Don't Predict Counterfactual Values, Predict Expected Values Instead

Jeremiasz Wołosiuk[1], **Maciej Świechowski**[2,3], **Jacek Mańdziuk**[3]

[1] Deepsolver
[2] QED Software
[3] Warsaw University of Technology
jeremi@deepsolver.com, maciej.swiechowski@qed.pl, jacek.mandziuk@pw.edu.pl

Abstract

Counterfactual Regret Minimization algorithms are the most popular way of estimating the Nash Equilibrium in imperfect-information zero-sum games. In particular, DeepStack - the state-of-the-art Poker bot – employs the so-called Deep Counterfactual Value Network (DCVN) to learn the Counterfactual Values (CFVs) associated with various states in the game. Each CFV is a multiplication of two factors: (1) the probability that the opponent would reach a given state in a game, which can be explicitly calculated from the input data, and (2) the expected value (EV) of a payoff in that state, which is a complex function of the input data, hard to calculate. In this paper, we propose a simple yet powerful modification to the CFVs estimation process, which consists in utilizing a deep neural network to estimate only the EV factor of CFV. This new target setting significantly simplifies the learning problem and leads to much more accurate CFVs estimation. A direct comparison, in terms of CFVs prediction losses, shows a significant prediction accuracy improvement of the proposed approach (DEVN) over the original DCVN formulation (relatively by $9.18 - 15.70\%$ when using card abstraction, and by $3.37 - 8.39\%$ without card abstraction, depending on a particular setting). Furthermore, the application of DEVN improves the theoretical lower bound of the error by $29.05 - 31.83\%$ compared to the DCVN pipeline when card abstraction is applied.

Additionally, DEVN is able to achieve the goal using significantly smaller, and faster to infer, networks.

While the proposed modification may seem to be of a rather technical nature, it, in fact, presents a fundamentally different approach to the overall process of learning and estimating CFVs, since the distributions of the training signals differ significantly between DCVN and DEVN. The former estimates CFVs, which are biased by the probability of reaching a given game state, while training the latter relies on a direct EV estimation, regardless of the state probability. In effect, the learning signal of DEVN presents a better estimation of the true value of a given state, thus allowing more accurate CFVs estimation.

1 Introduction

Poker is probably the most popular imperfect-information card game in contemporary Artificial Intelligence (AI) research (Billings et al. 2002; Rubin and Watson 2011). The game has been an object of intensive AI studies since Darse Billing's master thesis (Billings 1995). Soon after, online Poker platforms became popular and further boosted the interest in this game. In 2006, the first annual AAAI Computer Poker Competition was held (Bard et al. 2013).

The first notable Poker-playing systems were Loki (Billings et al. 1998) and Poki (Davidson et al. 2000). Both programs employed the enumeration approach, wherein each possible game state (with respect to hidden information) was assigned a heuristic score that denoted its winning chance. The heuristics relied on the given hand strength, hand potential, betting strategy and some elements of opponent modelling.

The next wave of programs operated by approximating Nash equilibrium strategies, which are commonly known as game theory optimal (GTO) strategies in the Poker community. To achieve this goal, Vexbot (Schauenberg 2006) employed expectimax search. PsOpti/Sparbot (Billings et al. 2003) applied linear programming on heavily abstracted game trees. Others used Monte Carlo simulation-based approaches (Schweizer et al. 2009; Broeck, Driessens, and Ramon 2009) including Monte Carlo Tree Search (Heinrich and Silver 2014), which is a very popular and successful technique in games in general (Świechowski et al. 2023).

A major milestone was the introduction of Counterfactual Regret Minimization algorithm (CFR) (Zinkevich et al. 2007), which allowed to outperform all winners of the AAAI 2006 Computer Poker Competition.

In 2017, the search for superhuman Poker playing bot has, is some sense, been concluded. Two bots: Libratus (Brown and Sandholm 2017) and DeepStack (Moravčík et al. 2017) were the first to consistently surpass professional human Poker players in 2-player No-Limit Texas Hold'em. Libratus is comprised of three parts: (1) precomputed "blueprint" strategies, (2) nested subgame solving and (3) self-improvement. For tasks (1) - (2) it employs Monte Carlo Counterfactual Regret Minimization (MCCFR) with Regret-Based Pruning (RBP) (Brown, Ganzfried, and Sandholm 2015). The self-improvement consists in an adaptive way of performing action abstractions based on the most frequent actions chosen by the opponent. DeepStack, briefly summarized in section 4, soundly transfers heuristic search methods from perfect to imperfect-information games. The system uses depth-limited search employs Deep Counter-

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

factual Value Network (DCVN) to learn the *Counterfactual Values* (CFVs) associated with different states in Poker.

Since 2017, the majority of the strongest Poker bots have been based on the CFR technique. Supremus (Zarick et al. 2020) introduced incremental modifications to the DeepStack approach, whereas Pluribus (Brown and Sandholm 2019) was an improved version of Libratus. However, in both these bots, the underlying SOTA idea, i.e. estimation of CFVs, remained unchanged. Later, DeepCFR (Brown et al. 2019) obviated the usage of abstraction and storing strategies in a tabular form by using neural networks to approximate the results of CFR reasoning.

Three exceptions from the above-mentioned research stream are ReBel (Brown et al. 2020), which combines Reinforcement Learning (RL) with search techniques, Alpha Holdem (Zhao et al. 2022), inspired by AlphaGo (Silver et al. 2016), which is an end-to-end Deep RL approach, and DREAM (Steinberger, Lerer, and Brown 2020), which is a model-free algorithm that uses an additional neural network to reduce the variance of the Monte Carlo CFR.

Lately, there appeared the Player Of Games algorithm (Schmid et al. 2021), which generalizes DeepStack approach to create an agent successful in both perfect and imperfect information games, by means of combining guided search, self-play learning, and game-theoretic reasoning.

In this paper, we consider the 2-player (a.k.a. *heads-up*) variant of No-Limit Texas Hold'em Poker, addressed by DeepStack, Libratus and their successors. Following Poker tradition, during the analysis of a situation, we distinguish one of the players, whose perspective is discussed, and will refer to him/her as a *hero*. For the complete set of game rules, please consult (Poker rules 2016).

Motivation

This work is motivated by certain limitations of a direct prediction of CFVs. Each predicted CFV is a multiplication of two factors: the probability of reaching a particular state in a game (called a *matchup*), which can be explicitly calculated from the input data, and the *expected value* (EV) of a payoff in that state, which is a complex function of the input data, hard to calculate.

To reduce the complexity of a CFVs prediction task, card abstraction techniques are applied that group hands of similar strength (i.e. the probability of winning against a random hand of the opponent). We observed that Poker hands of similar strength have similar EVs, whereas multiplying them by probabilities introduces a noise-like modifier to the hand's values – making CFVs different even for similar-strength hands.

Values other than CFVs are also used in DeepCFR and DREAM algorithms, i.e. instead of using CFVs explicitly as the target of a neural network training, their scaled values are considered. While the above scaling stems from the usage of Monte Carlo variant of CFR, an independent but similar idea is the basis of our work.

We propose to focus on the prediction of EVs and to apply card abstraction techniques to EVs, regardless of the matchup modifier, which is used at a later stage, to unabstracted hands. This new target setting significantly simplifies the learning problem and leads to much more accurate CFVs estimation.

While the proposed modification may seem to be of a rather technical nature, it, in fact, presents a fundamentally different approach to the process of learning and estimating CFVs, since the distributions of the training signals much differ between DCVN and DEVN. The former estimates CFVs, which are affected by the probability of reaching a given game state, while training the latter relies on a direct EV estimation, regardless of the associated probabilities.

Contribution

The main contribution of this work is fivefold.

- We propose a new approach (DEVN) to the problem of estimating Countefactual Values in Poker (and potentially in other related contexts / tasks) that relies on prediction of Expected Values and their subsequent multiplication by the matchup values.
- The efficacy of the proposed method is proven by its comparison with the DCVN approach, which is considered the state-of-the-art (SOTA) approach in the literature (Moravčík et al. 2017; Zarick et al. 2020; Brown and Sandholm 2019), in 7 experimental setups described in detail in section 6.
- The error introduced by applying card abstraction to EVs (in DEVN) is much smaller than in the case of CFVs (in DCVN). We show experimentally that DEVN lowers the theoretical error bound of estimating CFVs relatively by $29.05 - 31.83\%$ using an *encoding error* metric, introduced in section 6, thus leaving room for further improvement of results. Furthermore, the experiments show the relative advantage of $9.18 - 15.70\%$ of DEVN over DCVN with *unbucketed loss* metric, introduced in section 6. This practical result further cements the idea that using card abstractions in DEVN finally leads to more accurate CFVs predictions than in the case of DCVN.
- The experiments without using card abstraction show the relative advantage of $3.37 - 8.39\%$ of DEVN over DCVN, indicating that predicting EVs is generally a better-suited learning task than CFVs prediction.
- We show that unbucketed loss of 2-layer DEVNs is generally not worse than that of 7-layer DCVNs when card abstraction is used, and that the relative gain from expanding the network by adding additional layers is higher for DEVN than for DCVN.

2 Preliminaries

Game-Theoretic Basics Poker is an imperfect-information *extensive-form game* (Zinkevich et al. 2007), which means that it can be represented in the form of a game tree, using the concept of *information sets* (Karlin and Peres 2017; Frank and Basin 1998). An information set (or *infoset* for short) is a set of states in which the acting player could be at the moment, and which are indistinguishable for him/her. This ambiguity is a direct consequence of an imperfect-information nature of the game, where part of the state description (e.g. the opponents' hands in Poker) is

unknown to the player. Consequently, when performing an action in a given infoset the player needs to consider possible realizations of the infoset in terms of fully-described game states.

Although we consider a 2-player variant of Poker, in what follows the underlying concepts are introduced in n-player setting. Their transformation to a 2-player case is straightforward. Let's denote by P the set of players and by $\mathcal{I}_i$ a family of infosets in which player $i \in P$ is an active player, i.e. it is his/her turn to perform an action.

Each node in the game tree represents an infoset $I \in \mathcal{I}_i$ which is a set of *histories* $h \in H$. We overload the definition of I and say that $I(h) \in \mathcal{I}_{p(h)}$ is the infoset containing history h. Each non-terminal history h corresponds to player $p(h) \in P$ who is an active player in the state corresponding to $I(h)$. In each state corresponding to terminal history $h_t \in Z \subset H$ each player k obtains utility $u_k(h_t)$.

In Texas Hold'em, terminal histories are showdowns and folds. Each history contains information about particular hands of the players, public cards on the board, and a betting history. Therefore, the infoset $I(h)$ is a set of histories with a particular betting history, particular board and private cards of player $p(h)$ (identical to those in h), but with any possible combination of private cards held by other players (assuming particular private cards of player $p(h)$).

Texas Hold'em has a **zero-sum property**, i.e. for any terminal history h_t, $\sum_{i \in P} u_i(h_t) = 0$, because a pot consists only of players' money and the gain of one player is a loss of all the others.

A player's strategy $\sigma_i \in \Sigma$ is a function that assigns probabilities to all legal actions in all infosets. A strategy profile σ is a function that assigns a strategy to each player.

Nash Equilibrium – NE (Nash 1951; Zinkevich et al. 2007) is a strategy profile in which none of the players can increase his/her expected value by changing his/her strategy, if other players would not change theirs.

The NE $\sigma = (\sigma_1, \ldots, \sigma_n)$ can be defined as follows:

$$\begin{cases} u_1(\sigma) \geq \max\limits_{\sigma'_1 \in \Sigma_1} u_1((\sigma'_1, \sigma_2, \ldots, \sigma_n)) \\ \ldots \\ u_n(\sigma) \geq \max\limits_{\sigma'_n \in \Sigma_n} u_n((\sigma_1, \sigma_2, \ldots, \sigma'_n)) \end{cases} \quad (1)$$

Let $b_i(\sigma)$ be the payoff of player i in strategy σ. The *best response strategy* $br_i(\sigma)$ is the best strategy that player i can adopt against other players when they play strategies from σ. It stems immediately from the NE definition that $br_i(\sigma) = b_i(\sigma)$ when σ is the NE profile.

Exploitability is a popular metric in Poker that calculates the distance from the NE. Let $b_1(\sigma)$ and $b_2(\sigma)$ be best response utility values for the players, In 2-player variant Exploitability is defined as follows:

$$\epsilon_\sigma = (b_1(\sigma) + b_2(\sigma))/2 \quad (2)$$

Intuitively, ϵ_σ tells how much a given strategy profile would lose against a perfect opponent, if the player would apply the best response strategy and would play as the first player in half of the games (and as the second player in the other half). The exploitability of NE is equal to 0.

While playing the NE profile guarantees that a player cannot be exploited, it does not assure obtaining maximal utility (payoff). *Exploiting* is a process of derivation from NE strategy to maximize the player's utility. Knowing that the opponent plays according to some strategy σ and assuming he/she will not change it, the best answer strategy is $br(\sigma)$. However, playing according to strategy $br(\sigma)$ is exploitable itself, since the opponent can change his/her strategy to increase his/her utility. This is why professional Poker players aim to learn the GTO strategy as a non-exploitable baseline.

3 Counterfactual Regret (CFR) Minimization

Regret of a given action a is defined as a difference between the payoff of taking a and the payoff of the action actually taken. Hart and Mas-Colell (2000) proved that minimizing the cumulative regret for all players leads to playing strategies that converge to correlated equilibrium. However, due to the complexity of calculating the regret, a number of approximate regret minimization algorithms have been proposed (Hallak, Mertikopoulos, and Cevher 2021).

CFR Minimization (Zinkevich et al. 2007) is a family of methods that extend the base regret minimization methods with two ideas: (1) taking into account the probability of reaching a certain information set and (2) iterative and multi-step procedure to calculate the future regret and back-propagate it accordingly to information sets located earlier in the planned sequence. Zinkevich et al. (2007) have shown that CFR are of both space and time linear complexity w.r.t. the number of information sets.

Certain CFR variants are used by the top Poker bots including CFR+ in Cepherus (Bowling et al. 2017), MCCRM in Libratus (Brown and Sandholm 2017), and DCVN in DeepStack (Moravčík et al. 2017). CFR methods are also applied to other imperfect-information games, e.g. CFR-D in Leduc Hold'em (Burch, Johanson, and Bowling 2014) or MCCRM in Bluff (Burch et al. 2012).

Counterfactual Value (CFV) is a notion used in CFR. $CFV_i(\sigma, I)$ is equal to the expected payoff of player i at infoset I weighted by the probability of reaching I assuming that he/she plays according to the given strategy profile σ and aims to reach I. Following (Johanson et al. 2012), $CFV_i(\sigma, I)$ is defined as follows:

$$CFV_i(\sigma, I) = \sum_{h \in H_I} u_i(h) * \pi^\sigma_{-i}(h[I]) * \pi^\sigma(h[I], h) \quad (3)$$

where i - is the player; H_I - is a set of terminal histories passing through I; $h[I]$ is a prefix of h contained in I; $\pi^\sigma_{-i}(h)$ is the probability of reaching history h given strategy profile σ, assuming that the probability of player i to choose actions that lead to h is 1; $\pi^\sigma(h, h')$ is the probability of reaching history h' using strategy profile σ and given that state h was achieved (in particular $\pi^\sigma(h, h) = 1$).

Expected Value and Matchup The expected value (EV) of a payoff of player i at information set I is defined as follows:

$$EV_i(\sigma, I) = \frac{\sum_{h \in H_I} u_i(h) * \pi^{\sigma}_{-i}(h[I]) * \pi^{\sigma}(h[I], h)}{\sum_{h \in H_I} \pi^{\sigma}_{-i}(h[I])} \quad (4)$$

where the denominator is usually referred to as $matchup$. Precisely, $matchup_i(\sigma, I)$ denotes the joint probability that the opponents of player i will reach infoset I:

$$\sum_{h \in H_I} \pi^{\sigma}_{-i}(h[I]) = matchup_i(\sigma, I) \quad (5)$$

Please observe that combining (3), (4) and (5) leads to (6):

$$CFV_i(\sigma, I) = EV_i(\sigma, I) \cdot matchup_i(\sigma, I) \quad (6)$$

The above observation forms the basis for the CFR algorithm modification proposed in section 5.

4 DeepStack, Supremus and Deep Counterfactual Value Networks (DCVNs)

DeepStack (Moravčík et al. 2017) is a groundbreaking achievement in computer Poker. In short, the system performs CFR minimization via continuous re-solving and depth-limited search at every decision point in the game. In addition, it employs bucketing of player hands, which is an established abstraction method for imperfect information games (Johanson et al. 2013). A bucket denotes a group of player's hands that are functionally similar. The authors revealed that 1000 buckets were generated using k-means clustering, however, they did not make the generated buckets publicly available.

Supremus (Zarick et al. 2020) is essentially an up-scaled reimplementation of DeepStack. It uses much more training data, which is also finer-grained (due to less abstracted action space), and more iterations. In addition, it introduces a new CFR variant called CFR-D continual resolving. Finally, Supremus is implemented as an end-to-end GPU approach.

The novelty of the method proposed in this paper concerns one of fundamental aspects of DeepStack's and Supremus' architectures – the process of computing CFVs. In order to explain the proposed idea, the original approach to calculating CFVs is presented below, followed by a detailed description of the proposed modified method in section 5.

It is worth to underline that the problem of CFVs calculation extends beyond the game of Poker and the proposed method can also be applied in other domains, e.g. cybersecurity (Cotae and Reindorf 2021), resource allocation (Keith and Ahner 2021), or econometrics (Pesaran and Smith 2016)

Deep Counterfactual Value Networks

DCVNs are deep neural networks trained to predict the counterfactual values (eq. 3).

The DCVN Pipeline (DeepStack and Supremus)

1. The game state is defined by the the pot size, *public cards* and a *range* of each player (i.e., a probability distribution of having a certain hand by a player).
2. For all players, the ranges and public cards are clustered into buckets. Both DeepStack and Supremus use 1000 buckets, created using Potential Aware Card Abstraction (Ganzfried and Sandholm 2014). A vector of bucket ranges and the normalized pot size (divided by the stack size) serve as the DCVN input.
3. DCVN is a feedforward neural network with 7 fully connected hidden layers, each of them consisting of 500 neurons that use Parametric ReLU activation function.
4. The output from the network is a distribution of values over buckets, which are transformed by a single outer layer to satisfy the zero-sum property. Such transformed values are CFVs defined for buckets.
5. Finally, an inverse bucketing is performed to obtain the CFVs for specific players' hands. Such a vector of CFVs is then used in look-ahead search.

5 Deep Expected Value Networks (DEVNs)

In this work, we propose a different approach to obtaining reliable CFVs, required by the CFR algorithms. Instead of using DCVNs, we introduce DEVNs (i.e. neural networks that predict EVs) which outputs, multiplied by *matchups* (eq. 6), serve as CFVs approximations.

The idea undeprining the use of DEVNs differs from that of DCVNs. When predicting CFVs for a particular state and hero's cards, one asks *how valuable would it be for the hero to try reaching this state*, whereas when predicting EVs, the question is *how valuable is a given state assuming that the opponent would enter it*.

The probability of entering the state by the opponent, given a particular hero hand, is provided by the *matchup* value. The *matchup* for a given hero hand is a probability that an independently selected opponent's hand will not be blocked by the hero's hand, using opponent's range and relying on the fact that a particular card cannot appear more than once. The blocking relation is represented as a 1326 x 1326 matrix with *ones* if a hand in the column does not block the hand in the row, and *zeroes* otherwise. Assuming that ranges are normalized, calculating matchups is simply a multiplication of this matrix by the opponent's range vector.

DEVN Pipeline is similar to that of DCVN, except that:

1. There is no outer network to satisfy the zero-sum property during training (step 4 of DCVN). However, this property can be satisfied at prediction time as we discuss in Supplementary Material.
2. The final output (step 5 of DCVN) contains EVs instead of CFVs, for specific players' hands.
3. Due to step 2, to obtain CFVs, a calculation of matchups and their multiplication by EVs (eq. 6) is performed.

Training Procedure

The training process is performed similarly to DCVN, except that EVs (instead of CFVs) have to be provided as the *ground truth*. Depending on the implementation, EVs can be obtained aside with CFVs during solving, or can be calculated from CFVs by dividing them by the matchups.

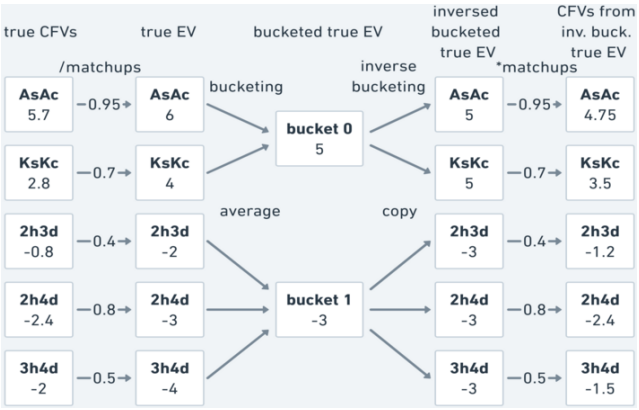


Figure 1: The processes of EVs bucketing and inverse bucketing performed on a subset of hands. During the bucketing, values for hands in a specific bucket are averaged. During inverse bucketing, the value for a hand assigned to a bucket is a copy of the bucket value. To transform EVs into CFVs, EVs are multiplied by matchups. Presented values are random. $encoding\_error = \sqrt{(5.7-4.75)^2+\ldots+(-2-(-1.5))^2} = 1.34$.
A similar visualization, picturing CFVs bucketing process, is presented in Supplementary Material.

Similarly to DeepStack and Supremus - a feedforward neural network with PReLU is used, however, the tests encompass various numbers of layers and neurons in a single layer. The dataset (see section 6) is split in the proportion 90/10 between training and validation sets. The training is performed for 400 epochs on batches of size 24000 samples, using Adam optimizer with learning rate equal to $3 \cdot 10^{-4}$.

6 Experiments

A comparison of DEVN with DCVN is performed in isolation from the methods of comparing Poker bots, since they are heavily affected by several external factors stemming from a particular implementation. Furthermore, our focus is on the CFVs prediction rather than creation of a full Poker bot. Therefore, we directly compare the validation losses (defined further in this section) for DEVN and DCVN.

Datasets Two different datasets of sizes $10M$ and $8.5M$, resp. were generated using the same approach as in DeepStack, In short, each sample was taken from a randomly generated turn subgame. The states on turn were solved using the regular CFR. This way, we are able to solve them until assumed exploitability of discovered strategies. To assure that the potential inaccuracy of CFVs in the datasets will not be a source of error, we performed preliminary tests to find a low enough threshold on the exploitability such that RMSE between vectors of found CFVs and ground truth CFVs (calculated until exploitability of 0.01% of the pot) will be with probability 0.9 smaller than 0.5, and with probability 0.98 smaller than 1. Please consult the DeepStack paper (Moravčík et al. 2017) for a detailed explanation of the data preparation process. CFVs are in the range [-0.5, 10], as they are normalized by the pot.

Bucketings Two different bucketings were used in the experiments to avoid a potential bias from a particular bucketing. As in the case of DeepStack, the method of generation was Potential Aware Card Abstraction (Ganzfried and Sandholm 2014).

Experiment Settings Both neural networks, i.e, DEVN and DCVN, resp. used in the experiments are composed of 7, 5, 3 or 2 layers. In each case the following 5 experimental settings are applied:
I: Dataset 1, 1000 buckets, 500 neurons in layer – a setup used by DeepStack and Supremus (with 7 layers);
II: Dataset 1, 1000 buckets, 1024 neurons in layer – this experiment was run twice to test stability;
III: Dataset 1, 2000 buckets, 1024 neurons in layer – to test scalability, I $\rightarrow$ II $\rightarrow$ III;
IV: Dataset 2, 1000 buckets, 1024 neurons in layer – to test repeatability between datasets, II $\rightarrow$ IV;
V: Dataset 2, 2000 buckets, 1024 neurons in layer – to test scalability and between-sets repeatability, III $\rightarrow$ V.
To further test the strength of the proposed approach, we created two more settings in which no bucketing is used. Instead, the network takes an information about the public cards as an input, in the form of one-hot encoding. The results are presented in Supplementary Material[1].

Methods of Comparison

The outputs of the DEVN and DCVN pipelines are CFVs which can be compared directly. At the same time, a direct comparison of the training losses of neural networks in both pipelines is not possible since the network outputs are different. Therefore, we use two additional loss measures: the *unbucketed loss* and the *encoding error*, both of which can be directly compared between the two approaches.

Encoding Error of a bucketing (Hopner and Loza Mencía 2018) represents the minimal theoretical error that DCVN/DEVN can achieve using a given bucketing. It can be viewed as an unbucketed loss of a perfectly performing neural network (which has the training loss equal to 0 and predicts bucketed values without any error). This error is calculated on the validation dataset using RMSE. It can be compared with unbucketed loss within each pipeline, and also between DEVN and DCVN. An example of encoding error calculation within the DEVN pipeline is shown in Fig. 1.

Bucketed Loss / Training Loss represents the error of a neural network on the training and validation sets and is defined as a difference between the bucketed ground truth and predicted values. Following (Moravčík et al. 2017) it is calculated using HuberLoss (Hopner and Loza Mencía 2018) with parameter $\delta = 1$.

Unbucketed Loss is the error of the entire DCVN/DEVN pipeline, including the error of the neural network and the error introduced by the bucketing. It represents the Root Mean Squared (RMS) difference between the ground truth CFVs and the neural network prediction after the inverse bucketing

[1] https://github.com/jwolosiuk/dont-predict-cfvs-predict-evs-instead (additional results and code)

Dataset	Buckets	DCVN	DEVN	Rel. impr.
1	1000	2.782	1.944	30.11%
1	2000	2.634	1.796	31.83%
2	1000	4.006	2.842	29.05%
2	2000	3.887	2.726	29.86%

Table 1: Encoding error (EE) for DCVN and DEVN pipelines for each unique pair (dataset, bucketing). Note that EE does not depend on the neural network used. The last column shows the relative improvement of DEVN over DCVN.

Method	Training loss	Validation loss
EVs (2 layers)	$0.000796 \pm 3e{-6}$	$0.001270 \pm 4e{-6}$
EVs (3 layers)	$0.000628 \pm 8e{-7}$	$0.001007 \pm 4e{-6}$
EVs (5 layers)	$0.000534 \pm 2e{-6}$	$0.000830 \pm 4e{-6}$
EVs (7 layers)	$0.000507 \pm 3e{-6}$	$0.000780 \pm 7e{-7}$
CFVs (2 layers)	$0.000746 \pm 1e{-6}$	$0.001093 \pm 5e{-6}$
CFVs (3 layers)	$0.000612 \pm 1e{-6}$	$0.000891 \pm 3e{-6}$
CFVs (5 layers)	$0.000540 \pm 2e{-7}$	$0.000761 \pm 9e{-9}$
CFVs (7 layers)	$0.000516 \pm 2e{-6}$	$0.000726 \pm 3e{-6}$

Table 2: Bucketed losses $\pm$ st.dev.in setting II. The results for other settings are similar (see Supplementary Material).

(multiplied by matchups in the case of DEVN) as a percent of the pot size. As discussed in (Moravčík et al. 2017) if this loss is low, the resulting strategies approximate NE well. The unbucketed loss is calculated every 10 epochs and the model with the lowest loss on the validation set is picked from each experiment run.

7 Results

Encoding Error values are presented in Table 1 for both datasets and both bucketings, and in Fig. 2b (Datatset 1) and Fig. 2c (Dataset 2). DEVN managed to lower the theoretical minimum error bound by $29.05\% - 31.83\%$ compared to the DCVN approach. It should be noted that, in practice, obtaining the unbucketed loss equal to or even close to the encoding error value may not be possible due to certain loss of information caused by the bucketing of the neural network input. Nevertheless, such a substantial gain creates relatively large room for accuracy improvements.

Bucketed (Training) Losses of the DEVN and DCVN network, resp. are shown in Table 2 and in Fig. 2a (DEVN). For both networks, a steep decline of both losses and the fact that the losses are close to each other suggest that neural networks learn their respective tasks effectively. While the values for both methods cannot be directly compared, training losses can be compared with the validation losses separately for each of the two methods.

Unbucketed Loss results are the most important in the context of the core claim of the paper, i.e. the advantage of EVs prediction instead of prediction of CFVs. The experiments were performed for five settings I-V introduced earlier in the previous section. The results are presented in Table 3 and Figs. 2b and 2c. Predicting EVs directly leads to

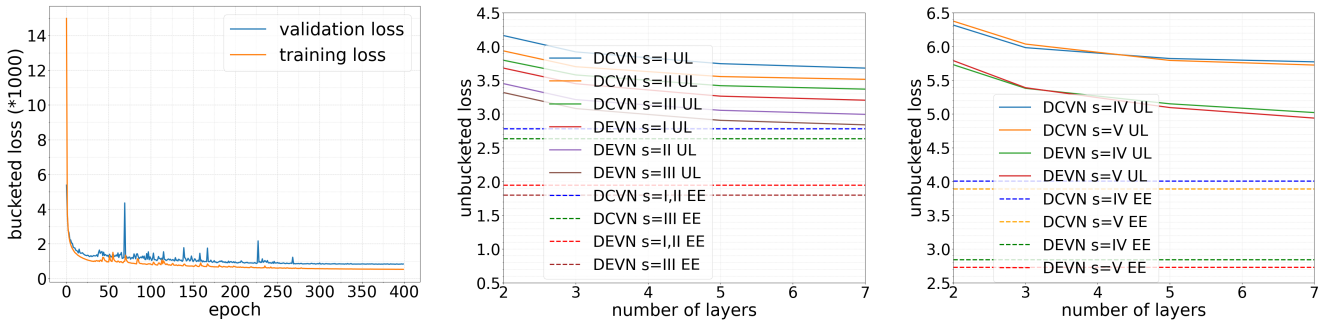
Size	DCVN	DEVN	Rel. impr.
I: Dataset 1; 1000 buckets; 500 neurons per layer			
2	4.161	3.679	11.59%
3	3.919	3.448	12.02%
5	3.745	3.264	12.85%
7	3.679	3.204	12.92%
EE	2.782	1.944	30.11%
II: Dataset 1; 1000 buckets; 1024 neurons per layer			
2	3.934 ± 0.010	3.449 ± 0.000037	12.33%
3	3.700 ± 0.001	3.212 ± 0.006	13.19%
5	3.554 ± 0.001	3.054 ± 0.004	14.07%
7	3.513 ± 0.002	2.994 ± 0.006	14.77%
EE	2.782	1.944	30.11%
III: Dataset 1; 2000 buckets; 1024 neurons per layer			
2	3.795	3.317	12.61%
3	3.578	3.080	13.92%
5	3.419	2.905	15.02%
7	3.368	2.839	15.70%
EE	2.634	1.796	31.83%
IV: Dataset 2; 1000 buckets; 1024 neurons per layer			
2	6.316	5.729	9.29%
3	5.982	5.380	10.07%
5	5.821	5.150	11.53%
7	5.772	5.021	13.02%
EE	4.006	2.842	29.05%
V: Dataset 2; 2000 buckets; 1024 neurons per layer			
2	6.375	5.790	9.18%
3	6.035	5.389	10.71%
5	5.793	5.095	12.05%
7	5.724	4.939	13.73%
EE	3.887	2.726	29.86%

Table 3: Unbucketed loss of DCVN and DEVN for various parameter settings. Experiments in setting II were run twice. The remaining results are based on a single run. EE denotes the encoding error (copied from Tab. 1), i.e. the minimum theoretical error bound, for each setting, respectively.

lower losses in each setting and for each considered numbers of layers. In general, it can be observed that the bigger the network, the better the relative improvement. The relative improvement varies from 9.18% (setting V with 2 hidden layers) to 15.7% for (setting III with 7 hidden layers).

In Figure 2b, it can be seen that doubling the number of neurons per layer (from 500 to 1024) improves the results for both DCVN and DEVN. Likewise, doubling the number of buckets (from 1000 to 2000) ameliorates the results in both cases. At the same time, however, DEVN with only 500 neurons per layer and 1000 buckets (setting I) performs better than DCVN with 1024 neurons per layer and 2000 buckets (setting III), for each tested number of layers.

In Dataset 2 (Figure 2c) the gain from using DEVN over DCVN is also clear, although the improvement from using 2000 buckets (V) instead of 1000 buckets (IV) is smaller. This difference is attributed to different distributions of gen-

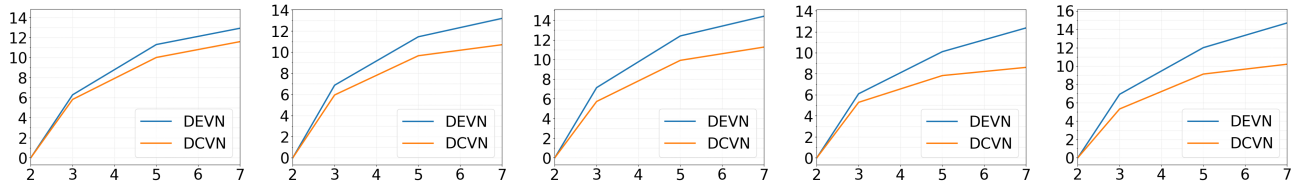


(a) Bucketed training and validation loss, resp. for a neural network predition of EVs (DEVN) with 5 layers in setting II

(b) Encoding errors (EE) and unbucketed losses (UL) for neural networks trained on Dataset 1. s stands for setting

(c) Encoding errors (EE) and unbucketed losses (UL) for neural networks trained on Dataset 2. s stands for setting

Figure 2: (a) DEVN learning curves (bucketed loss). (b), (c) DEVN and DCVN unbucketed losses and encoding errors



(a) Setting I (b) Setting II (c) Setting III (d) Setting IV (e) Setting V

Figure 3: Unbucketed loss for each neural network size relative to the 2-layer network, for DEVN and DCVN. The X axis denotes the number of layers, and the Y axis relative improvement of unbucketed loss in %.

erated public cards in Dataset 2 and the respective bucketing. Moreover, in 4 out of 5 cases, the loss of EV-predicting network with 2 layers is smaller or equal to the CFV-predicting network with 7 layers, which shows the DEVN potential to use simpler predictive models effectively.

Not only does DEVN better incorporate smaller networks, it also gains more when additional layers are used. The unbucketed loss gain, relative to the 2-layer network, for each neural network size and for both DEVN and DCVN, is illustrated in Figs. 3a- 3e, resp. for settings I-V. In all settings, the plot for DEVN is above the DCVN one, indicating that DEVN gains more from increasing the network size.

8 Conclusions and Future Work

The paper presents a new method of calculating CFVs that are used in CFR algorithms in game domain. The proposed approach consists of two steps. First, EVs are predicted using Deep Expected Value Networks - DEVN (introduced in Section 5) and then the obtained EVs are multiplied by matchups, which are relatively easy to calculate. DEVN is compared with the SOTA method that predicts CFVs directly using Deep Counterfactual Value Networks - DCVN.

Application of DEVN improves theoretical lower bound of CFVs prediction error by $29.05 - 31.83\%$ compared to the DCVN pipeline. This gain is attributed to the card abstraction techniques employed by both DEVN and DCVN, i.e. bucketing of the input and the output, and inverse bucketing. Card abstraction techniques group together hands of similar strength and average their EVs/CFVs values, resp., which causes certain information loss. Lower encoding error of DEVN than DCVN indicates that averaging of EVs is more effective than CFVs averaging.

The proposed modification allows to achieve significantly higher – between 9.18% and 15.7% – prediction accuracy (unbucketed loss), in relation to the SOTA approach. These results further back up the above conclusion (drawn from the analysis of the encoding error) about higher efficacy of applying card abstractions to EVs than CFVs.

In the case of bucketing, 2-layer DEVN achieves not worse results than 7-layer DCVN, which shows that the card abstraction error in CFVs cannot be mitigated by simply employing bigger networks (compared to the EVs prediction).

The $3.37 - 8.39\%$ improvement of DEVN over DCVN in the experiments without card abstraction (see Supplementary Material) shows that the increased performance of DEVN is not only attributed to the higher efficacy when employing card abstraction techniques to EVs instead of CFVs, but also to the fact that prediction of EVs is generally an easier task, compared to CFVs prediction.

The fact that DCVN pipeline does not improve meaningfully beyond 7 layers and 500 neurons per layer, makes the above conclusion even more meaningful.

In summary, in DCVN, the estimation of CFVs is affected by the probability of reaching a given state, which has a negative impact on generalization. Consequently, when predicting CFVs, our approach (DEVN) separates these two concepts, i.e. robust calculation of probabilities (matchups) and effective prediction of EVs.

Acknowledgments

JW was supported by Deepsolver. MŚ was supported by funding from Smart Growth Operational Programme 2014-2020, financed by European Regional Development Fund under GameINN project POIR.01.02.00-00-0207/20, operated by National Centre for Research and Development in Poland.

References

Bard, N.; Hawkin, J.; Rubin, J.; and Zinkevich, M. 2013. The annual computer poker competition. *AI Magazine*, 34(2): 112–112.

Billings, D. 1995. *Computer poker*. Master's thesis, University of Alberta.

Billings, D.; Burch, N.; Davidson, A.; Holte, R.; Schaeffer, J.; Schauenberg, T.; and Szafron, D. 2003. Approximating Game-Theoretic Optimal Strategies for Full-Scale Poker. In *IJCAI*, volume 3, 661.

Billings, D.; Davidson, A.; Schaeffer, J.; and Szafron, D. 2002. The challenge of poker. *Artificial Intelligence*, 134(1-2): 201–240.

Billings, D.; Papp, D.; Schaeffer, J.; and Szafron, D. 1998. Opponent Modeling in Poker. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference (AAAI '98, IAAI '98)*, 493–499.

Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2017. Heads-up limit hold'em poker is solved. *Communications of the ACM*, 60(11): 81–88.

Broeck, G. V. d.; Driessens, K.; and Ramon, J. 2009. Monte-Carlo tree search in poker using expected reward distributions. In *Asian Conference on Machine Learning*, 367–381. Springer.

Brown, N.; Bakhtin, A.; Lerer, A.; and Gong, Q. 2020. Combining Deep Reinforcement Learning and Search for Imperfect-Information Games. arXiv:2007.13544.

Brown, N.; Ganzfried, S.; and Sandholm, T. 2015. Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit Texas Hold'em agent. In *Workshops at the twenty-ninth AAAI conference on artificial intelligence*.

Brown, N.; Lerer, A.; Gross, S.; and Sandholm, T. 2019. Deep Counterfactual Regret Minimization. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 793–802. PMLR.

Brown, N.; and Sandholm, T. 2017. Libratus: The Superhuman AI for No-Limit Poker. In Sierra, C., ed., *IJCAI*, 5226–5228. ijcai.org. ISBN 978-0-9992411-0-3.

Brown, N.; and Sandholm, T. 2019. Superhuman AI for multiplayer poker. *Science*, 365: eaay2400.

Burch, N.; Johanson, M.; and Bowling, M. 2014. Solving Imperfect Information Games Using Decomposition. arXiv:1303.4441.

Burch, N.; Lanctot, M.; Szafron, D.; and Gibson, R. 2012. Efficient Monte Carlo Counterfactual Regret Minimization in Games with Many Player Actions. In Pereira, F.; Burges, C.; Bottou, L.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Cotae, P.; and Reindorf, N. E. A. 2021. Using counterfactual regret minimization and Monte Carlo tree search for cybersecurity threats. In *2021 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 1–6. IEEE.

Davidson, A.; Billings, D.; Schaeffer, J.; and Szafron, D. 2000. Improved opponent modeling in poker. In *International Conference on Artificial Intelligence, ICAI'00*, 1467–1473.

Frank, I.; and Basin, D. 1998. Search in games with incomplete information: A case study using Bridge card play. *Artificial Intelligence*, 100(1-2): 87–123.

Ganzfried, S.; and Sandholm, T. 2014. Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. *Proceedings of the National Conference on Artificial Intelligence*, 1: 682–690.

Hallak, N.; Mertikopoulos, P.; and Cevher, V. 2021. Regret minimization in stochastic non-convex learning via a proximal-gradient approach. In *International Conference on Machine Learning*, 4008–4017. PMLR.

Hart, S.; and Mas-Colell, A. 2000. A Simple Adaptive Procedure Leading to Correlated Equilibrium. *Econometrica*, 68(5): 1127–1150.

Heinrich, J.; and Silver, D. 2014. Self-play Monte-Carlo tree search in computer poker. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Hopner, P.; and Loza Mencía, E. 2018. Analysis and Optimization of Deep Counterfactual Value Networks. *KI 2018: Advances in Artificial Intelligence*, 305–312.

Johanson, M.; Bard, N.; Lanctot, M.; Gibson, R. G.; and Bowling, M. 2012. Efficient Nash equilibrium approximation through Monte Carlo counterfactual regret minimization. In van der Hoek, W.; Padgham, L.; Conitzer, V.; and Winikoff, M., eds., *AAMAS*, 837–846. IFAAMAS.

Johanson, M.; Burch, N.; Valenzano, R.; and Bowling, M. 2013. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 271–278.

Karlin, A. R.; and Peres, Y. 2017. *Game Theory, Alive*, volume 101. American Mathematical Soc.

Keith, A.; and Ahner, D. 2021. Counterfactual regret minimization for integrated cyber and air defense resource allocation. *European Journal of Operational Research*, 292(1): 95–107.

Moravčík, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. DeepStack: Expert-level Artificial Intelligence in heads-up no-limit Poker. *Science*, 356(6337): 508–513.

Nash, J. 1951. Non-cooperative Games. *Annals of Mathematics*, 54(2): 286–295.

Pesaran, M. H.; and Smith, R. P. 2016. Counterfactual analysis in macroeconometrics: An empirical investigation into the effects of quantitative easing. *Research in Economics*, 70(2): 262–280.

Poker rules. 2016. *Texas Hold'em Game Rulebook*. https://oag.ca.gov/sites/all/files/agweb/pdfs/gambling/BGC_texas.pdf (Accessed: 2022-08-14).

Rubin, J.; and Watson, I. 2011. Computer poker: A review. *Artificial Intelligence*, 175(5): 958–987. Special Review Issue.

Schauenberg, T. C. 2006. *Opponent Modelling and Search in Poker*. Master's thesis, University of Alberta.

Schmid, M.; Moravcik, M.; Burch, N.; Kadlec, R.; Davidson, J.; Waugh, K.; Bard, N.; Timbers, F.; Lanctot, M.; Holland, Z.; et al. 2021. Player of Games. *arXiv preprint arXiv:2112.03178*.

Schweizer, I.; Panitzek, K.; Park, S.-H.; and Fürnkranz, J. 2009. An exploitative Monte-Carlo poker agent. In *Annual Conference on Artificial Intelligence*, 65–72. Springer.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489.

Steinberger, E.; Lerer, A.; and Brown, N. 2020. DREAM: Deep regret minimization with advantage baselines and model-free learning. *arXiv preprint arXiv:2006.10410*.

Świechowski, M.; Godlewski, K.; Sawicki, B.; and Mańdziuk, J. 2023. Monte Carlo Tree Search: a review of recent modifications and applications. *Artificial Intelligence Review*, 56: 2497–2562.

Zarick, R.; Pellegrino, B.; Brown, N.; and Banister, C. 2020. Unlocking the Potential of Deep Counterfactual Value Networks. arXiv:2007.10442.

Zhao, E.; Yan, R.; Li, J.; Li, K.; and Xing, J. 2022. AlphaHoldem: High-Performance Artificial Intelligence for Heads-Up No-Limit Poker via End-to-End Reinforcement Learning. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, 4689–4697.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2007. Regret Minimization in Games with Incomplete Information. *Advances in neural information processing systems*, 20.