Can Deep Networks Learn to Play by the Rules? A Case Study on *Nine Men's Morris*

Federico Chesani, Andrea Galassi, Marco Lippi, and Paola Mello

***Abstract*—Deep networks have been successfully applied to a wide range of tasks in artificial intelligence, and game playing is certainly not an exception. In this paper, we present an experimental study to assess whether purely subsymbolic systems, such as deep networks, are capable of learning to play by the rules, without any *a priori* knowledge neither of the game, nor of its rules, but only by observing the matches played by another player. Similar problems arise in many other application domains, where the goal is to learn rules, policies, behaviors, or decisions, simply by the observation of the dynamics of a system. We present a case study conducted with residual networks on the popular board game of *Nine Men's Morris*, showing that this kind of subsymbolic architecture is capable of correctly discriminating legal from illegal decisions, just from the observation of past matches of a single player.**

***Index Terms*—Board games, deep learning, *Nine Men's Morris*, residual networks, rule learning.**

I. Introduction

GAME playing has been the source of inspiration and the testbed for many advancements and discoveries in artificial intelligence (AI). In the last decade, neural networks and especially deep learning techniques [1] have brought a revolution within AI and games. The application of deep reinforcement learning techniques to the development of agents playing Atari video-games has been one of the most successful deep learning stories [2]. The design of AlphaGo [3], [4], a computer system capable of beating several Go world champions[1] has become a milestone in the history of AI.

The development of AI techniques for game playing has long known the traditional dichotomy between symbolic and subsymbolic approaches [5]. Symbolic frameworks are based on an explicit and formal representation of the domain (often in a human understandable way) like, for example, in the form of logic facts and rules. Background knowledge can be encoded in the system, and reasoning techniques can exploit it to derive additional information and take decisions: a characterizing aspect is that the reasoning is performed at the level of symbols. Subsymbolic (also named connectionist) approaches consider instead reasoning as the result of the computation of a network of simple processing devices, without the need of explicitly representing knowledge through symbols. One major example of this class of models is clearly given by artificial neural networks. In game playing, both symbolic and subsymbolic approaches have historically been widely applied.

A great research effort has been put on learning strategies for winning games: the majority of the approaches takes as granted the game rules, often expressed in some formal description language. Trying to learn game playing without any kind of external hint of which are the actual game rules, instead, has rarely been addressed in the literature of AI in games. Arthur Samuel, in his seminal paper in 1959 on the application of machine learning techniques to the game of checkers [6], stated the following.

> "The rules of the activity must be definite and they should be known. Games satisfy this requirement. Unfortunately, many problems of economic importance do not. While in principle the determination of the rules can be a part of the learning process, this is a complication which might well be left until later."

Even the most recent applications to game playing, such as AlphaGo, or the system developed by Clark and Storkey [7], although heavily relying on the computational power of deep networks, still encode either in the network architecture or in the input features some information about the rules.

In this paper, we want to assess whether a deep learning approach can be exploited to learn to play by the rules a board game without any symbolic, background knowledge about the game rules. We consider as a case study *Nine Men's Morris*, a popular board game whose state space is not huge with respect to other games of the same kind. However, the complexity of the rules to be learned, and the large number of possible decisions to be taken by a player, make such a game a challenging benchmark.

Our main goal is not to be seen in terms of learning winning strategies, but rather in terms of learning legal moves. To this end, we have constructed a data set of game states and possible legal moves, by observing the behavior of an AI player based on a symbolic approach. Such a data set has been exploited to train a neural network system named Neural Nine Men's Morris (NNMM), based on residual networks [8]. For these reasons, we do not aim at comparing against any other system that exploits some (even partially) symbolic approach.

NNMM has been evaluated in terms of (percentage of) suggested legal moves, resulting in very good performance: in

Manuscript received May 22, 2017; revised October 31, 2017; accepted February 4, 2018. Date of publication February 8, 2018; date of current version December 13, 2018. *(Corresponding author: Marco Lippi.)*

F. Chesani, A. Galassi, and P. Mello are with the Department of Computer Science and Engineering, University of Bologna, Bologna 40126, Italy (e-mail: federico.chesani@unibo.it; a.galassi@unibo.it; paola.mello@unibo.it).

M. Lippi is with the Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Modena 41121, Italy (e-mail: marco.lippi@unimore.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TG.2018.2804039

[1] https://deepmind.com/research/alphago/

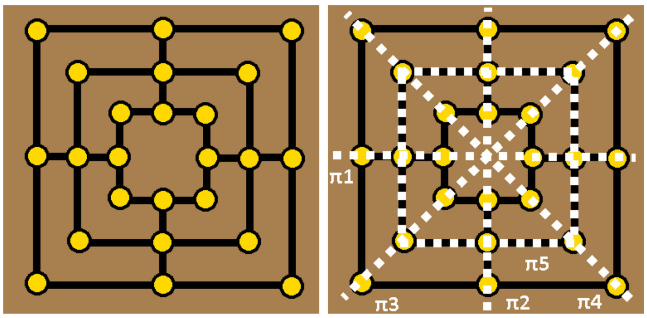

Fig. 1. *Nine Men's Morris* game board (left) and its symmetry axes (right).

almost the totality of the cases the network suggests, as a first choice, a legal move. Moreover, we have also evaluated, through a quantitative analysis, the "level" of legality expressed by our network over the first N suggested decisions, i.e., how much the networks are able to exhibit a compliant behavior with respect to the game rules. Again, results show that the presented approach is highly effective.

Natural extensions of this kind of analysis could be provided in other domains, such as decision making [9], anomaly detection [10], process mining [11]: when rules cannot be clearly identified or stated, a learning approach to compliance, based on observed behavior, could be an interesting alternative. Clearly, games represent the ideal scenario to first test our idea, as they provide frameworks where game rules are clearly defined and it is easy to check the legality of decisions.

This paper is structured as follows. In Section II, we briefly revise the game rules, whereas in Section III, we describe how we modeled the game, and which neural networks have been used in our experiments. In Section IV, we present the data set built for our purposes, while Section V will report the experimental results. Section VI will discuss related works, and finally Section VII will conclude the paper.

II. *Nine Men's Morris*

Nine Men's Morris, also known as *Mill Game*, *Merrils*, or *Cowboy Checkers*, is a perfect-information strategy board game for two players. It is a very ancient game, dating back around 1400 B.C. [12], but still very popular in many countries around the world. This game has long been analyzed from the point of view of AI and game theory, and its solution has been proven to be a draw [13]. Recently, theoretical results have been found for ultrastrong and extended solutions [14].

There exist several game variants that differ for either the game board or for the rules. We hereby briefly describe the most common setting, which has been used in our experiments. The game board, depicted in Fig. 1, consists of three concentric squares and four segments which connect the midpoints of the sides of the squares. The intersections of two or more lines create a grid of 24 points where checkers can be placed. Each player has nine checkers (also called stones or men). The game proceeds through the following three different phases that define the allowed moves.

1) Starting from an empty board, players alternately place a stone on an empty position.
2) When both players have placed all their stones, they must slide a checker along a line to a nearby empty position.
3) If a player remains with only three stones, then the constraint to move to an adjacent position is removed: checkers can be moved to any empty position in the board (the checker can "fly" or "jump") .

When a player succeeds in aligning three checkers along a line ("closing a mill"), he/she removes an opponent's checker from the board among those checkers not belonging to any mill.[2] The removed checker is said to be "eaten" or "captured."

The game ends when one of the following conditions occurs.

1) Player A removes seven stones of player B, thus leaving B with less than three stones (A wins).
2) Player A cannot make any legal move (A loses).
3) A configuration of the board is repeated (draw).[3]

Whereas the first two ending conditions can be detected by observing the game state, recognizing the third condition requires to keep track of the game history.

If we take into account only the game states during phases 2 and 3, each of the 24 cells of the board can be either occupied by a white checker, or by a black checker, or it can be empty. Thus, an upper bound for the number of possible states is 3^{24}, that is approximately 2.8×10^{11}. However, there are further constraints that limit the number of possible states: for example, if a player has closed a mill, the opponent cannot have all the 9 checkers on the board. If board symmetries are considered too, it can be shown that the game has 7 673 759 269 possible states in phase 2 and phase 3 [13]. The number of possible game configurations is thus not dramatically huge: as a comparison, consider that the chess game allows between 10^{43} and 10^{50} different configurations [15].

The number of moves that a player can do is quite large. In the most general case, the player faces three decisions: the checker to move, where to place it, and which adversary's stone to remove. As explained later in Section III-A, these decisions can lead to a quite large number of alternative moves.

Summing up, the game enjoys the following characteristics.

1) Symbolic approaches have been proven to successfully solve and play it. Therefore is a well-known case of study and we can rely on background knowledge.
2) The dimension and complexity of the state space is not huge: as a consequence, the process of training a subsymbolic approach does not require excessive resources in terms of time and hardware.
3) The choice of a move implies several decisions, and a legal move must satisfy constraints that affect both the single decisions and the move as a whole. As a consequence, the dimension of the space of legal moves is relatively small, when compared to the dimension of the space of possible moves, thus making the selection of legal moves a difficult and interesting problem (see Section V-A for details).

[2]In the case that all the opponent's checkers are aligned in at least one mill, one aligned checker is allowed to be removed. If two mills are closed at the same time, still only one checker can be removed.

[3]Repetitions can happen only during phases 2 and 3.

For these reasons, the *Nine Men's Morris* game represents a challenging case study for assessing whether a subsymbolic system is capable of learning to play by the rules.

III. System Architecture

In this section, we describe how we represent the game, and which architectures have been selected for the neural networks trained to play the game.

A. Game Modeling

The state of the game consists of four pieces of information: the board configuration and, for each player, the number of checkers he/she still has in his hand, the number of checkers that he/she has on the board and the phase of the game in which he/she is. The last two pieces of information can be deducted from the first two. The number of checkers in the hands of each player can obviously be represented with two numbers, each of which can assume values from 0 to 9. For the game board, several different representations could be chosen, by exploiting a 1-D array, a 2-D matrix, or a 3-D cube. For the sake of simplicity, we just used the most straightforward implementation, that is a plain enumeration of the board cells coded into a 1-D array (the ith element of the array representing the ith cell in the enumeration). Each cell can be occupied either by a white checker, or by a black checker, or it can empty. We can easily represent such configurations with three different values.

A move consists of following three distinct pieces of information.

TO: In every game phase, a checker is always placed somewhere. This will be either a newly introduced one during phase 1, or a checker that is already present in the board during phases 2 and 3. We name this information "TO". It can assume 24 possible values (the cells of the board).

REMOVE: If placing the checker causes the closing of a mill, another information that must be encoded in the move is the adversarial checker to be removed. We name this information "REMOVE" and it can assume 25 possible values: the 24 cells of the board plus the none value, in the case that the move implies no removal. During a single match, the maximum number of moves which imply a removal is 13, that is 7 by the winner and 6 by the loser.

FROM: During phases 2 and 3, the checker is moved from one position to another, so we have to encode also the initial position. We name this information "FROM." Therefore, it can assume 25 possible values: the 24 cells of the board plus the none value, in the event that the game is in phase 1.

Without knowing the constraints on these three information, i.e., without any knowledge on the game's rules, the number of possible combination, and therefore of possible moves, is $24 \times 25 \times 25 = 15\,000$. This number is quite big compared to other boardgames: for example, in the game of *Go*, a single decision has to be taken ($19 \times 19 = 361$ positions on the board), whereas in the game of chess, two decisions have to be taken[4] (initial and final positions of the moving piece – $64 \times 64 = 4096$ possible couples of positions).

TABLE I
Input Features for the Three Networks in NNMM

Feature	Bits	Description
Player's board	24	Position of player's checker on the board
Adversary's board	24	Position of adversary's checker on the board
Empty board	24	Empty position
Player's hand	9	The number of checkers in player's hand
Adversary's hand	9	The number of checkers in adversary's hand
First network choice[a]	25	Decision taken by the first network of the system
Second network choice[b]	25	Decision taken by the second network of the system

[a] Present only in the second and third networks.
[b] Present only in the third network.

B. Neural Nine Men's Morris

Our system thus consists of three different neural networks, each predicting one part of the move (TO, REMOVE, and FROM). We model the problem as a collection of three supervised learning tasks, where the target of each task is the partial decision to be taken by the player at a given board configuration. The output size of the TO network is 24 neurons, which represent the possible positions on the board. The REMOVE and FROM networks, instead, will also have an additional special output neuron (thus 25 output neurons) encoding the case in which no checker has to be moved or eaten, respectively. For this reason, we also let the TO network have 25 output neurons (with one extra neuron that is never used) so that the three networks have identical architectures. Any network will then provide a single position value among 25 possible ones. Note that exploiting a single neural network would have produced a number of output classes (all the possible moves) equal to 15 000 (see Section III-A), which would have made the training almost unfeasible.

Features have been represented with a binary encoding, thus exploiting different bits to represent different possible values of the game state variables. The feature vector thus composed is described in Table I. It simply consists of the board configuration and the number of checkers in hand for each player. In fact, the three networks operate in cascade, providing the positions chosen by the former ones as input to the latter ones. Since the output of each network is independent from the decision it has to make, its input/output structure of each network only depends on its position in the architecture. Since the decisions to be taken by the networks are clearly strongly correlated, exploiting independent networks rather than a cascade model, thus taking the three decisions (TO, FROM, and REMOVE) independently one from the other, would have certainly lead to worse performance. Fig. 2 illustrates the overall architecture of the system, which we name Neural *Nine Men's Morris*, when the cascade of the three

[4]The *castling* move can be indicated with the coordinates of the *king* involved. We are not taking into account the decision involved into *promotion*.

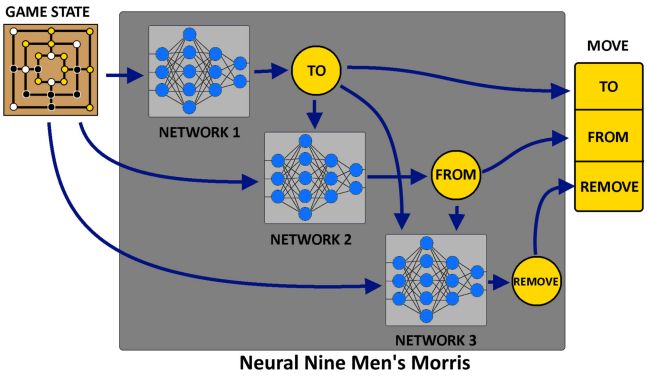

Fig. 2. Illustration of our NNMM system, exploiting the TO–FROM–REMOVE configuration. Each of the three networks takes as input both the board and the decision taken from the previous network(s) in the cascade.

networks is in the order TO–FROM–REMOVE (TFR). In Section V, we investigate different arrangements of the networks, and evaluate through experiments their performance. We hereby underline the fact that, at this point, no choice has been made on the internal neural network used for each step.

Finally, we remark that, for any game state, there are legality constraints both on the choices of each of the three networks, but also on the whole move. That is, the legality of the parts does not guarantee the legality of the full move.

Although the chosen network architecture may look specifically tailored for *Nine Men's Morris*, its cascade structure (where the decision on the nth part of the move depends on the first $n-1$) easily allows us to generalize it to other board games. More precisely, it could be immediately applied to any board game where one has to choose a piece to move (FROM), a position where to place it (TO), and possibly also an opponent's piece to remove (REMOVE).

C. Residual Networks

A preliminary experimental study [16] was conducted to choose the architecture for this case of study and to tune its parameters. As a result of this study, we decided to adopt residual networks, which had achieved the best performance. We hereby describe the final system that will be adopted in the experimental evaluation.

Residual networks [8] are a recently introduced architecture for deep networks that has been specifically designed to train networks with many layers. Such networks have been first applied to computer vision tasks, where they achieved state-of-the-art performance in many data sets: one of the most impressive results was obtained in the 2015 ImageNet competition, won by Microsoft Research team with a residual network with 152 layers. Nevertheless, the proposed framework is generic and thus such networks can be successfully applied also to other domains. The main idea behind residual networks is that of replacing a generic (possibly complex) objective function $\mathcal{H}(x)$ to be optimized with its residual, that is $\mathcal{F}(x) = \mathcal{H}(x) - x$. The original function is thus obtained from the residual by simply adding back the input, that is $\mathcal{H}(x) = \mathcal{F}(x) + x$. Further improvements, such as the use of preactivation of weight layers

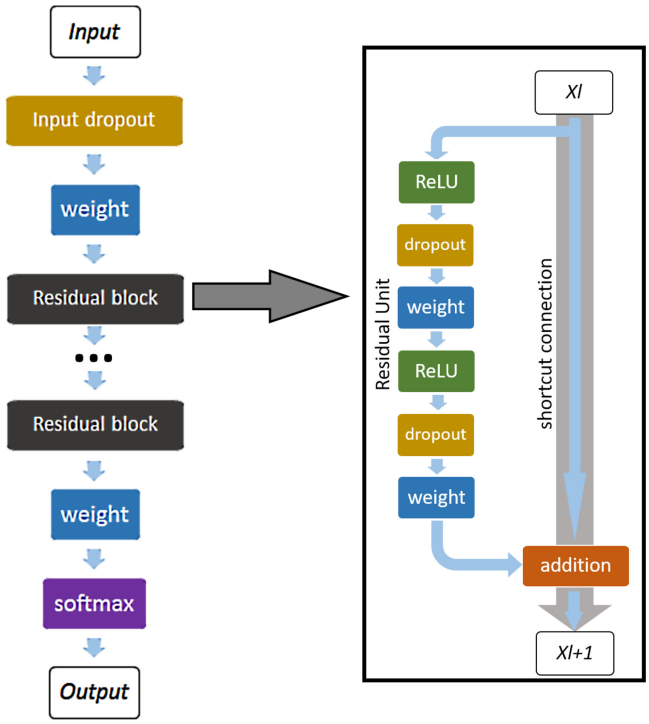

Fig. 3. Illustration of the residual network used in our experiments.

and of dropout inside the residual unit [17], [18], allow us to achieve even better results than the original model on challenging computer vision tasks. The optimization problem results to be easier to solve with respect to traditional (nonresidual) deep networks, in particular when the number of layers increases, which typically lets these networks achieve better performance.

All the three networks in NNMM are residual networks, and each of them presents several blocks as that depicted in Fig. 3. Each block follows a fully connected layer and is made by two subblocks, each consisting of a rectifier linear unit [19] preactivation layer, a dropout layer, and a fully connected layer. Each network has an initial layer of 200 neurons, followed by a set of residual units stacked one onto the other. In each residual unit, the first layer contains 300 neurons, whereas the second layer contains 200. TO and FROM networks have been made by 10 residual units whereas REMOVE networks have been made by 30 units. Each network terminates with an output layer made of 25 neurons, to which the softmax function is applied, so that we obtain a probability distribution over the possible positions in the board. The total depth is thus 22 layers for the TO and FROM networks, and 62 for the REMOVE one.

IV. Data Sets

To evaluate the capability of deep networks to learn legal moves in *Nine Men's Morris*, we built a data set of game matches to train our system. To this aim, we exploited a collection of AI players, all based on symbolic approaches. Such systems were developed by students for a competition within the "Foundations of Artificial Intelligence" course at the University of Bologna. The winner of the competition was chosen as the "trainer" of our deep networks. Thus, for each game board, the move of the

trainer is used as the supervision target. This implies that the target moves in the data set are not optimal moves according to some criterion, but rather good moves according to the trainer and, most importantly, they are legal moves. The decision to use this data set, rather than sampling from a database in which the optimal moves are indicated, such as the one presented in [14], came from the intention to verify if the subsymbolic system could learn to play by the rules, by simply observing matches played by another player.

For each state, the symmetries of the problem shown in Fig. 1 have been exploited to increase the number of examples. This "Matches Data Set" is composed by 1 628 673 state-move pairs. Each state in the data set is unique, therefore for any state only one move is present. To generate the data set, the symbolic trainer has played 7244 games against itself and the other AI players. We let some of the games start from initial random configurations, rather than from the conventional one (i.e., empty board and nine checkers in each player's hand). Some of these random states, and therefore, some of the states obtained during that match, are not reachable[5] in a proper game that starts from the conventional initial state. This feature of the data set makes it more general for the task of learning game rules, as it limits the possible problem of overfitting on a subset of the whole state space. At the same time, the generation of the initial board state takes into account some characteristics of the game to ensure that the rules of the game, and therefore that legality definition, still hold. Some of the constraints imposed on the randomly generated configuration are: the next player to move is the white one, each player must have at most 9 and at least 3 checkers (among the hand and the board), and both players must hold the same number of stones in their hands. With these constraints, the state of the game will always belong to one of the three phases described in Section II. Moreover, the symbolic players used to generate the data set have knowledge of the game rules, therefore, they are capable of making only legal decisions.

As an additional test set, a second data set was built. It contains random game states, that have been sampled as reachable board configurations, without any overlap with the ones present in the matches data set. In this way, 2 085 613 states have been gathered. We name this second data set "States Data Set." Such data set has been used only as a further test set for our system, with the goal of evaluating the generalization capabilities of the system on generic states that have not been reached by a match played by the NNMM's symbolic teacher.

Both data sets are available online.[6]

V. Experiments

In this section, we present our experimental results. After an analysis of the data sets, aimed to investigate the space of legal moves, we present three different system configurations. Then, we describe the tasks on which such systems have been tested, and we discuss the achieved results. The source code for the replication of these experiments is available online, together with the trained neural network models.[7]

A. Data Sets Analysis

In order to prove that the space of legal moves is very small in comparison with the space of possible moves, the two data sets have been analyzed, in particular measuring 1) the number of legal moves for each state; and 2) the number of states in which a move is legal.

For both data sets, the highest number of legal moves allowed by a state is 58, whereas the lowest is 1. The mean number of legal moves per state is 18 for the Matches Data Set and 22 for the States Data Set. Since our representation allows 15 000 different moves ($24 \times 25 \times 25$), the space of legal moves is at most the 0.39% of the space of possible moves, and on average it is less than 0.15%.

Moreover, for each of the 15 000 moves, we counted the number of states in each data set where such move is considered legal. On average, a move is legal in the 0.12% of the total number of states in the Matches data set, and in the 0.15% of the States data set. The number of moves which, are always illegal in both data sets is 1920, whereas only 32 moves are legal in more than 10% of both data sets. There is no move which is legal in more than 13% of either the Matches or the States data sets. These 32 moves that are more frequently legal are all characterized by a REMOVE value of 0 (thus, they do not remove any checker) and FROM values that represent those positions of the board connected with most other positions. No special pattern is observed in the TO values.

This analysis suggests that the problem of learning to play by the rules in *Nine Men's Morris*, without any background knowledge of the problem, is particularly challenging.

B. Setup

The Matches Data Set has been used as the development data set. It was partitioned into a training set, a validation set to monitor learning and to perform parameter tuning and early stopping, and a test set to evaluate the model at the end of the training phase. The test set consisted of 10% of the whole data set (about 163 000 pairs), while the validation set consisted in 5% of the data set (about 81 000 pairs), leaving about 1.4 million pairs for the training set. Each network was trained independently, using the game state and, eventually, a partial part of the move in the data set as inputs.[8] The desired partial move played by the symbolic trainer was used as target.

The loss function chosen as the objective of training was the negative log-likelihood of the target class, with an L1 regularization with weight 10^{-3}. Adam [20] was used as optimizer, with parameters $b_1 = 0.99$ and $b_2 = 0.999$. The initial learning rate $\alpha_0 = 2 \times 10^{-3}$, was progressively annealed through epochs with decay proportional to training epoch t, resulting in a learning rate $\alpha = \alpha_0(1 + k \times t)$, with $k = 0.01$ for TO and FROM networks, and $k = 0.02$ for REMOVE networks. Parameters were initialized with He initialization [21], specifically designed for ReLU activation. We employed minibatch

[5]For a reachable state we mean a state that can be reached, from the initial board configuration, with a sequence of legal moves only.

[6]http://ai.unibo.it/DatasetNineMenMorris

[7]http://ai.unibo.it/NNMM

[8]Because the second and the third networks in each configuration need a partial move as input.

optimization, with batch size equal to 20 000. For TO and FROM networks, dropout was applied to each layer, with $p = 0.1$, while it was not used for the REMOVE network. For early stopped, we used a patience value of 50 epochs. The whole system was implemented with the Lasagne [22] and Theano [23] frameworks.

Three different NNMM system configurations were compared, designed with different orders in deciding move parts. We name them after the order in which the decisions are made. The configurations are: TO–FROM–REMOVE (TFR), FROM–TO–REMOVE (FTR), and REMOVE–FROM–TO (RFT). The first configuration (TFR) is the one which appears to be more logical for a human player: in each phase of the game the system has to place a checker somewhere, so that it is the first decision that has to be taken, then it decides where that checker has to come from (to know if a mill has been closed) and if an opponent's checker should be removed. The second one (FTR) is an alternative to the first one, where the checker to be moved is considered the most important decision. The last one (RFT) can be considered as an extreme case study, as it seems illogical for a player to decide which opponent's checker has to be removed before even deciding which of his/her own checker should be moved and where.

C. Tasks

The system was tested to evaluate the following three different aspects.

1) Accuracy, that is the capability of reproducing the same complete move of its teacher.
2) Legality, that is the capability to suggest, as the best complete move, one that respects all the game rules.
3) Reliability, that is its capability to give legal decisions a higher probability than nonlegal decisions, thus, considering not only the top-ranked decision, but also the subsequent ordering.

The accuracy test simply measures how many times the move suggested by the subsymbolic system is equal to that provided by the symbolic trainer. Such a test does not give any hint about the quality of the system, because it does not evaluate whether the outputs of the subsymbolic system are better or worse than the choices of the symbolic system. A high accuracy, in this sense, is not necessary for our purpose of learning legal moves. Yet, it is a useful metric to assess that the training phase reached a reasonable network configuration.

The legality test is the most important one for our goals, since it measures whether the system has been able to learn to play by the rules, by counting how many times the move suggested by the subsymbolic system violates any of the game rules. It has been performed on both data sets.

Finally, the reliability test moves the legality test a step further. It is designed with the goal of assessing whether the system is able to correctly discriminate between legal and illegal decisions, and thus if it has learned a sort of *correct behavior*. To this aim, we separately consider the ranking of the output, partial or complete, moves for each network, according to their probability. For a partial move, we hereby mean the outcome either of the first network, or of the first and the second networks together. If the system has correctly learned the concept of game rules, then all the legal decisions should ideally appear before the illegal ones in such ranking. It has been performed both on the good moves and the testing data sets. To give a quantitative measure of this property, we build a sort of recall-precision curve as follows. For each board configuration in the test set, we consider the output of each network in the pipeline, ranking decisions by their probabilities, and we compute the percentage of legal moves (precision) as the number of retrieved legal moves (recall) increases. Finally, we average over all the states in the test set for each of the three networks separately. Clearly, for the first two networks in the pipeline the considered decisions will be partial moves, whereas for the last network these will be complete moves: each network in fact has to know the prediction of the previous network(s) in the pipeline. In the ideal case, when all legal decisions precede the illegal decisions, all the precision values are equal to 100%. Otherwise, if some illegal move is ranked higher than some legal move, the precision will decrease accordingly. Table II shows an example of how such an evaluation metric is computed. For the last network, the recall is thus the number of legal complete moves retrieved upon the number of existing legal complete move, whereas its precision is the number of legal retrieved complete moves upon the total number of retrieved complete moves. The same principle is applied to the second and first networks, considering only the legality of the partial moves. While the definition of legal complete move is straightforward, this is not the case for legal partial moves, which could be subject to different interpretations. For the purpose of our tests, we have listed all the possible legal complete moves for each state and decomposed them into partial ones, so as to define all the legal partial moves. In the case that one network takes an illegal decision, this makes every possible decision of the following networks illegal too. Therefore, those cases where there is no legal decision available to a network are discarded during the reliability evaluation of that network. Finally, note that for the FROM and REMOVE networks, we discarded the configurations for which no checker is moved/removed, respectively.[9]

TABLE II
EXAMPLE OF COMPUTATION OF THE RECALL-PRECISION CURVE

N	Decision ranking	Recall	Precision
1	1	0.20 (1/5)	1.00 (1/1)
2	1,7	0.40 (2/5)	1.00 (2/2)
3	1,7,9	0.60 (3/5)	1.00 (3/3)
4	1,7,9,**0**,**4**,5	0.80 (4/5)	0.67 (4/6)
5	1,7,9,**0**,**4**,5,12	1.00 (5/5)	0.71 (5/7)

For a given board configuration, suppose there exist only five legal decisions, namely outputs 1, 7, 9, 5, and 12. For each value of N from 1 to 5, we thus consider the decision ranking that is necessary to retrieve N legal decisions: in this way, for each value of N we compute recall and precision. Bold numbers represent illegal decisions.

D. Results and Discussion

Table III reports the accuracy on the Matches Data Set of each trained network in each configuration (TFR–FTR–RFT),

[9]Trivially, those cases do not add information with respect to the legality test, since there is only one legal move.

TABLE III
ACCURACY OF THE NETWORKS ON TRAINING, VALIDATION, AND TEST SETS FOR EACH OF THE THREE CONSIDERED CONFIGURATIONS

Configuration	Move part	Training	Validation	Test
TFR	TO	53.07%	51.77%	51.73%
	FROM	90.07%	89.43%	88.97%
	REMOVE	88.01%	86.54%	85.66%
FTR	TO	75.73%	73.72%	74.17%
	FROM	65.00%	63.90%	63.33%
	REMOVE	88.54%	86.73%	85.88%
RFT	TO	78.20%	76.40%	76.75%
	FROM	68.92%	68.38%	67.94%
	REMOVE	81.27%	80.25%	79.45%

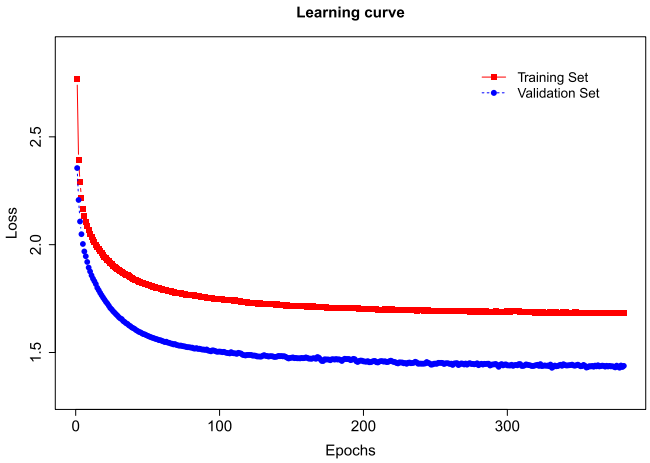

Fig. 4. Learning curve of the TO network in the TFR configuration.

TABLE IV
ACCURACY TEST RESULT

Configuration	All phases	Phase 1	Phase 2	Phase 3
TFR	37.20%	47.91%	36.27%	29.19%
FTR	38.13%	49.28%	37.75%	27.58%
RFT	37.52%	48.70%	37.31%	26.33%

thus considering the three partial moves separately. Results show that similar values of accuracy have been achieved on training, validation, and test sets, thus suggesting that the networks have maintained a good generalization, and that the phenomenon of overfitting does not have a strong impact here. Fig. 4 shows an example of learning curve, for the TO network within the TFR configuration.

In Table IV, we instead report the accuracy of the three configurations over the complete moves, highlighting the differences with respect to each phase of the game. It is interesting to note that the values of accuracy are very similar for the three network configurations. The game phase where the system achieves the best accuracy is phase 1.

The most impressive results have been obtained for the legality test. As depicted in Table V, the system demonstrates to have learnt to respect all the rules of the game in more than 99% of the cases, independently from the configuration of the networks and from the data set. To better investigate which are the rules that are more frequently broken by our system, the legality of partial moves has been tested too. For the TFR and FTR configurations, the mistakes mostly regard the constraints on the REMOVE part. The RFT configuration obtains a slightly higher legality percentage for the REMOVE network, which suggests that in that case the constraints that are more often broken regard the whole move.

TABLE V
LEGALITY TEST RESULTS

Configuration	Data Set	Whole move	TO	FROM	REMOVE
TFR	Matches	99.53%	99.94%	99.96%	99.62%
	States	99.53%	99.93%	99.96%	99.71%
FTR	Matches	99.53%	99.98%	100.00%	99.63%
	States	99.56%	99.98%	100.00%	99.73%
RFT	Matches	99.25%	99.91%	100.00%	99.86%
	States	99.19%	99.89%	100.00%	99.85%

The reliability test has confirmed that the system is able to differentiate between legal and illegal decision, holding very high average precision values for all the values of recall. As illustrated in Fig. 5, the results on the two data sets are similar, which confirms that the system has learnt to generalize well on previously unseen data. The FTR configuration is the best setting: all the networks maintain a precision of about 99% for all recall percentages (note that the y-axis in the plots in Fig. 5 starts at 0.9). The TFR configuration performs well for the second and third network, while the first one slightly loses accuracy as the recall increases. The RFT configuration not surprisingly results to be the worst (as it is difficult to first decide a checker to remove, before deciding which checker to move, and where), but still maintaining a 92% precision at 100% of recall on the States data set.

VI. RELATED WORK

In [13], the game of *Nine Men's Morris* is solved exploiting brute-force approaches, demonstrating that its solution is a draw. The study of the game has been pushed forward in [14], where the "extended strong solution" and the "ultrastrong solution" are found too. The former is the computation of the game-theoretic values of all the game states that could be reached in a match if the players have less than 9 checkers to place. The latter is the definition of a strategy that, against a fallible opponent, increases the chances to achieve a result which is better than the theoretic one. Even though these studies may have paved the way to our work, their purpose was very different from ours: their objective was to find optimal solutions to the game, whereas our focus has been on creating a system that is able to learn the game rules.

Game playing is a whole research field in AI, and a review of the many approaches available in the literature is out of the scope of this paper. The interested reader can refer to [24] for a panorama of the main AI techniques. In such a context, our paper could be classified as an instance of behavioral learning of nonplaying-characters. Usually, reinforcement learning

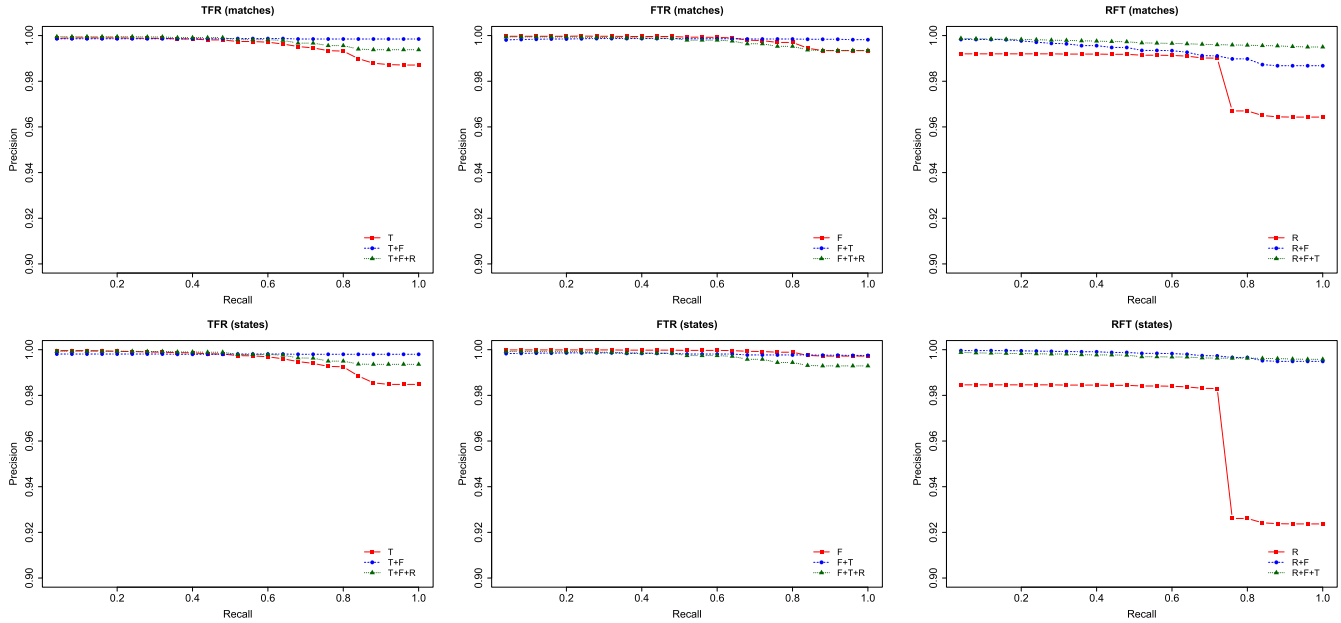

Fig. 5. Reliability of the different networks in each configuration (left to right: TFR, FTR, RFT), reporting precision of retrieved legal moves as a function of the recall of retrieved legal moves. Top/bottom charts refer to the Matches/States Data Sets, respectively. Note that y-axis starts at 0.9.

techniques or evolutionary computation are used to this end. This thematic is addressed in [25].

A main characteristic of our approach is that only subsymbolic techniques are exploited. Systems which used combinations of subsymbolic knowledge (acquired through learning) and symbolic knowledge (encoded into the system itself) have been proven extremely successful in playing many different games. *Backgammon*, *Chess*, *Checkers*, and *Go* are only few notable examples of the games which have been addressed with combination of artificial neural networks and symbolic techniques, resulting in artificial players capable of playing a specific game achieving very good results. Among the many, we can cite the following works: [3], [26]–[29]. Yet, such approaches do not use neural networks directly to decide the move to be played, but rather to rank a list of moves that are typically generated by some symbolic approach (and, thus, which are certainly legal).

Recent works that instead employ deep networks for game playing, such as AlphaGo Fan/Lee [3], AlphaGo Zero [4], or the system developed by Clark and Storkey [7], consider only legal moves, by forcing to zero all the illegal options in the last network weight layer before softmax or by excluding illegal moves during a symbolic exploration phase. Among the many features preprocessed and given as input to AlphaGo Fan/Lee's neural networks, there are the characteristics of the status of each intersection of the Go board: liberties, legality, stone color, number of opponent, and own stones that would be captured, and whether the move is a part of a ladder escape or capture. AlphaGo Zero, instead, uses only the last board configurations and players' choices as features. For the former, the game symmetries are handled by giving as input to the networks a minibatch of all the symmetric states, so that they can be computed in parallel, whereas the latter is trained with a data set, which is augmented considering the symmetries, in a similar fashion to our work.

Another famous application of deep networks to game playing was given in [2], where a reinforcement learning approach was undertaken to train a system to play Atari videogames, using only information of raw pixels in input, thus having no *a priori* knowledge of the game. Such a work, yet, does not have to deal with the concept of a legal move.

An orthogonal approach with respect to our work has been investigated in the General Game Playing (GGP) competition [30]. Instead of creating a player specialized in a single game, the GGP problem consists of creating a system able to play any kind of game, given its rules. Within this context, rules are thus explicit knowledge that systems receive as input, represented in an appropriate language called Game Description Language [30]. The competition is held once a year, since 2005, and it provides a benchmark for general approaches to AI and games. Even if many participants to this challenge rely only on symbolic techniques, some notable systems which combine symbolic and subsymbolic techniques have been among the participants. For example, [31], where neuroevolution is exploited to learn a game strategy. Although GGP considers the game rules as a known input, and asks participants to learn game strategies, it can be considered a very interesting point to further investigate the generality of our approach. For example, in the context of GGP, in [32], a symbolic algorithm capable of learning the rules of a simplified boardgame from a data set of matches is presented. The data set used for the training phase is made by game states and a nonexhaustive list of legal moves. Despite this similarity, *Nine Men's Morris* does not meet the definition of "simplified boardgame," thus making it unfeasible to apply such a solution to our case study.

In this paper, we used as training set a collection of (only) legal moves: in other words, we provided to the neural network only "positive" examples. Other approaches instead require both positive and negative examples, that is also a collection of illegal

decisions. For example, in [33], variant chess rules are learned as extended logic programming theories from both positive and negative examples, background knowledge and by applying theory revision. Although being a very interesting approach, the need for both types of examples might be not feasible in a number of domains, where only observations of correct system dynamics are available. This is a common situation in fields like, for example, process mining, anomaly detection, human behavior simulation, and profiling.

Finally, it is relevant to underline that many other models of artificial neural networks exist, and thus different system architectures could be employed, possibly leading to better results. Stochastic depth networks [34] randomly drop layers during training, allowing us to greatly increase the depth of the networks. Since we modeled the move as a sequence of decisions, recurrent neural networks [35] could also be a useful alternative architecture, as they are usually applied for the classification of data sequences (e.g., in speech recognition tasks). Dense networks [36] exploit the same intuition of residual networks, creating shortcuts between layers at different depth, and concatenating the outputs instead of summing them. A deeper investigation of different neural network architectures and training techniques applied to our context could be the subject for future works.

VII. Conclusion

Deep learning methods are widely employed in game playing. The aim of this paper was to analyze whether such subsymbolic systems are capable of learning to play a game by the rules just by observing a single player matches, without the need to explicitly model or encode any background knowledge of the game within the architecture of the network, nor providing any information about legality during the supervised training. Our analysis exploits residual networks, a particular type of deep networks specifically designed to learn models with many layers. Experimental results show that such systems are capable not only to suggest legal decision as best choices, but also of preferring legal decisions to illegal ones. Clearly, the chosen architecture and move encoding strongly affect the percentage of both possible and legal moves. Yet, it is worth remarking that, in the general case, it is not always possible to define an encoding that discards *a priori* illegal moves: in many board games, in fact, such as *Nine Men's Morris* but also chess or checkers, the legality of the move depends on the game status. In addition, looking forward beyond games, there are many applicative scenarios in the context of behavior compliance where it is just not possible to define in advance the concept of legality, and thus it certainly cannot be encoded within the move modeling. The proposed architecture is general enough to be employed with any board game where checkers are moved from a position to another, and opponent checkers are removed. Checkers and chess are other examples of such games. Thus, the impact of this kind of result goes beyond the application to game playing, opening the doors to the application of deep networks in many contexts where behavioral rules and decision policies could be learned directly from data, such as anomaly detection tasks.

Future works in this field may concern a quantitative evaluation of the impact of the data set in the learning: the use of a training set with better choices (i.e., a data set with the optimal move computed according to [13] and [14]) or with a different number of training data could enhance or decrease the system performance. The necessity of an high number of high quality data or a lack of it could heavily influence future application of these techniques. The next step for this work, before trying to apply it to more complex real-life or industrial contexts, could be to investigate its performances with other board games with complex rules, such as, for example, chess.

Acknowledgment

The authors would like to thank the authors of DeepMill and all the students who have participated to the Nine Men's Morris Challenge as part of the course of "Foundations of Artificial Intelligence" at the University of Bologna, providing the software used for the creation of the database. The authors would also like to thank the anonymous reviewers for their useful suggestions, which have contributed to improve the quality of this paper.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[3] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[4] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[5] J. Dinsmore, *The Symbolic and Connectionist Paradigms: Closing the Gap*. Mahwah, NJ, USA: Lawrence Erlbaum, 2014.
[6] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM J. Res. Develop.*, vol. 3, no. 3, pp. 210–229, Jul. 1959. [Online]. Available: http://dx.doi.org/10.1147/rd.33.0210
[7] C. Clark and A. J. Storkey, "Training deep convolutional neural networks to play go," in *Proc. 32nd Int. Conf. Mach. Learn., Lille, France,* 2015, pp. 1766–1774.
[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[9] C.-L. Hwang and K. Yoon, *Multiple Attribute Decision Making: Methods and Applications a State-of-the-Art Survey*, vol. 186. New York, NY, USA: Springer-Verlag, 2012.
[10] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Survey*, vol. 41, no. 3, 2009, Art. no. 15. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882
[11] W. van der Aalst *et al.*, *Process Mining Manifesto*. Berlin, Germany: Springer, 2012, pp. 169–194. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28108-2_19
[12] R. C. Bell, *Board and Table Games from Many Civilizations*, vol. 1. North Chelmsford, MA, USA: Courier Corp., 1979.
[13] R. Gasser, "Solving nine men's morris," *Comput. Intell.*, vol. 12, no. 1, pp. 24–41, 1996.
[14] G. E. Gévay and G. Danner, "Calculating ultrastrong and extended solutions for Nine Men's Morris, Morabaraba, and Lasker Morris," *IEEE Trans. Comput. Intell. AI Games*, vol. 8, no. 3, pp. 256–267, Sep. 2016.
[15] L. V. Allis *et al.*, *Searching for Solutions in Games and Artificial Intelligence*. Wageningen, The Netherlands: Ponsen & Looijen, 1994.
[16] A. Galassi, "Symbolic versus sub-symbolic approaches: A case study on training deep networks to play nine men's morris game," Master's thesis, Univ. Bologna, Bologna, Italy, Mar. 2017. [Online]. Available: http://amslaurea.unibo.it/12859/
[17] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.

[18] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting." *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[19] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, Fort Lauderdale, FL, USA, Apr. 2011, pp. 315–323. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a.html
[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, preprint arXiv:1412.6980, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980
[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.
[22] S. Dieleman *et al.*, "Lasagne: First release," Aug. 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.27878
[23] J. Bergstra *et al.*, "Theano: Deep learning on GPUs with Python," in *Proc NIPS*, BigLearning Workshop, Granada, Spain, vol. 3, 2011.
[24] G. N. Yannakakis and J. Togelius, "A panorama of artificial and computational intelligence in games," *IEEE Trans. Comput. Intell. AI Games*, vol. 7, no. 4, pp. 317–335, Dec. 2015.
[25] H. Muñoz-Avila, C. Bauckhage, M. Bida, C. B. Congdon, and G. Kendall, "Learning and game AI," in *Dagstuhl Follow-Ups*, vol. 6. Wadern, Germany: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.
[26] G. Tesauro, "Temporal difference learning and td-gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, Mar. 1995. [Online]. Available: http://doi.acm.org/10.1145/203330.203343
[27] M. Lai, "Giraffe: Using deep reinforcement learning to play chess," M.Sc. thesis, Imperial College London, preprint arXiv: 1509.01549, 2015. [Online]. Available: http://arxiv.org/abs/1509.01549
[28] O. E. David, N. S. Netanyahu, and L. Wolf, *DeepChess: End-to-End Deep Neural Network for Automatic Learning in Chess*. Cham, Switzerland: Springer, 2016, pp. 88–96. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-44781-0_11
[29] K. Chellapilla and D. B. Fogel, "Evolving neural networks to play checkers without relying on expert knowledge," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1382–1391, Nov. 1999.
[30] M. Genesereth, N. Love, and B. Pell, "General game playing: Overview of the aaai competition," AI Mag., vol. 26, no. 2, 2005, Art. no. 62.
[31] J. Reisinger, E. Bahceci, I. Karpov, and R. Miikkulainen, "Coevolving strategies for general game playing," in *Proc. 2007 IEEE Symp. Comput. Intell. Games*, Apr. 2007, pp. 320–327.
[32] Y. Björnsson, "Learning rules of simplified boardgames by observing," in *Proc. 20th Eur. Conf. Artif. Intell.*, 2012, pp. 175–180.
[33] S. Muggleton, A. Paes, V. Santos Costa, and G. Zaverucha, *Chess Revision: Acquiring the Rules of Chess Variants Through FOL Theory Revision From Examples*. Berlin, Germany: Springer, 2010, pp. 123–130. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13840-9_12
[34] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 646–661.
[35] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.
[36] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, vol. 1, pp. 2261–269.

Federico Chesani received the Ph.D. degree in informatics, telecommunications, and electronics engineering from the University of Bologna, Bologna, Italy, in 2007.

He is a Research Assistant at the Department of Computer Science and Engineering, University of Bologna, Bologna, Italy. He is an author of more than 70 papers published in international conferences and journals. He has been involved in several national and international projects. His research interests include rule-based systems, complex event processing, business process management, and theoretical and practical aspects of logic programming with a focus on abductive reasoning.

Andrea Galassi received the M.S. degree in computer engineering from the University of Bologna, Italy, in 2017. He is currently working toward the Ph.D. degree in computer science and engineering at the Department of Computer Science and Engineering, University of Bologna, Bologna, Italy.

His research interests include artificial intelligence and machine learning, focusing on deep networks and their applications to games, CSPs, and argumentation mining.

Marco Lippi received the Ph.D. degree in computer and automation engineering from the University of Florence, Florence, Italy, in 2010.

He is currently an Assistant Professor at the Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Modena, Italy. He previously held positions with the Universities of Florence, Siena, and Bologna, and was a Visiting Scholar at Université Pierre et Marie Curie, Paris. His research interests include machine learning and artificial intelligence, with applications in bioinformatics, time-series analysis, computer vision, game playing, and argumentation mining.

Dr. Lippi was the recipient of the "E. Caianiello" prize for the best Italian Ph.D. dissertation in the field of neural networks in 2012.

Paola Mello received the Ph.D. degree in informatics, telecommunications, and electronics engineering from the University of Bologna, Bologna, Italy, in 1988.

She is currently a Full Professor at the Department of Computer Science and Engineering, University of Bologna, Bologna, Italy. She is currently active in formal specification and on the automatic verification of (interaction) protocols, workflow patterns, medical guidelines, and Web Services. She is the author of several scientific papers published in important national end international conferences and journals, and has been involved in a number of national and international (EU) projects. Her research interests include artificial intelligence, knowledge representation and reasoning, logic programming, multiagent systems, and applications of expert systems with particular emphasis on medical domain.

Prof. Mello was the President of the Italian Association for Artificial Intelligence and the Head of the Department of Computer Science and Engineering, University of Bologna. She is a Fellow of the European Association for Artificial Intelligence.