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

Eric Steinberger[*†] **Adam Lerer**[*] **Noam Brown**[*]

Abstract

We introduce DREAM, a regret-based deep reinforcement learning algorithm that converges to an equilibrium in imperfect-information multi-agent settings. Our primary contribution 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 in order 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 (Mnih et al. 2015; van Hasselt, Guez, and Silver 2015; Wang, de Freitas, and Lanctot 2015; Schulman et al. 2017). 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 (Heinrich and Silver 2016; Srinivasan et al. 2018; Brown et al. 2019; Steinberger 2019; Li et al. 2020; Omidshafiei et al. 2019). Among these, neural forms of counterfactual regret minimization (CFR) (Zinkevich et al. 2008; Lanctot et al. 2009; Tammelin 2014; Brown and Sandholm 2019a) have achieved the best performance (Brown et al. 2019; Steinberger 2019; Li et al. 2020). This mirrors the success of tabular CFR, variants of which are state-of-the-art among tabular equilibrium-finding algorithms (Brown and Sandholm 2019a) and which have been used in every major AI milestone in the domain of poker (Bowling et al. 2015; Moravčík et al. 2017; Brown and Sandholm 2017; Brown and Sandholm 2019b).

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 (Ganzfried and Sandholm 2014; Brown, Ganzfried, and Sandholm 2015). 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 (Schmid et al. 2019; Davis, Schmid, and Bowling 2019).

DREAM minimizes regret and converges to an ϵ-Nash equilibrium in two-player zero-sum games with ϵ proportional to the neural modeling error, and more generally converges to an ϵ- 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 (Hansen, Bernstein, and Zilberstein 2004). We consider a game with $\mathcal{N} = \{1, 2, ..., N\}$ 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. $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times ... \times \mathcal{A}_N$ is the space of joint actions. $\mathcal{A}_i(w)$ denotes the legal actions for agent i at w and $a =$

[*]Facebook AI Research (FAIR)

[†]Climate Science (https://climate-science.com)

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

$(a_1, a_2, ..., a_N) \in \mathcal{A}$ 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 $\mathcal{R}_i(w)$. Upon transition from world state w to w' via joint action a, agent i receives an **observation** O_i from a function $\mathcal{O}_i(w, a, w')$.

A **history** (also called a **trajectory**) is a finite sequence of legal actions and world states, denoted $h = (w^0, a^0, w^1, a^1, ..., w^t)$. The reward i obtained in the last world state of a history h is $\mathcal{R}_i(h)$ 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 $s_i = (O_i^0, a_i^0, O_i^1, a_i^1, ..., O_i^t)$. 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 $h \in s_i$. An observation O_i^t 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 $h' \sqsubset h$ to indicate h' is a history reached on the path to h.

An agent's **policy** π_i is a function mapping from an infostate to a probability distribution over actions. A **policy profile** π is a tuple $(\pi_1, \pi_2, ..., \pi_N)$. The policy of all agents other than i is denoted π_{-i}. A policy for a history h is denoted $\pi_i(h) = \pi_i(s_i(h))$ and $\pi(h) = (\pi_1(s_1(h)), \pi_2(s_2(h)), ..., \pi_N(s_N(h)))$. We also define the transition function $\mathcal{T}(h, a_i, \pi_{-i})$ as a function drawing actions for $-i$ from π_{-i} to form $a = a_i \cup a_{-i}$ and to then return the history h' from $\mathcal{T}(w_{last}, a)$, where w_{last} 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 π 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, π^* 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 π is $e(\pi) = \sum_{i \in \mathcal{N}} \max_{\pi'_i} v_i(\pi'_i, \pi_{-i})$.

The **reach** $x^\pi(h)$ of a history h is the product of all action probabilities under π and all transition function probabilities leading to h. Formally, for $h^t = (w^0, a^0, w^1, a^1, ..., w^t)$, $x^\pi(h^t) = \Pi_{h^n \sqsubset h^t, i \in \mathcal{N}} \pi_i(h^n, a_i^n) \tau(w^n, a_i^n, w^{n+1})$. The **agent reach** $x_i^\pi(h^t)$ of a history h^t is the product of the probabilities for all agent i actions leading to h^t. Formally, $x_i^\pi(h^t) = \Pi_{h^n \sqsubset h^t}(h^n, a_i^n)$. We similarly define the agent reach $x_i^\pi(s_i)$ of infostate $s_i = (O_i^0, a_i^0, O_i^1, a_i^1, ..., O_i^t)$ as $x_i^\pi(s_i) = \Pi_t(a_i^t)$. 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, $x_{-i}^\pi(s_i) = \sum_{h^t \in \mathcal{H}(s_i)} \Pi_{h^n \sqsubset h^t, i \in \mathcal{N} \setminus \{i\}} \pi_i(h^n, a_i^n)$. If $x_i^\pi(s_i) > 0$ then external reach is also $\sum_{h \in \mathcal{H}(s_i)} \frac{x^\pi(h)}{x_i^\pi(s_i)}$.

3 Related Work

Tabular CFR methods are limited by the need to visit a given state to update the policy played in it (Zinkevich et al. 2008; Brown and Sandholm 2019a; Tammelin 2014; Lanctot et al. 2009). In large games that are infeasible to fully traverse, domain-specific abstraction schemes (Ganzfried and Sandholm 2014; Brown, Ganzfried, and Sandholm 2015) can shrink the game to a manageable size by clustering states into buckets. This *abstracted* version of the game is solved during training and then mapped back onto the full game. These techniques work in many games (Lisy and Bowling 2016), but they most often require expert knowledge to design well 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* (Moravčík et al. 2017). 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 (Hessel et al. 2018; Silver et al. 2017).

Neural Fictitious Self-Play (NFSP) (Heinrich and Silver 2016) approximates extensive-form fictitious play using deep reinforcement learning from single trajectories. NFSP was the first deep reinforcement learning 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 (Srinivasan et al. 2018; Lanctot et al. 2017).

Deep CFR is a fully parameterized variant of CFR that requires no tabular sampling (Brown et al. 2019). Deep CFR substantially outperformed NFSP (which in turn outperforms or closely matches competing algorithms (Srinivasan et al. 2018)). *Single Deep CFR* (SD-CFR) (Steinberger 2019) removed the need for an average network in Deep CFR and thereby enabled better convergence 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 (Li et al. 2020). *Advantage Regret Minimization (ARM)* (Jin, Keutzer, and Levine 2017) is a regret-based policy gradient algorithm designed for single agent environments. Before neural networks were considered, *Regression CFR (R-CFR)* (Waugh et al. 2015) used regression trees to estimate regrets in CFR and CFR$^+$. However, R-CFR seems to be incompatible with sparse sampling techniques (Srinivasan et al. 2018), rendering it less scalable.

4 Review of Counterfactual Regret Minimization (CFR)

Counterfactual Regret Minimization (CFR) (Zinkevich et al. 2008) is an iterative policy improvement algorithm that computes a new policy profile π^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 π_i^t 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 $r_i^t(s_i, a_i) = x_{-i}^{\pi^t}(s_i)\big(v_i^{\pi^t}(s_i, a_i) - v_i^{\pi^t}(s_i)\big)$. $r_i^t(s_i, a_i)$ 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 $R_i^T(s_i, a_i) = \frac{\sum_{t=1}^T r_i^t(s_i, a_i)}{T}$. CFR defines the policy for agent i on iteration $t+1$ based on their overall regret as $\pi_i^{t+1}(s_i, a_i) =$

$$\begin{cases} \frac{R_i^t(s_i,a_i)_+}{\sum_{a_i' \in \mathcal{A}_i(s_i)} R_i^t(s_i,a_i')_+} & \text{if} \sum_{a_i' \in \mathcal{A}_i(s_i)} R_i^t(s_i, a_i')_+ > 0 \\ \frac{1}{|\mathcal{A}_i(s_i)|} & \text{otherwise} \end{cases} \tag{1}$$

where $x_+ = max(x, 0)$ The initial policy is set to uniform random. The **average policy** $\bar{\pi}_i^T$ is

$$\bar{\pi}_i^T(s_i, a_i) = \frac{\sum_{t=1}^T x_i^{\pi^t}(s_i)\pi_i^t(s_i, a_i)}{\sum_{t=1}^T x_i^{\pi^t}(s_i)} \tag{2}$$

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

4.1 Linear CFR

Linear CFR (Brown and Sandholm 2019a) is a variant of CFR that weighs the updates to the regret and average policy on iteration t by t. Thus, under Linear CFR, $R_i^T(s_i, a_i) = \sum_{t=1}^T (t r_i^t(s_i, a_i))$ and $\bar{\pi}_i^T(s_i, a_i) = \frac{\sum_{t=1}^T (t x_i^{\pi^t}(s_i)\pi_i^t(s_i, a_i))}{\sum_{t=1}^T (t x_i^{\pi^t}(s_i))}$. This modification accelerates CFR by two orders of magnitude.

4.2 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)** (Lanctot et al. 2009). 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 $\mathcal{T}$). In ES, a single action is sampled for each non-traverser agent and a single outcome is sampled from $\mathcal{T}$, but every traverser action is explored. When agent i is the traverser in ES or OS, all other agents sample actions according to π_{-i}. In OS, we use a sampling policy $\xi_i^t(s_i)$ profile rather than π^t to choose actions during play. $\xi_i^t(s_i)$ is defined separately for the traverser i and any other agent j

$$\xi_i^t(s_i, a_i) = \epsilon \frac{1}{|\mathcal{A}_i(s_i)|} + (1-\epsilon)\pi_i^t(s_i, a_i)$$

$$\xi_j^t(s_j, a_j) = \pi_j^t(s_j, a_j) \tag{3}$$

For a thorough and more general description of MC-CFR see (Lanctot et al. 2009).

4.3 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 (Schmid et al. 2019; Davis, Schmid, and Bowling 2019). Given that OS ultimately reaches a terminal history z, the **baseline-adjusted sampled expected value** in history h is

$$\begin{aligned} &\tilde{v}_i^{\pi^t}(s_i(h), a_i|z) = \tilde{v}_i^{\pi^t}(h, a_i|z) \\ &= b_i(h, a_i) + \\ &\begin{cases} \frac{\sum_{h'}\{p(h'|\mathcal{T}(h,a_i,\pi_{-i}^t))\tilde{v}_i^{\pi^t}(h'|z)\} - b_i(h,a_i)}{\xi_i^t(s_i,a_i)} & \text{if } (h,a_i) \sqsubseteq z \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

$$\tilde{v}_i^{\pi^t}(s_i(h)|z) = \tilde{v}_i^{\pi^t}(h|z) = \begin{cases} \mathcal{R}_i(h) & \text{if } h = z \\ \sum_{a_i \in \mathcal{A}_i(h)} \pi_i^t(h, a_i)\tilde{v}_i^{\pi^t}(h, a_i) & \text{otherwise} \end{cases} \tag{5}$$

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 (Schmid et al. 2019; Davis, Schmid, and Bowling 2019).

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

Deep CFR (Brown et al. 2019) 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 B_i^d (Vitter 1985), 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 π_i^{t+1} 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 $D_i^T(s_i, a_i) = \frac{R_i^T(s_i,a_i)}{\sum_{t=1}^T (x_{-i}^{\pi^t}(s_i))}$. Deep CFR divides

$R_i^T(s_i, a_i)$ by $\sum_{t=1}^{T}(x^{\pi^t}_{-i}(s_i))$ 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 1 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 $\hat{S}_i$ is stored in a separate reservoir buffer B_i^s and collected during the same traversals that data for B_i^v 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 (RE)gret 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 (Burch, Moravcik, and Schmid 2019). On each iteration t at each encountered infostate s_i, we obtain estimated advantages $\hat{D}_i^t(s_i, a_i|\theta_i^t)$ for the actions $a_i \in \mathcal{A}_i(s_i)$ by inputting s_i into the advantage network parameterized by θ_i^t . From these estimated advantages, we obtain a policy $\pi_i^t(s_i)$ via regret matching as shown in Equation 1, except using $\hat{D}_i^t$ in place of R_i^t, and picking the action with the highest $\hat{D}_i^t$ 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 (Lanctot et al. 2009) with the sampling policy profile introduced in 4.3. where $\epsilon > 0$ is the exploration parameter. To compensate for the shifted sampling distribution due to ϵ, we weigh data samples stored in B_i^d by $\frac{1}{x_i^{\xi^t}(s_i)}$ 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 (Davis, Schmid, and Bowling 2019), the tabular form of which is described in section 4.3.

DREAM uses a baseline network $\hat{Q}_i^t(s^*(h), a_i|\phi_i^t)$ parameterized by ϕ_i^t as a baseline, where s^* is the set of infostates for all players at h, $s^*(h) = (s_0(h), ..., s_N(h))$.[1] $\hat{Q}_i^t$ is fine-tuned from $\hat{Q}_i^{t-1}$ when $t > 0$ and initialized with random weights on $t = 0$. It is trained using expected SARSA (Van Seijen et al. 2009) on a circular buffer B_i^q. Formally, it updates ϕ_i^t to minimize its mean squared error (MSE) to the target value sample $\mathcal{R}_i(h) + \sum_{a_i' \in \mathcal{A}_i} \pi^t(s_i(h'), a_i')\hat{Q}_i^t(s^*(h'), a_i')$, where h' is drawn from $\mathcal{T}(h, a_i, \pi^t_{-i})$.

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 played leading to h', is

$$\tilde{v}^{\pi^t}_{i,DREAM}(h, a_i|z) = \begin{cases} \hat{Q}_i^t(s^*(h), a_i) - \frac{\tilde{v}^{\pi^t}_{i,DREAM}(h'|z) - \hat{Q}_i^t(s^*(h), a_i)}{\xi^t(s_i, a_i)} & \text{if } a_i = a_i' \\ \hat{Q}_i^t(s^*(h), a_i) & \text{otherwise} \end{cases} \tag{6}$$

$$\tilde{v}^{\pi^t}_{i,DREAM}(h|z) = \begin{cases} \mathcal{R}_i(h) & \text{if } h = z \\ \sum_{a_i \in \mathcal{A}_i(h)} \pi_i^t(h, a_i)\tilde{v}^{\pi^t}_{i,DREAM}(h, a_i|z) & \text{otherwise} \end{cases} \tag{7}$$

and $\tilde{v}^{\pi^t}_{i,DREAM}(s_i(h), a_i|z) = \tilde{v}^{\pi^t}_{i,DREAM}(h, a_i|z)$ and $\tilde{v}^{\pi^t}_{i,DREAM}(s_i(h)|z) = \tilde{v}^{\pi^t}_{i,DREAM}(h|z)$.

We define the **sampled immediate advantage** as

$$\tilde{d}^t_{i,DREAM}(s_i, a_i) = \tilde{v}^{\pi^t}_{i,DREAM}(s_i, a_i) - \tilde{v}^{\pi^t}_{i,DREAM}(s_i) \tag{8}$$

Note that the expectation of $\tilde{d}^t_{i,DREAM}(s_i, a_i)$ is $\frac{r_i^t(s_i,a_i)}{x^{\pi^t_{-i}}(s_i,a_i)}$. Because we defined the sampling policy for $-i$ as $\xi^t_{-i}(s_i, a_i) = \pi^t_{-i}(s_i, a_i)$, the expectation of samples added to the advantage buffer B_i^d on iteration t in history h will be proportional to $r_i^t(s_i(h), a_i)$. Furthermore, because from agent i's perspective histories are sampled based on their information, the expectation for data added to B_i^d in infostate s_i is proportional to $r_i^t(s_i, a_i)$. Thus, training $\hat{D}_i^t$ on B_i^d means

[1] We 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.

it approximates a quantity proportional to the average regret as defined in section 4. To implement Linear CFR (Brown and Sandholm 2019a) in DREAM, we weigh the neural training loss of $\tilde{d}_i^t$ in B_i^d by t.

5.3 The average policy

In order to play according to the average policy, we follow the approach used in SD-CFR (Steinberger 2019), described in Section 4.4, in which the advantage model from each iteration, or a random subset thereof, is stored on disk in a set B_i^M and one is sampled randomly at test time. That model is then used to determine the policy for the whole game. When implementing linear CFR, the probability of sampling model t is proportional to t. When needed, the explicit average policy probabilities can be computed in $\mathcal{O}(|B_i^M|)$ time by evaluating and averaging strategies derived from all networks in B_i^M.

Deep CFR's method of computing the average policy by training a separate neural network on action probability vectors collected during training (Brown et al. 2019) is also compatible, but to correct for sampling bias, the neural training loss of samples from iteration t is weighted by $\frac{1}{x^{\xi}_{-i}}(h)$.

5.4 Convergence

The convergence of MC-CFR (Lanctot 2013) using function approximation to estimate infoset advantages has been previously studied in (Brown et al. 2019). 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 *approximately* minimizes the MSE, its convergence bound incurs an additional constant average regret term proportional to the maximum gap $\epsilon_{\mathcal{L}}$ between the model's MSE error on the samples and the error of the exact minimizer ((Brown et al. 2019), Theorem 1).

MC-CFR convergence results hold for any unbiased estimator of state-action values, and thus permit the use of arbitrary state-action baselines, although the rate of convergence is related to the variance of the baseline-adjusted estimator (i.e. the quality of the baseline). More details can be found in (Schmid et al. 2019). We derive a bound on the cumulative regret for DREAM in the in Appendix C.

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 (Brown et al. 2019), Single Deep CFR (SD-CFR) (Steinberger 2019), and Neural Fictitious Self-Play (NFSP) (Heinrich and Silver 2016). We chose Leduc poker (Southey et al. 2005) and Flop hold'em poker (Brown et al. 2019), two popular poker games, as benchmark domains. A description of these can be found in the appendix. We use Linear CFR (Brown and Sandholm 2019a) for DREAM and all variants of SD-CFR we evaluate. For all experiments, we plot the mean and standard deviation across three independent runs. Exploitability is measured in milli big blinds per game (mbb/g).

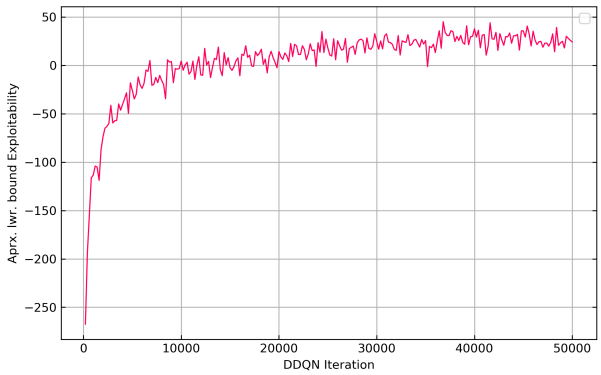

Figure 1: Convergence of DDQN as the approximate best response against DREAM on iteration 180 in Flop Hold'em Poker.

6.1 Approximate lower bound on the Best Response using RL

In FHP we measure an approximate lower bound on exploitability computed by training a DDQN agent (Wang, de Freitas, and Lanctot 2015; van Hasselt, Guez, and Silver 2015) against DREAM and running a number of hands between DREAM and the final iteration of DDQN and for evaluation.

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 (van Hasselt, Guez, and Silver 2015; Wang, de Freitas, and Lanctot 2015). 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 1 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.

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.

6.2 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 (Brown et al. 2019) 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 D.

On each iteration, $\hat{Q}_i^t$ is trained for 1000 minibatches of 512 samples using the Adam optimizer (Kingma and Ba 2014) with a learning rate of 0.001 and gradient norm clipping of 1 in both games. B_i^q has a capacity of 200,000. For comparibility, DREAM's value networks use the same training parameters as those SD-CFR (Brown et al. 2019; Steinberger 2019), unless otherwise stated. Both SD-CFR and DREAM use advantage buffers B_i^d 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_i^s 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 (Steinberger 2019). In FHP, we use 10,000 for SD-CFR as proposed by (Brown et al. 2019) 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 hyperparamters in Leduc as presented the original paper (Heinrich and Silver 2016). 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 3) and then compare DREAM to other algorithms (see figure 4). 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.

Though only loosely related to DREAM, we found it interesting that Deep CFR (Brown et al. 2019) 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 $\hat{D}_i^{t-1}$ 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 $\hat{D}_i^{t-1}$.

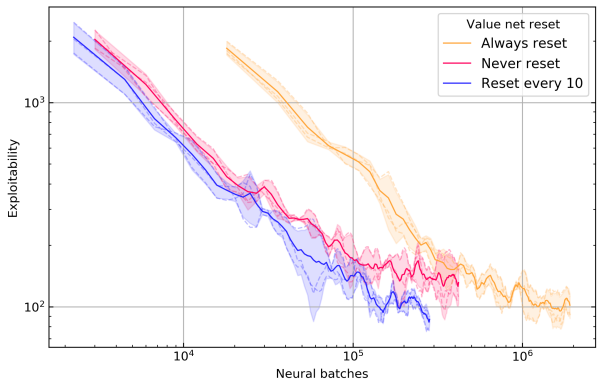

Figure 2: Modes of resetting DREAM's parameters of $\hat{D}$.

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 (Heinrich and Silver 2016) 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.

9 Ethics Statement

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 (Brown and Sandholm 2019a) and cooperative (Bard et al. 2019).

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 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 (Brown and Sandholm 2019b), so this publication should not add significant additional risk in that domain.

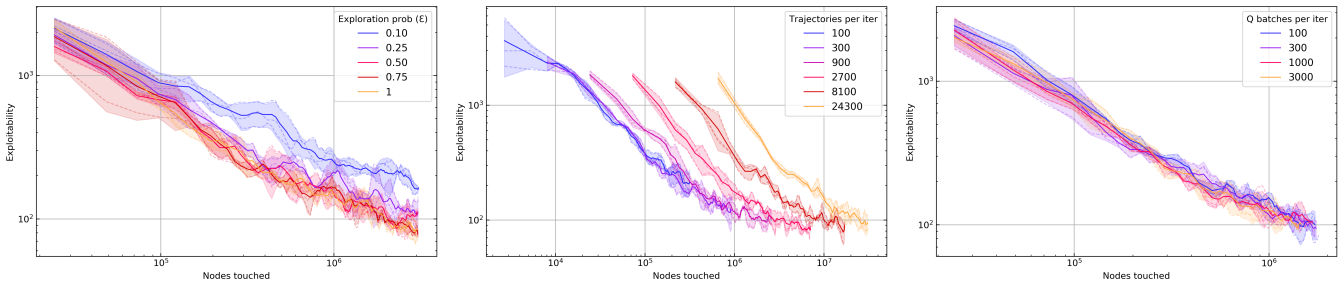

Figure 3: Ablation studies in Leduc. **Left**: ϵ for DREAM. **Middle**: Trajectories sampled per DREAM iteration. **Right**: Minibatch updates of DREAM's $\hat{Q}$ per iteration.

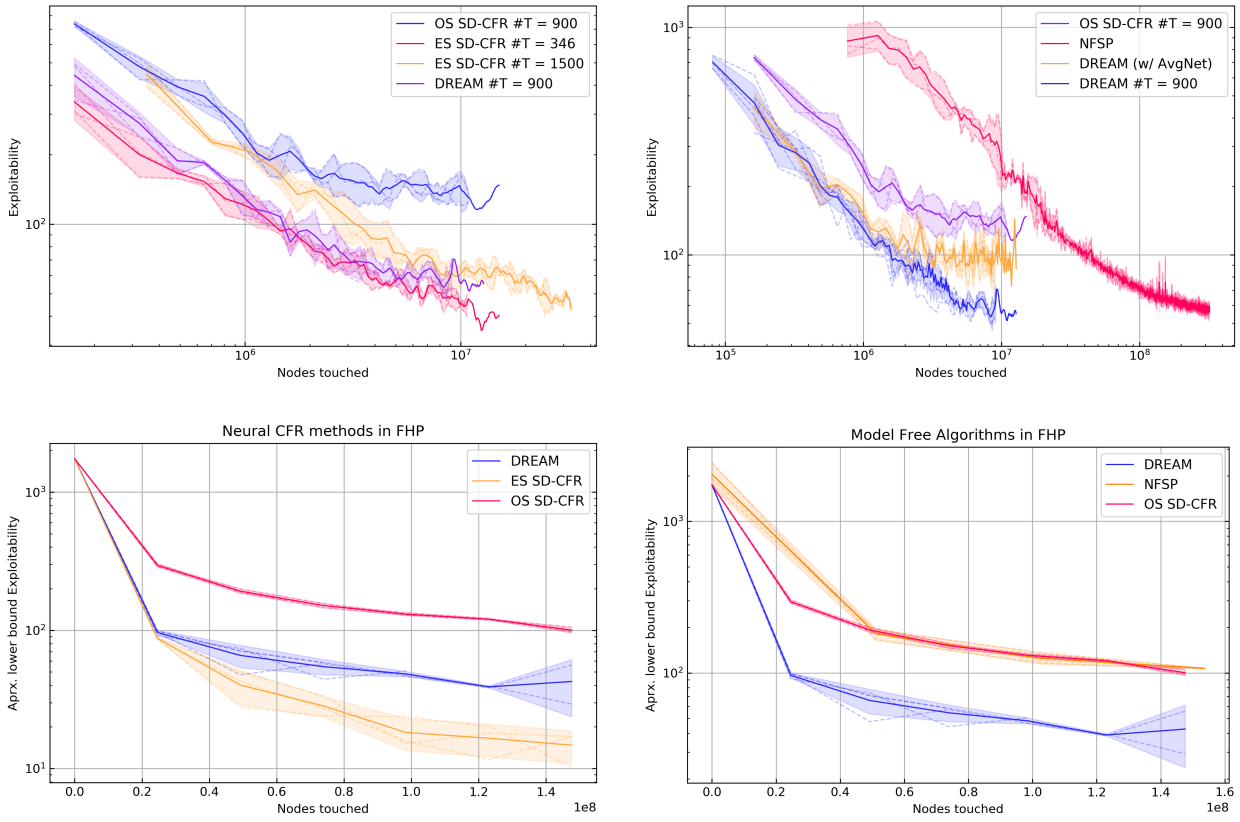

Figure 4: Exploitability of DREAM and other algorithms in Leduc poker (top) and Flop Hold'em Poker (bottom). DREAM performs competitively with non-model-free Neural CFR methods (left), and outperforms all other model-free neural methods (right). The number of traversals for SD-CFR variants were chosen such that the number of nodes traversed per iteration is approximately equal to that of DREAM.

References

[Bard et al. 2019] Bard, N.; Foerster, J. N.; Chandar, S.; Burch, N.; Lanctot, M.; Song, H. F.; Parisotto, E.; Dumoulin, V.; Moitra, S.; Hughes, E.; et al. 2019. The Hanabi challenge: A new frontier for AI research. *arXiv preprint arXiv:1902.00506*.

[Bowling et al. 2015] Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up limit hold'em poker is solved. *Science* 347(6218):145–149.

[Brown and Sandholm 2017] Brown, N., and Sandholm, T. 2017. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* eaao1733.

[Brown and Sandholm 2019a] Brown, N., and Sandholm, T. 2019a. Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1829–1836.

[Brown and Sandholm 2019b] Brown, N., and Sandholm, T. 2019b. Superhuman AI for multiplayer poker. *Science* eaay2400.

[Brown et al. 2019] Brown, N.; Lerer, A.; Gross, S.; and Sandholm, T. 2019. Deep counterfactual regret minimization. In *International Conference on Machine Learning*, 793–802.

[Brown, Ganzfried, and Sandholm 2015] 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 *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 7–15. International Foundation for Autonomous Agents and Multiagent Systems.

[Burch, Moravcik, and Schmid 2019] Burch, N.; Moravcik, M.; and Schmid, M. 2019. Revisiting cfr+ and alternating updates. *Journal of Artificial Intelligence Research* 64:429–443.

[Celli et al. 2020] Celli, A.; Marchesi, A.; Farina, G.; and Gatti, N. 2020. No-regret learning dynamics for extensive-form correlated and coarse correlated equilibria. *arXiv preprint arXiv:2004.00603*.

[Davis, Schmid, and Bowling 2019] Davis, T.; Schmid, M.; and Bowling, M. 2019. Low-variance and zero-variance baselines for extensive-form games. *arXiv preprint arXiv:1907.09633*.

[Farina, Kroer, and Sandholm 2020] Farina, G.; Kroer, C.; and Sandholm, T. 2020. Stochastic regret minimization in extensive-form games.

[Ganzfried and Sandholm 2014] Ganzfried, S., and Sandholm, T. 2014. Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. In *AAAI*, 682–690.

[Hansen, Bernstein, and Zilberstein 2004] Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, 709–715.

[Heinrich and Silver 2016] Heinrich, J., and Silver, D. 2016. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*.

[Hessel et al. 2018] Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Jin, Keutzer, and Levine 2017] Jin, P.; Keutzer, K.; and Levine, S. 2017. Regret minimization for partially observable deep reinforcement learning. *arXiv preprint arXiv:1710.11424*.

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

[Kovařík et al. 2019] Kovařík, V.; Schmid, M.; Burch, N.; Bowling, M.; and Lisỳ, V. 2019. Rethinking formal models of partially observable multiagent decision making. *arXiv preprint arXiv:1906.11110*.

[Lanctot et al. 2009] Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. 2009. Monte carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*, 1078–1086.

[Lanctot et al. 2017] Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Pérolat, J.; Silver, D.; and Graepel, T. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, 4190–4203.

[Lanctot 2013] Lanctot, M. 2013. *Monte Carlo sampling and regret minimization for equilibrium computation and decision-making in large extensive form games*. Ph.D. Dissertation, University of Alberta.

[Li et al. 2020] Li, H.; Hu, K.; Zhang, S.; Qi, Y.; and Song, L. 2020. Double neural counterfactual regret minimization. In *International Conference on Learning Representations*.

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

[Mnih et al. 2015] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

[Moravčík et al. 2017] Moravčík, M.; Schmid, M.; Burch, N.; Lisỳ, 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.

[Omidshafiei et al. 2019] Omidshafiei, S.; Hennes, D.; Morrill, D.; Munos, R.; Perolat, J.; Lanctot, M.; Gruslys, A.; Lespiau, J.-B.; and Tuyls, K. 2019. Neural replicator dynamics. *arXiv preprint arXiv:1906.00190*.

[Schmid et al. 2019] Schmid, M.; Burch, N.; Lanctot, M.; Moravcik, M.; Kadlec, R.; and Bowling, M. 2019. 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, 2157–2164.

[Schulman et al. 2017] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Silver et al. 2017] Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

[Southey et al. 2005] Southey, F.; Bowling, M. P.; Larson, B.; Piccione, C.; Burch, N.; Billings, D.; and Rayner, C. 2005. Bayes' bluff: Opponent modelling in poker. *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence* 550—-558.

[Srinivasan et al. 2018] Srinivasan, S.; Lanctot, M.; Zambaldi, V.; Pérolat, J.; Tuyls, K.; Munos, R.; and Bowling, M. 2018. Actor-critic policy optimization in partially observable multiagent environments. In *Advances in neural information processing systems*, 3422–3435.

[Steinberger 2019] Steinberger, E. 2019. Single deep counterfactual regret minimization. *arXiv preprint arXiv:1901.07621*.

[Tammelin 2014] Tammelin, O. 2014. Solving large imperfect information games using cfr+. *arXiv preprint arXiv:1407.5042*.

[van Hasselt, Guez, and Silver 2015] van Hasselt, H.; Guez, A.; and Silver, D. 2015. Deep reinforcement learning with double q-learning. *CoRR* abs/1509.06461.

[Van Seijen et al. 2009] Van Seijen, H.; Van Hasselt, H.; Whiteson, S.; and Wiering, M. 2009. A theoretical and empirical analysis of expected sarsa. In *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*, 177–184. IEEE.

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

[Wang, de Freitas, and Lanctot 2015] Wang, Z.; de Freitas, N.; and Lanctot, M. 2015. Dueling network architectures for deep reinforcement learning. *CoRR* abs/1511.06581.

[Waugh et al. 2015] Waugh, K.; Morrill, D.; Bagnell, J. A.; and Bowling, M. 2015. Solving games with functional regret estimation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Zinkevich et al. 2008] Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, 1729–1736.

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 thee 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 Proofs of Theorems

C.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 importnatly, what's usually referred to as A in CFR (Lanctot et al. 2009) is $\mathcal{A}_i$ in our notation. The transition function $\mathcal{T}$ 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.

C.2 Bound on DREAM's regret

This section is written in the notation typical for work on extensive form games. Due to the relation between the notation used in our paper and that used in prior work on extensive form games, the references in this proof remain theoretically sound (Kovařík et al. 2019).

Theorem 1. *Suppose a series of policies π^t are chosen via a CFR learning rule with outcome sampling, using the ϵ-regret-matching sampling policy ξ_i^t for the traversing agent described in 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_i^T for agent i over this series of policies is bounded by*

$$R_i^T \leq \frac{\Delta}{\sqrt{T}} \left(|\mathcal{I}_i| \sqrt{|\mathcal{A}_i|} + 2 \left(\frac{|\mathcal{A}_i|}{\epsilon} \right)^{d_i} \sqrt{2 \log \frac{1}{p}} \right) \quad (9)$$

with probability $1 - p$.

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

We start from the original CFR bound in (Zinkevich et al. 2008), Theorem 4: If agent i selects actions according to CFR then $\tilde{R}_i^T \leq \Delta |\mathcal{I}_i| \frac{\sqrt{|\mathcal{A}_i|}}{\sqrt{T}}$.

Next, we apply the bound from (Farina, Kroer, and Sandholm 2020), Proposition 1: If agent i selects actions according to MC-CFR then

$$P \left[R^T(u) \leq \tilde{R}^T(u) + (M + \tilde{M}) \sqrt{\frac{2}{T} \log \frac{1}{p}} \right] \geq 1 - p \quad (10)$$

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 (Farina, Kroer, and Sandholm 2020) Lemma 2, if agent i plays some sampling policy ξ_i^t and the opponent agents their true policy π_i^t, then $\tilde{M} \leq \max_{z \in \mathcal{Z}} \frac{\Delta}{\xi_i^t[z]}$, where $\xi_i^t[z]$ is the probability that ξ_i^t plays to terminal node z assuming the opponent and chance do.

DREAM uses an ϵ-regret matching sampling scheme for agent i; therefore, we can easily bound $\xi_i^t[z] \leq \left(\frac{\epsilon}{|\mathcal{A}_i|} \right)^{d_i}$, 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 $p \in [0, 1)$,

$$R_i^T \leq \Delta |\mathcal{I}_i| \frac{\sqrt{|\mathcal{A}_i|}}{\sqrt{T}} + \left(\Delta + \Delta \left(\frac{|\mathcal{A}_i|}{\epsilon} \right)^{d_i} \right) \sqrt{\frac{2}{T} \log \frac{1}{p}} \quad (11)$$

with probability $1 - p$.

□

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 π^t, 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 (Celli et al. 2020).

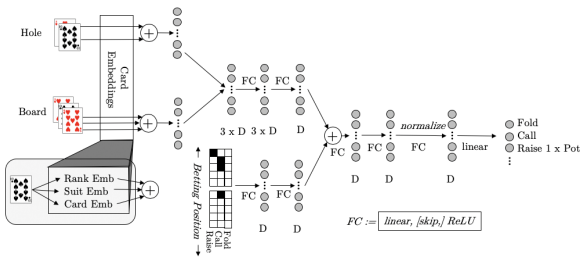

Figure 5: Neural network architecture as in (Brown et al. 2019). We set D=64 in all experiments.

D Neural Network Architecture

We use the neural architecture demonstrated to be successful with Deep CFR and NFSP (Brown et al. 2019) (see figure 5). 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.

References

[Bard et al. 2019] Bard, N.; Foerster, J. N.; Chandar, S.; Burch, N.; Lanctot, M.; Song, H. F.; Parisotto, E.; Dumoulin, V.; Moitra, S.; Hughes, E.; et al. 2019. The Hanabi challenge: A new frontier for AI research. *arXiv preprint arXiv:1902.00506*.

[Bowling et al. 2015] Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up limit hold'em poker is solved. *Science* 347(6218):145–149.

[Brown and Sandholm 2017] Brown, N., and Sandholm, T. 2017. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* eaao1733.

[Brown and Sandholm 2019a] Brown, N., and Sandholm, T. 2019a. Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1829–1836.

[Brown and Sandholm 2019b] Brown, N., and Sandholm, T. 2019b. Superhuman AI for multiplayer poker. *Science* eaay2400.

[Brown et al. 2019] Brown, N.; Lerer, A.; Gross, S.; and Sandholm, T. 2019. Deep counterfactual regret minimization. In *International Conference on Machine Learning*, 793–802.

[Brown, Ganzfried, and Sandholm 2015] 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 *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 7–15. International Foundation for Autonomous Agents and Multiagent Systems.

[Burch, Moravcik, and Schmid 2019] Burch, N.; Moravcik, M.; and Schmid, M. 2019. Revisiting cfr+ and alternating updates. *Journal of Artificial Intelligence Research* 64:429–443.

[Celli et al. 2020] Celli, A.; Marchesi, A.; Farina, G.; and Gatti, N. 2020. No-regret learning dynamics for extensive-form correlated and coarse correlated equilibria. *arXiv preprint arXiv:2004.00603*.

[Davis, Schmid, and Bowling 2019] Davis, T.; Schmid, M.; and Bowling, M. 2019. Low-variance and zero-variance baselines for extensive-form games. *arXiv preprint arXiv:1907.09633*.

[Farina, Kroer, and Sandholm 2020] Farina, G.; Kroer, C.; and Sandholm, T. 2020. Stochastic regret minimization in extensive-form games.

[Ganzfried and Sandholm 2014] Ganzfried, S., and Sandholm, T. 2014. Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. In *AAAI*, 682–690.

[Hansen, Bernstein, and Zilberstein 2004] Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, 709–715.

[Heinrich and Silver 2016] Heinrich, J., and Silver, D. 2016. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*.

[Hessel et al. 2018] Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Jin, Keutzer, and Levine 2017] Jin, P.; Keutzer, K.; and Levine, S. 2017. Regret minimization for partially observable deep reinforcement learning. *arXiv preprint arXiv:1710.11424*.

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

[Kovařík et al. 2019] Kovařík, V.; Schmid, M.; Burch, N.; Bowling, M.; and Lisỳ, V. 2019. Rethinking formal models of partially observable multiagent decision making. *arXiv preprint arXiv:1906.11110*.

[Lanctot et al. 2009] Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. 2009. Monte carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*, 1078–1086.

[Lanctot et al. 2017] Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Pérolat, J.; Silver, D.; and Graepel, T. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, 4190–4203.

[Lanctot 2013] Lanctot, M. 2013. *Monte Carlo sampling and regret minimization for equilibrium computation and decision-making in large extensive form games*. Ph.D. Dissertation, University of Alberta.

[Li et al. 2020] Li, H.; Hu, K.; Zhang, S.; Qi, Y.; and Song, L. 2020. Double neural counterfactual regret minimization. In *International Conference on Learning Representations*.

[Lisy and Bowling 2016] Lisy, V., and Bowling, M. 2016.

Equilibrium approximation quality of current no-limit poker bots. *arXiv preprint arXiv:1612.07547*.

[Mnih et al. 2015] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

[Moravčík et al. 2017] Moravčík, M.; Schmid, M.; Burch, N.; Lisỳ, 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.

[Omidshafiei et al. 2019] Omidshafiei, S.; Hennes, D.; Morrill, D.; Munos, R.; Perolat, J.; Lanctot, M.; Gruslys, A.; Lespiau, J.-B.; and Tuyls, K. 2019. Neural replicator dynamics. *arXiv preprint arXiv:1906.00190*.

[Schmid et al. 2019] Schmid, M.; Burch, N.; Lanctot, M.; Moravcik, M.; Kadlec, R.; and Bowling, M. 2019. 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, 2157–2164.

[Schulman et al. 2017] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Silver et al. 2017] Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

[Southey et al. 2005] Southey, F.; Bowling, M. P.; Larson, B.; Piccione, C.; Burch, N.; Billings, D.; and Rayner, C. 2005. Bayes' bluff: Opponent modelling in poker. *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence* 550—-558.

[Srinivasan et al. 2018] Srinivasan, S.; Lanctot, M.; Zambaldi, V.; Pérolat, J.; Tuyls, K.; Munos, R.; and Bowling, M. 2018. Actor-critic policy optimization in partially observable multiagent environments. In *Advances in neural information processing systems*, 3422–3435.

[Steinberger 2019] Steinberger, E. 2019. Single deep counterfactual regret minimization. *arXiv preprint arXiv:1901.07621*.

[Tammelin 2014] Tammelin, O. 2014. Solving large imperfect information games using cfr+. *arXiv preprint arXiv:1407.5042*.

[van Hasselt, Guez, and Silver 2015] van Hasselt, H.; Guez, A.; and Silver, D. 2015. Deep reinforcement learning with double q-learning. *CoRR* abs/1509.06461.

[Van Seijen et al. 2009] Van Seijen, H.; Van Hasselt, H.; Whiteson, S.; and Wiering, M. 2009. A theoretical and empirical analysis of expected sarsa. In *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*, 177–184. IEEE.

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

[Wang, de Freitas, and Lanctot 2015] Wang, Z.; de Freitas, N.; and Lanctot, M. 2015. Dueling network architectures for deep reinforcement learning. *CoRR* abs/1511.06581.

[Waugh et al. 2015] Waugh, K.; Morrill, D.; Bagnell, J. A.; and Bowling, M. 2015. Solving games with functional regret estimation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Zinkevich et al. 2008] Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, 1729–1736.