DREAM: Deep Regret minimization with Advantage baselines and Model-free learning

Eric Steinberger∗ † Adam Lerer∗ Noam Brown∗

Abstract

We introduce DREAM, a deep reinforcement learning algorithm that finds optimal strategies in imperfect-information games with multiple agents. Formally, DREAM converges to a Nash Equilibrium in two-player zero-sum games and to an extensive-form coarse correlated equilibrium in all other games. Our primary innovation is an effective algorithm that, in contrast to other regret-based deep learning algorithms, does not require access to a perfect simulator of the game to achieve good performance. We show that DREAM empirically achieves state-of-the-art performance among model-free algorithms in popular benchmark games, and is even competitive with algorithms that do use a perfect simulator.

1 Introduction

We consider the challenge of computing a Nash equilibrium in imperfect-information two-player zero-sum games. This means finding a policy such that both agents play optimally against the other and no agent can improve by deviating to a different policy.

There has been an explosion of deep reinforcement learning algorithms that compute optimal policies in single-agent settings and perfect-information games [22, 33, 36, 27]. However, these algorithms generally fail to compute an optimal policy in imperfect-information games.

More recently, deep reinforcement learning algorithms have been developed that either theoretically or empirically converge to a Nash equilibrium in two-player zero-sum imperfect-information games [13, 30, 4, 31, 20, 24, 25]. Among these, neural forms of counterfactual regret minimization (CFR) [38, 18, 32, 6] have achieved the best performance [4, 31, 20]. This mirrors the success of tabular CFR, variants of which are state-of-the-art among tabular equilibrium-finding algorithms [6] and which have been used in every major AI milestone in the domain of poker [2, 23, 5, 7].

However, in order to deploy tabular CFR in domains with very large state spaces, domain-specific abstraction techniques that bucket similar states together are needed [11, 3]. While these abstraction techniques have been successful in poker, they require extensive domain knowledge to construct and are not applicable to all games. This has motivated the development of CFR algorithms that use neural network function approximation to generalize with far less domain knowledge.

However, the existing forms of neural CFR only perform well with an exact simulator of the game, which allows them to sample and explore multiple actions at a decision point and thereby reduce variance. We show they perform poorly if only a single action is sampled, which is necessary in a model-free setting.

In this paper we introduce DREAM, a neural form of CFR that samples only a single action at each decision point. In order to keep variance low despite sampling only a single action, DREAM uses a sample-based neural form of learned baselines, a fundamental technique that has previously been shown to be successful in tabular CFR [26, 9].

∗Facebook AI Research (FAIR)
†Climate Science (https://climate-science.com)

Preprint. Under review.
DREAM minimizes regret and converges to an $\epsilon$-Nash equilibrium in two-player zero-sum games with $\epsilon$ proportional to the neural modeling error, and more generally converges to an $\epsilon$-extensive-form coarse correlated equilibrium (EFCCE) in all games.

We demonstrate empirically that DREAM achieves state-of-the-art performance among model-free self-play deep reinforcement learning algorithms in two popular benchmark poker variants: Leduc Hold’em and Flop Texas Hold’em.

2 Notation and Background

Our notation is based on that of partially observable stochastic games [12]. We consider a game with $\mathcal{N} = \{1, 2, \ldots, N\}$ agents. We use $i$ to denote the agent index and use $-i$ to refer to all agents except agent $i$.

A world state $w \in \mathcal{W}$ is the exact state of the world. $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \ldots \times \mathcal{A}_N$ is the space of joint actions. $\mathcal{A}_i(w)$ denotes the legal actions for agent $i$ at $w$ and $a = (a_1, a_2, \ldots, a_N) \in \mathcal{A}$ denotes a joint action. After each agent chooses a legal action, a transition function $\mathcal{T}$ determines the next world state $w'$ drawn from the probability distribution $\mathcal{T}(w, a) \in \Delta \mathcal{W}$. In each world state $w$, agent $i$ receives a reward $R_i(w)$. Upon transition from world state $w$ to $w'$ via joint action $a$, agent $i$ receives an observation $O_i$ from a function $O_i(w, a, w')$.

A history (also called a trajectory) is a finite sequence of legal actions and world states, denoted $h = (w_0^i, a_0^i, w_1^i, a_1^i, \ldots, w_t^i)^0$. The reward $r$ obtained in the last world state of a history $h$ is $R(h)$ and $i$’s legal actions are $\mathcal{A}_i(h)$. An infostate (also called an action-observation history (AOH)) for agent $i$ is a sequence of an agent’s observations and actions $s_i = (O_0^i, a_0^i, O_1^i, a_1^i, \ldots, O_t^i)$. The set of all infostates for agent $i$ is $\mathcal{I}_i$. The unique infostate containing history $h$ for agent $i$ is denoted $s_i(h)$. The set of histories that correspond to an infostate $s_i$ is denoted $\mathcal{H}(s_i)$. Note that all histories in an infostate are, by definition, indistinguishable to agent $i$, allowing us to define $\mathcal{A}(s_i) = \mathcal{A}(h)$ and $\mathcal{R}(s_i) = \mathcal{R}(h)$ for all $h \in \mathcal{S}_i$. An observation $O_i$ may and typically does include rewards achieved upon the state transitions as well as the opponents’ actions chosen in joint actions leading to the observation. We denote $h' \subset h$ to indicate $h'$ is a history reached on the path to $h$.

An agent’s policy $\pi_i$ is a function mapping from an infostate to a probability distribution over actions. A policy profile $\pi$ is a tuple $(\pi_1, \pi_2, \ldots, \pi_N)$. The policy of all agents other than $i$ is denoted $\pi_{-i}$. A policy for a history $h$ is denoted $\pi_i(h) = \pi_i(s_i(h))$ and $\pi(h) = (\pi_1(s_1(h)), \pi_2(s_2(h)), \ldots, \pi_N(s_N(h)))$. We also define the transition function $T(h, a, \pi_{-i})$ as a function drawing actions for $-i$ from $\pi_{-i}$ to form $a = a_i \cup a_{-i}$ and then to return the history $h'$ from $T(w_{t+1}, a)$, where $w_{t+1}$ is the last world state in $h$.

The expected sum of future rewards (also called the expected value (EV)) for agent $i$ in history $h$ when all agents play policy profile $\pi$ is denoted $v_i^\pi(h)$. The EV for an infostate $s_i$ is denoted $v_i^\pi(s_i)$ and the EV for the entire game is denoted $v_i(\pi)$. The EV for an action in a history and infostate is denoted $v_i^\pi(h, a_i)$ and $v_i^\pi(s_i, a_i)$, respectively. A Nash equilibrium is a policy profile such that no agent can achieve a higher EV by switching to a different policy. Formally, $\pi^*$ is a Nash equilibrium if for every agent $i$, $v_i(\pi^*) = \max_{\pi_i} v_i(\pi_i, \pi_{-i}^*)$. The exploitability of a policy profile $\pi$ is $e(\pi) = \sum_{\pi_i \in \Pi_i} \max_{\pi_{-i}} v_i(\pi_i, \pi_{-i})$.

The reach $x_i(h)$ of a history $h$ is the product of all action probabilities under $\pi$ and all transition function probabilities leading to $h$. Formally, for $h_i = (w_0^i, a_0^i, w_1^i, a_1^i, \ldots, w_t^i)$, $x_i(h_i) = \Pi_{h_i \subseteq h} \Pi_{t \in \mathcal{I}_i} \pi_{-i}(w_t, a_t)$. The agent reach $x_i^\pi(h_i)$ of a history $h_i$ is the product of the probabilities for all agent $i$ actions leading to $h_i$. Formally, $x_i^\pi(h_i) = \Pi_{h_i \subseteq h} \Pi_{t \in \mathcal{I}_i} \pi_{-i}(w_t, a_t)$. We similarly define the agent reach $x_i^\pi(s_i)$ of infostate $s_i = (O_0^i, a_0^i, O_1^i, a_1^i, \ldots, O_t^i)$ as $x_i^\pi(s_i) = \Pi_{t \in \mathcal{I}_i} (a_t)$. The external reach $x_i^\pi(s_i)$ of infostate $s_i$ is the probability that agent $i$ would reach $s_i$ if they had always taken actions leading to $s_i$ with 100% probability. Formally, $x_i^\pi(s_i) = \sum_{h_i \in \mathcal{H}(s_i)} x_i^\pi(h_i, a_i)$. If $x_i^\pi(s_i) > 0$ then external reach is also $\sum_{h \in \mathcal{H}(s_i)} x_i^\pi(h_i, a_i)$.

3 Related Work

Tabular CFR methods are limited by the need to visit a given state to update the policy played in it [38, 6, 32, 18]. In large games that are infeasible to fully traverse, domain-specific abstraction
Advantage Regret Minimization (ARM) \[11\] is a regret-based policy gradient algorithm designed for CFR. It is another algorithm aiming to approximate CFR using deep neural networks \[20\]. Double Neural CFR is a fully parameterized variant of CFR that requires no tabular sampling \[4\]. Deep CFR, Deep Neural CFR, and Single Deep CFR rely on a perfect simulator of the game to explore multiple actions at each decision point. These techniques work in many games \[21\], but they most often require expert knowledge to design and are not applicable to all games.

To address these issues, researchers started to apply neural network function approximation to CFR. The first such experiment was DeepStack \[23\]. While DeepStack used neural networks to predict a quantity called counterfactual value, it still relied on tabular solving for multiple stages of its training and evaluation process. The benefit of a parameterized policy is that it can make an educated guess for how to play in states it has never seen during training. Parameterized policies have led to performance breakthroughs in AI for perfect information games like Atari and Go \[14\], \[28\].

Neural Fictitious Self-Play (NFSP) \[13\] approximates extensive-form fictitious play using deep reinforcement learning from single trajectories. NFSP was the first algorithm to learn a Nash Equilibrium in two-player imperfect information games from self-play. Since then, various policy gradient and actor critic methods have been shown to have similar convergence properties if tuned appropriately \[30\], \[19\].

Deep CFR is a fully parameterized variant of CFR that requires no tabular sampling \[4\]. Deep CFR substantially outperformed NFSP (which in turn outperforms or closely matches competing algorithms \[30\]). Single Deep CFR (SD-CFR) \[31\] removed the need for an average network in Deep CFR and thereby enabled better coverage and more time- and space-efficient training. However, both Deep CFR and Single Deep CFR rely on a perfect simulator of the game to explore multiple actions at each decision point.

Double Neural CFR is another algorithm aiming to approximate CFR using deep neural networks \[20\]. Advantage Regret Minimization (ARM) \[15\] is a regret-based policy gradient algorithm designed for single agent environments. Before neural networks were considered, Regression CFR (R-CFR) \[37\] used regression trees to estimate regrets in CFR and CFR\(^+\). However, R-CFR seems to be incompatible with sparse sampling techniques \[30\], rendering it less scalable.

## 4 Review of Counterfactual Regret Minimization (CFR)

### 4.1 Counterfactual Regret Minimization (CFR)

Counterfactual Regret Minimization (CFR) \[38\] is an iterative policy improvement algorithm that computes a new policy profile \(\pi^t\) on each iteration \(t\). The average of these policies converges to a Nash equilibrium.

CFR can apply either simultaneous or alternating updates. If the former is chosen, CFR produces \(\pi^t_i\) for all agents \(i\) on every iteration \(t\). With alternating updates, CFR produces a policy only for one agent \(i\) on each iteration, with \(i = t \mod 2\) on iteration \(t\). We describe the algorithm in this section in terms of simultaneous updates.

The instantaneous regret for action \(a_i\) at infostate \(s_i\) is \(r_i^t(s_i, a_i) = x_{s_i}^\pi(s_i)(v_i^\pi(s_i, a_i) - v_i^\pi(s_i))\). \(r_i^t(s_i, a_i)\) is a measure of how much more agent \(i\) could have gained on average by always choosing \(a_i\) in \(s_i\), weighed by the external reach. The average regret on iteration \(T\) is defined as \(R_i^T(s_i, a_i) = \sum_{t=1}^T r_i^t(s_i, a_i)\). CFR defines the policy for agent \(i\) on iteration \(t + 1\) based on their overall regret as

\[
\pi_{i+1}^t(s_i, a_i) = \begin{cases} 
\max\{0, R_i^t(s_i, a_i)\} & \text{if } \sum_{a_i' \in A_i(s_i)} \max\{0, R_i^t(s_i, a_i')\} > 0 \\
\frac{1}{|A_i(s_i)|} & \text{otherwise}
\end{cases}
\]

The initial policy is set to uniform random. The average policy \(\bar{\pi}_i^T\) is

\[
\bar{\pi}_i^T(s_i, a_i) = \frac{\sum_{t=1}^T x_i^\pi_t(s_i)\pi_i^t(s_i, a_i)}{\sum_{t=1}^T x_i^\pi_t(s_i)}
\]

On each iteration \(t\), CFR traverses the entire game tree and updates the regrets for every infostate in the game according to policy profile \(\pi^t\). These regrets define a new policy \(\pi^{t+1}\). The average policy over all iterations converges to a Nash equilibrium.
4.2 Linear CFR

Linear CFR [6] is a variant of CFR that weights the updates to the regret and average policy on iteration $t$ by $t$. Thus, under Linear CFR, $R_t^i(s_i, a_i) = \sum_{t=1}^T (r_t^i(s_i, a_i))$ and $\bar{\pi}_t^i(s_i, a_i) = \frac{\sum_{t=1}^T (x_t^i(s_i)) \pi_t^i(s_i, a_i))}{\sum_{t=1}^T (x_t^i(s_i))}$. This modification accelerates CFR by two orders of magnitude.

4.3 Monte Carlo CFR (MC-CFR)

Monte Carlo CFR (MC-CFR) is a framework that allows CFR to only update regrets on part of the tree for a single agent, called the traverser. Two variants relevant to our work are External Sampling (ES) and Outcome Sampling (OS) [18]. In OS, regrets are updated for the traverser only along a single trajectory (i.e., a single action is sampled for each agent at each decision point, and a single outcome is sampled from $T$). In ES, a single action is sampled for each non-traverser agent and a single outcome is sampled from $T$, but every traverser action is explored. When agent $i$ is the traverser in ES or OS, all other agents sample actions according to $\pi_{-i}$. In OS, we use a sampling policy $\xi_t^j(s_i)$ profile rather than $\pi_t^j$ to choose actions during play. $\xi_t^j(s_i)$ is defined separately for the traverser $i$ and any other agent $j$.

$$\xi_t^j(s_i, a_i) = \epsilon \frac{1}{|A_i(s_i)|} + (1 - \epsilon) \pi_t^j(s_i, a_i) \quad \xi_t^j(s_j, a_j) = \pi_t^j(s_j, a_j)$$

For a thorough and more general description of MC-CFR see [18].

4.4 Variance-Reduced Outcome Sampling with Baselines (VR-MC-CFR)

ES typically produces lower-variance estimates of expected values than OS. Variance-reduced MC-CFR (VR-MCCFR) is a method to reduce variance in MC-CFR and is particularly helpful for OS [26, 9]. Given that OS ultimately reaches a terminal history $z$, the baseline-adjusted sampled expected value in history $h$ is

$$\hat{v}_t^\pi(s_i(h), a_i|z) = \hat{v}_t^\pi(h, a_i|z) = \begin{cases} b_i(h, a_i) + \sum_{h' \in A_i(h)} p(h'|T(h, a_i, \pi_t^i)) \hat{v}_t^\pi(h'|z) - b_i(h, a_i) \xi_t^i(s_i, a_i) & \text{if } (h, a_i) \sqsubseteq z \\ b_i(h, a_i) & \text{otherwise} \end{cases}$$

(4)

$$\hat{v}_t^\pi(h|z) = \hat{v}_t^\pi(h|z) = \begin{cases} R_i(h) \xi_t^i(h, a_i) \hat{v}_t^\pi(h, a_i) & \text{if } h = z \\ \sum_{a_i \in A_i(h)} \pi_t^i(h, a_i) \hat{v}_t^\pi(h, a_i) & \text{otherwise} \end{cases}$$

(5)

where $b$ is called the history-action baseline, a user-defined scalar function that ideally is closely correlated with $v$. Common choices in the tabular setting are a running average across visits to any given history, and domain-specific estimates of $v$ [26, 9].

4.5 Deep CFR and Single Deep CFR (SD-CFR)

Deep CFR [4] approximates the behavior of tabular CFR from partial game tree traversals. For each CFR iteration $t$, Deep CFR runs a constant $T$ external sampling traversals to collect samples of instantaneous regret and adds the data to a reservoir buffer $D^i_t$ [35], where agent $i$ is the traverser. Deep CFR then trains a value network $\hat{D}^i_t$ to predict the average regret (divided by external reach) for unseen infostate actions. $\hat{D}^i_t$ is then used to approximate the policy $\pi_{t+1}^i$ for agent $i$ like tabular CFR would produce given $R^i_t$. More data is collected by sampling based on this policy, and the process repeats until convergence.

Specifically, on each iteration $T$, the algorithm fits a value network $\hat{D}^i_T$ for one agent $i$ to approximate advantages defined as $R^i_T(s_i, a_i) = \frac{R^i_T(s_i, a_i)}{\sum_{i=1}^T (x^i(s_i))}$. Deep CFR divides $R^i_T(s_i, a_i)$ by $\sum_{i=1}^T (x^i(s_i))$ to erase the difference in magnitude arising from highly varying realisation probabilities across infostates, which makes the data easier for the value network to approximate. This division does not change the policy output by Equation [4]. Since every action in an infostate is divided by the same value.
In cases where Deep CFR predicts non-positive advantage for all infostate actions, it chooses the action with the highest advantage rather than a uniform random policy as vanilla CFR would do.

Since it is the average policy that converges to a Nash equilibrium in CFR, not the final strategy, Deep CFR trains a second neural network \(\hat{S}\) to approximate the average policy played over all iterations. Data for \(\hat{S}\) is stored in a separate reservoir buffer \(B^H_t\) and collected during the same traversals that data for \(B^R_t\) is being collected on.

Single Deep CFR (SD-CFR) is a modification of Deep CFR that instead stores all value networks from each CFR iteration to disk and mimics the average policy exactly during play by sampling one of them and using its policy for the entire game. This is mathematically equivalent to sampling actions from the average policy. SD-CFR eliminates the approximation error in Deep CFR resulting from training a network to predict the average policy, at the minor cost of using extra disk space to store the models from each CFR iteration (or a random sample of CFR iterations).

5 Description of DREAM

In this section, we will describe DREAM, (D)eep (R)egret minimization with (A)dvantage Baselines and (M)odel-free learning. The primary contribution of this paper is to effectively combine a deep neural network approximation of outcome sampling CFR with learned baselines.

Like Single Deep CFR, DREAM is trained using alternating CFR iterations. On each iteration \(t\) at each encountered infostate \(s_t\), we obtain estimated advantages \(D^j_t(s_t, a_t|\theta^j_t)\) for the actions \(a_t \in A_i(s_t)\) by inputting \(s_t\) into the advantage network parameterized by \(\theta^j_t\). From these estimated advantages, we obtain a policy \(\pi^j_t(s_t)\) via regret matching as shown in Equation 1, except using \(D^j_t\) in place of \(D^R_t\), and picking the action with the highest \(D^j_t\) probability 1 when all advantages are negative.

5.1 Incorporating outcome sampling with exploration

On each iteration \(t\), DREAM collects advantages from \(T\) trajectories of the game for agent \(i\), where \(i = t \mod N\) following the outcome sampling (OS) MC-CFR scheme [13] with the sampling policy profile introduced in [4,4], where \(\epsilon > 0\) is the exploration parameter. To compensate for the shifted sampling distribution due to \(\epsilon\), we weight data samples stored in \(B^R_t\) by \(\frac{1}{x^R_t(s_i)}\) during neural training. Storing advantage models from each iteration and sampling one randomly to determine a policy used during play, as is done in SD-CFR, results in an algorithm we call Outcome Sampling SD-CFR (OS-SD-CFR). We show in section 7 that OS-SD-CFR performs poorly.

5.2 Variance reduction using a learned Q-baseline

We hypothesised that the higher variance of advantage estimates in OS traversals is to blame for OS-SD-CFR’s bad performance. DREAM aims to reduce this source of variance by implementing a neural form of history baselines [9], the tabular form of which is described in section 4.4.

DREAM uses a baseline network \(\hat{Q}^j_t(s^*(h), a_i|\phi^j_t)\) parameterized by \(\phi^j_t\) as a baseline, where \(s^*\) is the set of infostates for all players at \(h\), \(s^*(h) = (s_0(h),...,s_N(h))\). \(\hat{Q}^j_t\) is fine-tuned from \(\hat{Q}^{j-1}_t\) when \(t > 0\) and initialized with random weights on \(t = 0\). It is trained using expected SARSA [34] on a circular buffer \(B^R_t\). Formally, it updates \(\phi^j_t\) to minimize its mean squared error (MSE) to the target value sample \(R_i(h) + \sum_{a'_i \in A_i} \pi^j_t(s_i(h'), a'_i)\hat{Q}^j_t(s^*(h'), a'_i)\), where \(h'\) is drawn from \(T(h, a_i, \pi^j_t)\).

VR-MCCFR computes history-action baselines both for player actions and ‘chance actions’ from the environment. The latter requires access to a perfect simulator of the environment’s transition function. Thus, DREAM only uses baselines for the player actions. Given that OS ultimately reaches a terminal history \(z\), the baseline-adjusted sampled expected value at history \(h\) where action \(a^*_i\) is

---

1We do not define \(\hat{Q}\) to take \(h\) directly as input since \(h\) is not available in a model-free setting. However, in the games we investigate, \(h\) and \(s^*(h)\) are equivalent.
We derive a bound on the cumulative regret for DREAM in Appendix D.

We define the weight of the neural training loss of \( \tilde{v} \).

The convergence of MC-CFR \cite{17} using function approximation to estimate infostate advantages has been previously studied in \cite{4}. They show that minimizing mean-squared error of predicted advantages on historical samples (as we do in DREAM) is equivalent to tabular MC-CFR. Furthermore, if a model only \( \approx \) the quality of the baseline. More details can be found in \cite{26}. We derive a bound on the cumulative regret for DREAM in Appendix D.

5.4 Convergence

The convergence of MC-CFR \cite{17} using function approximation to estimate infostate advantages has been previously studied in \cite{4}. They show that minimizing mean-squared error of predicted advantages on historical samples (as we do in DREAM) is equivalent to tabular MC-CFR. Furthermore, if a model only \( \approx \) the quality of the baseline. More details can be found in \cite{26}. We derive a bound on the cumulative regret for DREAM in Appendix D.

6 Experimental Setup

We explore a sweep of hyperparameters for DREAM, and compare its exploitability over the number of game states observed to Deep CFR \cite{4}, Single Deep CFR (SD-CFR) \cite{31}, and Neural Fictitious Self-Play (NFSP) \cite{13}. We chose Leduc Poker \cite{29} and Flop hold’em poker \cite{4}, two popular poker games, as benchmark domains. A description of these can be found in the appendix. We use Linear CFR \cite{6} for DREAM and all variants of SD-CFR we evaluate. For all experiments, we plot the mean and standard deviation across three independent runs.
In FHP we measure an approximate lower bound on exploitability computed by training a DDQN agent \[36, 33\] against DREAM and running a number of hands between DREAM and the final iteration of DDQN and for evaluation. Details are in appendix C.

6.1 Neural Networks

Our work does not focus on exploring new neural architectures. Instead, to enable easy reproducibility, we use the neural architecture demonstrated to be successful with Deep CFR and NFSP \[4\] for all networks in all algorithms and games. Details and notes on the required minimal modifications to input and output layers that were required for algorithmic reasons can be found in appendix E.

On each iteration, \( \hat{Q}^t_i \) is trained for 1000 minibatches of 512 samples using the Adam optimizer \[16\] with a learning rate of 0.001 and gradient norm clipping of 1 in both games. \( B^q_i \) has a capacity of 200,000. For comparability, DREAM’s value networks use the same training parameters as those SD-CFR \[4, 31\], unless otherwise stated. Both SD-CFR and DREAM use advantage buffers \( B^d_i \) with capacities of 40 million samples in FHP and 2 million in Leduc. Value networks for 3,000 batches of size 2,048 and 10,000 batches of size in Leduc and FHP, respectively. For DREAM with Deep CFR’s average network we allow an additional buffer \( B^s_i \) of 2 million samples per player. The average network is trained for 4,000 batches of 2,048 samples.

The number of traversals sampled per iteration for SD-CFR and DREAM was chosen such that both algorithms visit roughly the same number of states per iteration. In Leduc, we chose a successful value for DREAM (900) and then divide by 2.6 (the empirical average ratio between states visited by ES and OS in Leduc) to get 346 for SD-CFR. We compare this to the initially proposed setting of 1,500, which performs worse \[31\]. In FHP, we use 10,000 for SD-CFR as proposed by \[4\] and 50,000 for DREAM. Again, the factor of 5 difference cancels out the ratio of states seen by OS compared to ES, determined empirically in preliminary runs. However, in FHP we stared from the parameters for SD-CFR and adjusted DREAM accordingly.

Apart from the neural network architecture, NFSP uses the same hyperparameters in Leduc as presented the original paper \[13\]. Because the authors did not use FHP as a benchmark, we use the hyperparameters they applied in Limit Hold’em poker, a similar game.

7 Experimental Results

We first investigate settings and hyperparameters of DREAM (see figure 1) and then compare DREAM to other algorithms (see figure 2). We find that DREAM significantly outperforms other model-free algorithms in Leduc and FHP. DREAM’s performance matches that of SD-CFR in Leduc, despite not requiring a perfect simulator of the game.

We found that few minibatch updates per iteration are needed for \( \hat{Q} \) in DREAM. Thus, the overhead relative to OS-SD-CFR that the baseline network adds is not significant. This may be because training targets for \( \hat{Q} \) change only slightly on each iteration and \( \hat{Q}^t \) is started from \( \hat{Q}^{t-1} \)’s weights. Moreover, small modelling errors may be insignificant compared to the remaining game variance.

Deep CFR \[4\] performs better when value networks \( \hat{D}^t \) are trained from scratch on each iteration \( t \). We investigate three settings: 1) **Always reset**: train from scratch for 3,000 minibatches on every iteration, 2) **Never reset**: train from scratch for 3,000 minibatches on \( t = 0 \) but train 500 minibatches starting from \( \hat{D}^{t-1}_i \) if \( t > 0 \), 3) **Reset every 10 iterations**: train from scratch for 3,000 minibatches if \( t \mod 10 = 0 \) else finetune for just 500 minibatches starting from \( \hat{D}^{t-1}_i \).

8 Conclusions

We have introduced DREAM, a model-free self-play deep reinforcement learning algorithm. DREAM converges to a Nash equilibrium in two-player zero-sum imperfect-information games, and more broadly to an extensive-form coarse correlated equilibrium in all other games. In practise, DREAM achieves state-of-the-art performance among model-free RL algorithms in imperfect information games, beating the existing baseline NFSP \[13\] by two orders of magnitude with respect to sample complexity in Leduc Hold’em and reaching a far lower exploitability than NFSP after multiple days of training in Flop Hold’em Poker.
Figure 1: Ablation studies in Leduc. **Top-left:** Comparing $\epsilon$ for DREAM. **Top-right:** Various numbers of trajectories sampled per DREAM iteration. **Bottom-left:** Various numbers of minibatch updates of DREAM’s $\hat{Q}$ per iteration. **Bottom-right:** Modes of resetting the parameters of $\hat{D}$ in DREAM.

Figure 2: Exploitability comparison of DREAM with other algorithms in Leduc poker (top) and Flop Hold’em Poker (bottom). DREAM performs competitively with non-model-free Neural CFR methods (left), and outperforms all other model-free neural methods (right). The number of traversals for SD-CFR variants were chosen such that the number of nodes traversed per iteration is approximately equal to that of DREAM.
9 Broader Impact

DREAM seeks to find equilibria in imperfect-information games without relying on an explicit model of the interaction. In contrast to most other works in deep reinforcement learning, our algorithms are aiming to find equilibrium policies. This is particularly relevant for iterative decision-making processes - both competitive [6] and cooperative [11].

Beneficial future applications of DREAM after further research and development could include political and market management decision support, negotiation and agreement in self-driving car fleets, and other decision making processes. Equilibrium computation is an essential part of robust multi-agent decision making.

Potential adverse applications most prominently include providing easy access to tools that aid cheating in online imperfect-information games with our open-sourced implementation of DREAM. For poker, the most prominent imperfect-information game and the one where cheating would have the most significant consequences, successful domain-specific algorithms already exist [7], so this publication should not add significant additional risk in that domain.

10 Acknowledgements

We thank Yannis Wells for providing feedback on drafts of this paper and interesting discussions.

References

[1] Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The Hanabi challenge: A new frontier for AI research. arXiv preprint arXiv:1902.00506, 2019.

[2] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold’em poker is solved. Science, 347(6218):145–149, 2015.

[3] Noam Brown, Sam Ganzfried, and Tuomas Sandholm. Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit texas hold’em agent. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, pages 7–15. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[4] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. In International Conference on Machine Learning, pages 793–802, 2019.

[5] Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. Science, page eaao1733, 2017.

[6] Noam Brown and Tuomas Sandholm. Solving imperfect-information games via discounted regret minimization. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 1829–1836, 2019.

[7] Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. Science, page eaay2400, 2019.

[8] Andrea Celli, Alberto Marchesi, Gabriele Farina, and Nicola Gatti. No-regret learning dynamics for extensive-form correlated and coarse correlated equilibria. arXiv preprint arXiv:2004.00603, 2020.

[9] Trevor Davis, Martin Schmid, and Michael Bowling. Low-variance and zero-variance baselines for extensive-form games. arXiv preprint arXiv:1907.09633, 2019.

[10] Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Stochastic regret minimization in extensive-form games, 2020.

[11] Sam Ganzfried and Tuomas Sandholm. Potential-aware imperfect-recall abstraction with earth mover’s distance in imperfect-information games. In AAAI, pages 682–690, 2014.

[12] Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In AAAI, volume 4, pages 709–715, 2004.
[13] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[14] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[15] Peter Jin, Kurt Keutzer, and Sergey Levine. Regret minimization for partially observable deep reinforcement learning. *arXiv preprint arXiv:1710.11424*, 2017.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Marc Lanctot. *Monte Carlo sampling and regret minimization for equilibrium computation and decision-making in large extensive form games*. PhD thesis, University of Alberta, 2013.

[18] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*, pages 1078–1086, 2009.

[19] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4190–4203, 2017.

[20] Hui Li, Kailiang Hu, Shaohua Zhang, Yuan Qi, and Le Song. Double neural counterfactual regret minimization. In *International Conference on Learning Representations*, 2020.

[21] Viliam Lisy and Michael Bowling. Equilibrium approximation quality of current no-limit poker bots. *arXiv preprint arXiv:1612.07547*, 2016.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[23] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[24] Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Remi Munos, Julien Perolat, Marc Lanctot, Audrunas Gruslys, Jean-Baptiste Lespiau, and Karl Tuyls. Neural replicator dynamics. *arXiv preprint arXiv:1906.00190*, 2019.

[25] Julien Perolat, Remi Munos, Jean-Baptiste Lespiau, Shayegan Omidshafiei, Mark Rowland, Pedro Ortega, Neil Burch, Thomas Anthony, David Balduzzi, Bart De Vylder, et al. From poincaré recurrence to convergence in imperfect information games: Finding equilibrium via regularization. *arXiv preprint arXiv:2002.08456*, 2020.

[26] Martin Schmid, Neil Burch, Marc Lanctot, Matej Moravčik, Rudolf Kadlec, and Michael Bowling. Variance reduction in monte carlo counterfactual regret minimization (vr-mccfr) for extensive form games using baselines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2157–2164, 2019.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[28] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[29] Finnegan Southey, Michael P Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes’ bluff: Opponent modelling in poker. *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 550—558, 2005.

[30] Sriram Srinivasan, Marc Lanctot, Vinicius Zambaldi, Julien Pérolat, Karl Tuyls, Rémi Munos, and Michael Bowling. Actor-critic policy optimization in partially observable multi-agent environments. In *Advances in neural information processing systems*, pages 3422–3435, 2018.

[31] Eric Steinberger. Single deep counterfactual regret minimization. *arXiv preprint arXiv:1901.07621*, 2019.
[32] Oskari Tammelin. Solving large imperfect information games using cfr+. *arXiv preprint arXiv:1407.5042*, 2014.

[33] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

[34] Harm Van Seijen, Hado Van Hasselt, Shimon Whiteson, and Marco Wiering. A theoretical and empirical analysis of expected sarsa. In *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*, pages 177–184. IEEE, 2009.

[35] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[36] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015.

[37] Kevin Waugh, Dustin Morrill, James Andrew Bagnell, and Michael Bowling. Solving games with functional regret estimation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[38] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736, 2008.
A  Rules of Flop Hold’em Poker

Flop Hold’em Poker (FHP) is a two-player zero-sum game. The positions of the two players alternate after each hand. When acting, players can choose to either fold, call, or raise. When a player folds, the money in the pot is awarded to the other player. When a player calls, the player places money in the pot to equal to the opponent’s contribution. Raising means adding more chips to the pot than the opponent’s contribution.

A round ends when a player calls (and both players have acted). There can be at most three raises in each betting round. Raises are always in increments of $100. At the start of a hand, players are dealt two private cards from a standard 52-card deck. P1 must contribute $50 to the pot and P2 must contribute $100. A betting round then occurs starting with P1. After the round ends, three community cards are revealed face up. Another round of betting starts, but P2 acts first this time. At the end of this betting round, if no player has folded, the player with the strongest five-card poker hand wins, where the player’s two private cards and the three community cards are taken as their five-card hand. In the case of a tie, the pot is split evenly between both players.

B  Rules of Leduc Poker

Leduc Poker differs from Flop Hold’em poker in that the deck only six cards, made from two suits \{a, b\} and three ranks \{J, Q, K\}. Like FHP, the game consists of two rounds. At the start of the game, each player contributes $50, called the ante, to the pot and is handed one private card. There can be at most two raises per round, where the bet-size is fixed at $100 in the first round, and $200 in the second. When the game transitions from the first to the second betting round, one public card is revealed. If no player folded, the strongest hand wins. If the rank of a player’s private card matches that of the public card, they win. If not, \(K > Q > J\) determines the strongest hand. If both players’ private cards have the same rank, the pot is split between them.

C  Approximate lower bound on the Best Response using RL

When agent \(i\)’s policy is frozen, computing a best response policy for \(-i\) can be seen as a partially observable single-agent problem. We leverage this by running a deep RL algorithm to make best response computations feasible in bigger games. While almost any deep RL algorithm could be used, we chose DDQN \([33, 36]\). We train DDQN for 50,000 iterations with a batch size of 2048 and 512 steps per iteration. The circular buffer can hold 400,000 infostates. The target network is updated every 300 iterations. DDQN’s exploration parameter decays exponentially from 0.3 to 0.02.

After training, we run 1 million hands between the agent and the learned approximate BR to obtain a lower bound on the actual BR with minimal uncertainty (95% confidence interval of 5mBB/G in our experiments). This computation is run on 6 CPUs and takes 12 hours per datapoint.

Figure 3 shows the winnings during training, average across both agents. This sample was taken on DREAM iteration 180 (out of around 500) and is representative of the general trend. Note that this includes exploration and is thus not representative of the approximate BR computed in the end, where no exploration is done.

![Figure 3: Convergence of DDQN as the approximate best response against DREAM on iteration 180 in Flop Hold’em Poker.](image)
We also found by running multiple instances of DDQN that the variance in this evaluation is fairly low, with all DDQN replicas arriving at similar exploitabilities.

D Proofs of Theorems

D.1 Mapping notation principles

Our notation is more closely related to that of the reinforcement learning community than the conventional notation in the CFR literature. Most importantly, what’s usually referred to as $A$ in CFR \cite{18} is $A_i$ in our notation. The transition function $T$ used in our work is usually referred to as chance in the conventional notation. Moreover, the conventional notation assumes only one player acts in each node. We generalise this by considering opponent actions part of the transition function from the perspective of agent $i$.

D.2 Bound on DREAM’s regret

Theorem 1. Suppose a series of policies $\pi^t$ are chosen via a CFR learning rule with outcome sampling, using the $\epsilon$-regret-matching sampling policy $\xi^t_i$ for the traversing agent described in \cite{3}. Let $d_i$ be the maximum number of sequential decision points for agent $i$ (i.e. the game depth for $i$). For any $p \in [0, 1)$, the average regret $R^T_i$ for agent $i$ over this series of policies is bounded by

$$R^T_i \leq \frac{\Delta}{\sqrt{T}} \left( |I_i| \sqrt{|A_i|} + 2 \left( \frac{|A_i|}{\epsilon} \right)^{d_i} \sqrt{2 \log \frac{1}{p}} \right)$$

(9)

with probability $1 - p$.

Proof. Let $R^T_i$ and $\tilde{R}^T_i$ be the average regret under MC-CFR and CFR, respectively.

We start from the original CFR bound in \cite{38}, Theorem 4: If agent $i$ selects actions according to CFR then $\tilde{R}^T_i \leq \Delta |I_i| \sqrt{\frac{|A_i|}{T}}$.

Next, we apply the bound from \cite{10}, Proposition 1: If agent $i$ selects actions according to MC-CFR then

$$P \left[ R^T(u) \leq \tilde{R}^T(u) + (M + \tilde{M}) \sqrt{\frac{2}{T} \log \frac{1}{p}} \right] \geq 1 - p$$

(10)

where $M$ and $\tilde{M}$ are bounds on the difference between the minimum and maximum loss in the full and sampled game, respectively. Trivially, $M \leq \Delta$. According to \cite{10} Lemma 2, if agent $i$ plays some sampling policy $\xi^t_i$ and the opponent agents their true policy $\pi^t_i$, then $M \leq \max_z \Delta \xi^t_i[z]$, where $\xi^t_i[z]$ is the probability that $\xi^t_i$ plays to terminal node $z$ assuming the opponent and chance do.

DREAM uses an $\epsilon$-regret matching sampling scheme for agent $i$; therefore, we can easily bound $\xi^t_i[z] \leq \left( \frac{\epsilon}{|A_i|} \right)^{d_i}$, where $d_i^{\max}$ is the maximum number of sequential decision points for agent $i$ on any path of the game.

Putting this all together, we find that for any $p \in [0, 1)$,

$$R^T_i \leq \Delta |I_i| \sqrt{\frac{|A_i|}{T}} + \left( \Delta + \Delta \left( \frac{|A_i|}{\epsilon} \right)^{d_i} \right) \sqrt{\frac{2}{T} \log \frac{1}{p}}$$

(11)

with probability $1 - p$. 

\quare
Importantly, this theorem does not imply that the actions chosen by the MC-CFR learning procedure are themselves no-regret; indeed, since DREAM uses an off-policy sampling scheme its regret during "training" may be arbitrarily high. The theorem instead implies that the historical average of the CFR policies $\pi^*$, which is the policy returned by the DREAM algorithm, converges to a Nash equilibrium at this rate in two-player zero-sum games. In $n$-player general-sum games, it also converges to an extensive-form coarse correlated equilibrium at this rate [8].

### E Neural Network Architecture

We use the neural architecture demonstrated to be successful with Deep CFR and NFSP [4] (see figure 4). Slight differences between the networks are that we apply a softmax function to the last layer of NFSP’s and Deep CFR’s average networks like the original papers do. Moreover, DREAM’s $Q$ network requires the input of both player’s private cards and thus has a slightly adjusted input layer.

![Neural Network Architecture](image)

Figure 4: Neural network architecture as in [4]. We set D=64 in all experiments.