Deep Counterfactual Regret Minimization

Noam Brown * 1 2  Adam Lerer * 1  Sam Gross 1  Tuomas Sandholm 2 3

Abstract

Counterfactual Regret Minimization (CFR) is the leading framework for solving large imperfect-information games. It converges to an equilibrium by iteratively traversing the game tree. In order to deal with extremely large games, abstraction is typically applied before running CFR. The abstracted game is solved with tabular CFR, and its solution is mapped back to the full game. This process can be problematic because aspects of abstraction are often manual and domain specific, abstraction algorithms may miss important strategic nuances of the game, and there is a chicken-and-egg problem because determining a good abstraction requires knowledge of the equilibrium of the game. This paper introduces Deep Counterfactual Regret Minimization, a form of CFR that obviates the need for abstraction by instead using deep neural networks to approximate the behavior of CFR in the full game. We show that Deep CFR is principled and achieves strong performance in large poker games. This is the first non-tabular variant of CFR to be successful in large games.

1. Introduction

Imperfect-information games model strategic interactions between multiple agents with only partial information. They are widely applicable to real-world domains such as negotiations, auctions, and cybersecurity interactions. Typically in such games, one wishes to find an approximate equilibrium in which no player can improve by deviating from the equilibrium.

The most successful family of algorithms for imperfect-information games have been variants of Counterfactual Regret Minimization (CFR) (Zinkevich et al., 2007). CFR is an iterative algorithm that converges to a Nash equilibrium in two-player zero-sum games. Forms of tabular CFR have been used in all recent milestones in the benchmark domain of poker (Bowling et al., 2015; Moravčík et al., 2017; Brown & Sandholm, 2017) and have been used in all competitive agents in the Annual Computer Poker Competition going back at least six years. In order to deal with extremely large imperfect-information games, abstraction is typically used to simplify a game by bucketing similar states together and treating them identically. The simplified (abstracted) game is approximately solved via tabular CFR. However, constructing an effective abstraction requires extensive domain knowledge and the abstract solution may only be a coarse approximation of a true equilibrium.

In constrast, reinforcement learning has been successfully extended to large state spaces by using function approximation with deep neural networks rather than a tabular representation of the policy (deep RL). This approach has led to a number of recent breakthroughs in constructing strategies in large MDPs (Mnih et al., 2015) as well as in zero-sum perfect-information games such as Go (Silver et al., 2017; 2018). Importantly, deep RL can learn good strategies with relatively little domain knowledge for the specific game (Silver et al., 2017). However, most popular RL algorithms do not converge to good policies (equilibria) in imperfect-information games in theory or in practice.

Rather than use tabular CFR with abstraction, this paper introduces a form of CFR, which we refer to as Deep Counterfactual Regret Minimization, that uses function approximation with deep neural networks to approximate the behavior of tabular CFR on the full, unabstracted game. We prove that Deep CFR converges to an ε-Nash equilibrium in two-player zero-sum games and empirically evaluate performance in poker variants, including heads-up limit Texas hold’em. We show Deep CFR outperforms Neural Fictitious Self Play (NFSP) (Heinrich & Silver, 2016), which was the prior leading function approximation algorithm for imperfect-information games, and that Deep CFR is competitive with domain-specific tabular abstraction techniques.

1Equal contribution 1Facebook AI Research 2Computer Science Department, Carnegie Mellon University 3Strategic Machine Inc., Strategy Robot Inc., and Optimized Markets Inc. Correspondence to: Noam Brown <noamb@cs.cmu.edu>.

2. Notation and Background

In an imperfect-information extensive-form (that is, tree-form) game there is a finite set of players, \( P \). A node (or history) \( h \) is defined by all information of the current situation, including private knowledge known to only one player. \( A(h) \) denotes the actions available at a node and \( P(h) \) is either chance or the unique player who acts at that node. If action \( a \in A(h) \) leads from \( h \) to \( h' \), then we write \( h \cdot a = h' \). We write \( h \sqsupset h' \) if a sequence of actions leads from \( h \) to \( h' \). \( H \) is the set of all nodes. \( Z \subseteq H \) are terminal nodes for which no actions are available. For each player \( p \in P \), there is a payoff function \( u_p : Z \rightarrow \mathbb{R} \). In this paper we assume \( P = \{1, 2\} \) and \( u_1 = -u_2 \) (the game is two-player zero-sum). We denote the range of payoffs in the game by \( \Delta \).

Imperfect information is represented by information sets (infosets) for each player \( p \in P \). For any infoset \( I \) belonging to \( p \), all nodes \( h, h' \in I \) are indistinguishable to \( p \). Moreover, every non-terminal node \( h \in H \) belongs to exactly one infoset for each \( p \). We represent the set of all infosets belonging to \( p \) by \( I_p \). We call the set of all terminal nodes with a prefix in \( I \) as \( Z_I \) and we call the particular prefix \( z[I] \). We assume the game features perfect recall, which means if \( h \) and \( h' \) do not share a player \( p \) infoset then all nodes following \( h \) do not share a player \( p \) infoset with any node following \( h' \).

A strategy (or policy) \( \sigma(I) \) is a probability vector over actions for acting player \( p \) in infoset \( I \). Since all states in an infoset belonging to \( p \) are indistinguishable, the strategies in each of them must be identical. The set of actions in \( I \) is denoted by \( A(I) \). The probability of a particular action \( a \) in \( I \) is denoted by \( \sigma(I, a) \). We define \( \sigma_p \) to be a strategy for \( p \) in every infoset in the game where \( p \) acts. A strategy profile \( \sigma \) is a tuple of strategies, one for each player. The strategy of every infoset in the game where \( I \) in each of them must be identical. The set of actions in \( p \) of \( \sigma \) and the other players play according to \( \sigma_p \).

\[
\pi^p(h) = \Pi_{h' \sqsubset h} \sigma_p(h') \cdot (h', a) \text{ is called reach and is the probability \( h \) is reached if all players play according to } \sigma_p. \pi^p(h) \text{ is the contribution of } p \text{ to this probability. } \pi^{\sigma_p}(h) \text{ is the contribution of chance and other players according to } \sigma_p.
\]

For an infoset \( I \) belonging to \( p \), the probability of reaching \( I \) if \( p \) chooses actions leading toward \( I \) but chance and all other players other than \( p \) play according to \( \sigma_p \) is denoted by \( \pi_p^\sigma(I) = \sum_{h \in I} \pi^\sigma_p(h) \). For \( h \sqsubset z \), define \( \pi^\sigma(h \rightarrow z) = \Pi_{h' \sqsubset z, h' \sqsupset h} \sigma_p(h') \cdot (h', a) \).

A best response to \( \sigma_p \) is a player strategy \( BR(\sigma_p) \) such that \( u_p(BR(\sigma_p), \sigma_p) = \max_{\sigma_p} u_p(\sigma_p, \sigma_p) \). A Nash equilibrium \( \sigma^* \) is a strategy profile where everyone plays a best response: \( \forall p \), \( u_p(\sigma^*_p, \sigma^*_{-p}) = \max_{\sigma^*_p} u_p(\sigma^*_p, \sigma^*_{-p}) \) (Nash, 1950). The exploitability \( e(\sigma_p) \) of a strategy \( \sigma_p \) in a two-player zero-sum game is how much worse \( \sigma_p \) does versus \( BR(\sigma_p) \) compared to how a Nash equilibrium strategy \( \sigma^*_p \) does against \( BR(\sigma^*_p) \). Formally, \( e(\sigma_p) = u_p(\sigma^*_p, BR(\sigma^*_p)) - u_p(\sigma_p, BR(\sigma_p)) \). We measure total exploitability \( \sum_{p \in P} e(\sigma_p)^3 \).

2.1. Counterfactual Regret Minimization (CFR)

CFR is an iterative algorithm that converges to a Nash equilibrium in any finite two-player zero-sum game with a theoretical convergence bound of \( O(\frac{1}{\sqrt{T}}) \). In practice CFR converges much faster. We provide an overview of CFR below; for a full treatment, see Zinkevich et al. (2007). Some recent forms of CFR converge in \( O(\frac{1}{\sqrt{T}}) \) in self-play settings (Farina et al., 2019), but are slower in practice so we do not use them in this paper.

Let \( \sigma^t \) be the strategy profile on iteration \( t \). The counterfactual value \( v^\sigma(I) \) of player \( p = P(I) \) at \( I \) is the expected payoff to \( p \) when reaching \( I \), weighted by the probability that \( p \) would reached \( I \) if she tried to do so that iteration. Formally,

\[
v^\sigma(I) = \sum_{z \in Z_I} \pi_p(z|I)\pi^\sigma(z|I \rightarrow z)u_p(z)
\]

and \( v^\sigma(I, a) \) is the same except it assumes that player \( p \) plays action \( a \) at infoset \( I \) with 100% probability.

The instantaneous regret \( r^t(I, a) \) is the difference between \( P(I) \)’s counterfactual value from playing \( a \) vs. playing \( \sigma \) on iteration \( t \)

\[
r^t(I, a) = v^\sigma(I, a) - v^{\sigma^t}(I)
\]

The counterfactual regret for infoset \( I \) action \( a \) on iteration \( T \) is

\[
R_T(I, a) = \sum_{t=1}^{T} r^t(I, a)
\]

Additionally, \( R_T^L(I, a) = \max \{ R_T^L(I, a), 0 \} \) and \( R_T(I) = \max_{a} \{ R_T^L(I, a) \} \). Total regret for \( p \) in the entire game is \( R_T^p = \max_{\sigma_p} \sum_{t=1}^{T} (u_p(\sigma^t_p, a^t_p) - u_p(\sigma^*_p, a^*_p)) \).

CFR determines an iteration’s strategy by applying any of several regret minimization algorithms to each infoset (Littleston & Warmuth, 1994; Chaudhuri et al., 2009). Typically, regret matching (RM) is used as the regret minimization algorithm within CFR due to RM’s simplicity and lack of parameters (Hart & Mas-Colell, 2000).

In RM, a player picks a distribution over actions in an infoset in proportion to the positive regret on those actions. Formally, on each iteration \( t + 1 \), \( p \) selects actions \( a \in A(I) \)

---

Some prior papers instead measure average exploitability rather than total (summed) exploitability.
Appendix B for more details.

Vanilla CFR requires full traversals of the game tree, which (see Figure 4).

There exist a number of MCCFR variants (Gibson et al., 2012; Johanson et al., 2012; Jackson, 2017), but for this paper we focus specifically on the external sampling variant due to its simplicity and strong performance. In external-sampling MCCFR the game tree is traversed for one player at a time, alternating back and forth. We refer to the player who is traversing the game tree on the iteration as the traverser. Regrets are updated only for the traverser on an iteration. At infosets where the traverser acts, all actions are explored. At other infosets and chance nodes, only a single action is explored.

External-sampling MCCFR probabilistically converges to an equilibrium. For any \( \rho \in (0, 1] \), total regret is bounded by \( R_T^\rho \leq (1 + \frac{\rho^2}{\sqrt{\rho}}) |I| \Delta \sqrt{A} \sqrt{T} \) with probability \( 1 - \rho \).

3. Related Work

CFR is not the only iterative algorithm capable of solving large imperfect-information games. First-order methods converge to a Nash equilibrium in \( O(1/T) \) (Hoda et al., 2010; Kroer et al., 2018b,a), which is far better than CFR’s theoretical bound. However, in practice the fastest variants of CFR are substantially faster than the best first-order methods. Moreover, CFR is more robust to error and therefore likely to do better when combined with function approximation.

Neural Fictitious Self Play (NFSP) (Heinrich & Silver, 2016) previously combined deep learning function approximation with Fictitious Play (Brown, 1951) to produce an AI for heads-up limit Texas hold’em, a large imperfect-information game. However, Fictitious Play has weaker theoretical convergence guarantees than CFR, and in practice converges slower. We compare our algorithm to NFSP in this paper. Model-free policy gradient algorithms have been shown to minimize regret when parameters are tuned appropriately (Srinivasan et al., 2018) and achieve performance comparable to NFSP.

Past work has investigated using deep learning to estimate values at the depth limit of a subgame in imperfect-information games (Moravčík et al., 2017; Brown et al., 2018). However, tabular CFR was used within the subgames themselves. Large-scale function approximated CFR has also been developed for single-agent settings (Jin et al., 2017). Our algorithm is intended for the multi-agent setting and is very different from the one proposed for the single-agent setting.

Prior work has combined regression tree function approximation with CFR (Waugh et al., 2015) in an algorithm called Regression CFR (RCFR). This algorithm defines a number of features of the infosets in a game and calculates weights to approximate the regrets that a tabular CFR implementation would produce. Regression CFR is algorithmically similar to Deep CFR, but uses hand-crafted features similar to those used in abstraction, rather than learning the features. RCFR also uses full traversals of the game tree (which is infeasible in large games) and has only been evaluated on toy games. It is therefore best viewed as the first proof of concept that function approximation can be applied to CFR.
4. Description of the Deep Counterfactual Regret Minimization Algorithm

In this section we describe Deep CFR. The goal of Deep CFR is to approximate the behavior of CFR without calculating and accumulating regrets at each infoset, by generalizing across similar infosets using function approximation via deep neural networks.

On each iteration \( t \), Deep CFR conducts a constant number \( K \) of partial traversals of the game tree, with the path of the traversal determined according to external sampling MCCFR. At each infoset \( I \) it encounters, it plays a strategy \( \sigma^t(I) \) determined by regret matching on the output of a neural network \( V : I \to \mathbb{R}^{|A|} \) defined by parameters \( \theta^t \) that takes as input the infoset \( I \) and outputs values \( V(I, a|\theta^t) \). Our goal is for \( V(I, a|\theta^{t+1}) \) to be approximately proportional to the regret \( R^{t+1}(I, a) \) that tabular CFR would have produced.

When a terminal node is reached, the value is passed back up. In chance and opponent infosets, the value of the sampled advantage is the weighted average of all action values, where action \( a \)'s weight is \( \sigma^t(I, a) \). This produces samples of this iteration’s instantaneous regrets for various actions. Samples are added to a memory \( \mathcal{M}_{v, p} \), where \( p \) is the traverser, using reservoir sampling (Vitter, 1985) if capacity is exceeded.

Consider a nice property of the sampled instantaneous regrets induced by external sampling:

**Lemma 1.** For external sampling MCCFR, the sampled instantaneous regrets are an unbiased estimator of the advantage, i.e. the difference in expected payoff for playing \( a \) vs \( \sigma^t_p(I) \) at \( I \), assuming both players play \( \sigma^t \) everywhere else.

\[
E_{Q \subseteq \Omega, I \in Q} \left[ \tilde{r}^t_p(I, a) | Z_I \cap Q \neq \emptyset \right] = \frac{v^t(I, a) - v^t(I)}{\pi^t_p(I)}.
\]

The proof is provided in Appendix B.2.

Recent work in deep reinforcement learning has shown that neural networks can effectively predict and generalize advantages in challenging environments with large state spaces, and use that to learn good policies (Mnih et al., 2016).

Once a player’s \( K \) traversals are completed, a new network is trained from scratch to determine parameters \( \theta^t \) by minimizing MSE between predicted advantage \( V_p(I, a|\theta^t) \) and samples of instantaneous regrets from prior iterations \( t' \leq t \), \( \tilde{r}^t(I, a) \) drawn from the memory. The average over all sampled instantaneous advantages \( \tilde{r}^t(I, a) \) is proportional to the total sampled regret \( \tilde{R}^t(I, a) \) (across actions in an infoset), so once a sample is added to the memory it is never removed except through reservoir sampling, even when the next CFR iteration begins.

One can use any loss function for the value and average strategy model that satisfies Bregman divergence (Banerjee et al., 2005), such as mean squared error loss.

While almost any sampling scheme is acceptable so long as the samples are weighted properly, external sampling has the convenient property that it achieves both of our desired goals by assigning all samples in an iteration equal weight. Additionally, exploring all of a traverser’s actions helps reduce variance. However, external sampling may be impractical in games with extremely large branching factors, so a different sampling scheme, such as outcome sampling (Lancot et al., 2009), may be desired in those cases.

In addition to the value network, a separate policy network \( \Pi : I \to \mathbb{R}^{|A|} \) approximates the average strategy at the end of the run, because it is the average strategy played over all iterations that converges to a Nash equilibrium. To do this, we maintain a separate memory \( \mathcal{M}_\Pi \) of sampled infoset probability vectors for both players. Whenever an infoset \( I \) belonging to player \( p \) is traversed during the opposing player’s traversal of the game tree via external sampling, the infoset probability vector \( \sigma^t(I) \) is added to \( \mathcal{M}_\Pi \) and assigned weight \( t \).

If the number of Deep CFR iterations and the size of each value network model is small, then one can avoid training the final policy network by instead storing each iteration’s value network (Steinberger, 2019). During actual play, a value network is sampled randomly and the player plays the CFR strategy resulting from the predicted advantages of that network. This eliminates the function approximation error of the final average policy network, but requires storing all prior value networks. Nevertheless, strong performance and low exploitability may still be achieved by storing only a subset of the prior value networks (Jackson, 2016).

Theorem 1 states that if the memory buffer is sufficiently large, then with high probability Deep CFR will result in average regret being bounded by a constant proportional to the square root of the function approximation error.

**Theorem 1.** Let \( T \) denote the number of Deep CFR iterations, \( |A| \) the maximum number of actions at any infoset, and \( K \) the number of traversals per iteration. Let \( L \), be
the average MSE loss for $V_p(I, a|\theta^t)$ on a sample in $\mathcal{M}_{V,p}$ at iteration $t$, and let $\mathcal{L}_{V,p}$ be the minimum loss achievable for any function $V$. Let $\mathcal{L}_{V,p}^t - \mathcal{L}_{V,p} \leq \epsilon_L$.

If the value memories are sufficiently large, then with probability $1 - \rho$ total regret at time $T$ is bounded by

$$ R_p^T \leq \left( 1 + \frac{\sqrt{2}}{\sqrt{pK}} \right) \Delta|I_p|\sqrt{|A|} \sqrt{T} + 4T|I_p|\sqrt{|A|} \Delta \epsilon_L $$

with probability $1 - \rho$.

**Corollary 1.** As $T \to \infty$, average regret $R_p^T$ is bounded by

$$ 4|I_p|\sqrt{|A|} \Delta \epsilon_L $$

with high probability.

The proofs are provided in Appendix B.4.

We do not provide a convergence bound for Deep CFR when using linear weighting, since the convergence rate of Linear CFR has not been shown in the Monte Carlo case. However, Figure 4 shows moderately faster convergence in practice.

### 5. Experimental Setup

We measure the performance of Deep CFR (Algorithm 1) in approximating an equilibrium in heads-up flop hold’em poker (FHP). FHP is a large game with over $10^{12}$ nodes and over $10^9$ infosets. In contrast, the network we use has 98,948 parameters. FHP is similar to heads-up limit Texas hold’em (HULH) poker, but ends after the second betting round rather than the fourth, with only three community cards ever dealt. We also measure performance relative to domain-specific abstraction techniques in the benchmark domain of HULH poker, which has over $10^{12}$ nodes and over $10^{14}$ infosets. The rules for FHP and HULH are given in Appendix A.

In both games, we compare performance to NFSP, which is the previous leading algorithm for imperfect-information game solving using domain-independent function approximation, as well as state-of-the-art abstraction techniques designed for the domain of poker (Johanson et al., 2013; Ganzfried & Sandholm, 2014; Brown et al., 2015).

### 5.1. Network Architecture

We use the neural network architecture shown in Figure 5.1 for both the value network $V$ that computes advantages for each player and the strategy $\Pi$ that approximates the final average strategy. This network has a depth of 7 layers and 98,948 parameters. Infosets consist of sets of cards and bet history. The cards are represented as the sum of three embeddings: a rank embedding (1-13), a suit embedding (1-4), and a card embedding (1-52). These embeddings are summed for each set of permutation invariant cards (hole, flop, turn, river), and these are concatenated. In each of the $N_{rounds}$ rounds of betting there can be at most 6 sequential actions, leading to $6N_{rounds}$ total unique betting positions. Each betting position is encoded by a binary value specifying whether a bet has occurred, and a float value specifying the bet size.

The neural network model begins with separate branches for the cards and bets, with three and two layers respectively. Features from the two branches are combined and three additional fully connected layers are applied. Each fully-connected layer consists of $x_{i+1} = \text{ReLU}(Ax_{i} + x)$). The optional skip connection $[+x]$ is applied only on layers that have equal input and output dimension. Normalization (to zero mean and unit variance) is applied to the last-layer features. The network architecture was not highly tuned, but normalization and skip connections were used because they were found to be important to encourage fast convergence when running preliminary experiments on pre-computed equilibrium strategies in FHP. A full network specification is provided in Appendix C.

In the value network, the vector of outputs represented predicted advantages for each action at the input infoset. In the average strategy network, outputs are interpreted as logits of the probability distribution over actions.

### 5.2. Model training

We allocate a maximum size of 40 million infosets to each player’s advantage memory $\mathcal{M}_{V,p}$ and the strategy memory $\mathcal{M}_{\Pi}$. The value model is trained from scratch each CFR iteration, starting from a random initialization. We perform 4,000 mini-batch stochastic gradient descent (SGD) iterations using a batch size of 10,000 and perform parameter updates using the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.001, with gradient norm clipping to 1. For HULH we use 32,000 SGD iterations and a batch size of 20,000. Figure 4 shows that training the model from
Algorithm 1 Deep Counterfactual Regret Minimization

Function DeepCFR

- Initialize each player’s advantage network $V(I, a|θ_p)$ with parameters $θ_p$ so that it returns 0 for all inputs.
- Initialize reservoir-sampled advantage memories $M_{V,1}, M_{V,2}$ and strategy memory $M_{Π}$.

for CFR iteration $t = 1$ to $T$

for each player $p$ do

for traversal $k = 1$ to $K$ do

$\text{TRAVERSE}(h, p, θ_1, θ_2, M_{V,p}, M_{Π})$  
$\triangleright$ Collect data from a game traversal with external sampling

Train $θ_p$ from scratch on loss $L(θ_p) = E_{(I,t’,\hat{π}’)} [\sum_a (\hat{π}’(a) - V(I,a|θ_p))^2]$

Train $θ_{Π}$ on loss $L(θ_{Π}) = E_{(I,t',σ')} [\sum_a (σ'(a) - Π(I,a|θ_{Π}))^2]$

return $θ_{Π}$

Algorithm 2 CFR Traversal with External Sampling

Function $\text{TRAVERSE}(h, p, θ_1, θ_2, M_{V}, M_{Π}, t)$

Input: History $h$, traverser player $p$, regret network parameters $θ$ for each player, advantage memory $M_{V}$ for player $p$, strategy memory $M_{Π}$, CFR iteration $t$.

if $h$ is terminal then

return the payoff to player $p$

else if $h$ is a chance node then

$a \sim σ(h)$

return $\text{TRAVERSE}(h \cdot a, p, θ_1, θ_2, M_{V}, M_{Π}, t)$

else if $P(h) = p$ then

$V(a) \leftarrow \text{TRAVERSE}(h \cdot a, p, θ_1, θ_2, M_{V}, M_{Π}, t)$  
$\triangleright$ If it’s the traverser’s turn to act

$\text{for } a \in A(h) \text{ do}$

$v(a) \leftarrow \text{TRAVERSE}(h \cdot a, p, θ_1, θ_2, M_{V}, M_{Π}, t)$  
$\triangleright$ Traverse each action

else if $P(h) \neq p$ then

$\tilde{π}(I,a) \leftarrow v(a) - \sum_{a' \in A(h)} σ(I,a') \cdot v(a')$  
$\triangleright$ Compute advantages

Insert the infoset and its action advantages $(I,t,\tilde{π}(I))$ into the advantage memory $M_{V}$

else

$\text{Compute strategy } σ'(I) \text{ from predicted advantages } V(I(h),a|θ_{3-p}) \text{ using regret matching.}$

Insert the infoset and its action probabilities $(I,t,σ'(I))$ into the strategy memory $M_{Π}$

Sample an action $a$ from the probability distribution $σ'(I)$.

return $\text{TRAVERSE}(h \cdot a, p, θ_1, θ_2, M_{V}, M_{Π}, t)$

5.3. Linear CFR

There exist a number of variants of CFR that achieve much faster performance than vanilla CFR. However, most of these faster variants of CFR do not handle approximation error well (Tammelin et al., 2015; Burch, 2017; Brown & Sandholm, 2019; Schmid et al., 2019). In this paper we use Linear CFR (LCFR) (Brown & Sandholm, 2019), a variant of CFR that is faster than CFR and in certain settings is the fastest-known variant of CFR (particularly in settings with wide distributions in payoffs), and which tolerates approximation error well. LCFR is not essential and does not appear to lead to better performance asymptotically, but does result in faster convergence in our experiments.

LCFR is like CFR except iteration $t$ is weighed by $t$. Specifically, we maintain a weight on each entry stored in the advantage memory and the strategy memory, equal to $t$ when this entry was added. When training $θ_p$ each iteration $T$, we rescale all the batch weights by $\frac{2}{T}$ and minimize weighted error.

6. Experimental Results

Figure 2 compares the performance of Deep CFR to different-sized domain-specific abstractions in FHP. The abstractions are solved using external-sampling Linear Monte Carlo.
Deep Counterfactual Regret Minimization

Carlo CFR (Lanctot et al., 2009; Brown & Sandholm, 2019), which is the leading algorithm in this setting. The 40,000 cluster abstraction means that the more than 10^{10} different decisions in the game were clustered into 40,000 abstract decisions, where situations in the same bucket are treated identically. This bucketing is done using K-means clustering on domain-specific features. The lossless abstraction only clusters together situations that are strategically isomorphic (e.g., flushes that differ only by suit), so a solution to this abstraction maps to a solution in the full game without error.

Performance and exploitability are measured in terms of milli big blinds per game (mbb/g), which is a standard measure of win rate in poker.

The figure shows that Deep CFR asymptotically reaches a similar level of exploitability as the abstraction that uses 3.6 million clusters, but converges substantially faster. Although Deep CFR is more efficient in terms of nodes touched, neural network inference and training requires considerable overhead that tabular CFR avoids. However, Deep CFR does not require advanced domain knowledge. We show Deep CFR performance for 10,000 CFR traversals per step. Using more traversals per step is less sample efficient and requires greater neural network training time but requires fewer CFR steps.

Figure 2 also compares the performance of Deep CFR to NFSP, an existing method for learning approximate Nash equilibria in imperfect-information games. NFSP approximates fictitious self-play, which is proven to converge to a Nash equilibrium but in practice does so far slower than CFR. We observe that Deep CFR reaches an exploitability of 37 mbb/g while NFSP converges to 47 mbb/g.\footnote{We run NFSP with the same model architecture as we use for Deep CFR. In the benchmark game of Leduc Hold’em, our implementation of NFSP achieves an average exploitability (total exploitability divided by two) of 37 mbb/g in the benchmark game of Leduc Hold’em, which is substantially lower than originally reported in Heinrich & Silver (2016). We report NFSP’s best performance in FHP across a sweep of hyperparameters.} We also observe that Deep CFR is more sample efficient than NFSP. However, these methods spend most of their wallclock time performing SGD steps, so in our implementation we see a less dramatic improvement over NFSP in wallclock time than sample efficiency.

Figure 3 shows the performance of Deep CFR using different numbers of game traversals, network SGD steps, and model size. As the number of CFR traversals per iteration is reduced, convergence becomes slower but the model converges to the same final exploitability. This is presumably because it takes more iterations to collect enough data to reduce the variance sufficiently. On the other hand, reducing the number of SGD steps does not change the rate of convergence but affects the asymptotic exploitability of the model. This is presumably because the model loss decreases as the number of training steps is increased per iteration (see Theorem 1). Increasing the model size also decreases final exploitability up to a certain model size in FHP.

In Figure 4 we consider ablations of certain components of Deep CFR. Retraining the regret model from scratch at each CFR iteration converges to a substantially lower exploitability than fine-tuning a single model across all iterations. We suspect that this is because a single model gets stuck in bad local minima as the objective is changed from iteration to iteration. The choice of reservoir sampling to update the memories is shown to be crucial; if a sliding window memory is used, the exploitability begins to increase once the memory is filled up, even if the memory is large enough to hold the samples from many CFR iterations.

Finally, we measure head-to-head performance in HULH. We compare Deep CFR and NFSP to the approximate solutions (solved via Linear Monte Carlo CFR) of three differently sized abstractions: one in which the more than 10^{14} decisions are clustered into 3.3 \cdot 10^6 buckets, one in which there are 3.3 \cdot 10^7 buckets and one in which there are 3.3 \cdot 10^8 buckets. The results are presented in Table 1. For comparison, the largest abstractions used by the poker AI Polaris in its 2007 HULH man-machine competition against human professionals contained roughly 3 \cdot 10^8 buckets. When variance-reduction techniques were applied, the results showed that the professional human competitors lost to the 2007 Polaris AI by about 52 ± 10 mbb/g (Johanson, 2016). In contrast, our Deep CFR agent loses to a 3.3 \cdot 10^8 bucket abstraction by only −11 ± 2 mbb/g and beats NFSP by 43 ± 2 mbb/g.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.png}
\caption{Comparison of Deep CFR with domain-specific tabular abstractions and NFSP in FHP. Coarser abstractions converge faster but are more exploitable. Deep CFR converges with 2-3 orders of magnitude fewer samples than a lossless abstraction, and performs competitively with a 3.6 million cluster abstraction. Deep CFR achieves lower exploitability than NFSP, while traversing fewer infosets.}
\end{figure}
Deep Counterfactual Regret Minimization

<table>
<thead>
<tr>
<th>Model</th>
<th>Opponent Model</th>
<th>Deep CFR</th>
<th>NFSP</th>
<th>3.3 · 10^6</th>
<th>3.3 · 10^7</th>
<th>3.3 · 10^8</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFSP</td>
<td>-</td>
<td>-43 ± 2 mbb/g</td>
<td>-40 ± 2 mbb/g</td>
<td>-49 ± 2 mbb/g</td>
<td>-55 ± 2 mbb/g</td>
<td></td>
</tr>
<tr>
<td>Deep CFR</td>
<td>+43 ± 2 mbb/g</td>
<td>-</td>
<td>+6 ± 2 mbb/g</td>
<td>-6 ± 2 mbb/g</td>
<td>-11 ± 2 mbb/g</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Head-to-head expected value of NFSP and Deep CFR in HULH against converged CFR equilibria with varying abstraction sizes. For comparison, in 2007 an AI using abstractions of roughly $3 · 10^3$ buckets defeated human professionals by about 52 mbb/g (after variance reduction techniques were applied).

Figure 3. **Left:** FHP convergence for different numbers of training data collection traversals per simulated LCFR iteration. The dotted line shows the performance of vanilla tabular Linear CFR without abstraction or sampling. **Middle:** FHP convergence using different numbers of minibatch SGD updates to train the advantage model at each LCFR iteration. **Right:** Exploitability of Deep CFR in FHP for different model sizes. Label indicates the dimension (number of features) in each hidden layer of the model.

Figure 4. Ablations of Deep CFR components in FHP. **Left:** As a baseline, we plot 5 replicates of Deep CFR, which show consistent exploitability curves (standard deviation at $t = 450$ is 2.25 mbb/g). Deep CFR without linear weighting converges to a similar exploitability, but more slowly. If the same network is fine-tuned at each CFR iteration rather than training from scratch, the final exploitability is about 50% higher. Also, if the algorithm plays a uniform strategy when all regrets are negative (i.e. standard regret matching), rather than the highest-regret action, the final exploitability is also 50% higher. **Right:** If Deep CFR is performed using sliding-window memories, exploitability stops converging once the buffer becomes full$^9$. However, with reservoir sampling, convergence continues after the memories are full.

7. Conclusions

We describe a method to find approximate equilibria in large imperfect-information games by combining the CFR algorithm with deep neural network function approximation. This method is theoretically principled and achieves strong performance in large poker games relative to domain-specific abstraction techniques without relying on advanced domain knowledge. This is the first non-tabular variant of CFR to be successful in large games.

Deep CFR and other neural methods for imperfect-information games provide a promising direction for tackling large games whose state or action spaces are too large for tabular methods and where abstraction is not straightforward. Extending Deep CFR to larger games will likely require more scalable sampling strategies than those used in this work, as well as strategies to reduce the high variance in sampled payoffs. Recent work has suggested promising directions both for more scalable sampling (Li et al., 2018) and variance reduction techniques (Schmid et al., 2019). We believe these are important areas for future work.
8. Acknowledgments

This material is based on work supported by the National Science Foundation under grants IIS-1718457, IIS-1617590, IIS-1901403, and CCF-1733556, and the ARO under award W911NF-17-1-0082. Noam Brown was partly supported by an Open Philanthropy Project AI Fellowship and a Tencent AI Lab fellowship.

References


Deep Counterfactual Regret Minimization


Lanctot, M. Monte carlo sampling and regret minimization for equilibrium computation and decision-making in large extensive form games. 2013.


