



Predicting human behavior in size-variant repeated games through deep convolutional neural networks

Afrooz Vazifedan¹ · Mohammad Izadi¹

Received: 29 July 2020 / Accepted: 29 June 2021 / Published online: 4 August 2021
© Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract

We present a novel deep convolutional neural network (DCNN) model for predicting human behavior in repeated games. The model is the first deep neural network presented on repeated games that is able to be trained on games with arbitrary size of payoff matrices. Our neural network takes the players' payoff matrices and the history of the play as input, and outputs the predicted action picked by the first player in the next round. To evaluate the model's performance, we apply it to some experimental games played by humans and measure the rate of correctly predicted actions. The results show that our model obtains an average prediction accuracy of about 63% across all the studied games, which is about 6% higher than the best average accuracy obtained by the baseline models in the literature.

Keywords Convolutional neural networks · Deep learning · Repeated games · Bounded rationality · Action prediction · Behavioral game theory

1 Introduction

Predicting the behavior of individuals in strategic interactions has always been considered as an important task in economics, politics, psychology and social behavior analysis. In the last decades, numerous game-theoretic learning models have been proposed to model the players' bounded rationality observed in empirical interactions. These models aim to explain the players' deviations from Nash equilibrium, while they are tended to be as accurate as possible [1–7].

Learning models in game theory target both repeated and unrepeated games. In unrepeated games, the goal is usually to predict the aggregate behavior of a population of players in an unseen game. However, in repeated games, the task is to predict the *step-by-step* actions played by the players [8]. This can be a real challenge due to many different possible scenarios the game may encounter to. An ideal model in these settings should be able to recognize patterns and

structures emerged in the players' behavior throughout the play. As an example, it should be able to predict the *alternation strategy* (alternation between some actions) as the most frequent outcome of empirical experiments on games such as *Battle of the Sexes* [9] or *Common-Pool Resources* [10].

Existing learning models in game theory can be divided into three groups. In the first group, the models try to directly identify the factors influencing humans' decision-making process, such as *the players' psychological features* or *features extracted from the game's payoff matrices*. Inputs to these models are the game's payoff matrices along with some parameters indicating the players' behavioral characteristics (Fig. 1, left). These parameters are obtained individually for each specific game, or are estimated based on some other games with similar payoff matrices. Examples from the models in the first group include *EWA* [11] and *SBN* [2] for repeated games and *Level-k* [12] and *Decision-Tree-based* [4] for unrepeated games.

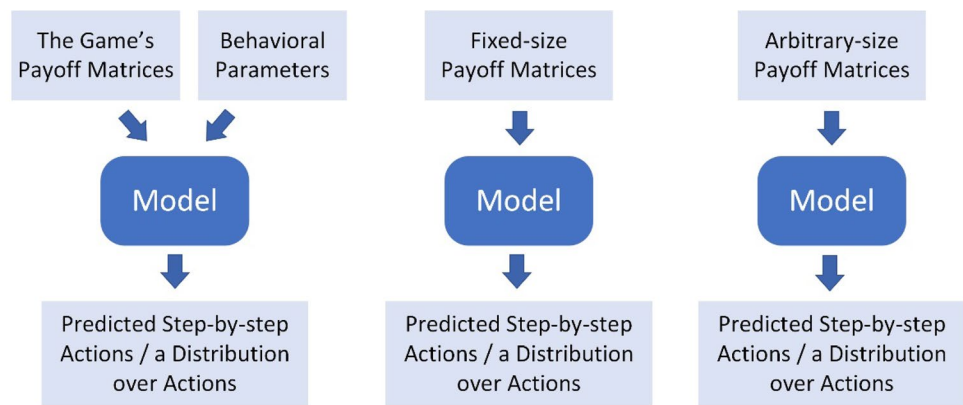
The major challenge in the first group is detecting the principal parameters involved in human interactive learning (such as the players' level of intelligence). To eliminate the need for these parameters, *neural networks* are utilized to automatically model the players' decision process. Neural networks allow prediction of future actions of the players, without any knowledge of their psychological features or

✉ Afrooz Vazifedan
af.vazifedan@gmail.com

✉ Mohammad Izadi
izadi@sharif.edu

¹ Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Fig. 1 Three kinds of learning models in Game theory, considering the models' inputs and outputs



preferences; only with the use of the payoffs and history of the play.

The second group of learning models consists of those neural networks that take the payoff matrices of the game *with a fixed size* as input, and output the predicted probability distributions for one player's actions (Fig. 1, middle). In fact, every entry of each payoff matrix is a separate input to the neural network. These models try to answer this question: "How does a player learn to play any $M_1 \times \dots \times M_n$ game by playing a finite set of other $M_1 \times \dots \times M_n$ games?" Examples from the second group include the model proposed by Kolumbus and Noti [8] for repeated games, and the model proposed by Bhatia and Golman [13] for unrepeated ones.

The third group of learning models consists of those neural networks that are able to take games with arbitrary size as input, and predict one player's actions as output (Fig. 1, right). As long as we know, the only existing model in this category is proposed by Hartford et al. [14]. Their model takes the players' payoff matrices, and in each layer of the neural network, it maps a tensor of the outputs from the previous layer into some matrix outputs, rather than a scalar one. So, the model can generalize across different input dimensions. This model is proposed for unrepeated games. In this paper, we extend it to repeated ones. Our work is also an extension of Kolumbus and Noti's neural network (2019) to a model that allows size-variant inputs.

Our proposed model is based on a *deep convolutional neural network (DCNN)* that automatically performs action prediction without requiring any analysis of human preferences or any expert feature engineering. Our task is a supervised learning problem, where the input is the payoff matrices of the players, as well as the history of the play up to round r , and the output is a prediction of the first player's action in round $r + 1$.

To evaluate the predictive power of our proposed model and compare it to other learning methods, we apply it to 25 behavioral games previously used in the literature. The games differ in the number of actions, pure-strategy

equilibria and playing rounds. This brings the chance to evaluate the model's performance in a diverse range of games. We report the rate of correctly predicted actions as a criterion to estimate the model's performance. The results show that our model outperforms the baseline methods in the literature.

The paper is organized as follows. In Sect. 2, we discuss the related works. In Sect. 3, the architecture of our proposed DCNN model is presented in details. In Sect. 4, we present the results of implementing the model on the empirical games and we compare the outputs with those of previous methods. In Sect. 5, we discuss the effect of the neural network's hyper parameters on the predicted outcomes. Finally, in Sect. 6, we conclude.

2 Related works

Two main conventional approaches to strategy learning in game theory are *Reinforcement Learning* and *Fictitious Play*. Reinforcement Learning, originally proposed by Erev and Roth [15], assumes that players learn to play optimal strategies through reinforcing their own past actions by the obtained payoffs. So, they increase the probability of playing actions that have met with success in previous rounds. Fictitious Play, a kind of *belief learning* proposed by Brown [16], assumes each player takes the empirical distribution of the actions played in the previous rounds by the opponents to be his/her belief about the opponents' mixed strategy. In each round, the player plays a best response to this belief.

Many other learning models such as *weighted reinforcement learning* or *stochastic fictitious play* are just variants of the two above approaches [17–21]. There are also some other methods that *combine* these two learning rules. For example, Ansari et al. [22] suggest a *hidden Markov chain* that consists of two states, each representing one of the reinforcement or belief learning rules. Players stochastically make transitions between the states according to a first-order Markov process. So, players use either one of the

above rules to choose an action in each round, but they may switch among them over the course of the game.

Another example of hybrid learning models is proposed by Chen et al. [2]. They present a *sequential Bayesian network (SBN)* framework for building game-playing agents. They show reinforcement/belief learning rules have a *Bayesian* explanation within this framework. Then, they derive a new algorithm, which is a hybrid of these two methods and can handle the specific setting in which a player cannot observe the opponent's strategy and payoff. They employ the *EM* algorithm to help infer unavailable information from the observed data.

Hybrid models also include *sophisticated* ones which consider the fact that real human players often assume that other players are also learning, so they respond to their *forecasts* of the others' actions. An example is the method presented by Mengel [7], which models forward-looking players that form beliefs by learning from past behavior of the other players and then best respond to these beliefs looking k rounds ahead. The model is described by a *stationary Markov chain*, whose states consist of the tuple of the game's history and the players' collective memory; and the transition matrix shows the probability of moving from one state to another.

The most famous and efficient mixture of reinforcement and belief learning approaches is *EWA* or *Experienced-Weighted-Attractions*, proposed by Camerer and Ho [11]. *EWA* is based on this idea that every player assigns a level of attraction to each of his/her strategies. Attractions are updated after every round of the play, according to the player's experiences, and determine the player's probability distribution over his/her choice set. This model has five parameters to capture the psychological properties of individual reasoning, such as *best response sensitivity* or the *discount factor of past experiences*. To make the model simple to estimate statistically, *self-tuning EWA* [23] was presented by the authors. By fixing two of the parameters at plausible values and replacing two others by deterministic functions, this variant of *EWA* is left with just one free parameter. *EWA* and self-tuning *EWA* are both *adaptive* (not sophisticated) models.

Another adaptive model of playing games is called *Regret Matching* [24]. It assumes players may depart from their current play with probabilities proportional to measures of regret for not having used other strategies in the past. The empirical distributions of the play based on this assumption converge to the set of Correlated equilibrium of the game. *QRE* or *Quantal Response Equilibrium* is another learning model which imposes the assumption that beliefs about choices of other players are accurate, but responding to them may be imperfect [25]. So, noisy responses are allowed and actions with higher expected payoffs are chosen more often based on a logit function.

As we mentioned in the introduction, the limitation of all the above models is their need of psychological analysis of human learning process. We take a different approach and employ *neural networks* to overcome this need. Most closely related to our work are those of Kolumbus and Noti [8] and Hartford et al. [14], which incorporate convolutional neural networks to predict players' actions in repeated and stage games respectively.

Our proposed model extends the model presented by Kolumbus and Noti [8] to allow *size-variant inputs*. We have the same learning setting and prediction task as their work. Our contribution over their model is proposing additional network architectures and performing a more extensive assessment of the model and comparison to the baselines. Their network is evaluated on only one dataset containing similar 2×2 games, as it simply takes every entry of each payoff matrix as a separate input, and outputs one of the two actions which is predicted to be played by the first player. They also incorporate a history vector of length 20 as input to their model and evaluate the results for rounds greater than 20. Such history may not be available in many behavioral games. So, we confine our model to use a history of length up to 10. We also evaluate our proposed network on more diverse games and compare our results with the most efficient models from the literature.

Our proposed model can also be considered as an extension of the network presented by Hartford et al. to be applied to *repeated games*. Their neural network intends to predict initial play. It takes the players' payoff matrices as input, and in each layer, it maps a tensor of the previous layer's outputs into some matrix outputs. The network consists of two parts: *feature* layers and *action-response* layers. Each hidden unit in a *feature* layer is a nonlinear transformation of the weighted sum of the preceding layer's units, consisting of the corresponding hidden units and the max-pooled quantities from each row and column. In the last feature layer, each unit sums uniformly over the columns of the matrices and then applies a *softmax* function to convert the vectors to normalized distributions. So, the output of feature layers (or *features*) will be some distributions over the row player's actions.

Action-response layers in Hartford's model represent iterative strategic reasoning: the second player responds to the features; then, each player responds to the other player's response, and so on to some finite depth. The output of the network is a weighted sum of the outputs of the action-response layers. The final trained network on empirical data consists of two feature layers of 50 units each and just *one* action-response layer, suggesting there is no need of iterative strategic reasoning for predicting the outcome.

We have a similar approach to that of Hartford's in forming features based on payoff matrices; however, our target is to predict the step-by-step actions of the players in repeated

games. So, we make use of the history of the play as well as the input payoff matrices. This extends our architecture to perform some other convolutions on the history matrices and then obtain the final probability distribution over the actions of one player.

3 Convolutional neural network model

We present the proposed model for two-player games, but it can be easily extended to the games with more than two players. Our model takes as input the players' payoff matrices (of size $m \times n$) as well as the history of the play (h previously played actions by the players). It outputs the predicted m -dimensional probability distribution over the first player's actions. The action with the most probability will be considered as the predicted action by the model. Figure 2 shows the architecture of the proposed model.

Our neural network has three parts. In the first part, the input payoff matrices go through several convolutional and pooling layers to produce some feature matrices, which we call *extended-payoffs*. In the second part, we perform a convolution on the history matrix of the opponent to extract an aggregated *history vector*. After that, we take a dot product between this history vector and the extended-payoff matrices to obtain the *priority vectors* which indicate some probability distributions on the actions of the first player. In the third part of our architecture, the history matrix of the first player goes through a convolutional layer and *self-history vectors* are extracted. The final probability distribution will be the output of another convolutional layer on the self-history vectors, the priority vectors and the row-pooled features extracted from the extended-payoff matrices.

3.1 Convolutional and max-pooling layers

Extended-payoff matrices at each hidden layer are the same size as the input payoff matrices and are the result of some convolutional and pooling operations on the payoffs. Convolutional layers in neural networks make use of kernels to detect features present in the input. A kernel is just a matrix of weights that are trained throughout the process of fitting the network with the data. It carries out a convolutional operation consisting of an element-wise product and sum between the kernel and the input matrices. In the first part of our neural network, we use 1×1 convolutional kernels. Then, all it does is adding nonlinearity: each element of the output matrix will be a nonlinear transformation of the weighted sum of the inputs. So, the dimensions of the input matrices do not change through the convolutional layers.

In a max-pooling layer of a neural network, a window passes over the input matrix, and the maximum value within the window is pooled into the output matrix. In the first part

of our model, after each convolutional operation on payoff matrices, we perform max-pooling operations along each row and column of the input so that each element of the output matrix would depend on the corresponding elements in the input as well as the maximum values in their row and column. As a result, each action of the players is associated with the maximum payoff that could be earned by playing that action. Finally, an extra max-pooling layer along the rows is placed after the last convolutional layer to represent the features of the actions of the first player extracted from the extended-payoff matrices.

3.2 History matrices

The proposed model receives the $m \times h$ and $n \times h$ history matrices of the first player and the opponent, respectively. These matrices contain the players' played actions in the last h rounds of the play. Each column of every matrix is dedicated to the history of one round of the play, and the element corresponding to the picked action in that round is set to *one* and the other elements are set to *zero*.

Then, the opponent's history matrix goes through a convolutional layer that aggregates the matrix into one vector of the size of the number of the opponent's actions. This vector represents a distribution over the opponent's actions and shows how frequently each of the actions has been picked in the last h rounds. The actions that have been picked more recently may get more weight by the convolutional kernel. After that, the result vector is multiplied by the extended-payoff matrices by the *dot product* layer to produce some m -length vectors that show the *priorities* with which the first player's actions should be selected, in order to *respond to the opponent*.

In the third part of our proposed architecture, the self-history matrix goes through a convolutional layer, and as a result, some aggregate vectors are produced to indicate some distributions on the first player's actions. The final layer of our DCNN model is another convolutional layer which is in charge of generating the probability of picked actions by the first player, given three input layers: the *feature vectors* extracted from the extended-payoff matrices, the *priority vectors* extracted from the opponent's history matrix and the *self-history vectors* extracted from the first player's history matrix.

4 Results

In this section, we report how well the proposed model, compared to the other baseline learning models, can predict the picked actions by the players in different human-subject games. As the evaluation criterion, we measure the predictive *hit-rates* of the models, which is defined as the fraction

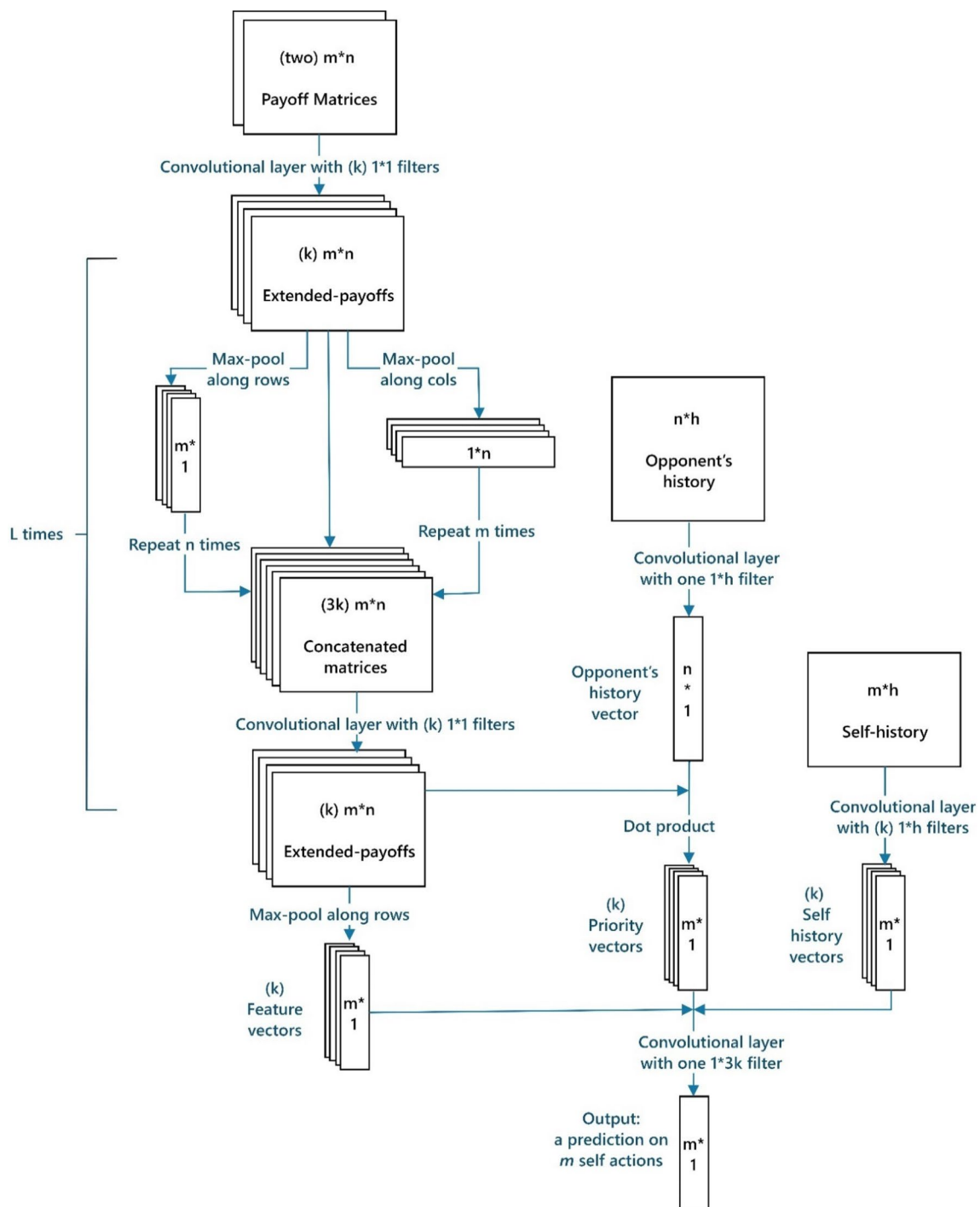


Fig. 2 The architecture of our proposed DCNN model

of choices that are correctly picked by the models in each round. We apply the models to 25 games from 7 different datasets which are described as the following. The models' performance on the datasets is reported afterwards.

4.1 Data description

Table 1 provides some details about the games incorporated in our evaluation process. The details are: the number of strategies among which the players can choose, the number of pure Nash equilibrium of the game, the number of rounds of the play, the number of subjects (samples of each repeated

Table 1 Details of the games used in the experiments for measuring the predictive power of our model

Game	Number of strategies	Number of pure Nash eq.	Number of rounds	Number of subjects	References
GRP1	4	0	40	20	[28]
GRP2	6	0	40	20	[28]
GRP3	4	0	40	20	[28]
GRP4	6	0	40	20	[28]
Patent	5,6	0	80	72	[29]
DSG3	3	1	20	32	[18]
nDSG3	3	1	20	32	[18]
DSG4	4	1	20	20	[18]
nDSG4	4	1	20	20	[18]
CPR1	2	1	20	52	[9]
CPR2	2	1	20	52	[9]
MDN	7	7	10	54	[31]
PD	2	1	10	280	[1]
Selten (12 games)	2	1	Not fixed	780	[30]

game available in the corresponding dataset) and the reference of that dataset. A brief description about each of the games comes in the following. The games' payoff matrices can be found in "Appendix 1".

GRPs are four constant-sum games with unique mixed strategy equilibrium. Each game refers to a lottery where one player wins and the other loses. *Patent race* is an investment game in which the player who is investing the largest amount wins a fixed reward. *DSG3* and *DSG4* are two dominance solvable games with *three* and *four* number of actions, respectively. *nDSG3* and *nDSG4* are two similar games to these two games, which are not dominance solvable. These four games have been chosen to have Nash payoffs that are on the Pareto frontier, and the Nash equilibrium payoffs not symmetric.

CPR1 and *CPR2* are two assignment games that have been influential in the study of common-pool resources (CPR). In these games, most of the players play according to the *alternation* strategy between their actions, as taking turns in holding each of the resources is the most efficient way for the players to cooperate. Inferring this kind of history-dependent rules of behavior from experimental plays can be a challenge for learning models.

MDN is a coordination game where players choose among some integers and each player's payoff decreases in the distance between his/her choice and the *median* of all choices. *PD* is the Prisoners' Dilemma game, which is the most famous game known as a social dilemma in game theory. Ultimately, Selten games are 12 two-player 2×2 games, each with a unique mixed Nash equilibrium. We use these twelve fixed-size games in a *separate training and test phase* as the other games to compare our model's performance to

that of Kolumbus and Noti's [8]. The corresponding results come in the last subsection of this section.

4.2 The model's predictive performance

To tune the network's hyper parameters, we use 10% of each game's subjects as the validation set. Then, we use fivefold cross-validation on the rest of the data in order to train and test the model. This procedure consists of partitioning the dataset into five groups, training the model on four of the groups (the training set) and validating the analysis on the remaining group (the test set). Results are then averaged over the five subsets to give an estimate of the model's predictive performance. We report *hit-rates* as the accuracy of the prediction separately for each game. Hit-rate is the rate of the correctly picked actions by the model, which are the actions corresponding to the largest (the most probable) output of the network in each round.

The values for the network's hyper parameters leading to the best performance have been achieved to be 2 for L (number of the hidden layers), 45 for k (number of the kernels of the convolutional layers), 10 for h (the history length) and tanh as the activation function of all the network layers. The *Adam* optimization algorithm is used to train our neural network. The model is regularized using *dropout* layers after each convolution layer on the payoff matrices with drop rate of 0.2. The learning metric of the network is selected to be *accuracy*. Before passing the payoff matrices of the games to the network, we normalize them, so all the payoffs would be between zero and one.

Table 2 shows the hit-rate percentages of our proposed deep convolutional neural network (DCNN) along with some other baseline models in the literature, including weighted

Table 2 Hit-rate percentages of the proposed (DCNN) and some other learning models on the studied behavioral games

Games models	GRP1	GRP2	GRP3	GRP4	Patent	DSG3	nDSG3	DSG4	nDSG4	CPR1	CPR2	MDN	PD	Game average	Step average
Weighted fictitious play	36	26	28	24	43	15	54	16	26	21	32	12	55	29.8	39.2
Reinforcement learning	30	20	29	19	46	67	61	58	60	56	73	71	75	51.2	62.4
QRE	21	18	23	16	49	49	45	40	35	62	62	20	70	39.2	52.9
EWA	40	34	34	30	57	67	63	61	61	73	87	58	75	56.9	66.2
Regret matching	33	27	32	23	48	67	57	58	61	59	79	77	78	53.8	65.2
Average aggregate	36	34	35	31	56	68	63	57	62	67	83	74	72	56.8	65.3
Cognitive hierarchy	25	15	24	16	14	31	39	33	32	78	87	14	57	35.8	45.3
DCNN (proposed)	43	36	36	26	49	72	65	63	64	95	95	89	85	62.9	74.6
Average of above models	33.0	26.2	30.1	23.1	45.4	54.6	55.9	48.2	50.1	63.9	74.8	51.9	70.9	48.3	58.9
Max of all tested models	43	37	38	33	58	72	68	63	66	96	96	89	85	64.9	76.1

fictitious play (WFP), reinforcement learning (RL), quantal response equilibrium (QRE), experience weighted attractions (EWA), regret matching (RM), average aggregate (AA) and cognitive hierarchy (CH). The *average aggregate* (AA) model simply predicts the next action based on the average aggregate distribution of the first player's actions in the previous rounds of the plays, according to the data. It provides a *sanity check* that the network learns the dynamics and not only the average action in the data.

The *cognitive hierarchy* model [26] is an initial play predictor that assumes players choose their actions according to some cognitive levels: level-0 players play randomly; and level- k players best-respond to others, assuming they belong to level $k - 1$. Motivated by empirical evidence [26], we constrain all players to be level-2 in our experiments. To be able to compare results of this model with the other ones, we assume that cognitive hierarchy predicts the same action (as for the initial play) for the other steps of the play.

Table 2, in the two last columns, includes the average hit-rates obtained by each of the models. The *Game average* column takes the average across all the games, and the *Step average* column shows the average hit-rate over all the prediction steps. In the two last rows, the table reports the average hit-rate of all the models for each specific game, as well as the maximum hit-rate ever obtained on each game by any of the models we tested in our experiments (including other variants of the proposed network). The best fits for each game are printed in bold. As improved by results, our DCNN model achieves the best average hit-rate among all the studied models. It has also the best performance in most of the games, especially in *CPRs*, *MDN* and *PD* games. Figure 3 demonstrates the same results as Table 2.

The reason we report the *maximum* hit-rate obtained on each game is to have a better estimate of the performance of the models. Reaching a certain maximum level of accuracy in a certain game may not be possible by even an ideal model, due to different strategies and approaches the players may take during the play or even due to possible noise in reporting the data. Models that we have tested include more than twenty different architecture variants of our model (for example, the one that replaces the convolutional layers on the history matrices and the final layer by just one convolutional layer). The variants also include models incorporating other possible values for hyper parameters, or metric and optimization functions.

4.3 The model's performance on fixed-size games

In order to compare our results to those of Kolumbus and Noti's [8], we train our model on the same dataset as theirs, and employ the same training and test settings. They train their model on 11 games of Selten dataset [27] and evaluate its prediction accuracy on the 12th game. Then, they take the

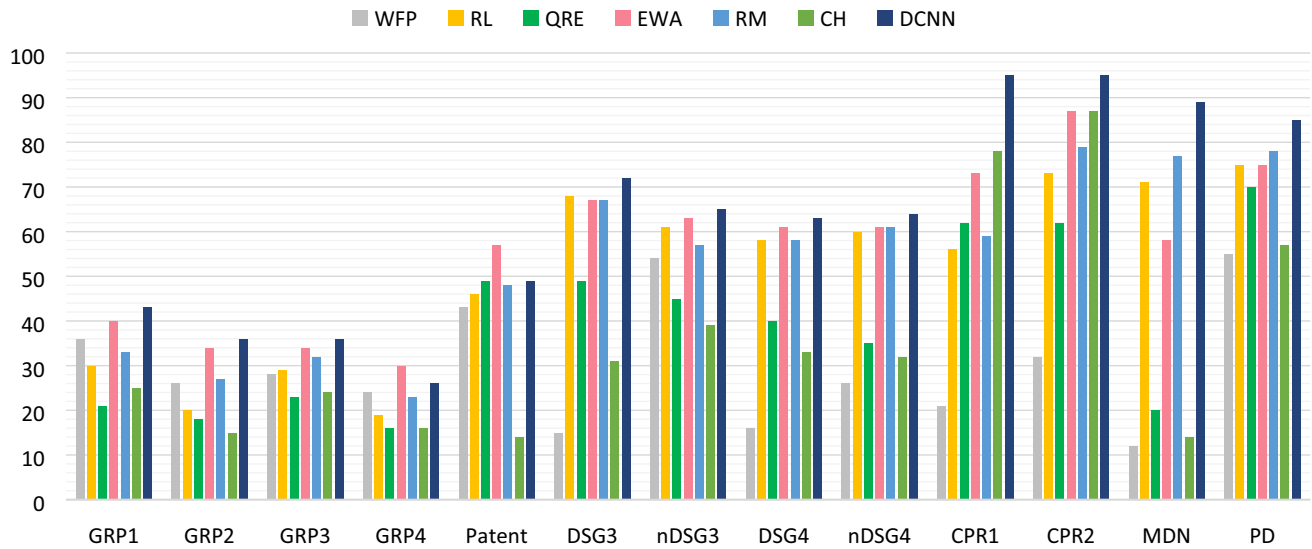


Fig. 3 Hit-rate percentage of the proposed (DCNN) and some baseline models from the literature on the experimental games

average performance over all the games. We use the same method along with the hyperparameters obtained in above experiments. Subsequently, an average hit-rate of more than 98% is obtained by our DCNN model for every game. We think this happens due to the similarity of the games in the dataset, as well as the uniformity of the selected actions in most of the consecutive rounds of the empirical plays.

5 Parameter analysis

This section explores the effect of the hyper parameters of the proposed model on its average prediction accuracy across the 13 behavioral games of Table 2. We vary the number of hidden layers (L) among integer values in the range $[1, 20]$ and the number of kernels of the convolutional layers (k) in the range $[1, 50]$ and measure hit-rate percentages of the network *on the validation set*. Figure 4 shows the corresponding results. The best fit is achieved by $L = 2$ and $k = 45$. The length h of the history is kept at 10 in these experiments.

Fig. 4 Effect of the number of kernels in the convolutional layers (k) and the number of hidden layers (L) of the proposed model on its average prediction hit-rates (reported in the bubbles), based on the validation set

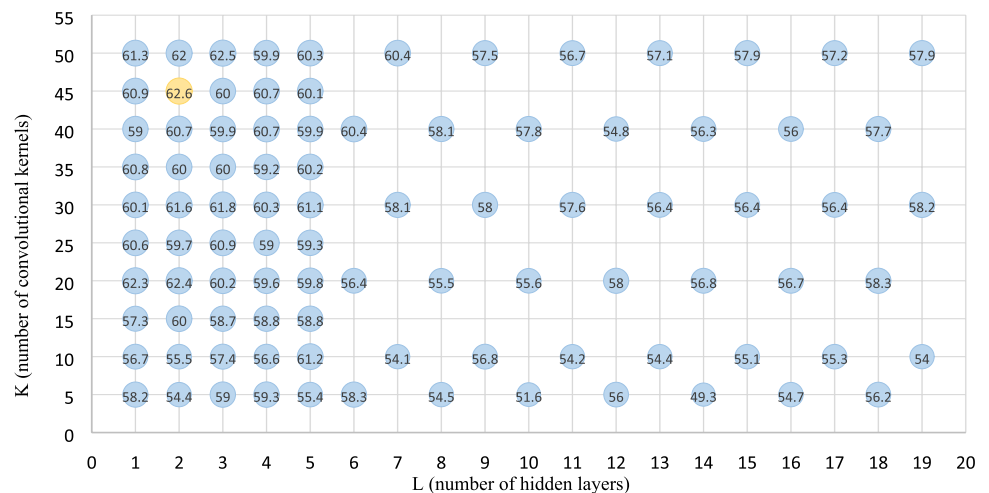
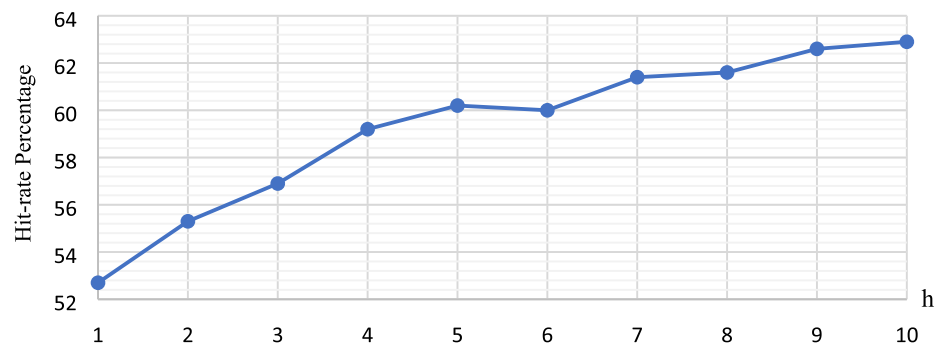


Fig. 5 Effect of parameter h (the history length) on the average hit-rate percentage of the proposed model, based on the test sets



We also measure the influence of parameter h on the performance of the model. This parameter shows the number of previous rounds of the play whose information is put in the history matrices. Figure 5 demonstrates the results *on the test sets*. The figure illustrates that while we increase the size of the history matrices, the performance improves. We have limited parameter h to 10, so that the model can be tested on games that have been repeated for maximum 10 rounds.

6 Conclusion

In this paper, we present a novel convolutional neural network architecture to model the bounded rational behavior of human subjects while playing repeated games. Our model maps the game's payoff matrices and the history of the play to the predicted action of the first player in the next step. It improves the prediction performance upon state-of-the-art models and achieves an average hit-rate of near 63% in the tested behavioral games.

Our proposed network can be applied to games with distinct sizes of payoff matrices in *one* training phase, as it processes matrix units in each layer, rather than scalar

ones. So, the number of the network's parameters depends only on the number of hidden layers of the network, and not the size of input or output matrices. As a result, our approach reduces the training time of the network and also its possibility of overfitting.

Deep neural networks have the ability to generalize over the messy data from the real world. They help automatically discover the features and patterns in the data, and form highly nonlinear representations of the input. A successful example of this ability is demonstrated by our model. As we show in the *Results* section, our model can predict the correctly picked actions 95% of the time in two of the studied games where the *alternation* strategy is empirically observed.

Some future works related to this paper are to extend the proposed model to work with games with imperfect information. Another extension is to consider the *end-game effect* (the fact that human players usually act more selfish in the last rounds of the repeated games). Other types of neural networks, such as an RNN for processing the past actions of the players, could be examined within the framework, too.

Appendix 1

This appendix is dedicated to the payoff matrices of the behavioral games used in our experiments.

	A	B	C	D
A	5,5	0,0	0,0	5,0
B	0,0	0,0	5,5	5,0
C	0,0	5,5	5/3,5/3	5/3,10/3
D	0,5	0,5	10/3,5/3	5,5

GRP1

	A	B	C	D
A	10,10	0,0	0,0	10,0
B	0,0	0,0	10,10	10,0
C	0,0	10,10	10/3,10/3	10/3,20/3
D	0,10	0,10	20/3,10/3	10,10

GRP3

	A	B	C	D	E	F
A	5,5	0,0	0,0	0,0	0,0	5,0
B	0,0	0,0	5,5	5,5	5,5	5,0
C	0,0	5,5	0,0	0,5	5,0	0,5
D	0,0	5,5	5,0	0,0	0,5	0,0
E	0,0	5,5	0,5	5,0	0,0	5,5
F	0,5	0,5	5,0	0,0	5,5	5,5

GRP2

	A	B	C	D	E	F
A	10,10	0,0	0,0	0,0	0,0	10,0
B	0,0	0,0	10,10	10,10	10,10	10,0
C	0,0	10,10	0,0	0,10	10,0	0,10
D	0,0	10,10	10,0	0,0	0,10	0,0
E	0,0	10,10	0,10	10,0	0,0	10,10
F	0,10	0,10	10,0	0,0	10,10	10,10

GRP4

	A	B	C	D	E	F
A	4,5	4,14	4,13	4,12	4,11	4,10
B	13,5	3,4	3,13	3,12	3,11	3,10
C	12,5	12,4	2,3	2,12	2,11	2,10
D	11,5	11,4	11,3	1,2	1,11	1,10
E	10,5	10,4	10,3	10,2	0,1	0,10

Patent

	A	B	C
A	51,30	35,43	93,21
B	35,21	25,16	32,94
C	68,72	45,69	13,62

DSG3

	A	B	C
A	12,83	39,56	42,45
B	24,12	12,42	58,76
C	89,47	33,94	44,59

nDSG3

	A	B	C	D
A	44,72	51,30	35,43	93,21
B	36,18	35,21	25,16	32,94
C	16,65	68,72	45,69	13,62
D	52,68	58,67	39,78	11,72

DSG4

	A	B	C	D
A	12,83	39,56	42,45	47,54
B	24,12	12,42	58,76	49,61
C	61,63	79,24	21,13	13,81
D	89,47	33,94	44,59	31,11

nDSG4

	A	B	C	D	E	F	G
A	70,70	75,65	70,50	55,25	30,-10	-5,-55	-50,-110
B	65,75	80,80	85,75	80,60	65,35	40,0	5,-45
C	50,70	75,85	90,90	95,85	90,70	75,45	50,10
D	25,55	60,80	85,95	100,100	105,95	100,80	85,55
E	-10,30	35,65	70,90	95,105	110,110	115,105	110,90
F	-55,-5	0,40	45,75	80,100	105,115	120,120	125,115
G	-110,-50	-45,5	10,50	55,85	90,110	115,125	130,130

Median

	A	B
A	49,49	98,42
B	42,98	21,21

CPR1

	A	B
A	60,60	120,20
B	20,120	10,10

CPR2

	A	B
A	7,7	0,12
B	12,0	4,4

Prisoners' Dilemma

	A	B
A	10,8	0,18
B	9,9	10,8

Selten 1

	A	B
A	9,4	0,13
B	6,7	8,5

Selten 2

	A	B
A	8,6	0,14
B	7,7	10,4

Selten 3

	A	B
A	7,4	0,11
B	5,6	9,2

Selten 4

	A	B
A	7,2	0,9
B	4,5	8,1

Selten 5

	A	B
A	7,1	1,7
B	3,5	8,0

Selten 6

	A	B
A	10,12	4,22
B	9,9	14,8

Selten 7

	A	B
A	9,7	3,16
B	6,7	11,5

Selten 8

	A	B
A	8,9	3,17
B	7,7	13,4

Selten 9

	A	B
A	7,6	2,13
B	5,6	11,2

Selten 10

	A	B
A	7,4	2,11
B	4,5	10,1

Selten 11

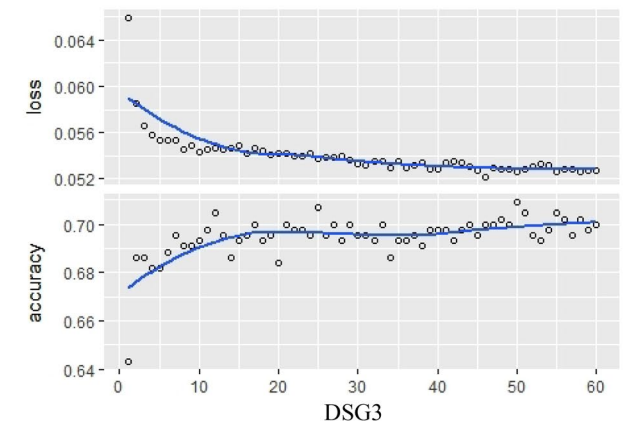
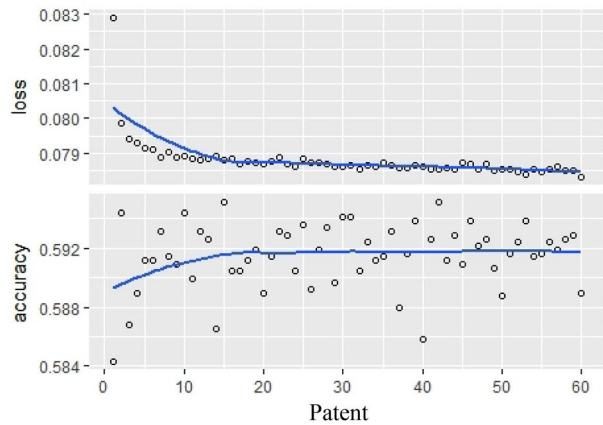
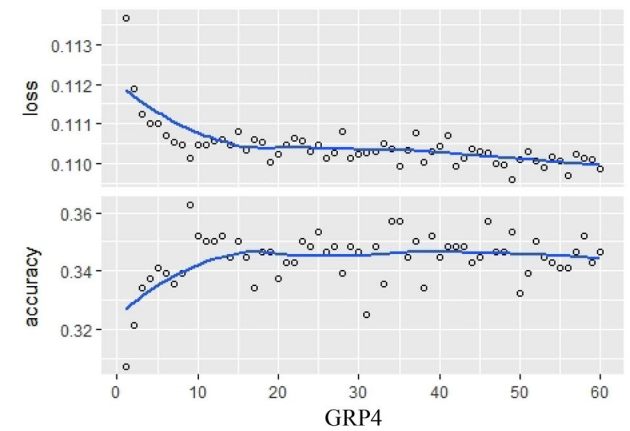
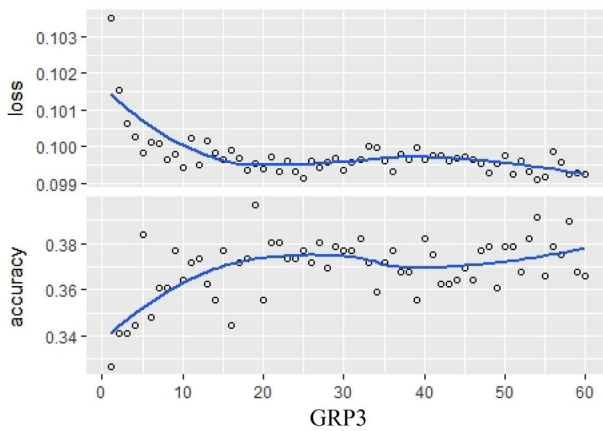
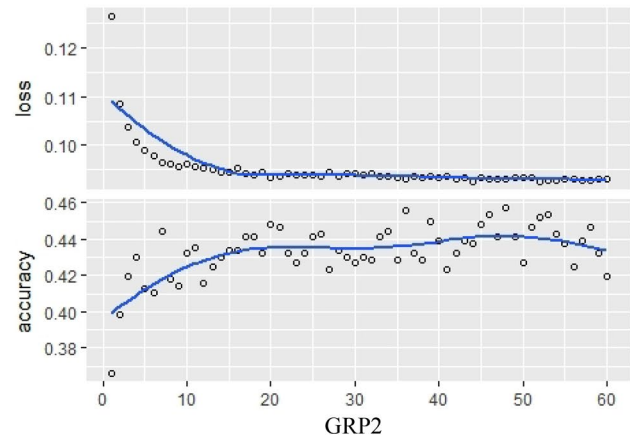
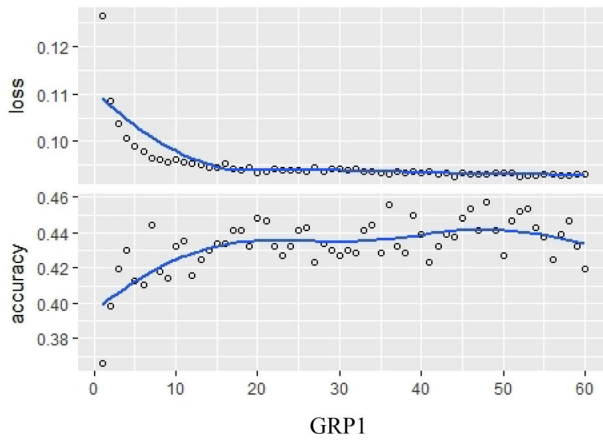
	A	B
A	7,3	3,9
B	3,5	10,0

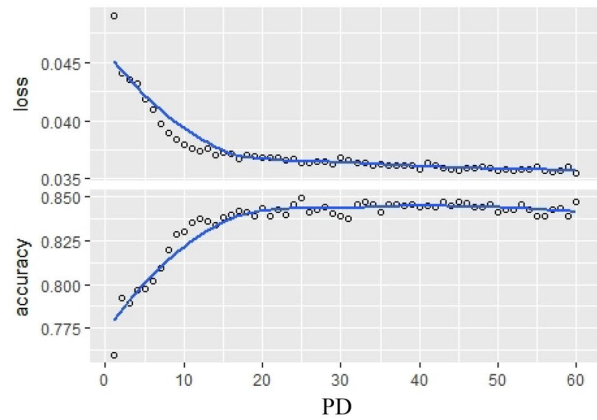
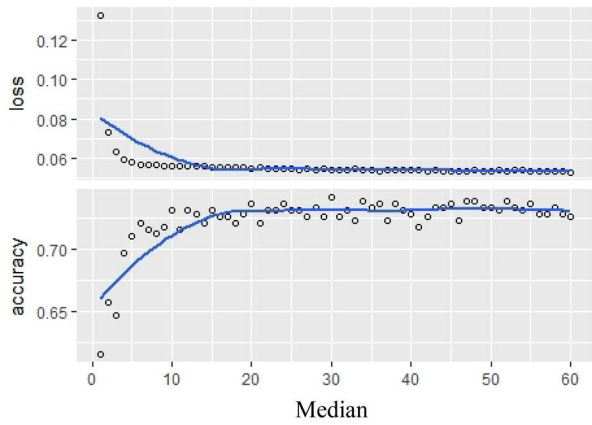
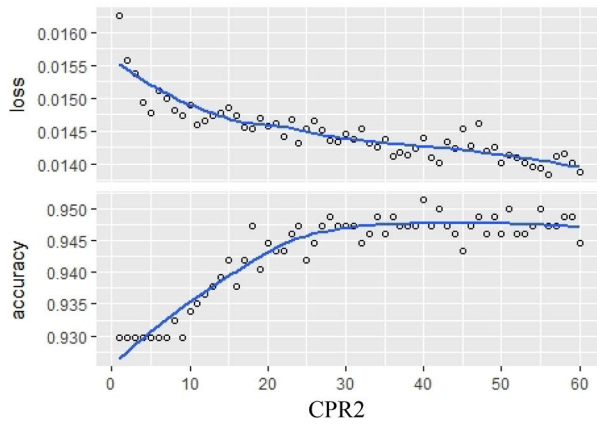
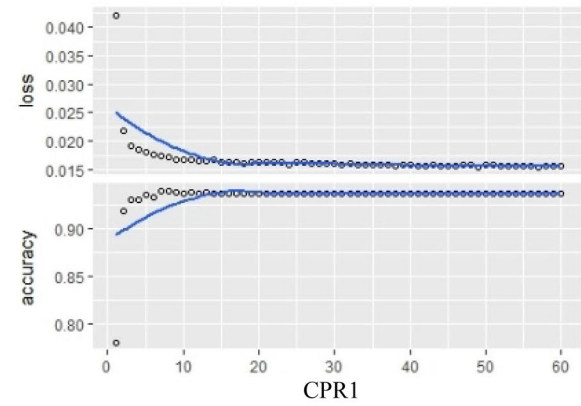
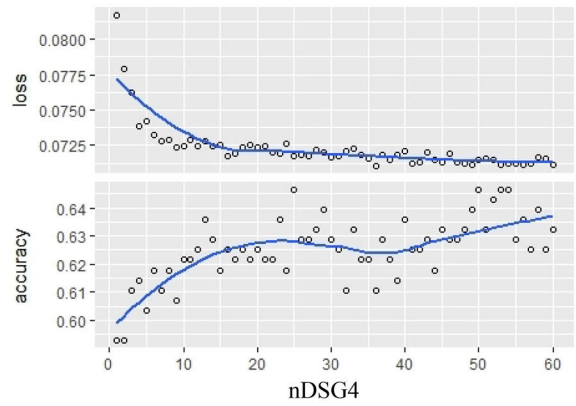
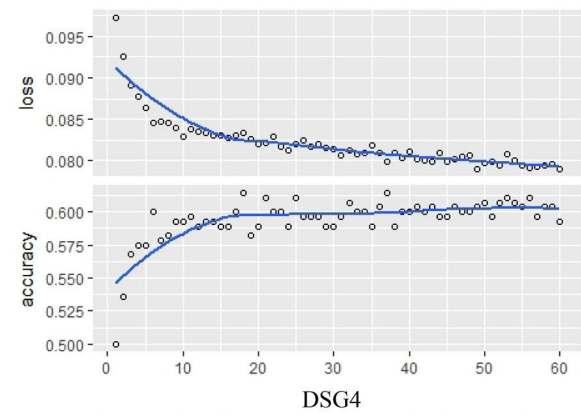
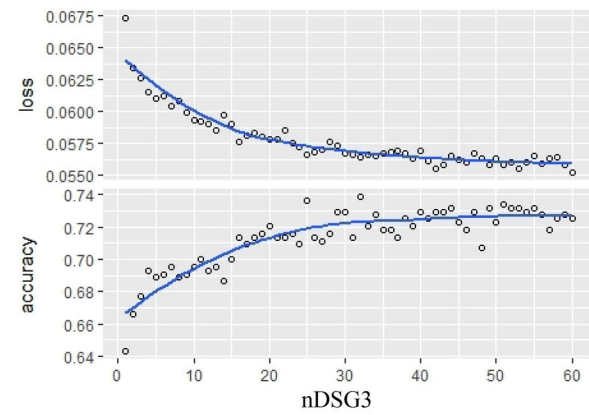
Selten 12

Appendix 2

This appendix is dedicated to the learning curves of training the proposed network on the behavioral games in Table 2. The curves demonstrate *loss* and *accuracy* of the network obtained in 60 epochs of one training phase on the whole

train set, reported individually for each game. However, in the *Results* section of the paper, the test results are reported based on the network trained for 6 times on the train set, each consisting of one epoch on every subject.





References

1. Camerer, C.F., Ho, T.H.: Behavioral game theory experiments and modeling (chapter 10). In: *Handbook of Game Theory with Economic Applications* (Vol. 4, pp. 517–573). Elsevier (2015).
2. Chen, W., Chen, Y., Levine, D.K.: A unifying learning framework for building artificial game-playing agents. *Ann. Math. Artif. Intell.* **73**(3–4), 335–358 (2015)
3. Fudenberg, D., Levine, D.K.: Whither game theory? Towards a theory of learning in games. *VOPROSY ECONOMIK I*, 5 (2017)
4. Fudenberg, D., Liang, A.: Predicting and understanding initial play. *Am. Econ. Rev.* **109**(12), 4112–4141 (2019)
5. Hyndman, K., Ozbay, E.Y., Schotter, A., Ehrblatt, W.Z.E.: Convergence: an experimental study of teaching and learning in repeated games. *J. Eur. Econ. Assoc.* **10**(3), 573–604 (2012)
6. Izquierdo, L.R., Izquierdo, S.S., & Vega-Redondo, F.: Learning and evolutionary game theory. In: *Encyclopedia of the Sciences of Learning*. Springer, Boston (2012)
7. Mengel, F.: Learning by (limited) forward looking players. *J. Econ. Behav. Organ.* **108**, 59–77 (2014)
8. Kolumbus, Y., Noti, G.: Neural networks for predicting human interactions in repeated games. arXiv preprint arXiv:1911.03233 (2019).
9. Mathevet, L., & Romero, J. (2012). Predictive repeated game theory: Measures and experiments.
10. Cason, T.N., Lau, S.H.P., Mui, V.L.: Learning, teaching, and turn taking in the repeated assignment game. *Econ. Theor.* **54**(2), 335–357 (2013)
11. Camerer, C.F., Ho, T.H.: Experience-weighted attraction learning in normal form games. *Econometrica* **67**(4), 827–874 (1999)
12. Camerer, C., Ho, T., Chong, K.: Models of thinking, learning, and teaching in games. *Am. Econ. Rev.* **93**(2), 192–195 (2003)
13. Bhatia, S., Golman, R.: A recurrent neural network for game theoretic decision making. In: *Proceedings of the Annual Meeting of the Cognitive Science Society* (Vol. 36, No. 36) (2014).
14. Hartford, J.S., Wright, J.R., Leyton-Brown, K.: Deep learning for predicting human strategic behavior. In: *Advances in Neural Information Processing Systems* (2016).
15. Erev, I., Roth, A.E.: Predicting how people play games: reinforcement learning in experimental games with unique, mixed strategy equilibria. *Am. Econ. Rev.* 848–881 (1998).
16. Brown, G.W.: Iterative solution of games by fictitious play. *Act. Anal. Prod. Allocat.* **13**(1), 374–376 (1951)
17. Elie, R., Pérolat, J., Laurière, M., Geist, M., Pietquin, O.: Approximate fictitious play for mean field games. arXiv preprint arXiv:1907.02633 (2019).
18. Kamra, N., Gupta, U., Wang, K., Fang, F., Liu, Y., Tambe, M.: Deep fictitious play for games with continuous action spaces. In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (pp. 2042–2044). International Foundation for Autonomous Agents and Multiagent Systems (2019).
19. Lanctot, M., Lockhart, E., Lespiau, J.B., Zambaldi, V., Upadhyay, S., Pérolat, J., et al.: Openspiel: a framework for reinforcement learning in games. arXiv preprint arXiv:1908.09453.
20. Ma, W.C., Huang, D.A., Lee, N., Kitani, K.M.: Forecasting interactive dynamics of pedestrians with fictitious play. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 774–782) (2017).
21. Wang, H., Yu, C., Wu, L.: A neighborhood correlated empirical weighted algorithm for fictitious play. In: *Life System Modeling and Intelligent Computing* (pp. 305–311). Springer, Berlin (2010)
22. Ansari, A., Montoya, R., Netzer, O.: Dynamic learning in behavioral games: a hidden Markov mixture-of-experts approach. *Quant. Mark. Econ.* **10**(4), 475–503 (2012)
23. Ho, T.H., Camerer, C.F., Chong, J.K.: Self-tuning experience weighted attraction learning in games. *J. Econ. Theory* **133**(1), 177–198 (2007)
24. Hart, S., Mas-Colell, A.: A simple adaptive procedure leading to correlated equilibrium. *Econometrica* **68**(5), 1127–1150 (2000)
25. McKelvey, R.D., Palfrey, T.R.: Quantal response equilibria for normal form games. *Games Econom. Behav.* **10**(1), 6–38 (1995)
26. Camerer, C.F., Ho, T.H., Chong, J.K.: A cognitive hierarchy model of games. *Q. J. Econ.* **119**(3), 861–898 (2004)
27. Selten, R., Chmura, T.: Stationary concepts for experimental 2x2-games. *Am. Econ. Rev.* **98**(3), 938–966 (2008)
28. Mookherjee, D., Sopher, B.: Learning and decision costs in experimental constant sum games. *Games Econom. Behav.* **19**(1), 97–132 (1997)
29. Rapoport, A., Amaldoss, W.: Mixed strategies and iterative elimination of strongly dominated strategies: an experimental investigation of states of knowledge. *J. Econ. Behav. Organ.* **42**(4), 483–521 (2000)
30. Van Huyck, J.B., Battalio, R.C., Beil, R.O.: Tacit coordination games, strategic uncertainty, and coordination failure. *Am. Econ. Rev.* **80**(1), 234–248 (1990)
31. Andreoni, J., Miller, J.H.: Rational cooperation in the finitely repeated prisoner’s dilemma: experimental evidence. *Econ. J.* **103**(418), 570–585 (1993)
32. Marchiori, D., Warglien, M.: Predicting human interactive learning by regret-driven neural networks. *Science* **319**(5866), 1111–1113 (2008)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.