1.0)	96.07 | 80.88 | 34.66
suit contracts (NS: 0.5, WE: -0.5)	98.68 | 87.88 | 40.11
tested on *notrump* contracts	91.64 | 69.21 | 26.06
notrump and suit contracts (NS: 0.5, WE: -0.5)	97.72 | 84.90 | 37.56
tested on suit contracts only	98.57 | 87.24 | 39.43
tested on *notrump* contracts only	94.30 | 75.50 | 30.09
♠ contracts (NS: 1.0, WE: -1.0)	98.77 | 88.00 | 40.13
tested on ♡ contracts	59.18 | 39.09 | 14.12
tested on ◇ contracts	58.89 | 38.67 | 13.51
tested on ♣ contracts	58.86 | 38.90 | 13.77
♡ contracts (NS: 1.0, WE: -1.0)	98.65 | 87.81 | 40.18
◇ contracts (NS: 1.0, WE: -1.0)	98.66 | 87.68 | 39.96
♣ contracts (NS: 1.0, WE: -1.0)	98.73 | 87.90 | 40.02

for details). The availability of as many as over 700 000 precomputed deals allowed making various experiments in order to compare the four proposed input representations and draw several interesting conclusions concerning *notrump* and suit DDBP contracts.

A. Notrump Contracts

Notrump contracts seem to be potentially simpler than suit ones, because there is no possibility to ruff a card of a high rank with a trump card. It simplifies the rules, but does not mean simplification of the strategy. In *notrump* contracts, there is no guarantee that a card will take a trick; even *Aces* are useless in tricks of other suits. The success of a contract often lies in the hand making the opening lead. Hence, even knowing the location of all cards may sometimes be not sufficient to indicate the cards that will take tricks. This is probably the reason of worse results achieved for *notrump* than for suit contracts (see Table V).

B. Suit Contracts

In the second group of the experiments, all four suit contracts were used. It means that a deal was included four times in the training or testing set, once for each possible trump suit.

Rules of the game for suit contracts are more complicated than for *notrump* contracts. Also, the play in suit contracts seems to be more subtle and difficult. However, neural networks were able to achieve visibly better results when there was a trump suit.

Table IV presents the results obtained by the $52 - 25 - 1$ networks. The results achieved for suit contracts are substantially better than the ones for *notrump* contracts. The results of the network trained using both *notrump* and suit contracts (sixth row) are comparable to the results of specialized networks (rows 1–4), and the overall result is close to the average of the results for *notrump* and suit contracts.

It is surprising that the results achieved by the network trained using all suit contracts (fourth row) were comparable to the results of all four networks trained exclusively using one suit (rows 9 and 13–15). On the other hand, these networks that were trained based on the individual suits were useless for other suit contracts (rows 10–12).

C. Spades Contracts

The results presented in Table IV confirm the intuition that there is no significant difference among trump suits. For the sake of brevity of the presentation, in the remainder of this paper, *spades* will be used to present the results for (single) suit contracts. In the case when all four suits are considered, the respective description and results will be labeled *all_suits*.

D. Influence of A Hand Making the Opening Lead

For the *spades* contracts, in 7% of the considered deals, the number of tricks to be taken by the *NS* pair depends on which hand makes the opening lead, i.e., the number of tricks after the opening lead from the *West* side differs from the respective number after the opening lead from the *East* side. Enlarging both training and testing sets by duplicating all deals and exchanging hands in pairs ($N \leftrightarrow S, W \leftrightarrow E$) improved results by about 2.5 percentage points (see the last column of Table V).

E. Comparison of Results

Table V contains the results obtained by various networks for *notrump* and *spades* contracts (in the latter case, both with and without changing a hand making the opening lead). All tested networks, regardless of the size and the used way of input coding, achieved significantly better results for suit contracts.

Almost all presented networks were able to take advantage of additional information coming from changing a hand making the opening lead and improved their results. The only exception was the $52 - 25 - 1$ network, which achieved the best result when input values representing players in a pair were equal to each other (see Table IV). Since in the 52 representation, a value presented to an input neuron indicated a hand, each card in a deck was represented by only one real value. Differentiating players in a pair (e.g., by applying the following input values: 1.0 for N, 0.8 for S, -1.0 for W, and -0.8 for E) misled the network, since it suggested that the cards of one player in a pair were more important. The network using the 104 representation, which was invented as a response to this problem, was able to assimilate additional information and improve the results.

In all categories of the experiments, the advantage of the (52×4) representation is unquestionable. However, two details arising from Table V should be explained. A comparison of the results of two networks using the (52×4) representation for *notrump* contracts shows that the smaller of them achieved better results. It can be easily explained by verifying results on the training sets—$98.02 \,|\, 86.10 \,|\, 39.18$ for the smaller network $[(52 \times 4) - (13 \times 4) - 13 - 1]$ and $98.95 \,|\, 89.30 \,|\, 42.35$ for the bigger one $[(52 \times 4) - (26 \times 4) - 26 - 13 - 1]$, which indicate that the smaller network was able to perform better generalization, and the bigger one was too deeply adapted to the training data (in both cases, the same training set containing 100 000 deals was used).

The same problem of too high adaptation of the $(52 \times 4) - (26 \times 4) - 26 - 13 - 1$ network to the training set explains a relatively small advantage over the smaller network for *spades* contracts without changing a hand making the opening lead.

TABLE V
COMPARISON OF RESULTS OBTAINED FOR *NOTRUMP* CONTRACTS AND FOR SUIT CONTRACTS WITH AND WITHOUT CHANGING A HAND MAKING THE OPENING LEAD

The Network	Results for $Notrump$ Contracts	Results for $Spades$ Contracts	Results for $Spades$ Contracts with Changing a Hand Making the Opening Lead
(26x4)-(13x4)-(7x4)-13-1	93.87 | 75.70 | 31.04	97.67 | 84.24 | 36.82	98.76 | 88.00 | 39.90
52-25-1	96.07 | 80.88 | 34.66	98.77 | 88.00 | 40.13	98.49 | 87.15 | 39.29
104-30-4-1	95.64 | 79.63 | 33.74	98.61 | 87.17 | 39.21	99.09 | 89.79 | 41.92
(52x4)-(13x4)-13-1	97.34 | 84.31 | 37.80	99.78 | 95.00 | 50.03	99.79 | 95.49 | 50.62
(52x4)-(26x4)-26-13-1	96.89 | 83.64 | 37.31	99.80 | 95.54 | 50.91	99.88 | 96.48 | 53.11

TABLE VI
RESULTS FOR SUBSETS OF A TESTING SET ACHIEVED BY A NETWORK $52-25-1$ TRAINED ON *SPADES* CONTRACTS

Target Number of Tricks	Number of Deals	Results
0	1,138	93.32 | 66.61 | 12.30
1	2,725	97.39 | 81.21 | 34.53
2	5,156	98.10 | 86.66 | 40.73
3	8,043	98.93 | 88.96 | 41.41
4	10,447	98.94 | 89.04 | 40.36
5	12,201	98.85 | 88.67 | 40.80
6	12,927	99.03 | 88.75 | 41.32
7	12,709	99.10 | 88.99 | 40.50
8	11,467	99.28 | 89.29 | 40.46
9	9,618	99.14 | 89.19 | 42.14
10	6,866	98.89 | 88.45 | 40.58
11	4,225	97.94 | 85.87 | 42.32
12	1,935	97.57 | 81.71 | 31.94
13	543	94.66 | 73.85 | 9.39
Total	100,000	98.77 | 88.00 | 40.13

Also in this case, the results of the bigger network on the training set were significantly better $(99.95 \mid 97.61 \mid 56.19)$ than those of the smaller network $(99.84 \mid 95.91 \mid 51.68)$. In the experiments with changing a hand making the opening lead, the size of the training set was doubled by duplicating all deals and rotating hands (see Section IV-E), what prevented overfitting.

F. Reliability of Results

In order to verify the reliability of results, four networks, each with one hidden layer composed of 25 neurons, differing only by initial, randomly chosen weights, were trained based on the same set of deals. The experiment was aimed at checking the number of training deals for which all four networks would learn the same number of tricks to be taken by *NS*.

For *notrump* contracts, all four networks estimated the same number of tricks in 61.23% of deals. In 37.93% of them, the estimated numbers of tricks differed by one trick, in 0.81% by two tricks, and in 0.03% by three tricks. The same experiment for suit contracts output the following results: for 63.40% of deals, all networks were unanimous, for 36.56%, there was one trick difference, and for 0.04%, there were two tricks.

Recall that in all the experiments the value of a single output neuron was restricted to the interval $[0.1, 0.9]$. The final number of tricks was calculated by dividing the output interval into 14 subintervals of pairwise equal lengths (≈ 0.06). Considering that, it was further checked that in 98.13% of testing deals for *notrump* contracts, and in 99.53% for suit contracts, real output values of all four trained networks differed by no more than 0.06.

The above experiments confirmed that the attained results are repeatable and independent of initial choice of weights. Consequently, some effort was dedicated to defining decision boundaries in a more flexible way, e.g., by enlarging the subintervals defining the middle values of the number of tricks (5, 6, 7, 8, and 9) at the cost of shortening the remaining subintervals. Also some concepts involving partly overlapping intervals were verified, but at a general level, no further improvement was achieved.

G. Results by the Target Number of Tricks

Results of the $52-25-1$ network trained on *spades* contracts were investigated in more detail in order to test whether the efficacy of the system varies for different numbers of target tricks. The testing set containing 100 000 deals was divided into 14 subsets, according to the target number of tricks, which were then tested separately.

The results are presented in Table VI. Only the figures for 0, 1, 12, and 13 tricks are significantly worse than the result attained for the whole test set (the last row of the table). The reason for this phenomenon is probably twofold: the relatively small numbers of examples and "specificity" of the deals in these categories—they all represent either grand slam or slam deals. The results for the remaining subsets are on pairwise similar levels, in spite of the considerable differences in the number of deals belonging to particular subsets (the second column of the table).

VII. ANALYSIS OF TRAINED NETWORKS

Besides the analysis of numerical results, the main focus of the paper was the exploration of how the bridge knowledge possessed during training is represented in connection weights of the trained networks. Three figures (Figs. 2–4) present visual representations of connection weights in the $52-25-1$ network trained, respectively, for *notrump*, *all_suits*, and *spades* contracts. All of these figures represent the weights of connections in the following way: as black (for negative values) or white (for positive values) circles with radius depending on weight's value—the bigger the absolute value, the bigger the radius. In each figure, the left column represents the weights of connections from hidden neurons (numbered to the left of the column) to the output neuron. The right, big area of circles represents the weights of connections from all 52 input neurons (assigned to cards from a deck, as depicted below the area) to 25 hidden neurons (numbered in the left column). For example, in Fig. 3, the connection weight from the 15th input neuron (representing 3♡ to the 22nd hidden neuron is positive and greater or equal

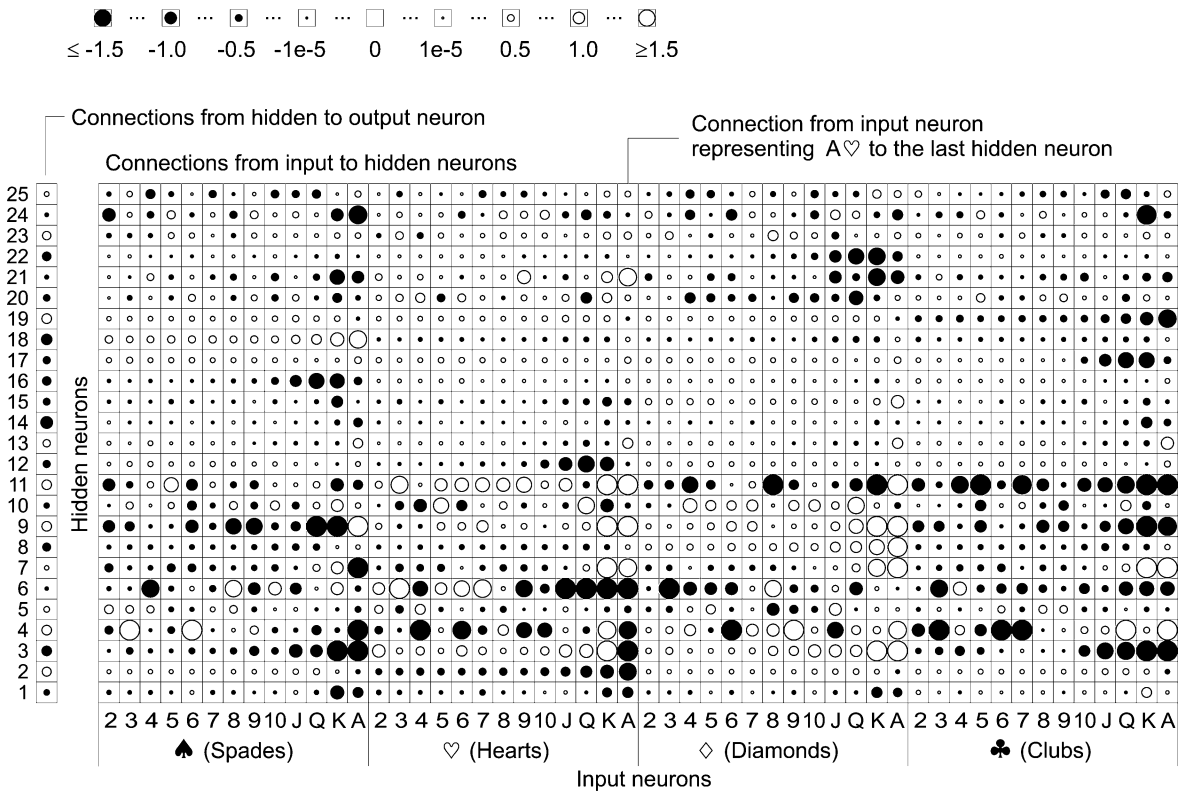


Fig. 2. Visualization of connection weights of the $52 - 25 - 1$ network trained on *notrump* contracts. Each circle represents the weight of one connection. If the circle is placed in the leftmost column, it represents the weight of connection from a hidden to an output neuron, otherwise, from an input to a hidden neuron. The radius of the circle represents the absolute value of the connection weight. Black circles denote the negative and white positive weights.

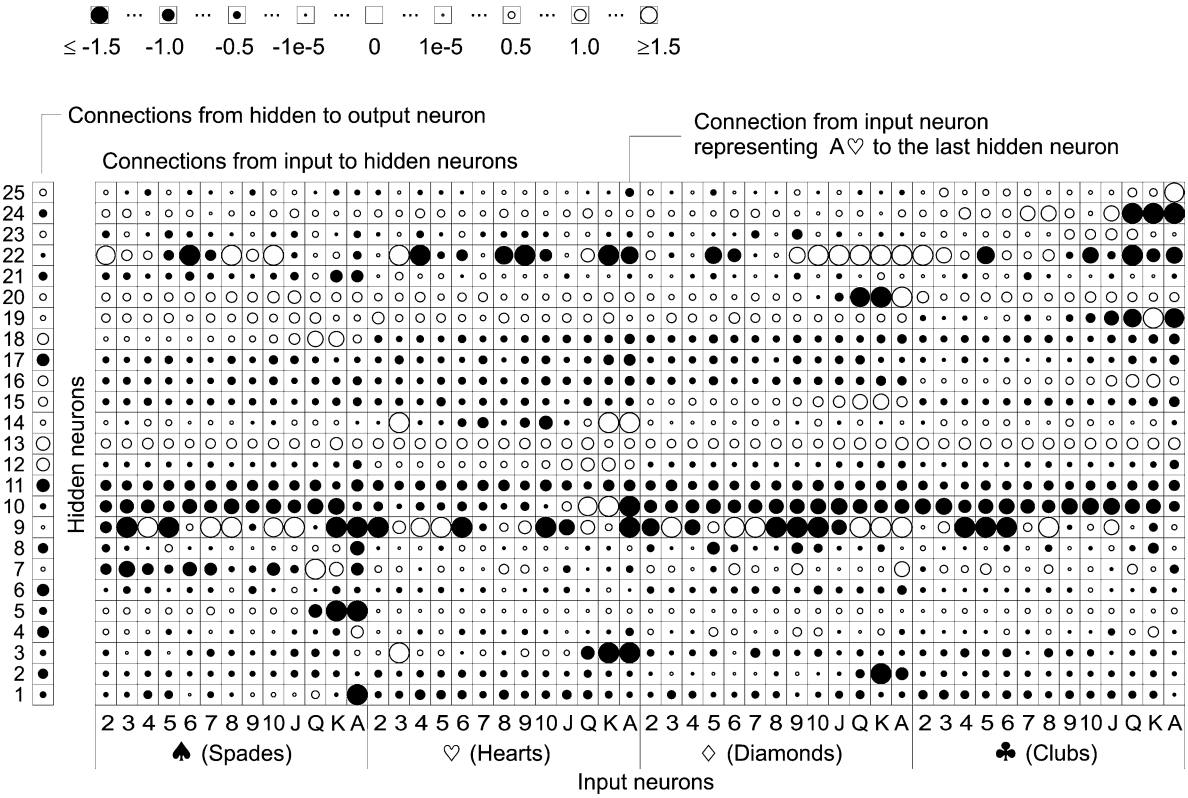


Fig. 3. Visualization of connection weights of the $52 - 25 - 1$ network trained on *all_suits* contracts. See the description of Fig. 2.

1.5 (the largest possible empty circle), whereas the connection weight from the 22nd hidden neuron to the output one has a small negative value, close to zero (represented by a small black circle in the "additional" leftmost column).

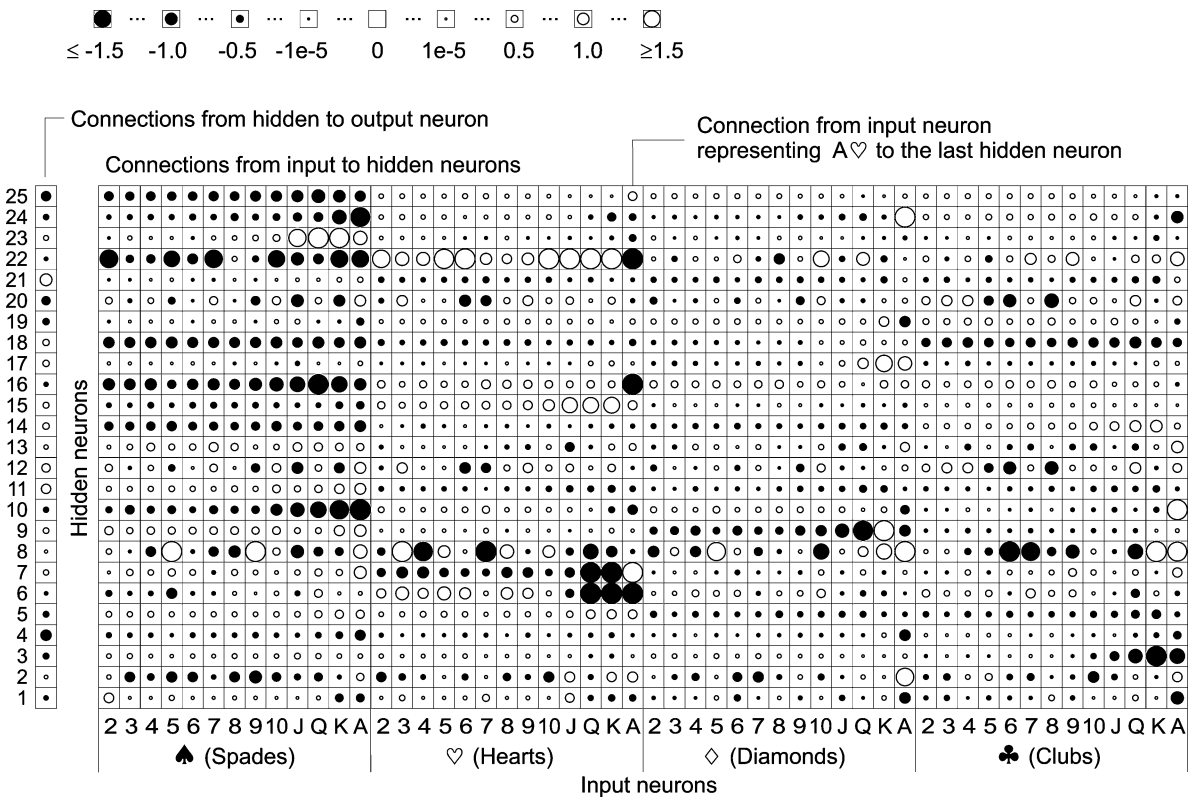


Fig. 4. Visualization of connection weights of the $52-25-1$ network trained on *spades* contracts. See the description of Fig. 2.

A. *Patterns Found in Hidden Layers for Notrump Contracts*

Fig. 2 visualizes the weights of connections of the $52-25-1$ network trained for *notrump* contracts. Three other networks of the same size, trained using the same training set differed only by the initial random values of connection weights, were also investigated. All types of patterns described in this section were found in all four networks of this group.

The first observation from the figure is the existence of hidden neurons with rather "random" values of input connections and very small absolute values of output connections, e.g., neurons number 6, 10, or 24. These neurons seem to be useless. The number of such "useless" hidden neurons depends on the total number of hidden neurons. There were no such neurons in the $52-8-1$ networks and more than ten in the $52-52-1$ ones.

The second pattern that can be found in the figure is the concentration of relevant connections (with biggest absolute values of weights) for honors (i.e., *Aces*, *Kings*, *Queens*, *Jacks*, and *Tens*). Additionally, for each investigated network, it was possible to point out exactly four connections with absolute values much greater than the others. All these favored connections had input neurons assigned to *Aces*. In the network presented in Fig. 2, these connections were the following: for $A\spadesuit$ to hidden neuron number 9 (with value 24.19), for $A\heartsuit$ to neuron number 3 (-26.89), for $A\diamondsuit$ to neuron number 11 (26.71), and for $A\clubsuit$ to neuron number 4 (26.69). For comparison, the biggest absolute value of the remaining connections was equal to 6.47.

The third found pattern also emphasizes the importance of honors in the game of bridge. For each suit, one hidden neuron specialized in honors of that suit can be selected: the hidden neuron number 16 for ♠, number 12 for ♡, number 22 for ♢, and number 17 for ♣. All these four neurons have the same type of weight pattern in the input connections: only connections from neurons representing honors of the respective suit have significant values. All the other weights are much smaller. It is very interesting that for these neurons not *Aces*, but *Queens* and *Kings* are the most important. Even *Jacks* are more important (for these neurons) than *Aces*. For human bridge players, it is obvious that the presence of all figures makes the suit much more powerful and simplifies taking tricks (especially in *notrump* contracts where there is no possibility to ruff) without a need of a *finesse* (i.e., playing on the assumption, which of the opponents possesses the missing figure).

Another pattern reveals when the input connections to hidden neurons with numbers: 2, 8, 18, and 19 in Fig. 2 are compared. As in the previous patterns, each of these neurons specializes in one suit, respectively, ♡, ♢, ♠, and ♣. This specialization is based on favoring all cards of the suit and *Aces* from other suits—these cards have the values of connection weights of the same sign, and connections from all the other cards have the opposite sign. Moreover, the importance of cards in the relevant suit is graded according to the rank of the card: the *Ace* has the biggest absolute value of connection, the *King* relatively smaller, etc. It is interesting that for these four hidden neurons, the importance of the *Aces* of other suits is smaller than the importance of the *two* of the specific suit, however still noticeable, since value of connection weight from the input neuron representing the *Ace* has the same sign as the values of weights of all the connections from the cards of the specific suit.

TABLE VII
VALUES OF CONNECTIONS OF THREE $52-1$ NEURAL NETWORKS TRAINED, RESPECTIVELY, ON *NOTRUMP*, *ALL_SUITS*, AND *SPADES* CONTRACTS. VALUES WERE LINEARLY SCALED INTO THE INTERVAL $(0,4)$

Card's Value	*Notrump* Contracts				*All_Suits* Contracts				*Spades* Contracts			
	♠	♡	♢	♣	♠	♡	♢	♣	♠	♡	♢	♣
2	0.342	0.327	0.329	0.342	1.660	1.670	1.668	1.667	1.371	0.008	0.010	0.004
3	0.340	0.334	0.328	0.353	1.664	1.667	1.663	1.660	1.370	0.004	-0.006	0.000
4	0.347	0.314	0.351	0.345	1.669	1.655	1.669	1.685	1.380	-0.002	0.007	0.015
5	0.341	0.332	0.341	0.344	1.660	1.673	1.676	1.663	1.372	-0.003	0.012	-0.006
6	0.356	0.349	0.339	0.329	1.684	1.685	1.680	1.688	1.405	0.006	0.001	0.026
7	0.380	0.331	0.354	0.356	1.680	1.684	1.687	1.697	1.413	0.007	0.011	0.007
8	0.358	0.361	0.375	0.400	1.709	1.719	1.718	1.723	1.468	0.024	0.023	0.019
9	0.496	0.469	0.461	0.473	1.782	1.791	1.780	1.783	1.591	0.062	0.060	0.058
10	0.660	0.663	0.671	0.684	1.921	1.916	1.918	1.938	1.810	0.145	0.148	0.179
J	1.047	1.032	1.056	1.030	2.174	2.167	2.177	2.172	2.167	0.331	0.359	0.363
Q	1.676	1.688	1.675	1.656	2.569	2.569	2.572	2.565	2.666	0.715	0.708	0.711
K	2.643	2.643	2.677	2.655	3.207	3.210	3.220	3.216	3.314	1.399	1.403	1.416
A	3.975	3.971	3.966	3.989	3.982	3.984	3.973	3.995	3.998	2.319	2.300	2.326

B. Patterns Found in Hidden Layers for All_Suits Contracts

Fig. 3 presents the weights of connections in the $52-25-1$ network trained on suit contracts. Recall that in the *all_suits* case and the 52 representation, the trump suit was indicated by multiplication of input values representing cards of the trump suit by 2 and each deal was repeated in the training (testing) set four times, once for each trump suit. Considering such representation, it is not surprising that the relative importance of the lowest cards in a deal is visibly bigger compared to *notrump* contracts (Fig. 2).

Similar conclusions, as in the *notrump* case, can be drawn with respect to the existence of "useless neurons" (e.g., hidden neurons 7, 9, or 22 have very strange values of weights of input connections and weights of their output connections are close to zero) and the presence of hidden neurons specialized in honors of single suits. Honors of ♠ are definitely the most important for the hidden neuron 18, honors of ♡ for the hidden neuron 12, ♢ for the hidden neuron 15, and ♣ for the hidden neuron 16. Specialization of these neurons is very clear: absolute values of the weights of connections of all other cards are significantly smaller. Additionally, these neurons favor one specific suit: weights of all cards from other suits have the opposite sign.

The relevance of the three highest cards of one suit is more visible for suit contracts. For *spades*, only one hidden neuron (number 5) interested in the *Queen*, *King*, and *Ace* can be pointed out, but for other suits, there are more such neurons—the ones with numbers 3, 10, and 14 for ♡, 2 and 20 for ♢, and 19 and 24 for ♣.

C. Patterns Found in Hidden Layers for Spades Contracts

Fig. 4 shows the weights of connections of a network of the same architecture $(52-25-1)$, but trained on *spades* contracts. Again, a few examples of such networks differing only by the initial random weights were trained using the same training set. All patterns described here were observed in each of them. Recall that for *spades* contracts no information about the trump suit was presented to the network. Each input neuron represented one fixed card. The input value equal to 1.0 denoted *NS* pair and a value of -1.0 *WE* pair. Hence, individual hands within each pair were not directly and precisely specified.

The importance of *spades* (i.e., the trump suit) is visible in Fig. 4 at first glance. Connections from input neurons representing cards of *spades* have significantly bigger absolute values. Especially the lowest cards of *spades* have noticeably more importance compared to the lowest cards in other suits. An interest in honors is also more obvious for *spades* than for any other suit.

Similarly to *notrump* and *all_suits* contracts, also for *spades* contracts, there exist hidden neurons specializing in honors of one suit: these are the neurons number 23, 15, 17, and 3 for ♠, ♡, ♢, and ♣, respectively.

In comparison with previous cases, for *spades* contracts, there are more hidden neurons interested in *Aces* and, what is new, this interest is much more focused (other cards from the same suit, especially the *King* and the *Queen* are much less important). It is interesting that this pattern can be observed for all suits other than the trump suit (e.g., hidden neurons number 16 and 25 for ♡; number 2, 4, and 24 for ♢; and number 1, 10, and 24 for ♣). The weights of connections from input neurons "dedicated" to *spades* follow human intuition: growing importance of cards from the lowest to the highest one can be easily noticed (e.g., hidden neurons 4, 10, or 24). An interesting observation is that these neurons consider also *Aces* from suits other than the trump one as being important.

D. Similarities and Differences Between Networks Trained for Notrump, All_Suits, and Spades Contracts

Concluding the similarities and differences found in the patterns of connection weights in networks trained on *notrump*, *all_suits*, and *spades* contracts, the following observations can be stated.

- Favoring *Aces*. *Aces* are definitely the most important cards for neural networks in all three cases.
- Specialization in single suits. Such specialization is visible in all cases, but only for *notrump* contracts, there existed hidden neurons grading cards in a suit from the *two* to the *Ace*.
- Specialization in honors of single suits. This pattern was visible in all cases, but it differed between *notrump* and suit contracts. For *notrump* contracts, the most important

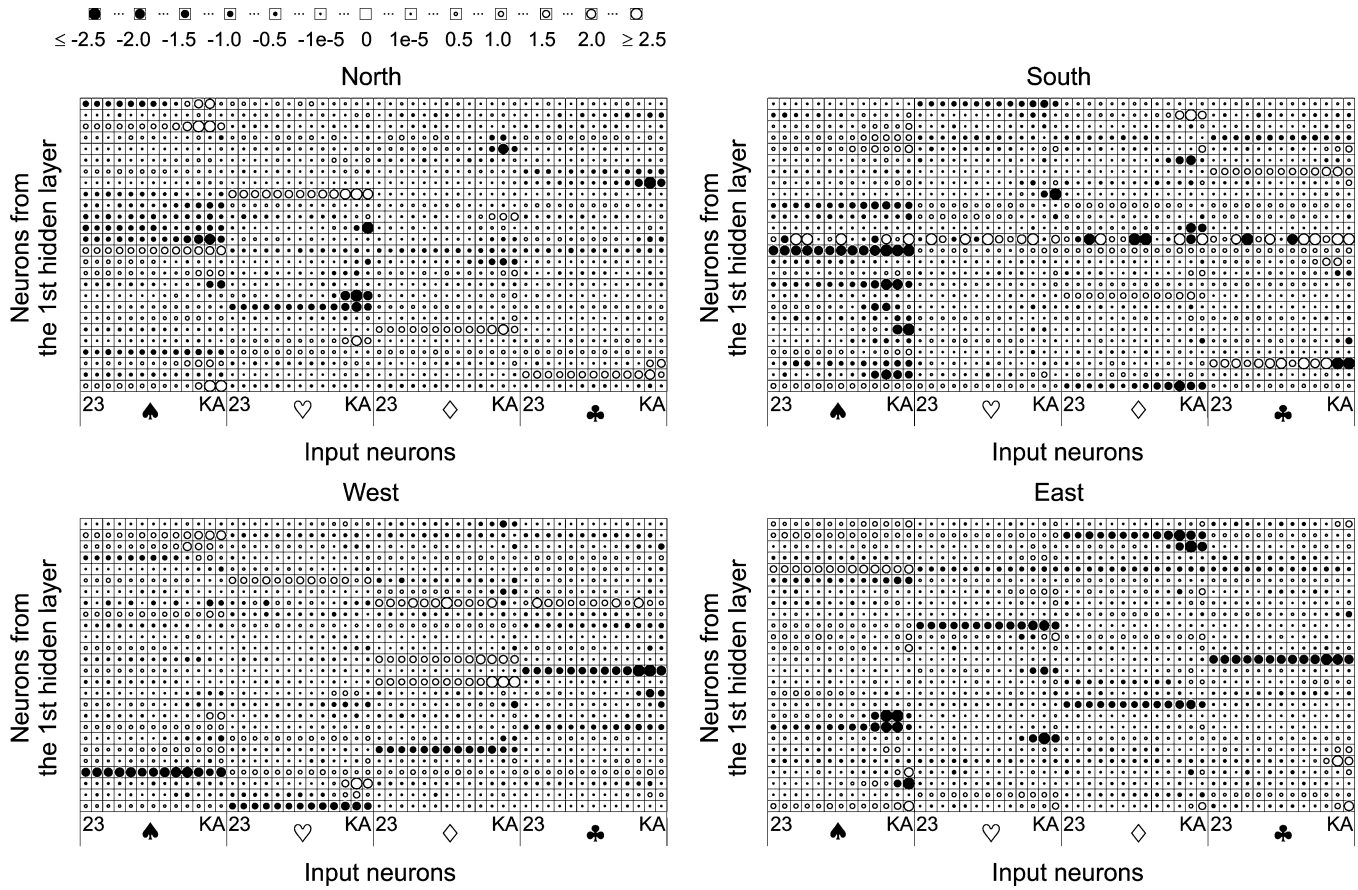


Fig. 5. Visualization of connection weights between input neurons and the first hidden layer ones in the $(52 \times 4) - (26 \times 4) - 26 - 13 - 1$ network trained on *spades* contracts with changing a hand making the opening lead. Each of the four subfigures presents the connections from 52 input neurons representing one hand to the respective 26 hidden neurons. The radius of a circle represents an absolute value of the weight. Black circles denote the negative weights and white ones denote the positive weights.

were *Jacks*, *Queens*, and *Kings*, and for *spades* contracts, the networks fixed their attention on *Kings* and *Aces*.

- Importance of lower cards. For *notrump* contracts, the networks focused on honors, and lower cards were noticed only by hidden neurons specialized in single suits. The networks trained on *all_suits* contracts paid much more attention to cards of lower rank, and the networks trained only on *spades* contracts showed the intuitive solution: all trumps, even the lowest ones, were relevant, and lower cards of other suits were not important.
- Differences between the suits. In two cases, for *notrump* and *all_suits* contracts, there was no visible difference between the connection weights of individual suits. Only for *spades* contracts, one suit (naturally *spades*, the trump suit) had much more importance than any other suit.

Table VII presents the values of connection weights of the networks without hidden units $(52 - 1)$ trained, respectively, on *notrump*, *all_suits*, and *spades* contracts. These values confirm the conclusions drawn from the visualization of connection weights: there is no significant difference between the suits for *notrump* and *all_suits* contracts, and cards from the trump suit are definitely the most important for *spades* contracts.

Another interesting conclusion, which is obvious for human bridge players, is that for *spades* contracts, the three most important cards are: $A\spadesuit$, $K\spadesuit$, and $Q\spadesuit$. $A\heartsuit$, $A\diamondsuit$, and $A\clubsuit$ are less important than $Q\spadesuit$ (the *Queen* of trumps) and $K\heartsuit, K\diamondsuit$, and $K\clubsuit$ have smaller absolute values of output connections than $8\spadesuit$.

E. Patterns Found in Hidden Layers of Networks Using Other Deal Representations

For two other representations of a deal using fixed assignment of cards from the deck to the input neurons (104 and 52×4), similar patterns were observed.

Fig. 5 presents a graphical representation of connection weights from input neurons to neurons in the first hidden layer of the $(52 \times 4) - (26 \times 4) - 26 - 13 - 1$ network. The first two layers were not fully connected, but restricted to one hand.

For each group of connections (one group per hand, as presented in Fig. 5), similar patterns were found. These patterns are analogous to the ones observed in the weights of the $52-25-1$ network. There are hidden neurons specialized in single suits, which grade the rank of cards from the *two* to the *Ace*. Also, hidden neurons fixing their attention on honors of particular suits can be easily pointed out.

VIII. Human Methods of Hand's Strength Estimation

All the results presented in the previous sections were achieved by artificial neural networks trained only on examples, without presenting human knowledge about the game of

TABLE VIII
HUMAN POINT COUNT METHODS

Method	A	K	Q	J	10
Work Point Count	4	3	2	1	0
Bamberger Point Count	7	5	3	1	0
Collet Point Count	4	3	2	0.5	0.5
Four aces points	3	2	1	0.5	0
Polish points	7	4	3	0	0
Reith Point Count	6	4	3	2	1
Robertson Point Count	7	5	3	2	1
Vernes Point Count	4	3.1	1.9	0.9	0
AKQ points	4	3	2	0	0

bridge in any form. These experiments constitute the *first group* of tests.

In this section, the results of training *with* additional human knowledge are presented and discussed. The human knowledge is represented by various numerical estimators of hand's strength used by experienced human bridge players in order to declare the optimal possible contract. These experiments are divided into two groups referred to as the *second* and the *third group* of tests, respectively. In the second group, the inputs from human hand's strength estimators were added to previously used deal representation. In the third group of tests, for comparison purposes, only human estimators were used during training, without accompanying the presentation of a deal.

Human estimators of hand's strength can be divided into two categories: point count methods and distributional points methods.

A. Point Count Methods

Human point count methods are based on calculating the strength of a hand as a sum of single cards' strengths [8], [67]. In these methods, the value of each card depends only on card's rank. The most widely used points counting system is called work point count (WPC), which scores four points for an *Ace*, three points for a *King*, two points for a *Queen*, and one point for a *Jack*. Table VIII presents other popular human point count methods, which were used in the experiments.

B. Distributional Points Methods

The other category of human hand's strength estimators contains the so-called distributional points [8], [67]. These methods score patterns that can be found in a set of cards assigned to one hand. The most important patterns are: suits' lengths and existence of groups of honors in one suit. Even novice bridge players know that a void (lack of cards in a suit) or a singleton (single card in a suit) are very valuable in suit contracts (certainly, it does not regard the trump suit), so almost all distributional points methods award such shortness.

Another very important pattern that is appreciated is a group of honors in one suit located in the cards of both players in a pair. Having a group of top honors in a suit allows to predict more precisely the number of tricks available in this suit.

Human distributional points methods used in our experiments are listed in Table IX. Most of them join WPC scoring with some rewarding for short and long suits.

TABLE IX
HUMAN DISTRIBUTIONAL POINTS METHODS

Method	Short Description
Honor Trick	Potential number of tricks to be taken by honors in one suit (e.g. 2 points for AK, 1.5 for AQ, 1.0 for KQ, etc.)
Playing Trick	Similar to the Honor Trick, with additional bonuses for short and long suits ($+1.0$ for 5 cards in the suit with honors, $+2.0$ for 6 cards, $+3.0$ for 7 or more cards), for figures in the trump suit ($+1.0$ for A, K, or QJ, $+0.5$ for Q or $J10$), and for a long trump suit ($+0.5$ for 4 cards in the trump suit, $+1.0$ for 5 cards, and $+2.0$ for 6 or more cards).
Losing Trick Count	Potential number of tricks to loose in one suit (e.g 1 point for Ax, 2 points for Qx, 3 points for $J10x$).
Asset System	the WPC's enhancement with bonuses for short ($+2$ for a void, $+1$ for a singleton) and long ($+1$ for 5 or more cards) suits.
Stayman Point Count	the WPC method with rewarding short suits and lowering the status of single honors in suits.
Rule of three and four	the WPC method with additional $+1$ point for 5th, 6th, etc. card in the trump suit and for 4th, 5th, etc. card in any other suit.
Moins-value	Modification of the WPC: -1.0 for no *Aces*, -0.5 for no *Tens*, -1.0 when there are too few cards in a suit to make honor(s) potential trick(s), and -0.5 when there are less than 3 honors in a suit or there is only one out of the three top honors: AKQ.
Plus-value	Modification of the WPC: $+0.25$ for each *Ace*, $+0.5$ for a *Ten* with a honor or *nine*, and $+0.5$ when there are 3 honors or at least two of the three following top honors: AKQ in a suit.

Zar points [68] is another method of estimating hand's strength, which combines the elements of point count methods with ideas of distributional points. In this method, each hand is scored by adding the following:

- points for honors: six points for each *Ace*, four points for a *King*, two points for a *Queen*, and one point for a *Jack*;
- the difference between the lengths of the longest and the shortest suits;
- the sum of the lengths of the two longest suits.

C. Representation of Human Estimators

The values of human estimators of hand's strength were coded as real numbers from the range $[0.1, 0.9]$. For each estimator, the minimum and maximum possible values were determined, and evaluator's value was linearly mapped to the destination range. Usually, four input neurons were assigned to each estimator, one per hand. In some experiments, two additional input neurons were assigned to the sums of estimator's values for pairs of players (*NS* and *WE*).

In the case of 52 and 104 codings, input neurons representing human estimators were connected to the first hidden layer exactly in the same way as all other input neurons. In the (52×4) representation, there existed a special group of neurons in the first hidden layer, devoted to processing this additional input information. For example, in the $(52 \times 4 + 84) - (13 \times 4 + 21) - 26 - 1$ network, there were 21 neurons that belonged to such group. These neurons received values exclusively from input neurons assigned to human estimators (in this particular case,

TABLE X
COMPARISON OF THE RESULTS OBTAINED BY NETWORKS WITH AND WITHOUT USING EXPLICIT HUMAN KNOWLEDGE. THE FIRST FIVE ROWS DENOTE THE RESULTS OF TRAINING WITHOUT HUMAN KNOWLEDGE DISCUSSED IN SECTION VI INITIALLY PRESENTED IN TABLE V. ROWS 6-10, 12, 14, AND 18 PRESENT THE EFFECTS OF TRAINING BASED EXCLUSIVELY ON HUMAN ESTIMATORS (WITHOUT EXAMPLE DEALS). THE REMAINING ROWS REFER TO TRAINING RELYING ON "COMBINED" INFORMATION, I.E., THE RAW EXAMPLE DEALS TOGETHER WITH HUMAN HAND'S STRENGTH ESTIMATORS

The Network	Inputs	Results for $Notrump$ Contracts	Results for $Spades$ Contracts with Changing a Hand Making the Opening Lead
(26x4)-(13x4)-(7x4)-13-1	Deals in $(26x4)$ representation	93.87 | 75.70 | 31.04	98.76 | 88.00 | 39.90
52-25-1	Deals in 52 representation	96.07 | 80.88 | 34.66	98.49 | 87.15 | 39.29
104-30-4-1	Deals in 104 representation	95.64 | 79.63 | 33.74	99.09 | 89.79 | 41.92
(52x4)-(13x4)-13-1	Deals in $(52x4)$ representation	97.34 | 84.31 | 37.80	99.79 | 95.49 | 50.62
(52x4)-(26x4)-26-13-1	Deals in $(52x4)$ representation	96.89 | 83.64 | 37.31	99.88 | 96.48 | 53.11
1-1	Sum of WPC values for the NS pair of players	93.73 | 76.41 | 31.37	76.22 | 49.55 | 16.93
4-1	WPC values for each hand	93.73 | 76.34 | 31.31	76.14 | 49.57 | 16.79
20-1	WPC values for hands (4 inputs) and suit lengths (16 inputs)	93.73 | 76.34 | 31.32	96.98 | 82.43 | 35.36
20-10-5-1	WPC values for hands (4 inputs) and suit lengths (16 inputs)	94.24 | 77.78 | 32.78	98.67 | 87.98 | 40.62
36-25-1	9 human point count methods, 4 inputs per method (see Table VIII)	94.87 | 78.30 | 32.39	76.84 | 49.65 | 16.76
(52+36)-25-1	Deals in 52 representation and 9 point count human methods	96.33 | 81.39 | 35.01	98.78 | 88.20 | 40.53
32-25-1	8 human distributional points methods, 4 inputs per method (see Table IX)	94.94 | 77.71 | 32.50	98.53 | 87.64 | 39.93
(52+32)-25-1	Deals in 52 representation and 8 distributional points human methods	96.86 | 83.02 | 36.67	99.67 | 94.18 | 48.40
68-25-1	9 point count methods and 8 distributional points methods (4 inputs for each method)	96.03 | 81.34 | 35.41	98.99 | 90.06 | 42.67
(104+68)-50-10-1	Deals in 104 representation, 9 point count human methods, and 8 distributional points human methods (4 inputs per method)	95.68 | 80.08 | 34.00	99.46 | 92.40 | 45.54
(52+102)-77-38-19-1	Deals in 52 representation, 9 point count human methods, and 8 distributional points human methods (6 inputs per method - 4 for hands and 2 for pairs of players)	96.06 | 81.21 | 35.15	99.50 | 92.67 | 45.80
(52x4+84)-(13x4+21)-26-1	Deals in $(52x4)$ representation, 9 point count human methods, 8 distributional points human methods (4 inputs per method), and lengths of all suits from all hands (16 inputs)	97.37 | 84.99 | 38.78	99.84 | 96.12 | 52.47
4-1	Zar Points values for each hand	91.98 | 71.77 | 28.21	77.45 | 49.98 | 16.76
(52+4)-25-1	Deals in 52 representation and Zar Points values for each hand	96.47 | 81.61 | 35.43	99.45 | 91.90 | 44.69

there were 84 such neurons). In the second hidden layer, all information was combined.

D. Comparison of the Results

Table X contains the results achieved by neural networks in all three aforementioned types of experiments.

The main conclusion that can be drawn from this table is the difference between *notrump* and suit contracts. Whenever full deals were presented in the input layer (regardless of the way of coding), the networks achieved significantly better results for *spades* contracts than for *notrump* ones.

Further comparison of the results obtained for *notrump* and *spades* contracts reveals an important difference between human methods of hands' strength estimation. The results obtained by the networks taking as input data only values of estimators ($36-25-1$ with nine point count methods and $32-25-1$ with eight distributional points methods) are comparable for *notrump* contracts, but for *spades* contracts, the advantage of using distributional points methods over point count methods is unquestionable.

It is also interesting to compare the results achieved by networks using the 52 representation with additional information coming from human estimators. The network using a deal representation and point count methods $[(52+36)-25-1]$ improved the results for *spades* contracts only slightly (compared to the result of the $52-25-1$ network). When distributional points methods were used $[(52+32)-25-1]$, the improvement was significant.

For *spades* contracts, the best results were achieved by the $(52\times 4)-(26\times 4)-26-13-1$ architecture. This network defeated all networks using additional input from human estimators (including the $(52\times 4+84)-(13\times 4+21)-26-1$ one, which used the same, superior representation of deals). For *notrump* contracts, the results attained by this network were not superior, due to overfitting. Two other architectures achieved better results: $(52\times 4)-(13\times 4)-13-1$ and $(52\times 4+84)-(13\times 4+21)-26-1$.

The results achieved by the $20-10-5-1$ network are worth noticing. This architecture uses very simple input data: WPC values for hands and the lengths of suits on each hand (20 input values in total). Only networks using the 52×4 representation were able to achieve visibly better results without providing additional input data from human estimators.

Also the results obtained by the $68-25-1$ network (which uses human estimators only) are interesting. This network was outperformed only by (52×4) architectures, among those trained solely on example deals.

Based on the results presented in this section, it can be concluded that human estimators (especially the distributional ones) are strong and reliable indicators of hand's strength. The results suggest that these estimators provide "the essence of the game" and can be regarded as efficient, compact representation of the

problem (deal to be assessed) for neural network training. On the other hand, in case of *spades* contracts, the so-far superior architecture $[(52 \times 4) - (26 \times 4) - 26 - 13 - 1]$ remained the leading one, which in turn implies that this architecture is capable to extract relevant knowledge straight from the raw data, with no need to use additional estimators in the input layer.

IX. Assessment of the Results

Due to relatively low interest in the game of bridge within AI community (compared, e.g., to chess, checkers, Othello, or go), the objective assessment of the results is not straightforward. In the following subsections, three points of reference are proposed: comparison with the literature (Section IX-A), comparison with the use of selected human estimators discussed in Section VIII, but applied directly to the DDBP deals, not as the input for neural networks (Section IX-B), and finally, comparison with results accomplished by professional human bridge players (Section IX-C).

A. Comparison With Literature

As it was mentioned in the Introduction, the DDBP can be solved exactly with the use of sophisticated AI methods [14], [1], [24], [61] including fast search algorithms and hash tables. These attempts are mentioned in Section III-C. Another approach was proposed by Ginsberg who combined the efficient DDBP solver with partition search mechanism and rollout simulations as a part of his famous GIB program [45], [11].

None of the above papers, however, was connected with the computational intelligence methods and specifically neural networks. To our knowledge, the only attempt to solve the DDBP with the use of neural nets was reported in [2] and [3]. These papers are discussed in Section III-D. Unfortunately, no numerical evidence about the efficiency of proposed approach (neither in the case of using the raw input data nor in the case of using additional "precomputed feature points") is provided in the cited papers. Only high-level conclusions about inefficiency of raw deal representation are given. Our results contradict the above statement and suggest that with appropriate input representation neural networks are capable to extract relevant, domain-specific knowledge straight from the raw data without the need for initial preprocessing of the training/testing deals.

All the above cited papers address the classical version of the DDBP with all four hands uncovered. In practice, from the point of view of bridge rules, it is very interesting to consider the case in which the hands of the opponents are hidden. Such a situation is more realistic, since during the bidding phase, an experienced player can learn about the strengths and weaknesses of his partner's hand quite precisely (under the assumption that the players have a great deal of experience in playing together). Thus, the assumption that partner's cards are known to the player is to some extent justified. Certainly some information concerning cards distribution on the opponents hands can also be inferred from the bidding phase, but the hands of the opponents should be treated, in principle, as hidden.

In the reminder of this paper, in order to distinguish between these two versions of the DDBP, they will be referred to as DDBP-4 (or simply DDBP) in case of fully uncovered version of the problem and DDBP-2 in case of the opponents' hands being hidden. The latter variant is implemented in the paper by the $52-25-1$ architecture with the following hand coding: 1.0 for *NS* and -1.0 for *WE*, as discussed in Section VI-E (cf., the results of $52 - 25 - 1$ architecture in Table V).

Since no reference to the DDBP-2 have been found in the literature, in order to asses the networks' results, a specially designed experiment with human professional bridge players has been performed (see Section IX-C).

B. Comparison With Raw Human Estimators

As discussed in Section VIII, professional bridge players use several numerical estimators in order to assess their hand's strength and consequently bid the optimal contract. These estimators depend mainly on the points gathered on both hands in each pair as well as the distribution (length) of the suits on each hand. One of the conclusions drawn is Section VIII-D refers to the ability of neural networks to evaluate the possible contract based exclusively on numerical information represented by the human estimators, i.e., *without* having access to particular deal distribution. This suggest that the estimators are highly reliable, and also the neural nets are very efficient in utilizing this aggregated information. However, the best neural architectures, when applied to sample deals, can still improve the results by a few percent. In other words, there is more in neural networks than a simple statistical estimation of the points and cards distribution.

In order to further verify the above hypothesis, a simple statistical evaluation of the human estimators is proposed in this section. Instead of presenting the values of human estimators as inputs to neural architectures, it is proposed to use them straight in the same way as human players do. Three basic estimators, namely, the WPC system, the Asset system, and the Stayman system, are taken into account. In each case, the test covers the same 100 000 deals, which were used in evaluation of the neural networks.

In case of the WPC estimator, the results are equal to $86.44|62.07|22.76$ and $74.64|48.25|16.41$, respectively, for *notrump* and *spades* contracts. Application of the Asset estimator leads, respectively, to $79.99|55.14|19.68$ and $72.32|48.42|16.93$. Finally, the use of the more elaborate Stayman system yields the following results: $89.10|67.63|26.02$ and $74.11|50.96|18.32$.

The above results confirm that straight application of human estimators is much less effective than estimations delivered by neural networks appropriately trained on sufficient number of deals.

The final assessment of the quality of neural solutions of the DDBP was made based on a comparative experiment organized among highly qualified human bridge players. The results are presented in the next section.

C. Comparison With Human Bridge Players

The third and the most reliable comparison was performed with the help of human bridge players. A group of 24 professional players, the members of The Polish Bridge Union, took part in the experiment through the Internet. Depending on time availability, each of them solved between 27 and 864 instances

TABLE XI
RESULTS OF HUMAN PLAYERS VERSUS SELECTED NEURAL ARCHITECTURES

Type of the player	DDBP-4. Results for $Notrump$ Contracts	DDBP-4. Results for $Spades$ Contracts	DDBP-2. Results for $Notrump$ Contracts	DDBP-2. Results for $Spades$ Contracts
Group-1	94.74 | 88.30 | 73.68	88.34 | 81.63 | 53.06	93.17 | 79.18 | 43.32	93.68 | 81.20 | 38.63
Group-2	92.94 | 84.71 | 60.78	93.87 | 82.95 | 48.66	84.00 | 69.71 | 34.86	88.46 | 73.59 | 30.59
(52x4)-(26x4)-26-13-1	96.89 | 83.64 | 37.31	99.88 | 96.48 | 53.11		
52-25-1 ($NS = 1.0$, $WE = -1.0$)			96.07 | 80.88 | 34.66	98.77 | 88.00 | 40.13

of the problem grouped into 27 deal chunks.[3] The players, based on their professional accomplishments (the bridge titles and the ranks of competitions they took part in at the international and national level) were divided into two groups. The first group was composed of ten upper classified players (four Grand Masters, three International Masters, and three Masters, playing in the First or the Second Polish Bridge League) including one player from the top ten players in Poland in 2007 ranking. The remaining 14 players (members of the lower ranked bridge teams, each having a professional bridge title) composed the second group. These groups of players will be denoted by "group-1" and "group-2," respectively.

The participants of the experiment were faced with both types of deals, i.e., DDBP-4 and DDBP-2. In each 27-deal chunk, only deals of one type were served. For each deal, the information about the type of contract (*notrump* or *spades*) and the hand making the opening lead were provided. The *NS* pair was either the declaring pair or the defending one. There was a 30-s slot allotted for each test problem. This amount of time was chosen after some preliminary experiments and it seems to be sufficient for experienced players. Before starting the experiment, the players had a chance for some training in order to get used to the environment and the rules of the experiment.

Various statistics were collected from the experiment. The overview of the main results is presented in Table XI. Generally speaking, human players were able to outperform neural networks only in the case of DDBP-4 and *notrump* contracts, where the score of the Group-1 players is respectful. In the remaining three categories, the respective networks (52×4 in case of DDBP-4 and 52 in case of DDBP-2) are competitive to humans, especially when the one-trick and two-trick margins are considered.

Analysis of the results leads to two specific observations: first, humans are visibly better at solving the *notrump* contracts than the suit ones. Second, the opposite conclusion is valid in the case of neural networks.

The reason for better human performance in *notrump* contracts than in the suit ones is twofold. First, the tournament statistics show that a distribution of contracts being played is approximately equal to 60%, 35%, and 5%, respectively, for *notrump*, *spades* and *hearts*, and *diamonds* and *clubs* [69]. Hence, humans are more used to playing *notrump* contracts. Second, *notrump* contracts are easier to be played than the suit ones (roughly speaking, playing the suit contract includes all the techniques used for playing the *notrump* contract and additionally several other maneuvers related to the use of trump cards) [69].

The opposite effect observed in the case of neural networks can be attributed to the fact that human way of solving the DDBP is very different from that of neural networks. Humans, when analyzing a deal, despite scoring the hands and locating the honors also try to virtually simulate the play phase. Neural networks are restricted to a thorough analysis of cards distribution, which includes the location of honors and the lengths of suits, but does not include the play phase simulations. Due to a different specificity of *notrump* versus suit contracts, the point count methods and distributional estimators (when used alone) are more effective in the case of suit contracts than the *notrump* ones, which require some amount of rollout simulations [69]. In this sense, the suit contracts are "better suited" for simulation-free estimations made by neural networks than the *notrump* contracts.

X. SAMPLE DEALS

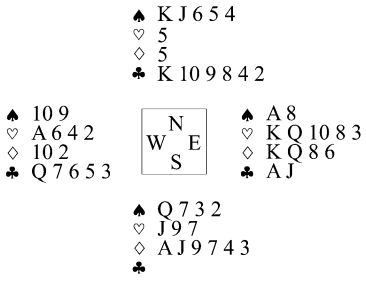
In this section, six examples of deals from the testing set were chosen in order to illustrate the strong and weak points of trained networks depending on deal codings used in the experiments. For each deal, also the fraction of correct answers given by top human players (group-1) taking part in the experiment is provided. Note that since the order of deals in the human contest was partly random, the numbers of answers for each of the deals were not pairwise equal. The first four deals (Sections X-A–X-D) were chosen before the human contest was organized, and hence, the criteria of choosing them were mainly to illustrate the differences between the neural architectures. The last two examples (presented in Section X-E) were chosen so as to show the possible advantage of using neural networks in this task. All human and network answers concern the DDBP-4 version of the problem.

A. 4♠ *With 15 WPC*

The first sample deal, presented in Fig. 6, shows the advantage of the (52×4) architecture over the networks using the 52 or 104 codings. In this deal, the *NS* pair is able to take ten tricks when playing *spades* contract. However, 52 and 104 networks estimated only seven and eight tricks, respectively. Both tested networks using the (52×4) coding were perfectly right. Adding human estimators to the network's input (by using nine point count and eight distributional points methods) and enlarging 104 network's size to $(104+68)-50-10-1$ allowed to estimate the correct number of tricks.

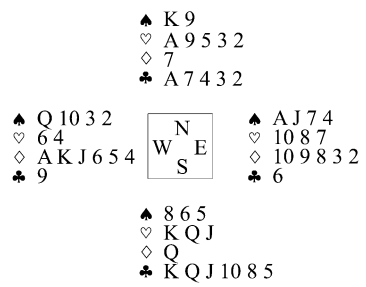
Analysis of this deal shows that *NS* pair has together only 15 WPC. There is a void in *clubs* on *South* and two singletons

[3]In fact, there were more players involved in the experiment. After the contest, the results were restricted to the players having at least the lowest possible professional bridge title, and at the same time, the ones that solved at least one full chunk of deals.



Correct number of tricks	10
Networks' estimations	
$(52x4)-(13x4)-13-1$	10
$(52x4)-(26x4)-26-13-1$	10
$104-30-4-1$	8
$52-25-1$	7
$(104+68)-50-10-1$	10
$(52x4+84)-(13x4+21)-26-1$	10
Group-1 human players (% of correct answers)	40%

Fig. 6. First sample deal. The estimations of a number of tricks to be taken by the *NS* pair in *spades* contract with *West* opening lead.



Correct number of tricks	3
Networks' estimations	
$(52x4)-(13x4)-13-1$	5
$(52x4)-(26x4)-26-13-1$	3
$104-30-4-1$	5
$52-25-1$	5
$(104+68)-50-10-1$	5
$(52x4+84)-(13x4+21)-26-1$	4
Group-1 human players (% of correct answers)	80%

Fig. 7. Second sample deal. The estimations of a number of tricks to be taken by the *NS* pair in *spades* contract with *North* opening lead.

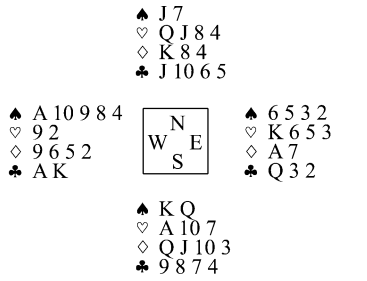
in *hearts* and *diamonds* on *North*. These short suits extremely strengthen *NS* and enable them to hold ten tricks.

Note that simple "human" WPC-based scoring system would estimate only five tricks in this case. This may be one of the reasons why this deal appeared to be quite demanding for humans. The correct answer was given by 40% of the players.

B. Only Three Tricks in Defense Despite 25 WPC and Two Singletons

In the second example, presented in Fig. 7, *WE* can hold ten tricks in *spades* contract, so NS are able to hold three tricks only. Most of the networks claimed five tricks being wrong by two tricks. The $(52\times4+84)-(13\times4+21)-26-1$ architecture overestimated one trick, and only the $(52\times4)-(26\times4)-26-13-1$ one answered properly.

The power of *NS* pair (25 WPC including the *King* of trumps and two singletons) promises more than three tricks. Closer analysis shows some weaknesses. $K\spadesuit$ is not able to take a trick because it is placed on the *N* hand and will be beaten by $A\spadesuit$ placed on the *E* hand. Also $Q\diamondsuit$ cannot take a trick because it



	Opening lead	
	North	*South*
Correct number of tricks	4	3
Networks' estimations		
$(52x4)-(13x4)-13-1$	3	4
$(52x4)-(26x4)-26-13-1$	4	3
$104-30-4-1$	4	4
$52-25-1$	4	4
$(104+68)-50-10-1$	3	4
$(52x4+84)-(13x4+21)-26-1$	3	4
Group-1 human players (% of correct answers)	80%	50%

Fig. 8. Third sample deal. The estimations of a number of tricks to be taken by the *NS* pair in *spades* contract.

is a singleton. Twenty points in *hearts* and *clubs* can take only three tricks due to shortnesses on the opponents hands.

Humans managed to do much better than in the previous case. The correct answers were provided in 80% of the cases.

C. The Number of Tricks Depending on Which Hand Makes the Opening Lead

In the third example (presented in Fig. 8), the number of tricks to be taken by the *NS* pair in the *WE* contract in *spades* depends on which hand makes the opening lead. If it is *North*, the *NS* pair is able to hold four tricks (one in ♠, two in ♡, and one in ◇). When *S* makes defender's lead, *NS* pair can hold only one trick in ♡, so three in total.

The $(52\times4)-(26\times4)-26-13-1$ was the only network that correctly estimated the number of tricks in both cases. The remaining architectures were mistaken in at least one of the cases. The case with the opening lead from *S* was generally more difficult.

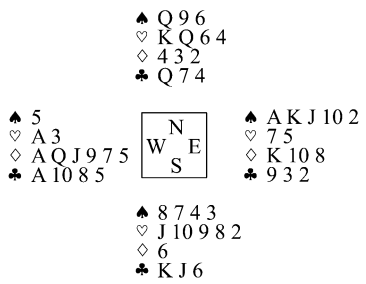
The results among human players follow similar pattern. In the variant with opening lead from *N*, the 80% of the players were correct, but in the case of the rotated opening lead (from the *S* side), the correct answer was given by only half of them.

D. Undefended Grand Slam

The next example (shown in Fig. 9) is one of the deals for which all networks, regardless of the applied way of coding, made a significant error. Perfectly fitted *WE* hands are able to hold all the tricks (the grand slam) thanks to very favorable distribution of *spades* on *NS* hands. The strength of the *NS* pair is noticeable: 14 WPC, seven trumps (including the *Queen*), and a singleton in *diamonds*, but still not enough to hold any trick in this deal.

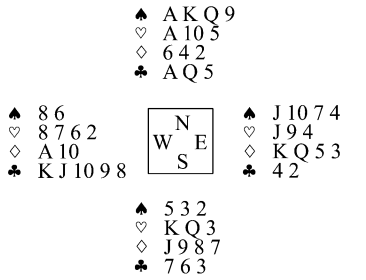
Under these circumstances, the networks' estimations seem to be somehow "justified," but certainly wrong.

In this example, the number of correct answers given by humans highly "diverged" between DDBP-4 (80%) and DDBP-2 (0.0%). The partly hidden variant of the deal appeared to be



Correct number of tricks	0
Networks' estimations	
$(52x4) - (13x4) - 13 - 1$	4
$(52x4) - (26x4) - 26 - 13 - 1$	3
$104 - 30 - 4 - 1$	3
$52 - 25 - 1$	4
$(104 + 68) - 50 - 10 - 1$	3
$(52x4 + 84) - (13x4 + 21) - 26 - 1$	4
Group-1 human players (% of correct answers)	80%

Fig. 9. Fourth sample deal. The estimations of a number of tricks to be taken by the *NS* pair in *spades* contract with *North* opening lead.



Correct number of tricks	9
Networks' estimations	
$(52x4) - (13x4) - 13 - 1$	9
$(52x4) - (26x4) - 26 - 13 - 1$	9
$104 - 30 - 4 - 1$	9
$52 - 25 - 1$	9
$(104 + 68) - 50 - 10 - 1$	9
$(52x4 + 84) - (13x4 + 21) - 26 - 1$	9
Group-1 human players (% of correct answers)	20%

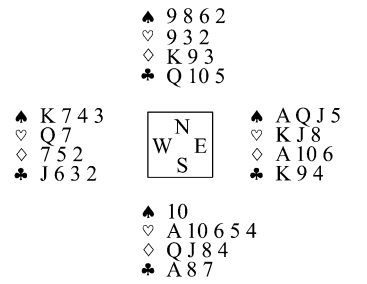
Fig. 10. Fifth sample deal. The estimations of a number of tricks to be taken by the *NS* pair in *spades* contract with *South* opening lead.

highly demanding. The closest answer was three, which means three tricks error. Surprisingly, the problem was relatively easy in its fully uncovered version.

E. Advantage of Networks' Estimations

The following two examples were chosen among those that appeared to be relatively easy for neural networks (all tested architectures predicted correct numbers of tricks), but at the same time quite demanding for human players. In the first example, presented in Fig. 10, it is quite easy to point out eight tricks for *NS* pair (three in ♠, three in ♡, and two in ♣), and "eight tricks" was the most frequent answer given by humans. The correct number of tricks is nine. Since the WPC estimation also suggests eight tricks (the *NS* pair plays on 25 points), it is quite interesting that the networks were able to find this "missing" trick (in ◇).

The other example of networks' "supremacy" over humans is presented in Fig. 11, where the correct answer is four (one trick in ♡, two in ◇, and one in ♣), but humans were largely inclined to estimate five tricks, possibly due to 16 WPC points on the *NS* hand.



Correct number of tricks	4
Networks' estimations	
$(52x4) - (13x4) - 13 - 1$	4
$(52x4) - (26x4) - 26 - 13 - 1$	4
$104 - 30 - 4 - 1$	4
$52 - 25 - 1$	4
$(104 + 68) - 50 - 10 - 1$	4
$(52x4 + 84) - (13x4 + 21) - 26 - 1$	4
Group-1 human players (% of correct answers)	25%

Fig. 11. Sixth sample deal. The estimations of a number of tricks to be taken by the *NS* pair in *spades* contract with *West* opening lead.

Based on the above analysis of example deals, it can be concluded that even though the results accomplished by selected neural networks are comparable with those of humans (in case of *spades* contracts), there exist deals in which human players are clearly surpassed by neural networks and *vice versa*. It should be interesting to explore this issue in more detail.

XI. Conclusion

The results presented in this paper allow to make some observations concerning the abilities of neural networks of autonomous discovering of the properties of the game of bridge and the influence of input representation on the efficacy of this process. Moreover, several observations related to the internal representation of the knowledge acquired during the learning process in the networks' connections can be formulated.

Generally speaking, artificial neural networks turned out to be very effective in estimating the number of tricks to be taken by one pair of players in the DDBP.

The quality of the attained results strongly depends on the way of coding a deal in the input layer. The best tested architectures were capable of discovering knowledge concerning the game based exclusively on the sample training deals. The process of training was so effective that adding explicit human bridge knowledge (in the form of well-known human estimators of hands' strength) did not cause further improvement (such an improvement was, however, observed for less sophisticated neural architectures).

In several cases, it is quite difficult, even for experienced human bridge players, to answer the question about the number of tricks to be taken by a playing pair, even with all cards revealed. In some deals, such an answer depends on the location of the defender's lead hand. Despite these difficulties, the most efficient neural network $[(52\times4)-(26\times4)-26-13-1]$ trained exclusively on example deals, without any human knowledge or awareness of nuances of the play (e.g., finesses), and with no information about the rules of the game, achieved a respectful result: in suit contracts, it was perfectly right in 53.11% of the test deals and mistaken by more than one trick in only 3.52% out of

100 000 test cases. The results for *notrump* contracts were equal to 37.80% and 16.36%, respectively.

From the game of bridge's perspective, the most interesting observation is the difference between *notrump* and suit contracts. The $1-1$ network (i.e., without hidden neurons), which takes as an input only one value—a sum of points of one pair of players, for *notrump* contracts achieved the results comparable to the bigger and more complicated architectures. Certainly, the best networks obtained visibly better results, but the difference was not as big as in the case of *spades* contracts.

For suit contracts, the simplest set of input values that allowed to obtain the results at decent level used the WPC scoring and the lengths of suits on hands. This observation is backed up by the results of networks using as inputs the human estimators of hands' strength only. An advantage of using the set of input values based on distributional points methods (i.e., mainly using information about lengths of suits on hands) for *spades* contracts is undisputable.

The way of coding a deal clearly has an impact on the quality of the results. The three deal representations [$(26 \times 4), 52$, and 104] achieved comparable results when trained solely on the example deals, and after adding human estimators of hands' strength to their input data, the results improved. Hence, it may be concluded that the networks using these representations were not able to extract all corresponding knowledge from the examples. The fourth way of coding was superior in two aspects: first, the networks using this representation achieved the best results. Second, additional knowledge from human estimators did not improve the results, which suggests that all the relevant information about the game was already detected by these networks straight from example deals.

The final assessment of the efficacy of proposed neural approach was made through a comparative experiment organized among professional bridge players. The top ten of them were holding international titles (four Grand Masters, three International Masters, and three Masters) and were playing in either the First or the Second Polish Bridge League. This selective group of players visibly outperformed neural network approach in case of *notrump* contracts, but accomplished a comparable (or even slightly worse) result in the case of *suit* contracts. What is more important, the results degraded in the case of DDBP-2 (a version of DDBP in which only two hands, our and our partner's, are revealed, whereas the opponents' hands are hidden). On the contrary to humans, no degradation of neural networks' efficiency was observed in this partly hidden variant (actually, a slight improvement in $52-25-1$ network's results was noted).

Except for detailed analysis of neural networks applicability to solving the DDBP, the main contribution of this paper is an in-depth analysis of connection weights of trained networks, which revealed the existence of weight patterns "responsible" for particular aspects of the overall solution of the problem. These patterns can be easily recognized and explained by experienced human bridge players. In other words, it turned out that the representation of problem-specific knowledge gained through the learning process, although redundant, is highly specialized and several aspects indispensable for high playing competency in the game of bridge can be explicitly pointed out in the network's weight space.

Analysis of the connection weights of trained networks revealed patterns that can be explained using human knowledge of the game. The most common patterns (found practically in all networks "large enough") are as follows: preference for honors with special attention put to *Aces*, favoring trump suit cards, and gradual importance of cards from *two* to the *Ace* in each suit. Specialization of particular neurons in the above features is very clear. Weight patterns are repeatable for the whole ensemble of randomly initialized networks. All of the above relationships are crucial and well known even for novice human players. All of them, on the other hand, were discovered by the networks themselves in the blind example-based training regime, without explicitly adding any domain knowledge.

In future work, we plan to continue the analysis of internal representation of bridge-specific knowledge in the neural architectures in order to provide the guidelines about when neural-network-based estimator of a bridge contract can be practically applied with high level of confidence.

Another interesting direction for future research is extraction of rules from the trained networks. Since neural networks appeared to be efficient in solving the DDBP and several human-type patterns were found in the networks' weights, it will be interesting to formally define and numerically quantify the bridge-specific features underlying their high performance. These extracted numerical features may be compared with human hand scoring systems and potentially lead to development of some new ideas in human bridge playing and be helpful for novice and semiprofessional players in improving their bridge skills.

Furthermore, we would like to extend the proposed approach to other classification problems having multidimensional, binary data representation.

ACKNOWLEDGMENT

The authors would like to thank P. Dybicz, the International Bridge Master, for fruitful discussions regarding human versus computer bridge playing styles and for his great help in organizing the human DDBP contest described in Section IX-C. They would also like to thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] M.-S. Chang, "Building a fast double-dummy bridge solver," New York Univ., New York, Tech. Rep. TR1996-725, 1996 [Online]. Available: citeseer.ist.psu.edu/chang96building.html

[2] B. Gambäck and M. Rayner, "Contract Bridge as a micro-world for reasoning about communication agents," Swedish Inst. Comput. Sci., Kista, Sweden, Tech. Rep. SICS/R-90/9011, 1990 [Online]. Available: ftp://sics.Se/pub/SICS-reports/Reports/SICS-R-90-ll-SE.ps.Z

[3] B. Gambäck, M. Rayner, and B. Pell, "Pragmatic reasoning in bridge," Comput. Lab., Univ. Cambridge, Cambridge, U.K., Tech. Rep. 299, Apr. 1993 [Online]. Available: citeseer.ist.psu.edu/gamback93pragmatic.html

[4] K. Mossakowski and J. Mańdziuk, "Artificial neural networks for solving double dummy bridge problems," in *Lecture Notes in Artificial Intelligence*, L. Rutkowski, J. H. Siekmann, R. Tadeusiewicz, and L. A. Zadeh, Eds. Berlin, Germany: Springer-Verlag, 2004, vol. 3070, pp. 915–921.

[5] J. Mańdziuk and K. Mossakowski, "Looking inside neural networks trained to solve double-dummy bridge problems," in *Proc. 5th Game-On Int. Conf. Comput. Games: Artif. Intell., Design Educat.*, Reading, U.K., 2004, pp. 182–186.

[6] K. Mossakowski and J. Mańdziuk, "Neural networks and the estimation of hands strength in contract bridge," in *Lecture Notes in Artificial Intelligence*, L. Rutkowski, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds. Berlin, Germany: Springer-Verlag, 2006, vol. 4029, pp. 1189–1198.

[7] J. Mańdziuk and K. Mossakowski, "Example-based estimation of hands strength in the game of bridge with or without using explicit human knowledge," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, Honolulu, HI, 2007, pp. 413–420.

[8] H. Francis, A. Truscott, and D. Francis, *The Official Encyclopedia of Bridge*, 5th ed. Memphis, TN: American Contract Bridge League, 1994.

[9] W. H. Root, *The ABCs of Bridge*. New York: Three Rivers, 1998.

[10] D. N. Levy, "The million pound bridge program," in *Heuristic Programming in Artificial Intelligence: The First Computer Olympiad*, D. Levy and D. Beal, Eds. Chichester, U.K.: Ellis Horwood, 1989, pp. 95–103.

[11] M. L. Ginsberg, "GIB: Imperfect information in a computationally challenging game," *J. Artif. Intell. Res.* vol. 14, pp. 303–358, 2001 [Online]. Available: citeseer.ist.psu.edu/ginsberg01gib.html

[12] M. L. Ginsberg, Library of Double-Dummy Results, [Online]. Available: http://www.cirl.uoregon.edu/ginsberg/gibresearch.html

[13] G. Carley, "A program to play contract bridge," M.S. thesis, Dept. Electr. Eng., Massachusetts Inst. Technol., Cambridge, MA, Jun. 1962.

[14] E. R. Berlekamp, "Program for double-dummy bridge problems, a new strategy for mechanical game playing," *J. Amer. Comput. Mach.*, vol. 10, no. 3, pp. 357–364, 1963.

[15] C. N. Napjus, "Declarer: A learning bridge-playing program which generalizes and infers," Ph.D. dissertation, Univ. Washington, Seattle, WA, 1969.

[16] C. N. Napjus, "Declarer: A learning bridge-playing program which generalizes and infers," *Behav. Sci.*, vol. 16, no. 4, pp. 404–410, 1971.

[17] A. Wasserman, "Realization of a skillful bridge bidding program," in *Proc. Fall Joint Comput. Conf.*, Houston, TX, Jan. 1970, vol. 37, pp. 433–444.

[18] A. M. Stanier, "Bribip: A bridge bidding program," in *Proc. 4th Int. Joint Conf. Artif. Intell.*, Tbilisi, Georgia, 1975, pp. 374–378.

[19] A. M. Stanier, "Planning to make tricks at bridge," in *Proc. AISB Summer Conf.*, D. Levy, Ed., New York, 1988, pp. 256–265.

[20] J. R. Quinlan, "A knowledge-based system for locating missing high cards in bridge," in *Proc. 6th Int. Joint Conf. Artif. Intell.*, Tokio, Japan, 1979, pp. 705–707.

[21] T. A. Throop, *Computer Bridge*. Rochelle Park, NJ: Hayden Book, 1983.

[22] T. Lindelöf, *COBRA - The Computer Designed Bidding System*. London, U.K.: Gollancz, 1983.

[23] D. L. S. Berlin, "Span: Integrating problem solving tactics," in *Proc. Int. Joint Conf. Artif. Intell.*, Los Angeles, CA, 1985, pp. 1047–1051.

[24] R. Wheen, "Brute force programming for solving double dummy bridge problems," in *Heuristic Programming in Artificial Intelligence: The First Computer Olympiad*, D. Levy and D. Beal, Eds. Chichester, U.K.: Ellis Horwood, 1989, pp. 88–94.

[25] J. M. MacLeod, "Microbridge—A computer developed approach to bidding," in *Heuristic Programming in Artificial Intelligence: The First Computer Olympiad*, D. Levy and D. Beal, Eds. Chichester, U.K.: Ellis Horwood, 1989, pp. 81–87.

[26] I. Frank, "Finesse—An adaptation of proof-planning techniques to declarer play in the game of bridge," 1991 [Online]. Available: citeseer.ist.psu.edu/frank91 finesse.html

[27] I. Frank, D. A. Basin, and A. Bundy, "An adaptation of proof-planning to declarer play in bridge," in *Proc. Eur. Conf. Artif. Intell.*, 1992, pp. 72–76 [Online]. Available: citeseer.ist.psu.edu/frank92adaptation.html

[28] I. Frank, "Search and planning under incomplete information, a study using bridge card play," Ph.D. dissertation, Dept. Artif. Intell., Univ. Edinburgh, Edinburgh, U.K., 1996.

[29] I. Frank, "Computer bridge survey," Electrotech. Lab., Tsukuba, Japan, Tech. Rep. TR-97-3, 1997.

[30] I. Frank, D. A. Basin, and H. Matsubara, "Monte-Carlo sampling in games with imperfect information: Empirical investigation and analysis," 1997 [Online]. Available: citeseer.ist.psu.edu/frank97montecarlo.html

[31] I. Frank, D. A. Basin, and H. Matsubara, "Finding optimal strategies for imperfect information games," in *Proc. 15th Nat. Conf. Artif. Intell./10th Conf. Innovative Appl. Artif. Intell.*, 1998, pp. 500–507.

[32] I. Frank and D. A. Basin, "Search in games with incomplete information: A case study using bridge card play," *Artif. Intell.* vol. 100, no. 1–2, pp. 87–123, 1998 [Online]. Available: citeseer.ist.psu.edu/frank98search.html

[33] I. Frank and D. A. Basin, "Optimal play against best defence: Complexity and heuristics," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 1999, vol. 1558, pp. 50–73 [Online]. Available: citeseer.ist.psu.edu/frank98optimal.html

[34] I. Frank and D. A. Basin, "A theoretical and empirical investigation of search in imperfect information games," *Theor. Comput. Sci.* vol. 252, no. 1–2, pp. 217–256, 2001 [Online]. Available: citeseer.ist.psu.edu/frank99theoretical.html

[35] I. Frank and D. A. Basin, "Strategies explained," in *Proc. 5th Game Programm. Workshop*, Hakone, Japan, 1999, pp. 1–8 [Online]. Available: citeseer.ist.psu.edu/frank99strategie.html

[36] I. Frank, D. A. Basin, and A. Bundy, "Combining knowledge and search to solve single-suit bridge," in *Proc. 17th Nat. Conf. Artif. Intell./12th Conf. Innovative Appl. Artif. Intell.*, 2000, pp. 195–200.

[37] I. Frank, D. A. Basin, and A. Bundy, "Solving single-suit bridge play: Winning and knowing why," [Online]. Available: citeseer.ist.psu.edu/frankOOsolving.html

[38] S. J. J. Smith and D. S. Nau, "Strategic planning for imperfect-information games," in *Proc. Fall Symp. Games: Planning Learn.*, 1993, pp. 84–91 [Online]. Available: citeseer.ist.psu.edu/smith93strategic.html

[39] S. J. J. Smith and D. S. Nau, "An analysis of forward pruning," in *Proc. Nat. Conf. Artif. Intell.*, 1994, pp. 1386–1391 [Online]. Available: citeseer.ist.psu.edu/smith94analysis.html

[40] S. J. J. Smith, D. S. Nau, and T. A. Throop, "A planning approach to declarer play in contract bridge," *Comput. Intell.* vol. 12, pp. 106–130, 1996 [Online]. Available: citeseer.ist.psu.edu/smith96planning.html

[41] S. J. J. Smith, D. S. Nau, and T. A. Throop, "Computer bridge—A big win for AI planning," *Artif. Intell. Mag.* vol. 19, no. 2, pp. 93–106, 1998 [Online]. Available: citeseer.ist.psu.edu/smith98computer.html

[42] S. J. J. Smith, D. S. Nau, and T. A. Throop, "Success in spades: Using AI planning techniques to win the world championship of computer bridge," *AAAI/IAAI* pp. 1079–1086, 1998 [Online]. Available: citeseer.ist.psu.edu/smith98success.html

[43] M. L. Ginsberg, "How computers will play bridge," *The Bridge World*, vol. 67, no. 9, pp. 3–7, 1996.

[44] M. L. Ginsberg, "Partition search," in *Proc. 13th Nat. Conf. Artif. Intell./8th Innovative Appl. Artif. Intell. Conf.*, H. Shrobe and T. Senator, Eds., Menlo Park, CA, 1996, vol. 2, pp. 228–233 [Online]. Available: citeseer.ist.psu.edu/ginsberg96partition.html

[45] M. L. Ginsberg, "GIB: Steps toward an expert-level bridge-playing program," in *Proc. 16th Int. Joint Conf. Artif. Intell.*, 1999, pp. 584–589 [Online]. Available: citeseer.ist.psu.edu/ginsberg99gib.html

[46] R. F. Chris Vogt, "A rule based approach to bidding bridge," 1994 [Online]. Available: http://citeseer.ist.psu.edu/126924.html

[47] A. Amit and S. Markovitch, "Learning to bid in bridge," *Mach. Learn.*, vol. 63, no. 3, pp. 287–327, 2006.

[48] W. W. Cohen, "Abductive explanation-based learning: A solution to the multiple inconsistent explanation problem," *Mach. Learn.*, vol. 8, pp. 167–219, 1992.

[49] B. Yegnanarayana, D. Khemani, and M. Sarkar, "Neural networks for contract bridge bidding," *Sadhana*, vol. 21, no. 3, pp. 395–413, Jun. 1996.

[50] M. Sarkar and B. Yegnanarayana, "Feedforward neural networks configuration using evolutionary programming," in *Proc. IEEE Int. Conf. Neural Netw.*, 1997, vol. 1, pp. 438–443.

[51] M. Sarkar and B. Yegnanarayana, "An evolutionary programming-based probabilistic neural networks construction technique," in *Proc. IEEE Int. Conf. Neural Netw.*, 1997, vol. 1, pp. 456–461.

[52] M. Sarkar and B. Yegnanarayana, "Rough-fuzzy set theoretic approach to evaluate the importance of input features in classification," in *Proc. IEEE Int. Conf. Neural Netw.*, 1997, vol. 3, pp. 1590–1595.

[53] W. Jamroga, "Modelling artificial intelligence on a case of bridge card play bidding," in *Proc. 8th Int. Workshop Intell. Inf. Syst.*, Warsaw, Poland, 1999, pp. 267–277 [Online]. Available: citeseer.ist.psu.edu/jamroga99modelling.html

[54] T. Ando, Y. Sekiya, and T. Uehara, "Partnership bidding for computer bridge," *Syst. Comput. Jpn.*, vol. 31, no. 2, pp. 72–82, 2000.

[55] T. Ando and T. Uehara, "Reasoning by agents in computer bridge bidding," *Comput. Games*, vol. 2063, pp. 346–364, 2001.

[56] T. Ando, N. Kobayashi, and T. Uehara, "Cooperation and competition of agents in the auction of computer bridge," *Electron. Commun. Jpn.*, vol. 86, no. 12, pt. 3, pp. 76–86, 2003.

[57] D. Kibler and K. Schwamb, "Complete contingency planners," Univ. California, Irvine, CA, Tech. Rep. ICS-TR-92-24, 1992 [Online]. Available: citeseer.ist.psu.edu/kibler92complete.html

[58] D. Khemani, "Planning with thematic actions," in *Proc. 2nd Int. Conf. Artif. Intell. Planning Syst.*, 1994, pp. 287–292.

[59] S. L. Epstein and J. Shih, "Sequential instance-based learning," in *Lecture Notes in Computer Science*, R. E. Mercer and E. Neufeld, Eds. Berlin, Germany: Springer-Verlag, 1998, vol. 1418, pp. 442–454.
[60] Y. Nygate and L. Sterling, "Aspen—Designing complex knowledge based systems," in *Proc. 10th Israeli Symp. Artif. Intell.*, 1993, pp. 51–60.
[61] L. Sterling and Y. Nygate, "Python: An expert squeezer," *J. Log. Program.*, vol. 8, no. 1–2, pp. 21–39, 1990.
[62] M. Köhle and F. Schönbauer, "Erfahrung mil einem Neuralen Netz, das Bridge spielen lernt," in *Proc. 5th Austrian Meeting Artif. Intell.*, J. Retti and K. Leidlmair, Eds., 1989, pp. 224–229.
[63] H. Yo, Z. Xianjun, Y. Yizheng, and L. Zhongrong, "Knowledge acquisition and reasoning based on neural networks—The research of a bridge bidding system," in *Proc. Int. Neural Netw. Conf.*, Paris, France, 1990, pp. 416–423.
[64] M. Sarkar, B. Yegnanarayana, and D. Khemani, "Application of neural network in contract bridge bidding," in *Proc. Nat. Conf. Neural Netw. Fuzzy Syst.*, 1995, pp. 144–151.
[65] Java Neural Network Simulator [Online]. Available: http://www-ra.informatik.uni-tuebingen.de/software/JavaNNS/welcome_e.html
[66] M. Riedmiller and H. Braun, "Rprop—A fast adaptive learning algorithm," 1992 [Online]. Available: citeseer.ist.psu.edu/riedmiller92rprop.html
[67] B. Seifert, Ed., *Encyclopedia of Bridge* (in Polish). Warsaw, Poland: Polish Scientific Publishers PWN, 1996.
[68] Z. Petkov, "Zar points aggressive bidding," [Online]. Available: http://www.zarpoints.com
[69] P. Dybicz, International Bridge Master and Trainer, 2008, private communication.

Krzysztof Mossakowski was born in Warsaw, Poland, in 1975. He received the M.S. degree in computer science (with honors) from the Faculty of Technical Physics and Applied Mathematics, Warsaw University of Technology, Warsaw, Poland, in 1999.

In 1999-2007, he was an Assistant at the Warsaw University of Technology, where he is currently a Lecturer at the Faculty of Mathematics and Information Science. He is the coauthor of six journal and conference papers. His scientific interests are computational intelligence and game playing.

Jacek Mańdziuk (M'05) received the M.Sc. (with honors) and Ph.D. degrees in applied mathematics from Warsaw University of Technology, Warsaw, Poland, in 1989 and 1993, respectively, and the D.Sc. degree in computer science from the Polish Academy of Sciences, Warsaw, Poland, in 2000.

He is currently an Associate Professor at the Warsaw University of Technology and Deputy Head of Computer Science Department, Mazovian Gas Company, Warsaw, Poland. He is the author or coauthor of two books and over 70 scientific papers. His research interests include application of computational intelligence methods to mind game playing, bioinformatics, financial modeling, time-series prediction, and development of constructive learning methods for artificial agents.

Dr. Mańdziuk has been a member of the International Neural Networks Society since 1995. He served as Program Committee Member for over 40 international conferences, was a Program Co-Chair of the 2002 International Workshop on Adaptive Systems in Soft Computing and Life Sciences and a panelist in the Computational Intelligence and Games panel at the 2008 IEEE World Congress on Computational Intelligence, Hong Kong.

He is a Reviewer for several scientific journals devoted to computational intelligence and artificial intelligence. He is the Associate Editor of the newly launched IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES, member of the Games Technical Committee of the IEEE Computational Intelligence Society and Chair of the IEEE CIG *Task Force on Neural Networks for Games*. He is a recipient of the 1996 Senior Fulbright Research Grant.