# Towards Cognitively-Plausible Game Playing Systems

Jacek Mańdziuk

Faculty of Mathematics and Information Science,
Warsaw University of Technology,
Plac Politechniki 1, 00-661 Warsaw, Poland
Email: {mandziuk}@mini.pw.edu.pl

**Abstract**

We propose to return to the roots of Artificial/Computational Intelligence applicability to board games domain by attempting to mimic human way of playing (or human intelligence in a broader perspective). The paper provides the argumentation for potential virtues of developing cognitively-plausible pattern-based human-like playing systems. Such systems, besides playing games, may *in principle*, also be applied to other domains related to general problem-solving.

In order to support theoretical considerations a cognitively-inspired game-playing framework is presented, which is not focused on applying explicitly defined numerical evaluation function or performing extensive search, but instead takes advantage of context-based pattern templates and knowledge generalization techniques in a way that, to some extent, is alike to that of human players. Experimental evaluation of proposed approach in Connect Four proved its potential, which also gives hope for successful application of the method to other, more demanding game environments (chess, checkers or Othello).

Certainly, reaching the level of play exhibited by the state-of-the-art programs in the above-mentioned games, if at all attainable, would require substantial time and effort, but as advocated in the paper, this research avenue seems to be worth exploring.

**keywords** cognitive approach, intuitive playing, human-like problem solving, example-based learning, Connect Four.

## I Introduction

Playing games has always been an important part of human activity and the oldest games still played in their original form (Go and backgammon) date back to 1,000 - 2,000 BC.

Games also became a fascinating topic for Artificial Intelligence (AI). Presumably the first genuine attempt at artificial chess playing was made around 1890 by Torres y Quevedo who constructed a device capable of playing king and rook versus king endings (the machine was playing the more powerful side). The machine check-mated the opponent regardless of their play. Since the problem is relatively simple - a rule-based algorithm can easily be developed for endings of this type - the device may possibly be not regarded as sophisticated achievement according to today's standards, but was definitely advanced for the period of its origin.

Serious, scientific attempts to invent “thinking machines” able to play popular board games began in the middle of the previous century. Thanks to seminal papers devoted to programming chess [1, 2, 3] and checkers [4] in the 1950s, games through decades remained an interesting topic for both classical AI and CI (Computational Intelligence) based approaches.

The long-lasting (and sometimes even dramatic) AI/CI research struggle to build better-than-humans playing systems in all widely-known board and card games led to spectacular accomplishments in most of them and to developments of various search algorithms. Notable examples in this area include minimax search, alpha-beta pruning (with various enhancements), SSS$^*$, B$^*$, SCOUT, Principal Variation Search, Negascout, MTD($f$), or Monte Carlo sampling-based simulations, e.g. the UCT algorithm [5, 6, 7], currently dominating the scene of game-playing.

Due to space limits the above search methods are not discussed any further in the paper nor the references to them are listed. The interested reader may obtain in-depth information concerning these algorithms from the source publications or from a concise overview presented in [8, chapter 3]. Even though the research ideas presented in this paper are not related to the above search algorithms (and actually they advocate *only limited use of search algorithms*) we have no doubts that the methods listed above deserve respect and attention, since in many cases they proved to be remarkably efficient. In other words, the efforts to promote playing systems that are less search-dependent are by no means motivated by the presumably low assessment of currently applied methods. On the contrary, the so-far accomplishments of the AI systems in the area of game playing are unquestionable and speak for themselves. On the other hand, despite high efficacy of the state-of-the-art AI game-playing systems, we would be rather reluctant in calling them *intelligent*, at least according to human standards.

As previously stated, the focus of this paper is on classical board games, that is two-player deterministic perfect-information turn-based zero-sum board ones, which would be simply termed *games*. Henceforth, unless stated otherwise, any reference to "games" or "game playing" will address games of this type.

The mainstream research in the field of games was, for decades, focused on maximization of the playing efficiency of artificial agents, practically neglecting the issue of similarity (or actually *the lack of similarity*) of the applied methods to the way in which humans reason, analyze and play games. Spectacular accomplishments of AI systems in chess (e.g. Deep Blue II [9] or Rybka [10]) or checkers (Chinook [11, 12]) stemmed largely from increasing computational power of computers, supported by efficacy of the search algorithms, extensive opening and ending databases, and time-consuming, manual selection of weights in multi-stage hand-crafted evaluation functions, rather than any facet of human-type intelligence.

The most renowned example of efficacy of AI application to games is supercomputer Deep Blue II, the winner of the 1997 world championship match against the then human world chess champion Garry Kasparov. In this contest, Deep Blue II employed the evaluation function consisting of approximately 8 000 parameters and possessed computational power allowing for effective analysis of between 50 and 330 million positions per second (alternatively about 50 billion positions per three minutes, which is the average time allotted for one move in the top-level chess matches [13]).

On the contrary to machines, top human players usually consider between a few hundreds and a few thousands moves during the entire game. A comparison of the numbers of moves analyzed by humans and machines in chess (and also in other popular board games) leads to evident conclusion that the way of position representation and the mechanisms of its analysis significantly differ between human and artificial players. The underlying distinction is attributed to much higher involvement of the mechanisms of abstraction and generalization in human play. Moreover, humans take advantage of topological ($2D$-geometrical) similarities between positions or their crucial components, which allows effective qualitative assessment of the board (in terms of goals and challenges) rather than purely numerical estimation of strengths and weaknesses of both playing sides. The above capacity to perform efficient goal-oriented qualitative analysis is further enhanced by human players' ability to make intuitive, though highly reliable, move pre-selection, which substantially narrows the search space (size of the game tree) that need to be explored. This quality, in turn, enables people to accomplish high-quality play without the need for performing exhaustive search (in contrast to machine players).

The main goal of the paper is introduction of a context-based approach to board games, which relies on some cognitive principles while representing and analyzing game positions in a way similar to that of human players. This resemblance is exhibited on a functional level, i.e. by means of the *effects* of the system's performance. In this respect the proposed system can be, to some degree, linked to the research stream known as BICAs (Biologically Inspired Cognitive Architectures) [14], which do not attempt to imitate low-level neurophysiological implementation of cognitive functions, but focus on their high-level functionality and consistency.

The cognitively-inspired game playing system introduced in the paper is tested on a simple board game, Connect Four. Even though the game is not very demanding, the quality of results and their closer examination promise potential of the method also in more challenging games, e.g. chess, which is one of our next targets.

The remainder of the paper is organized as follows. In section II, we briefly present and discuss neurobiological foundations of building human-like playing systems. In section III, a list of selected facets that an artificial playing system must posses in order to be considered as *playing in a human way* is proposed and briefly motivated. Design principles of the aforementioned game playing system, partly based on the above list of postulated critical features, are presented in subsection $B$, along with experimental results of the system's application to Connect Four. Conclusions and possible future research directions are placed in the last section.

## II Inspirations from Neurobiological and Cognitive Experiments

One of the fundamental inspirations for the proposed research comes from the studies of de Groot [15] and Chase and Simon [16, 17] who analyzed the internal specificity of the human way of playing chess. According to their findings, the superiority of master chess players over the intermediate and novice ones in the task of memorizing chess positions (after a limited time of exposure) can be observed only in the case when positions to be memorized come from real games (the ones actually played). Such positions are coherent in the sense of possessing "internal chess logic", and can be decomposed into meaningful chunks (templates), that serve as universal building blocks of the memorized positions. These features of real chess positions allow master players taking advantage of their expertise in the game. On the other hand, in the case of randomly-generated positions (unlikely or impossible to be accessed in real play) the verbal ability to memorize them was only slightly higher among chess grandmasters than among less-skilled players.

The above experiments support a hypothesis that superior chess playing capabilities of grandmasters are not related to their generally better memory functioning, but stem from more efficient internal position representation they use. This representation is in the form of a set of interconnected relevant chunks. The ability to immediately compare the current game position under examination with this collection of chunks stored in the player's memory seems to be a key factor in achieving high playing competency. The chunks as well as the internal representation of position in the player's memory are developed by them during years of professional career (experience in chess playing).

Subsequent studies related to saccadic eye movement analysis [18, 19] and to verification of the check status in specially designed positions [20] confirmed the above conclusions about position decomposition into meaningful chunks, each of them performing a particular "abstract" role specifically implemented in the context of a given position. Furthermore, the above studies confirmed that advanced and master chess players perform parallel analysis of a position, as oppose to sequential analysis applied by novice and less advanced players (the time required for analysis of more complicated positions was increasing only slightly in the case of master players, whereas the respective difference was significant in the case of less skilled players).

In particular, de Groot and Gobet confirmed in [19] that during a single eye fixation professional players analyze the relations between two to five elements of a position (being either chess pieces or empty squares), which potentially compose a meaningful piece of information (a chunk).

## III The Main Facets of Cognitively-Plausible Playing Systems

Allen Newell [21] introduced several criteria to define/verify cognitive systems, which refer mainly to their behavioral, learning, and knowledge-related properties. Duch et al. [22] proposed a simplified taxonomy of cognitive systems based on two main criteria: *memory* and *learning*, which according to the authors are the *"two key design properties* *that underlie the development of any cognitive architecture"*. In the view of Duch et al.' taxonomy, the system proposed in this paper fits the *Emergent Cognitive Architectures* category, which is inspired by connectionist ideas. These architectures are frequently implemented with the help of neural networks (as is the case of our system) with either associative or competitive learning being applied. The memory in emergent cognitive architectures can be organized either in globalist or localist manner. In our case, a guided (supervised) associative learning method is used. Due to distributed, local data processing, the localist memory structure is utilized.

In this section, selected underlying concepts of human-like playing systems are presented with the focus on their functional rather than implementation side. Each of these issues represents a characteristic feature of human play and as such may serve as a guideline for the human-like systems research and development. Certainly, this list is not complete and may easily be extended by adding other human-specific capabilities. On the other hand, fulfilling implementation postulates listed below in this section would already lead to development of advanced and innovative game playing systems and would narrow the gap between human and machine ways of playing. Additional aspects may be subsequently added in the incremental process of development of a truly human-like game player.

The human-like playing concepts presented below are divided into two categories: those related to learning mechanisms and those related to acquisition, representation and management of data. On the conceptual level, these postulates reflect to some extent the aforementioned neurobiological and cognitive experimental evidence (chunk-based representation, chunk sharing, hierarchy of knowledge representation, associative pattern-based learning). They are also partly inspired by the work of Linhares [23] and by the design principles of Morph [24, 25] and SYLPH [26] - the two most renowned systems among those that attempted to functionally imitate human way of intuitive reasoning and playing. Due to space limits, we are not able to elaborate more on these accomplishments. A critical and thorough discussion on artificial systems that emulate human-like intuitive way of playing can, for instance, be found in [8, chapter 12] or [27]. A general discussion on machines' intuition and creativity is presented in [28].

## A Learning-Related Postulates

P1: ***Learning should be implemented as an incremental development process.***

Humans generally learn in an incremental manner. A learning process usually starts off with understanding simple ideas, and afterwards the level of complication gradually increases. As the learning progresses, the learner is capable to acquire more and more complex concepts built upon already possessed knowledge. Learning new concepts is interleaved with the process of reviewing already learnt knowledge.

P2: ***Learning should be implemented in a parallel (multitasking) mode.***

Humans are not "single track" learners. They learn simultaneously, albeit, as stated above, with gradually increasing complexity of the learning tasks. Hence, a game playing system should be capable of learning several or many game ideas at the same time, assuming that their level of complexity is adequately suited to the advancement of the learner.

P3: ***Learning system should be capable of suitable decomposition of game patterns into meaningful subpatterns without the need for external intervention.***

Ideally, a game-learning system should be able to autonomously decompose learning examples into meaningful chunks. Furthermore, each chunk should be classified and assigned to a cluster of chunks representing the same learning motif. In chess, for example, such a cluster might be defined as *possible attacks on square F7* (which is a common motif of attacking black king), and all *subpatterns* that *directly* relate to the task of threatening black on square F7 would belong to this cluster/category.

P4: ***Learning process should be performed on several levels of detail.***

This requirement stems from the observations that advanced human players perform analysis in a repetitive and gradually more detailed way. Position analysis usually starts with identifying its general strengths and weaknesses based on rough assessment of material and positional features. This first approximation allows discarding obviously non-interesting moves and leaving only a fraction of all legal moves for further analysis. The next level of selection relies on a more detailed analysis and is performed with more careful consideration. The number of such stages of analysis varies, depending on the position's complexity as well as other, nonmeasurable factors, e.g. the player's situation in a tournament, their current frame of mind, etc., but as a rule, examination of position at subsequent levels is gradually more thorough.

## *B Data Representation and Management*

P5: ***Game-related concepts may be effectively represented and processed with the use of pattern-based representation.***

Postulate P5 is related to human ability to handle hundreds of thousands of game patterns efficiently. Consequently, the use of pattern-based representations seems to be a viable alternative to traditional rule-based, symbolic representations. The above claim is experimentally supported by the findings of de Groot and Gobet [19], Chase and Simon [16, 17], and other authors not mentioned in this paper, who strongly suggest that human players define relations among pieces and empty squares in a pattern-based manner, i.e. based on spacial location of these objects and their geometrical interrelations.

P6: ***Knowledge acquired in the learning process should be represented in a hierarchical structure with various inter- and intra-level connections.***

Even though hypotheses concerning the issue of how knowledge is represented in human brains vary significantly in particular details, they generally agree on the principle that more complex observations are constructed based on primal concepts and that knowledge is represented in a hierarchical manner. Furthermore, this hierarchy of information is not a purely linear structure, but rather involves feedback loops and intra-connections. At the bottom of this hierarchy, local representations of atomic concepts are placed.

P7: ***Acquisition of knowledge as well as its further processing in the system should take into account symmetries existing in the game.***

Symmetries exist in many classical board games, and they are efficiently, almost effortlessly handled by humans in their analysis of a game position. It seems reasonable to assume that artificial human-like playing system should possess a similar competency which, above all, greatly simplifies categorization of information chunks and substantially reduces the search space.

## IV Cognitively Motivated System for Connect Four

The rest of the paper presents a Connect Four system which is based on some of the above-mentioned principles, capable of **effective search-free sorting of moves, thus exhibiting a human-like ability to intuitive move pre-selection**. Consequently, the optimality of system's play is not treated here as a goal *per se*. Instead our focus is on functional emulation of a human cognitive process of move ordering, which is applied *before* the actual search for the optimal move takes place.

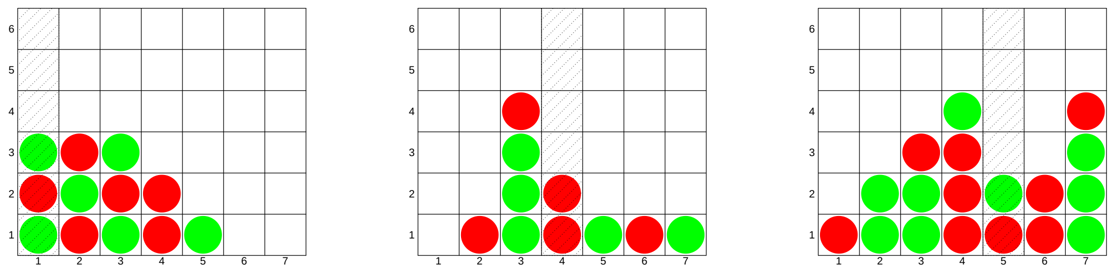
On the other hand, effective move sorting is certainly not a purely academic issue. As it is well known, the ordering of moves has a crucial impact on the efficacy of alpha-beta search methods [29], and may lead to significant speed-up of the search process.

Our main concern, when designing the system, was devoted to the implementation of postulates related to data representation and management (P5, P6 and P7). Furthermore, since the learning process is performed with the use of a hierarchical, multi-layer neural network-based system, postulates P1 and P2 are also partly addressed.

## A Connect Four

The rules of Connect Four are relatively simple. The following description comes from Wikipedia [30]: "Connect Four (also known as Four Up, Plot Four, Find Four, Four in a Row, and Four in a Line) is a two-player game in which the players first choose a color and then take turns dropping their colored discs from the top into a seven-column, six-row vertically-suspended grid. The pieces fall straight down, occupying the next available space within the column. The object of the game is to connect four of one's own discs of the same color next to each other vertically, horizontally, or diagonally before one's opponent can do so."

Let us assume that there are two players: Red and Green. In the game situation presented in Figure 1(a), Red can win immediately by playing in column 1 (*playing 1* for short) and building a diagonal-four in columns 1-4. In the next example (Figure 1(b)) Red can win in two moves by playing 4 and posing a double threat for Green (column-four in column 4 and diagonal-four in columns 3-6) which cannot be simultaneously defended. Similarly, in position 1(c) Red can win in two moves by playing 5 and posing a double threat of building-up a winning four in the third row (columns 2-5 or columns 3-6).



(a) Red to win in one move (b) Red to win in two moves (c) Red to win in two moves

Figure 1: Three examples of Connect Four positions winning by Red (if they are to move). The column to be played by Red is shadowed (columns 1, 4 and 5, respectively).

Connect Four is obviously not a very complicated game. It was solved in 1988 by Victor Allis [31], who showed that the winner is the one who makes the first move (under the assumption of their perfect play) and the winning initial move is 4 (the central column). On the other hand, the game is by no means trivial for the search-free or shallow-search-based methods. In many game positions, appropriate assessment of particular moves requires performing quite a deep search. In this respect, for a search-free pattern-based approach proposed in this paper, the game poses several research challenges.

## *B The Main Components of the System*

In order to provide a formal description of the system, let us denote the player who is to move as the active player (ACT) and their opponent as the passive player (PAS). For the sake of clarity of the presentation, unless stated otherwise, the ACT player will be the Red player and the PAS player will be the Green one.

### 1 *First-Level Contexts, their Features and Attributes*

The primary facet of the proposed system is making use of the so-called *first-level context* (or *context* in short), which defines the first-level cognitive feature aggregate. Contexts are defined based solely on the rules of the game and definition of its goal (description of the terminal states). Contexts make use of partial board information gathered from a particular area of the board. In the case of Connect Four, the first-level contexts include the quadruples of board consecutive squares located either horizontally or vertically or diagonally as presented in Figure 2.

Each context is coded as a 4-element vector (representing four consecutive squares) whose values belong to the set $\{-1, 0, 1\}$, where 0 denotes that the respective field is empty, $-1$ denotes the PAS's disc and 1 the ACT's disc, respectively.

For each context there are some number of *features* assigned to it, which define its basic characteristics. In the current implementation of the system, there are four such features denoted by [F1]-[F4]:

- [F1]: the fact of **being a context** (a trivial, though pertinent, feature which is possessed by each context);

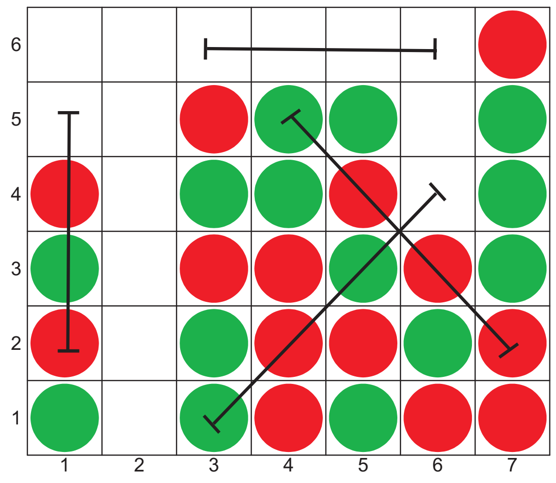


Figure 2: The notion of a context considered in this paper refers to any four consecutive squares in a row, column or any of the two diagonal directions (\-type or /-type). One example of each type of contexts is presented in the figure.

- [F2]: the **type of a context** - one of the following four {*row*, *column*, */-diagonal*, *\\-diagonal*};
- [F3]: the **column location** (*c-location* in short) of a context, which for row and /-diagonal contexts is defined as the leftmost column out of these in which the context is placed; for the column context it is the number of the column which includes the context, and for the \\-diagonal context it is the rightmost column among these in which the context is placed;
- [F4]: **depth of a context**, which is defined as the biggest number of discs that need to be placed in one of the columns containing the context so as to fill the context (in that column).

For example, context $K$ presented in Figure 3 is represented by a vector [-1,0,0,-1] (Green, empty, empty, Green) and has the following four features: [F1] - the fact of being a context, [F2] - type is /-diagonal, [F3] - c-location = 3, [F4] - depth = max{0,2,3,0} = 3 (since the maximum number of discs that must be placed in any of the context's columns to fill the context equals 3, in column 5).

Each context possesses also four *attributes* denoted by [A1]-[A4]:

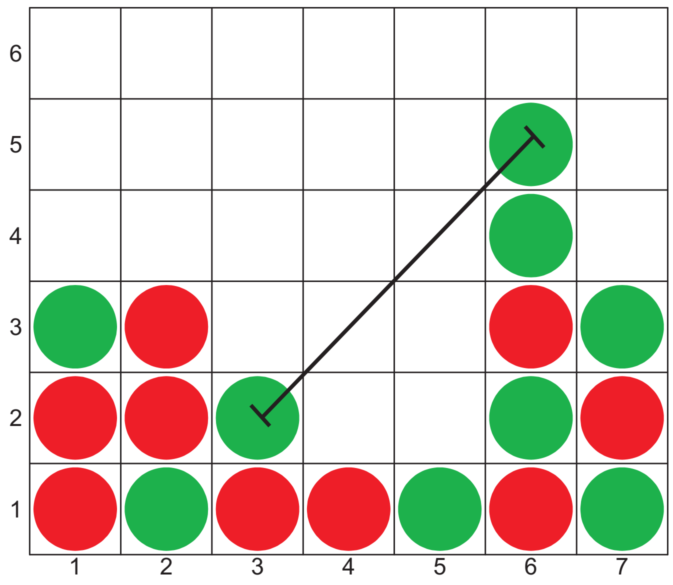


Figure 3: Features of the context. Context $K$ is in fact a context [F1], is of /-diagonal type [F2], is located in the 3rd column [F3], and its depth equals 3 [F4].

- [A1]: whether the **context is empty** (i.e. includes only empty squares); then the corresponding attribute is set to 1, otherwise it is set to 0;
- [A2]: whether **any of the playing sides can potentially win the game by completing the context** (i.e. whether the context is empty or contains discs of one player only); then the attribute is set to 1, otherwise it is set to 0;
- [A3]: whether **the context is closed** (all its squares are occupied); then the respective attribute is set to 1, and it is set to 0, otherwise;
- [A4]: **which of the playing sides can win the context**; if it is the ACT player then the attribute is set to 1, if the PAS player, then it is set to -1; if none of the playing sides can win the context or the context is empty, then the attribute's value is set to 0.

The vector of attributes $A(K)$ of our exemplar context $K$ from Figure 3 is of the form [0,1,0,-1]: the context in not empty [A1], it can be won [A2], it is not closed [A3], and the side which may potentially win by completing this context is Green (the PAS player) [A4].

The first-level contexts are combined into more complex structures (higher-level contexts) constructed with the help of a connectionist feed-forward neural network with shared weights (described in section 4 below). The overall (unfolded) network is relatively large, but since most of the weights are shared, the effective number of connections actually used in a given moment is usually much smaller than the maximum capacity of the network.

## 2 *Locality of Board Representation*

Locality of information, which is one of the postulated crucial concepts in human data acquisition and memorization is assured not only through the context-based representation, but also by restricting the system's focus to particular regions of the board.

In the current implementation of the system, the entire board is divided into three overlapping frames, as depicted in Figure 4, which will be referred to as $frame_1$ (the leftmost one), $frame_2$ (the middle one) and $frame_3$ (the rightmost one). Each frame

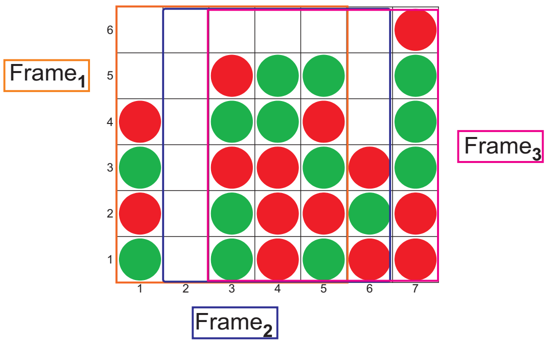


Figure 4: Division of the entire board into 3 overlapping frames, each of size $6 \times 5$. Initially contexts extracted from each of the frames are processed independently from the other contexts (except for the weight sharing mechanism). Only in the penultimate layer of the system the information from all three frames is combined (see section 4 for details).

contains 39 contexts (please note that only the contexts which are entirely included in the frame are counted as belonging to that frame), among which there are 12 contexts in rows (2 in each row), 15 column contexts (5 in each column), 6 /-diagonal contexts, and 6\-diagonal ones.

Observe that the number of frames that a given context belongs to vary among contexts. Some contexts belong to one frame only (e.g. the ones which originate in the first column belong exclusively to $frame_1$), the majority of contexts belong to two frames (either $frame_1$ and $frame_2$ or $frame_2$ and $frame_3$), and some contexts (the column ones located in columns 3, 4 or 5) belong to all three frames.

## 3 *Context-Based Local Feature Detectors*

Each context when input to the system (connectionist neural network) is preprocessed by specialized local feature detectors placed in the first hidden layer of the system. Each feature detector is implemented as a simple Perceptron network with 9 inputs (a context, its attributes, and a bias) and 3 outputs. If we denote by $V_K$, a vector representing context $K$, its attributes, and additional bias input i.e.

$$V_K = [K, A(K), 1] = [v_K^1, \ldots, v_K^9], \tag{1}$$

by $f^*$ the nonlinear transformation implemented by the feature detector $*$ (see below), by $w_{ij}^*, i = 1, 2, 3, j = 1, \ldots, 9$ the weights of this feature detector and by $O^* = [o_1^*, o_2^*, o_3^*]$ its output, then

$$o_i^* = g(\sum_{j=1}^{j=9} w_{ij}^* v_K^j), \quad i = 1, 2, 3, \tag{2}$$

where $g$ is a non-linear s-shaped transfer function.

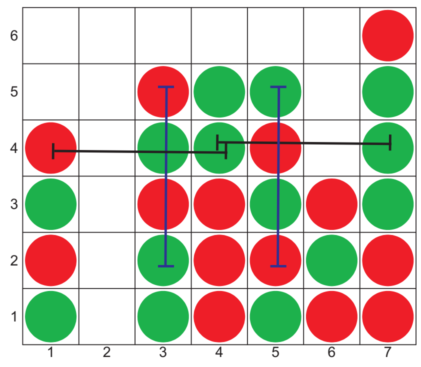
Due to a variety of possible context's features an adequate variability of feature detectors is required. In the current system's implementation the following feature detectors, denoted by [D1]-[D4], are used:

- [D1]: a detector which is applied to all contexts (**corresponding to the trivial feature [F1]**) whose transformation function will be denoted by $f^{-|/\backslash}$. In the full (unfolded) network all contexts share a common set of weights assigned to this detector;

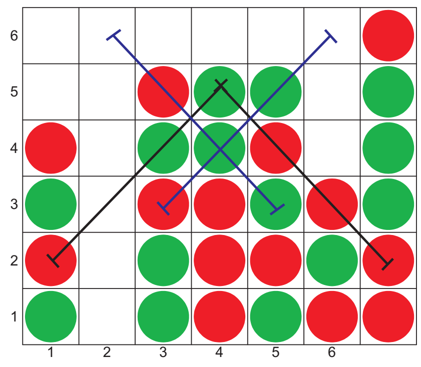
- [D2]: 3 detectors **corresponding to feature [F2]**, whose transformation functions will be denoted by $f^*$, where $* \in \{-, |, /\backslash\}$, respectively for the row, column, and diagonal contexts (with weight sharing between the two diagonal types of contexts). All row contexts share the same set of weights of $f^-$ detector, likewise all column contexts and all diagonal ones;

- [D3]: 10 detectors **corresponding to combinations of features [F2] and [F3]** with symmetrization (weight sharing) with respect to column 4 (see Figure 5); the respective transformations will be denoted by $f^*_{\{x,y\}}$, where $* \in \{-, |, /\backslash\}$ and $\{x, y\} \in \{\{1, 7\}, \{2, 6\}, \{3, 5\}, \{4, 4\}\}$, and all combinations of $*$ and $\{x, y\}$ are valid except for $f^-_{\{3,5\}}$ and $f^-_{\{4,4\}}$. Similarly to the previous cases, the respective weights are shared among contexts. For example the weights of column contexts located in columns two and six share the weights of $f^{|}_{\{2,6\}}$ detector;

- [D4]: 21 detectors **corresponding to combinations of features [F2] and [F4]** (depth of a context); the respective transformations will be denoted by $f^*_n$, where $* \in \{-, |, /\backslash\}$ and $n = 0, \ldots, 6$. Again, all contexts of a certain type $*$ and a certain depth $n$ share a common set of weights of $f^*_n$ feature detector.



(a) Examples of row and column context symmetrization



(b) Examples of diagonal context symmetrization

Figure 5: Four examples of context symmetrization is the system: row and column symmetrization (Figure 5(a)), and symmetrization of diagonal contexts (Figure 5(b)).

All together there are 35 feature detectors, each of them (except for the "universal" one) devoted to a specific type of a context, or a combination of a type and c-location, or a combination of a type and depth, with appropriate weight sharing. Depending on the particular design of the system each context may be processed by either 1 or 2 feature detectors. Since each feature detector has 3 outputs, then each context is represented in the first hidden layer by either 3 or 6 neurons per each frame to which the context belongs to.

As an example, let us consider context $K$ presented in Figure 3. Due to its location, it will be included in $frame_2$ and $frame_3$, but not in $frame_1$. Depending on the system's setup, context $K$ will be preprocessed by a selected subset of the following detectors: the "universal" detector [D1] (function $f^{-|/\backslash}$), the diagonal one [D2] (function $f^{/\backslash}$), the combined (type and c-location) detector [D3] (function $f^{/\backslash}_{\{3,5\}}$), the combined (type and depth) detector [D4] (function $f^{/\backslash}_{3}$). This choice of detectors will be duplicated (since $K$ belongs to two frames) with appropriate sharing of weights among corresponding detectors.

## 4 ***System Architecture and Training Scheme***

Let us assume that there are $M$ training patterns $P^i, i = 1, \ldots, M$ and that each $P^i$ is a pair $(B^i, m^i)$, where $B^i$ represents the training position and $m^i$ the optimal move (a number of a column, in which the ACT player should place their disc $(1 \leq m \leq 7)$). Each board pattern $B^i$ is first decomposed into 3 frames and depicted in Figure 4, denoted by $B^i_j, j = 1, 2, 3$. From each frame $B^i_j$, all possible contexts $K^i_{j,n}, n = 1, \ldots, 39$ are extracted (in a priori defined order which is fixed during the whole experiment). Hence, each training pattern $P^i$ is decomposed into $3 \times 39$ contexts $K^i_{j,n}$ used for training the network's weights.

In the first step each context $K^i_{j,n}$ (together with its attributes) is preprocessed by appropriate specialized feature detectors. In the simplest case, only one feature detector [D1] is applied leading to a three-element representation of a context. In more complex cases (depending on the system's design) [D1] is accompanied by one of the detectors of type [D2]-[D4] leading to a six-element context representation (per each frame the context belongs to).

The system is represented as a connectionist feed-forward neural network with three layers, consisting of sigmoidal neurons. In the first hidden layer all contexts of a given training pattern (preprocessed in the above described way) are placed in a predefined order. If we denote the number of feature detectors used in the preprocessing phase by $N, (N = 1, 2)$, then the size of the first hidden layer of the system equals $3 \times (39 \times N \times 3)$ (39 contexts, each represented by $N \times 3$ neurons, multiplied by 3 frames).

Each $39 \times N \times 3$ section of the first hidden layer defines a convolutional sublayer of the network (see Figure 6). Each of the three sublayers is fully connected to the respective part of the second hidden layer by the shared set of $39 \times N \times 3 \times 5$ weights.

Finally, the $5 \times 3$ second hidden layer neurons are fully connected to the output layer composed of 7 sigmoidal units providing system's output. The higher the value of the output neuron the greater the preference of the system for placing a disc in the respective column.

As the training algorithm, we use backpropagation method without momentum and with weight sharing. Sharing of weights is implemented by calculating the sum of the gradient values (from all processed contexts) of the weights that are shared and applying this cumulative value to update each of these shared weights. A standard mean square error function is used. The learning rate is equal to 0.1.

## *C Experiment Design and Results*

The task of the system is to sort columns of a Connect Four position according to the priority of respective moves with no use of search mechanism whatsoever. In the ideal case, the highest output should be associated to the column representing the best move, the second highest output to the second-best move, etc.

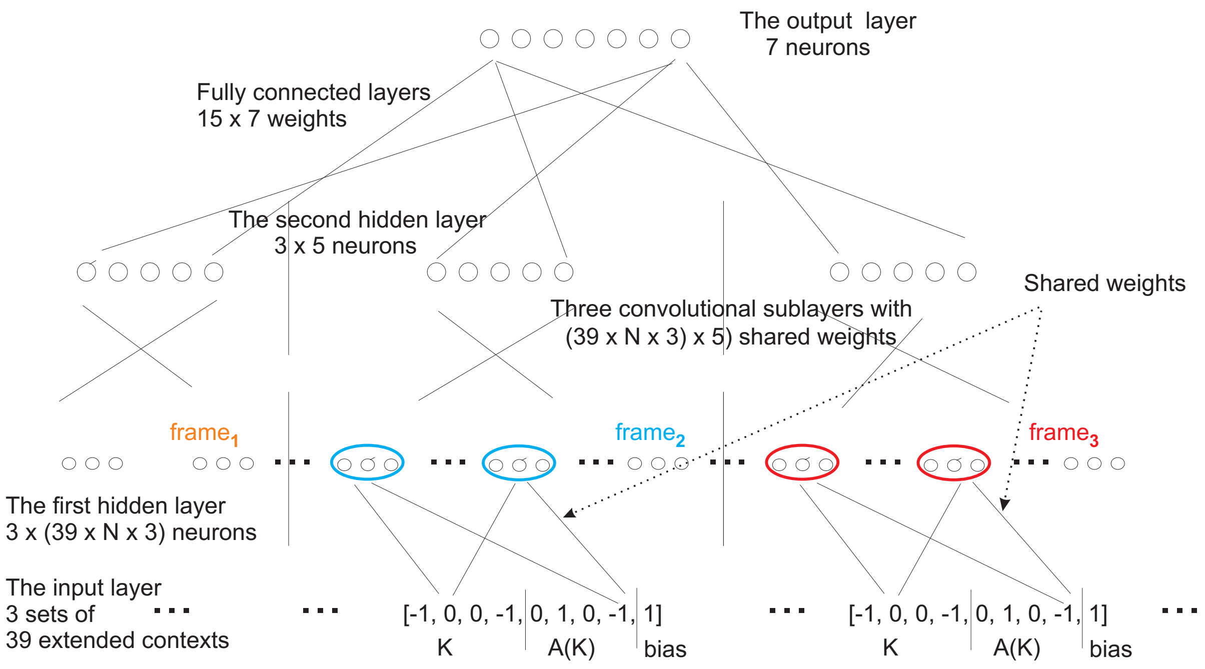


Figure 6: Overview of the system. First, each extended context is preprocessed by $N$ ($N = 2$ in the figure) appropriately chosen specialized feature detectors (the lower part of the figure). The weights of the respective detectors are shared among contexts and among frames - see description within the text. The preprocessed contexts (their $N \times 3$-value representations) constitute the first hidden layer of the system. Each of the three sublayers representing particular frames in the first hidden layer is fully connected to the respective 5-neuron second hidden sublayer through the convolutional set of weights. The outputs of these three 5-tuples are fully connected to the final 7-unit output layer. For example, context $K$ presented in Figure 3 belongs to $frame_2$ and $frame_3$, but does not belong to $frame_1$.

## 1 *Training and Testing Data*

In order to train and test various configurations of the feature detectors in the proposed system a set composed of over 420 000 Connect Four positions, together with the associated best moves was generated with the help of publicly available, presumably perfectly playing program Velena [32] written by Victor Allis. Out of these positions, 10 non-overlapping sets, each containing 30 000 positions were chosen and subsequently used in 10 independent runs of the experiment. From each set, half of the positions were used for training and the other half for testing. In each run of the experiment, training was performed until the learning curve entered a plateau (which in most cases happened already after a few thousand epochs).

The following selections of feature detectors were tested:

- [S1]: only the universal detector (applied to all contexts) was used;
- [S2]: detector [S1] was used together with 3 detectors corresponding to feature [F2], with weight sharing between two types of diagonal contexts;
- [S3]: detector [S1] was used together with 10 detectors corresponding to combinations of features [F2] and [F3], with respective weight sharing;
- [S4]: detector [S1] was used together with 21 detectors corresponding to combinations of features [F2] and [F4], with respective weight sharing.

Each of system's versions [S1]-[S4], as described in section $B$, depends heavily on the *locality* of context-based feature detectors and on the mechanism of *weight sharing.* In order to truly verify the advantage of both these architectural aspects, the following neural architectures were trained and tested on the same data sets for the sake of direct comparison with systems [S1]-[S4]:

- [N20]: a fully connected feed-forward network with one hidden layer composed of 20 units and with 7 output neurons; in the input layer the entire game positions were presented coded as 1 for ACT's disc, $-1$ for PAS's disc, and 0 otherwise;
- [N40]: as above, but with 40 hidden units;
- [N60]: as above, but with 60 hidden units;
- [C]: a convolutional network composed of three sublayers (corresponding to three frames, in the same way as in the main system) with shared weights, with 7 outputs and with the same board coding in the input layer as in the case of the three above described architectures.

The first three neural architectures were selected to test the efficacy of *global vs. local context-based representation.* The fourth one was chosen to assess the importance of the *convolutional mechanism.*

## 2 *Results*

The main results are presented in Tables 1 and 2, respectively for training and testing data. In both tables, each neural architecture is described in two rows. The upper one presents the fraction of positions in which the best move was pointed out in the first, second, ..., seventh place, averaged over 10 trials. The lower row shows standard deviation of these values. The most indicative is the first value, i.e. the fraction of the cases in which the best move was found. The value denoted as *AvRank* in the rightmost column presents the aggregated estimation of the system's efficacy in the form of the *average rank* of the best move (if it were equal to 1, it would mean the best move was found in all cases). For comparison purposes, for all 300 000 samples, the average rank of the uniform random choice strategy was calculated, which represents the expected quality of the blind choice algorithm. Not surprisingly, this value was close to 4 (equal to 3.961 to be exact).

| system | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | *AvRank* |
|---|---|---|---|---|---|---|---|---|
| [S1] | 0.55832 | 0.18964 | 0.10459 | 0.06645 | 0.04386 | 0.02588 | 0.01122 | 1.970 |
| | 0.00736 | 0.00602 | 0.00375 | 0.00313 | 0.00174 | 0.00301 | 0.00267 | |
| [S2] | 0.59086 | 0.17135 | 0.09327 | 0.06024 | 0.04059 | 0.02748 | 0.01618 | 1.935 |
| | 0.00468 | 0.00782 | 0.00316 | 0.00300 | 0.00246 | 0.00420 | 0.00512 | |
| [S3] | 0.60276 | 0.16357 | 0.0895 | 0.05919 | 0.04217 | 0.02706 | 0.01576 | 1,918 |
| | 0.00676 | 0.00757 | 0.00315 | 0.00292 | 0.00275 | 0.00275 | 0.00413 | |
| [S4] | **0.63721** | 0.15930 | 0.08346 | 0.05268 | 0.03507 | 0.02119 | 0.01109 | **1.797** |
| | 0.00529 | 0.00607 | 0.00150 | 0.00178 | 0.00173 | 0.00283 | 0.00411 | |
| [N20] | 0.48555 | 0.19949 | 0.12255 | 0.08204 | 0.05817 | 0.03652 | 0.01571 | 2.200 |
| | 0.00441 | 0.00928 | 0.00495 | 0.00169 | 0.00476 | 0.00561 | 0.00267 | |
| [N40] | 0.55181 | 0.18302 | 0.10409 | 0.06958 | 0.04842 | 0.02961 | 0.01348 | 2.022 |
| | 0.00486 | 0.00504 | 0.00286 | 0.00286 | 0.00337 | 0.00409 | 0.00360 | |
| [N60] | 0.59532 | 0.15713 | 0.08887 | 0.06433 | 0.04521 | 0.02899 | 0.02010 | 1.974 |
| | 0.00602 | 0.00399 | 0.00285 | 0.00236 | 0.00332 | 0.00140 | 0.00586 | |
| [C] | 0.38428 | 0.22982 | 0.15566 | 0.10365 | 0.06787 | 0.04050 | 0.01819 | 2,435 |
| | 0.02457 | 0.01564 | 0.01069 | 0.01153 | 0.00913 | 0.00831 | 0.00826 | |

Table 1: Training results. Each neural architecture is described in two rows. The upper one presents the fraction of positions in which the best move was pointed out in the first, second, ..., seventh place, averaged over 10 trials. The lower row shows standard deviation of these values. *AvRank* in the rightmost column presents the aggregated estimation of the system's efficacy in the form of the *average rank* of the best move (the closer this value to 1.000, the better).

| system | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | *AvRank* |
|---|---|---|---|---|---|---|---|---|
| [S1] | 0.53211<br>0.00962 | 0.19894<br>0.00619 | 0.11193<br>0.00444 | 0.06955<br>0.00396 | 0.04555<br>0.00252 | 0.02942<br>0.00309 | 0.01248<br>0.00405 | 2.035 |
| [S2] | 0.52837<br>0.00616 | 0.19634<br>0.00495 | 0.10889<br>0.00298 | 0.07053<br>0.00317 | 0.04768<br>0.00172 | 0.03048<br>0.00294 | 0.01768<br>0.00460 | 2.074 |
| [S3] | 0.52796<br>0.00556 | 0.19409<br>0.00497 | 0.10943<br>0.00553 | 0.07092<br>0.00265 | 0.04864<br>0.00339 | 0.03174<br>0.00341 | 0.01720<br>0.00465 | 2.082 |
| [S4] | **0.58621**<br>0.00930 | 0.18107<br>0.00536 | 0.09746<br>0.00240 | 0.06171<br>0.00284 | 0.03780<br>0.00154 | 0.02349<br>0.00332 | 0.01223<br>0.00475 | **1.903** |
| [N20] | 0.43894<br>0.00816 | 0.21677<br>0.00936 | 0.13356<br>0.00754 | 0.09102<br>0.00510 | 0.06340<br>0.00447 | 0.03913<br>0.00506 | 0.01716<br>0.00294 | 2.309 |
| [N40] | 0.45198<br>0.00831 | 0.22054<br>0.00558 | 0.13598<br>0.00378 | 0.08671<br>0.00227 | 0.05723<br>0.00372 | 0.03268<br>0.00379 | 0.01490<br>0.00261 | 2.234 |
| [N60] | 0.44537<br>0.00514 | 0.22343<br>0.00508 | 0.13725<br>0.00334 | 0.08516<br>0.00218 | 0.05407<br>0.00406 | 0.03247<br>0.00257 | 0.02228<br>0.00489 | 2.265 |
| [C] | 0.37807<br>0.02365 | 0.22950<br>0.01637 | 0.15804<br>0.01301 | 0.10489<br>0.00922 | 0.06915<br>0.00931 | 0.04090<br>0.00954 | 0.01945<br>0.00937 | 2.458 |

Table 2: Testing results. See description below Table 1.

A general conclusion that can be drawn from presented results is that cognitively-inspired system proposed in this paper proved to be effective and seems to have potential for further development. Based on the analysis of testing results (Table 2), it is evident that all implementations of the system (coded as [S1] to [S4]) outperformed the competitive neural architectures with a wide margin. All of them achieved over 52.7% rate in finding the best move, compared to $37.8\% - 45.2\%$ range accomplished by [N$xx$] and [C] architectures. The supremacy is especially visible in the case of [S4] implementation which accomplished over 58.6% result in finding the best move. The fact that neither classical neural networks nor a convolutional architecture managed to come close to context-based implementations suggests that the superiority of the proposed approach stems from the specificity of local, context-based position representation, rather than neural network mechanisms as such. This observation offers prospects for further research development in this area.

Despite very encouraging results in general, there are a few aspects that require further investigation. First of all, in around 1% of training and testing positions, the best move was chosen as the last one (rank 7). We are not yet convinced what the reasons for such a poor move selection in these cases were, but one of the possible explanations is related to the "vague" notion of optimality in the move selection process. In some game positions, there is more than one move which can lead to a win. Most probably, the program that we used to generate game positions and the associated optimal moves assigns the label "best move" to the one which leads to accomplishing the ultimate goal in the fastest way. Hence, alternative paths of play, which led to the best achievable position (with respect to the current game state), but in a greater number of moves, were not considered to be the optimal ones (neither during training nor in the testing phase). Such a specific notion of the "best move" might potentially impair system's performance. Since detailed examination of this issue requires laborious manual verification of doubtful positions, we didn't manage to complete this task yet, but preliminary investigations partly support this hypothesis.

Observe that in the tests on the training data (Table 1) one of the "classical" neural architectures, namely [N60], was comparably effective to the context-based architectures [S1]-[S4]. It should be noted, however, that [N60], and to some degree also [N40], were clearly *overfitted* to the training data. This conclusion becomes apparent after analysis of the testing results (Table 2), where [N$xx$] architectures (for $xx \in \{20, 40, 60\}$) are far inferior to [S$y$] ones (for $y \in \{1, 2, 3, 4\}$).

## 3 ***Hand-Coded Examples***

The results presented in the previous section proved advantage of [S1]-[S4] system's implementations (in particular the [S4] one) over classical architectures on statistical ground. In order to illustrate the performance of the system from another perspective, we hand-crafted three Connect Four positions and tested on them all architectures after each training run of the experiment (i.e. 10 times each).

In the first position, depicted in Figure 7, Red should play 4 with a threat for completing a /-diagonal context in columns 1-4. After Green's move in column 1, Red plays 7, which forces Green to play 6, and then Red after playing 6 wins a \-diagonal context in

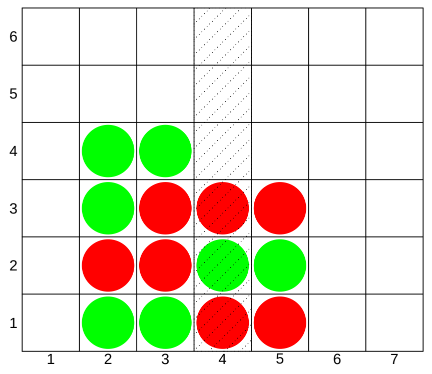


Figure 7: Red to move and win. There are two winning moves: in column 4 and in column 7.

columns 4-7. The position is not simple and requires a few-ply analysis. The alternative winning combination starts with 7 and proceeds through 6 (Green) and 4 (Red).

The average ranks for all 10 trials are presented in Table 3.

| system | move 1 | move 2 | move 3 | **move 4** | move 5 | move 6 | **move 7** |
|---|---|---|---|---|---|---|---|
| [S1] | 5.6 | 3.3 | 4.3 | **1.7** | 2.0 | 5.3 | 5.8 |
| [S2] | 4.9 | 2.8 | 4.6 | 2.5 | **1.8** | 5.6 | 5.8 |
| [S3] | 5.5 | 3.6 | 4.1 | **1.8** | 2.3 | 4.4 | 6.3 |
| [S4] | 4.4 | 4.1 | 3.9 | **1.1** | 4.7 | 4.8 | 5.0 |
| [N20] | 5.9 | 5.0 | 2.2 | **1.4** | 2.7 | 4.2 | 6.6 |
| [N40] | 5.7 | 5.15 | 3.1 | 2.2 | **1.8** | 3.7 | 6.35 |
| [N60] | 6.35 | 4.2 | 2.9 | **1.5** | 2.8 | 3.7 | 6.55 |
| [C] | 5.4 | 5.6 | **1.4** | 2.3 | 2.5 | 3.9 | 6.9 |

Table 3: Test results for position depicted in Figure 7. There are two winning moves for Red in this position: 4 and 7. For each tested architecture the average ranks of all possible moves (from column 1 to column 7) are presented. Ranks in bold are the best (lowest) ones.

In case of majority of architectures (5 out of 8), the best move was in average pointed with the lowest (best) rank. The best performance was accomplished by [S4] system, which managed to choose the optimal move in 9 out of 10 cases (the average rank was equal to 1.1).

This test position confirmed the hypothesis that some inefficiencies in the learning process, associated with the choice of the best move in the training positions, must undoubtedly exist. The alternative continuation - move 7 (although "less obvious", but anyway leading to a win) was ranked between 5.0 and 6.9. Such property, as we discussed above, should most probably be attributed to the "single best move presentation" in the training process, which potentially hinders the ability to appropriately value the alternative winning moves.

The second position is depicted in Figure 8.

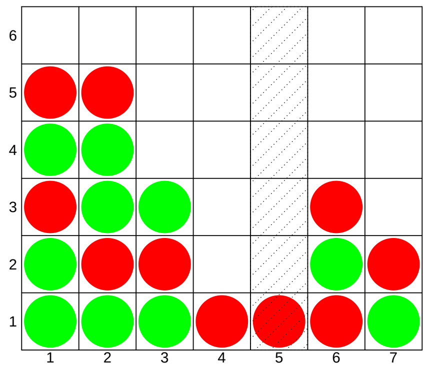


Figure 8: Red to move and win. The winning move is in column 5.

The position is rather challenging. At first glance there exist two equally strong moves: 4 and 5. Apparently only move 5 leads to a win, but the winning combination (from pattern-based search-free method perspective) is quite demanding. After Red's playing 5, Green is forced to play 4. Red then plays 7 forcing Green to respond 7, and after Red's 4 the position is won (a simultaneous \-diagonal and row threat). Please note, that a tempting alternative of playing 4 (if consequently continued) leads to Green's win. Although after Red's 4 Green has to play in defence 5, but if Red continues their attack in the 4-th column, Green plays 4 and poses a double threat (the 4-th row and \-diagonal 4-7).

The average ranks for all 10 trials are presented in Table 4.

The position appeared to be complicated enough to mislead all tested architectures,

| system | move 1 | move 2 | move 3 | move 4 | **move 5** | move 6 | move 7 |
|---|---|---|---|---|---|---|---|
| [S1] | 5.1 | 6.4 | 2.7 | **2.4** | 2.7 | 2.9 | 5.8 |
| [S2] | 4.3 | 4.8 | **2.7** | 3.0 | 4.1 | 3.5 | 5.6 |
| [S3] | 4.3 | 4.7 | **1.7** | 3.3 | 3.6 | 3.6 | 6.8 |
| [S4] | 4.9 | 5.35 | 3.0 | **1.5** | 2.3 | 4.65 | 6.3 |
| [N20] | **2.0** | 5.6 | 2.1 | 3.8 | 4.0 | 3.6 | 6.9 |
| [N40] | **2.5** | 5.4 | 3.0 | 3.6 | 3.4 | 3.5 | 6.6 |
| [N60] | 2.9 | 5.7 | **2.5** | 3.9 | 2.9 | 3.5 | 6.6 |
| [C] | 5.9 | 5.5 | **1.2** | 2.9 | 3.4 | 2.8 | 6.3 |

Table 4: Test results for position depicted in Figure 8. The winning move for Red is 5. A potentially viable alternative (move 4) does not lead to Red's win (in a foreseeable perspective). For each tested architecture the average ranks of all possible moves (from column 1 to column 7) are presented. Ranks in bold are the best (lowest) ones.

including [S4] - the most effective one in the main experiment. System [S4] was visibly trapped by the "false beauty" of move 4, giving it a clear preference (6 choices out of 10, the remaining ones being 5 - twice, and 3 - twice). Move 5 was ranked by [S4] as the second-best choice (with a rank 2.3).

The third manually chosen position is depicted in Figure 9.

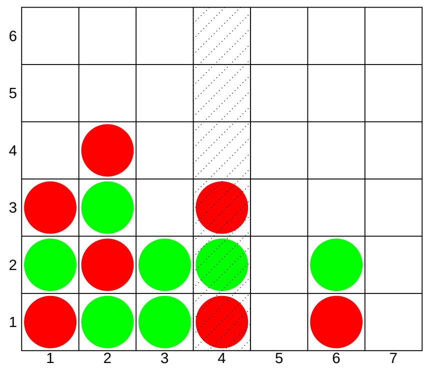


Figure 9: Red to move and win. The winning move is in column 4.

This position was intended to be a challenge of a similar quality as the previous one. The optimal move for Red is 4, which requires Green to respond with 3, and after Red's playing 3 and Green's response 1, Red plays in the 7-th column, forcing Green's move in

5, after which Red by playing 5 completes a \-diagonal context in columns 3-6. Note that it takes 7 moves (7-ply) to complete the winning sequence. The position is even more interesting because it additionally offers an alternative path of play which leads to Red's win in all cases except for one, and this defence line for Green is not forceful. Namely, if Red starts with 7, Green answers with 5, which in turn forces Red to play 5, then Green has all options to play except for 3, which leads to immediate loss. However, only if they play 4 can they defend the position from immediate loss, since otherwise (assume e.g. move 5 for Green) Red plays 4 and Green is faced with a stark choice: either to play 3 and lose after Red's plays 3 or to play any another move, and ... lose after Red's 3. All in all the position seems to be far from trivial.

The average ranks for all 10 trials are presented in Table 5.

| system | move 1 | move 2 | move 3 | **move 4** | move 5 | move 6 | move 7 |
|---|---|---|---|---|---|---|---|
| [S1] | 4.1 | 4.8 | 4.1 | **1.1** | 4.5 | 5.0 | 4.4 |
| [S2] | 3.6 | 4.6 | 3.0 | **1.7** | 5.1 | 4.6 | 5.4 |
| [S3] | 3.15 | 4.5 | 2.8 | **1.5** | 5.15 | 4.8 | 6.1 |
| [S4] | 4.0 | 5.8 | 3.5 | **1.5** | 4.8 | 3.4 | 5.0 |
| [N20] | 4.3 | 4.7 | 2.8 | **1.8** | 3.2 | 4.2 | 7.0 |
| [N40] | 4.1 | 4.2 | **2.3** | 2.8 | 3.7 | 3.9 | 7.0 |
| [N60] | 4.8 | 4.8 | 2.6 | **1.5** | 4.4 | 3.3 | 6.6 |
| [C] | 5.2 | 5.4 | **1.8** | 2.0 | 2.9 | 3.9 | 6.8 |

Table 5: Test results for position depicted in Figure 9. The winning move for Red is 4. For each tested architecture the average ranks of all possible moves (from column 1 to column 7) are presented. Ranks in bold are the best (lowest) ones.

Surprisingly enough, all networks except for [N40] and [C] were able, in average, to assign the lowest rank to the best move. Closer examination reveals that the frequency of making the optimal choice differed among the tested architectures. The best by far was [S1] with 9 out of 10 score, followed by [S4] with 8 out of 10 best choices, [S3] (7 out of 10) and [N60] (6 out of 10).

## *D Summary of Results*

The experiments have shown that the method relying on context-based feature detection, inspired by cognitive foundations clearly outperformed "classical" feed-forward network-based systems as well as the convolutional network. The best overall performance was exhibited by [S4] system, which utilized the highest number of context-based feature detectors among [S1]-[S4]. Recall that data representation in [S1] relied on a one-dimensional (or alternatively single-type) feature detector, i.e. all contexts shared one type of detector. In the case of [S2] or [S3], two classes of detectors were applied: the aforementioned "universal" detector along with 3 orientation-based or 10 orientation-with-column-based ones, respectively. The [S4] implementation of the system also utilized two classes of detectors, all of which, except for the "universal" one were based on a combination of orientation and depth. These detectors constituted a more diverse set, composed of 21 elements.

Analysis of results suggests that diversity of context detectors is advantageous - [S4] was undoubtedly a superior implementation, outperforming the remaining ones by a visible margin. On the other hand, it should be noted that gradual complication and increasing variability of the context detectors is not monotonically correlated with the increase of their performance: [S1], [S2] and [S3] exhibited comparable performance, despite their various levels of complexity. One of the possible explanations of this phenomenon is related to potential importance of the depth aspect in the detectors' design, which was taken advantage of only in [S4] system.

On a more general note, since [S1]-[S4] architectures differ from [N20]-[N60] and [C] mainly by using local, context-based input representation with shared weights, and additionally from [N20]-[N60] (but not [C]) by having a convolutional layer, the visibly higher efficacy of these systems should be attributed to the locality of representation and specialization of context-detectors rather than to general neural network-related representation and learning capabilities. Certainly, Connect Four is not complex enough a game to validate firm conclusions, but experimental results presented in the paper bring hope for further successful application of the context-based approach also in more challenging environments.

Despite the above promising insights, there are several open questions, which need to be addressed in future research. One of the weaknesses of the proposed approach is “too selective” learning which is manifested by completely ignoring alternative moves leading to a win in a predictable perspective. As previously stated, this type of behavior most probably stems from the imperfection of the learning process, which does not include alternative options (winning moves) in the course of training. Investigation of this issue and adequate modification of the training procedure is one of our research targets at the moment.

## V Conclusions and Future Research

We have presented the argumentation for potential virtues of cognitively-plausible, pattern-based playing systems. In particular, several basic concepts which in our opinion should underline the development of such systems are listed and briefly commented.

In order to illustrate these underpinning ideas of cognitively-plausible data representation, they were applied to the game Connect Four. Certainly, in order to make the judgement of results fair, one needs to keep in mind that the underlying ideas of the proposed approach are located in the framework of human-like pattern-base search-free methods, which by no means (at least at this stage of development) should be compared with established AI approaches. The above remark is particularly true in the case of Connect Four, which is simple enough (for AI and mathematical analysis) to be solved, and consequently, the omnipotent, perfect players exist for this game. On the other hand, the relative simplicity of the game does not necessarily reflect in the area of CI-based solutions, especially those that do not rely on search and do not apply explicit evaluation function, as is the case of our system.

In this perspective, we believe that experimental results presented in the paper proved the usefulness of context-based approach and indicated its further potential in game domain.

Certainly, there is still a lot of room for modifications and improvements of the proposed system. Possible future research directions can be roughly concentrated along four dimensions, which are briefly summarized below.

## *A Direct Improvement of the Connect Four System*

One of our current concerns is in-depth analysis of the training data in the view to revealing the reasons for making a frequent choice of one particular best continuation in case more than one best move exists in a given position. We would also like to investigate the reasons for catastrophically poor performance in about 1% of the test positions, in which the expected move was ranked as the last one.

Furthermore, we plan to investigate the usefulness of context-based feature-detectors other than the ones already developed, as well as, assess the possible advantage of combining more than two types of detectors simultaneously, thus potentially extending dimensionality and flexibility of representation.

## *B Application to More Challenging Games*

The underlying facet of the proposed system is making use of the *context*, which defines the first-level cognitive aggregate (feature). Context is defined based *solely on the rules of the game and definition of its ultimate goal* (description of terminal states) and makes use of *local*, partial board information gathered from a particular board's area. This idea is general enough to be implemented also in other, more complicated board games, with a word of warning, however, that the more complicated the game rules are, the potentially more sophisticated the context definition and the higher number of feature detectors required. Nevertheless, thanks to the above mentioned flexibility of context-based representation, we hope to be able to demonstrate the strength of this approach in another, more demanding game, in particular chess.

In the case of chess, the shapes of the first-level contexts associated with the chess pieces should correspond to syntactical rules of their moves (rows and columns for the rooks, diagonals for bishops, etc.). The basic contexts will therefore be defined with respect to the *locations* of the pieces on the board and represent the *rules of their moving.* Similarly to Connect Four, the elements comprising the context (squares of the board) should have their specific *attributes*, which would form another aspect of context's definition. These first-level contexts should be combined into more complex structures - the second-level contexts, each of which including several first-level contexts. For instance, two chess figures attacking or defending a given square would have their contexts combined in the second-level context layer. These constructs will be combined into more complex structures in the subsequent layers of the system. Such hierarchical representation should allow gradual building of the entire board's representation from the lower-level concepts. Certainly, the above considerations only sketch out possible implementation of context-based approach in as complicated a game as chess is. A lot of conceptual problems need to be solved on the way before any valid and plausible result will be achieved, but despite the apparent difficulty of this task, we believe that the idea of using local, context-based representation is universal and strong enough to prove efficient also in challenging game domains, including chess.

## C Automatization of the Feature-Detectors Definition Process

The list of the long-term objectives of the proposed research includes also automatic definition of suitable context representation and related feature-detectors for any particular game at hand. Fulfilling this task, however, would require, above all, using a specific description language, which would enable formal, symbolic game description in the form of states, rules and goals.

One of the tempting alternatives is to concentrate on the subset of game repository within the General Game Playing (GGP) framework. GGP [33] was proposed at Stanford University and considers *finite, synchronous, multi-player* games. Such games can, technically, be represented as state machines. Their rules are defined in first-order logic, with some extensions. Game descriptions are provided with the use of Game Description Language (GDL) [34] in the form of logical predicates that must be true in every state of the game. For consistency reasons, we should restrict the above framework to two-player games with $2D$ (possibly rectangular) representation [1].

In theory, these logical predicates seem to be perfect candidates for low-level contexts, as they address local specificity of game positions. In practice, however, one is faced with a problem of visualization ($2D$ geometrical representation) of positions since a sequence of logical statements (moves played so far) does not lead to geometrical interpretation in a straightforward manner. What is more, game descriptions in GGP repository does not allow for automatic detection of a board square on which a move is performed. Some efforts were devoted to devise an algorithm to infer board-related predicates from game descriptions [35, 36], but as yet, unsuccessfully. Similarly, our initial efforts towards building an autonomous parser capable to transform Datalog-based logical description of the game into its $2D$ board-related representation, are so-far of limited success only.

## D Self-Adaptability of the System

One of the potential limitations of proposed approach is fixed architecture of the system. The number of layers, the intra-layer connections, as well as specification of the context-based detectors are defined prior to the learning process, and once set remain unchanged (except for adaptation of system's internal parameters - i.e. connection weights).

In a long-term perspective, we plan to adopt a more connectionist-type approach to the system's implementation, which assumes that the system's architecture is adaptable

[1]Please note, that GGP definition does not imply that the games are played on boards. Moreover, the number of players may vary from one to many.

and may be changed in the course of the learning process. Such a system may be built in an incremental manner with the capability of adding new elements (detectors, layers, connections, etc.) whenever necessary, on demand from the internal mechanism monitoring the training process.

## Acknowledgments

I would like to thank my Ph.D. student Cezary Dendek for his help in carrying out the experiments described in the paper.

## References

[1] C. E. Shannon, "Programming a computer for playing chess," *Philosophical Magazine*, vol. 41 (7th series), no. 314, pp. 256–275, 1950.

[2] A. M. Turing, "Digital computers applied to games," in *Faster than thought: a symposium on digital computing machines*, B. V. Bowden, Ed. Pitman, London, UK, 1953, ch. 25.

[3] A. Newell, J. C. Shaw, and H. A. Simon, "Chess-playing programs and the problem of complexity," *IBM Journal of Research and Development*, vol. 2, no. 4, pp. 320–335, 1958.

[4] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[5] L. Kocsis and C. Szepesvari, "Bandit based Monte Carlo planning," in *Proceedings of the 15th European Conference on Machine Learning (ECML'06)*, 2006, pp. 282–293.

[6] J.-B. Hoock, A. Rimmel, F. Teytaud, O. Teytaud, C.-S. Lee, and M.-H. Wang, "Intelligent agents for the game of Go," *IEEE Computational Intelligence Magazine*, vol. 5, no. 4, pp. 28–42, 2010.

[7] A. Rimmel, O.Teytaud, C.-S. Lee, S.-J. Yen, M.-H. Wang, and S.-R. Tsai, "Current frontiers in computer Go," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 4, pp. 229–238, 2010.

[8] J. Mańdziuk, *Knowledge-Free and Learning-Based Methods in Intelligenet Game Playing*, ser. Studies in Computational Intelligence. Berlin, Heidelberg: Springer-Verlag, 2010, vol. 276.

[9] F. Hsu, *Behind Deep Blue.* Princeton, NJ: Princeton University Press, 2002.

[10] H. Fietz, "Beyond the 3000 Elo barrier. a glance behind the scenes of the Rybka chess engine," *Chess Magazine*, pp. 18–21, May 2007.

[11] J. Schaeffer, *One Jump Ahead: Challenging Human Supremacy in Checkers.* New York: Springer-Verlag, 1997.

[12] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen, "Checkers is solved," *Science*, vol. 317, pp. 1518–1522, 2007.

[13] M. Campbell, A. J. Hoane Jr., and F. Hsu, "Deep Blue," *Artificial Intelligence*, vol. 134, pp. 57–83, 2002.

[14] B. Goertzel, R. Lian, I. Arel, H. de Garis, and S. Chenl, "A world survey of artificial brain projects, Part II: Biologically inspired cognitive architectures," *Neurocomputing*, vol. 74, pp. 30–49, 2010.

[15] A. D. de Groot, *Thought and Choice in Chess.* The Hague: Mouton Publishers, 1965. (Original work published in 1946 under the title *Het denken van den schakar*).

[16] W. G. Chase and H. A. Simon, "The mind's eye in chess," in *Visual information processing*, W. G. Chase, Ed. New York: Academic Press, 1973, pp. 215–281.

[17] ——, "Perception in chess," *Cognitive Psychology*, vol. 4, pp. 55–81, 1973.

[18] N. Charness, E. M. Reingold, M. Pomplun, and D. M. Stampe, "The perceptual aspect of skilled performance in chess: Evidence from eye movements," *Memory & Cognition*, vol. 29, no. 8, pp. 1146–1152, 2001.

[19] A. D. de Groot and F. Gobet, *Perception and memory in chess.* Assen, the Netherlands: Van Gorcum, 2002.

[20] E. M. Reingold and N. Charness, "Perception in chess: Evidence from eye movements," in *Cognitive processes in eye guidance*, G. Underwood, Ed. Oxford: Oxford University Press, 2005, pp. 325–354.

[21] A. Newell, *Unified Theories of Cognition.* Harvard University Press, 1990.

[22] W. Duch, R. Oentaryo, and Pasquier, "Cognitive architectures: where do we go from here?" in *Proceedings of the Artificial General Intelligence (AGI 2008)*, ser. Frontiers in Artificial Intelligence and Applications, P. Wang, B. Goertzel, and S. Franklin, Eds., vol. 171. IOS Press, 2008, pp. 122–136.

[23] A. Linhares, "An active symbols theory of chess intuition," *Minds and Machines*, vol. 15, pp. 131–181, 2005.

[24] R. A. Levinson, "A self-learning, pattern-oriented chess program," *ICCA Journal*, vol. 12, no. 4, pp. 207–215, 1989.

[25] R. A. Levinson and R. Snyder, "Adaptive pattern-oriented chess," in *Proceedings of the 8th International Workshop on Machine Learning*, L. Birnbaum and G. Collins, Eds. Morgan Kaufmann, 1991, pp. 85–89.

[26] L. Finkelstein and S. Markovitch, "Learning to play chess selectively by acquiring move patterns," *International Computer Chess Association Journal*, vol. 21, pp. 100–119, 1998.

[27] J. Mańdziuk, "Computational Intelligence in Mind Games," in *Challenges for Computational Intelligence*, ser. Studies in Computational Intelligence, W. Duch and J. Mańdziuk, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, vol. 63, pp. 407–442.

[28] W. Duch, "Intuition, insight, imagination and creativity," *IEEE Computational Intelligence Magazine*, vol. 2, no. 3, pp. 40–52, August 2007.

[29] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta algorithm," *Artificial Intelligence*, vol. 6, pp. 293–326, 1975.

[30] "Wikipedia - the free encyclopedia," http://en.wikipedia.org/wiki/Connect_Four.

[31] V. Allis, "A knowledge-based approach of Connect-Four," Department of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, The Netherlands, 1988.

[32] "Velena - a freeware Connect Four playing program," http://www.ce.unipr.it/ gbe/velena.html.

[33] M. Genesereth, N. Love, and B. Pell, "General Game Playing: Overview of the AAAI Competition," *AI Magazine*, vol. 26, no. 2, pp. 62–72, 2005.

[34] N. Love, M. Genesereth, and T. Hinrichs, "General game playing: Game description language specification," Stanford University, Stanford, CA, Tech. Rep. LG-2006-01, 2006, http://logic.stanford.edu/reports/LG-2006-01.pdf.

[35] G. Kuhlmann, K. Dresner, and P. Stone, "Automatic heuristic construction in a complete General Game Player," in *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence (AAAI-06).* Boston, MA: AAAI Press, 2006, pp. 1457–1462.

[36] S. Schiffel and M. Thielscher, "Automatic construction of a heuristic search function for general game playing," in *Proceedings of the Seventh IJCAI International*

*Workshop on Nonmontonic Reasoning, Action and Change (NRAC07)*, Hyderabad, India, 2007.