# Backplay: ‘Man muss immer umkehren’

**Cinjon Resnick**[*]
NYU
cinjon@nyu.edu

**Roberta Raileanu**[*]
NYU
rr3009@nyu.edu

**Sanyam Kapoor**
NYU
sanyam@nyu.edu

**Alexander Peysakhovich**
FAIR
alexpeys@fb.com

**Kyunghyun Cho**
NYU, FAIR
kyunghyun.cho@nyu.edu

**Joan Bruna**
NYU, FAIR
bruna@cims.nyu.edu

## Abstract

A long-standing problem in model-free reinforcement learning (RL) is that it requires a large number of trials to learn a good policy, especially in environments with sparse rewards. We explore a method to increase the sample efficiency of RL when we have access to demonstrations. Our approach, Backplay, uses a single demonstration to construct a curriculum for a given task. Rather than starting each training episode in the environment's fixed initial state, we start the agent near the end of the demonstration and move the starting point backwards during the course of training until we reach the initial state. We perform experiments in a competitive four-player game (Pommerman) and a path-finding maze game. We find that Backplay provides significant gains in sample complexity with a stark advantage in sparse reward settings. In some cases, it reached success rates greater than 50% and generalized to unseen initial conditions, while standard RL did not yield any improvement.

## 1 Introduction

An important goal of AI research is to construct agents that can enter new environments and learn how to achieve desired goals (Levine et al. 2016). A key paradigm for this task is deep reinforcement learning, which has succeeded in many environments, including complex zero-sum games such as Go, Poker, and Chess (Silver et al. 2016; Moravcík et al. 2017; Silver et al. 2017). However, training an RL agent can take a very long time, particularly in environments with sparse rewards. In these settings, the agent typically requires a large number of episodes to stumble upon positive rewards and learn even a moderately effective policy that can then be refined. Reward sparsity is often resolved via hand-engineering a dense reward function. Such reward shaping, while effective, can also change the set of optimal policies and have unintended side effects (Ng, Harada, and Russell 1999; Clark and Amodei 2016).

We consider an alternative technique for accelerating RL in sparse reward settings. The idea is to create a curriculum for the agent via reversing a single trajectory (i.e. state sequence) of reasonably good, but not necessarily optimal, behavior. That is, we start our agent at the end of a demonstration and let it learn a policy in this easier setup. We then move the starting point backward until the agent is training only on the initial state of the task. We call this technique **Backplay** (Figure 1).

[*]These two authors contributed equally.

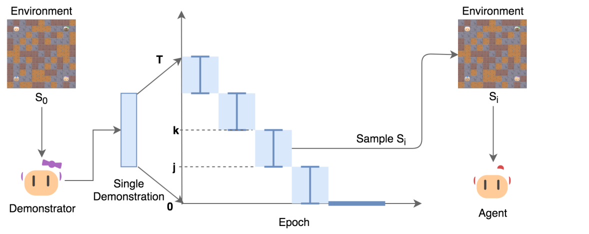


Figure 1. Backplay: We first collect a demonstration, from which we build a curriculum over the states. We then sample a state according to that curriculum and initialize our agent accordingly.

Our experiments are in a maze environment where the agent must find a path to a goal and Pommerman (MultiAgentLearning 2018), a complex stochastic four-player competitive game. We use those to empirically show that Backplay vastly decreases the number of samples required to learn a good policy. In addition, we see that a Backplay agent can outperform its corresponding demonstrator and even learn an optimal policy when using a sub-optimal demonstration. Notably, the Backplay agent is learns just as well in environments with a single sparse reward as in those with a dense shaped reward. This extends to a challenging setup where it successfully generalizes to unseen Pommerman scenarios while the baselines fail to learn at all.

## 2 Related Work

The most related work to ours is an independent and concurrent blog post describing a method similar to Backplay used to train policies on the challenging Atari game Montezuma's Revenge (Salimans and Chen 2018). While both propose training an RL agent by starting each episode from a demonstration state based on a reverse curriculum, there are a number of key differences between the two reports.

First, our work considers the problem of learning effective policies in environments with various initial configurations, each corresponding to a different demonstration. Second, we analyze whether the technique can find optimal policies when trained using sub-optimal demonstrations and the extent to which those policies can generalize to unseen initial conditions. Third, we do not use an adaptive schedule based on the agent's success rate to advance its starting state because we did not find this to improve the overall performance or sample complexity. Fourth, we include both empirical and theoretical analysis on the conditions under which this approach is more effective than standard RL. Finally, we also evaluate the method on a stochastic multiplayer game, which has additional complexities compared to a deterministic single agent environment.

Behavioral cloning explicitly encourages a policy to mimic an expert policy (Ross, Gordon, and Bagnell 2011; Daumé, Langford, and Marcu 2009; Zhang and Cho 2016; Laskey et al. 2016). Many such algorithms require access to the expert's policy or use supervision from the expert's actions and/or states (Nair et al. 2017; Hester et al. 2017; Ho and Ermon 2016; Aytar et al. 2018; Lerer and Peysakhovich 2018; Peng et al. 2018). In contrast, Backplay only requires access to a single trajectory of states visited by the expert and only uses these states to initialize a curriculum (Bengio et al. 2009). Thus, it allows for the action space of the agent to be different from that of the expert and could be used with third-person demonstrations.

Other algorithms (Ranzato et al. 2015; Li et al. 2016; Das et al. 2017a; Das et al. 2017b), primarily in dialog, use a Backplay-like curriculum, albeit they utilize behavioral cloning for the first part of the trajectory. This is a major difference as we show that for many classes of problems, we *only* need to change the initial state distribution and do not see any gains from warm-starting with imitation learning. Backplay is more similar to Conservative Policy Iteration (Kakade and Langford 2002), a theoretical paper which presents an algorithm designed to operate with an explicit restart distribution.

(McAleer et al. 2018) frame the problem of solving the Rubik's cube as a sparse reward model-based RL task with inputs generated by taking random actions from the goal state. (Hosu and Rebedea 2016) use uniformly random states of an expert demonstration as starting states for a policy. Like Backplay, (Gao et al. 2018) show that using a single loss function to learn a policy from both demonstrations and rewards can outperform the demonstrator and is robust to sub-optimal demonstrations. However, none of those works impose a curriculum over the demonstration.

(Zhu et al. 2018) use a curriculum but manually tune it for each 'stage' of the environment. Within each stage, they use what we call Uniform training, which fails in our most challenging environments.

The idea of using a reverse curriculum for sparse reward reinforcement learning problems was proposed by (Florensa et al. 2017), which uses a sequence of short random walks to gradually initialize an RL agent from a set of start states increasingly far from a known goal location. In contrast, Backplay uses the demonstration to bias the start distribution towards a more realistic one, thus increasing the training time spent in worthwhile parts of the state space. Moreover, their approach requires the environment to be reversible, which may not be the case in many scenarios of interest, including popular research gaming environments like Starcraft. Finally, their method only applies to tasks with a unique fixed goal state, while our technique could be applied in settings with multiple non-stationary targets such as multi-agent learning games.

To improve sample efficiency of RL algorithms, (Goyal et al. 2018) and (Edwards, Downs, and Davidson 2018) simultaneously proposed the use of a learned backtracking model to generate traces that lead to high value states. In order to effectively use the traces, their methods rely on either having the agent visit high reward states or learning a model of the environment capable of generating the states, both of which are challenging in environments in which the dynamics near starting states are very different from those near goal states.

Finally, (Ivanovic et al. 2018) use a known (approximate) dynamics model to create a backwards curriculum for continuous control tasks. Their approach requires a physical prior which is not always available and often not applicable in multi-agent scenarios. In contrast, Backplay automatically creates a curriculum fit for any resettable environment with accompanying demonstrations.

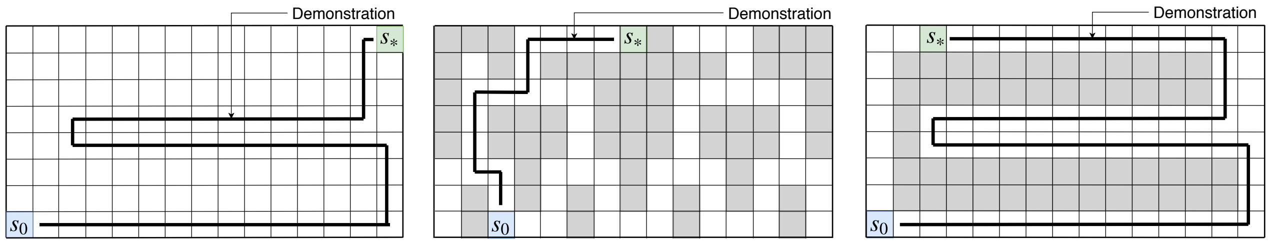
## 3 Backplay

Consider the standard formalization of a single agent Markov Decision Process (MDP) defined by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, and a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ which gives the probability distribution of the next state given a current state and action. If $\mathcal{P}(\mathcal{A})$ denotes the space of probability distributions over actions, the agent chooses actions by sampling from a stochastic policy $\pi : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$, and receives reward $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ at every time step. The agent's goal is to construct a policy which minimizes its discounted expected return $R_t = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right]$ where $r_t$ is the reward at time $t$ and $\gamma \in [0, 1]$ is the discount factor, and the expectation is taken with respect to both the policy and the environment.

The final component of an MDP is the distribution of initial starting states $s_0$. The key idea in Backplay is that we do not initialize the MDP in only a fixed $s_0$. Instead, we assume access to a demonstration which reaches a sequence of states $\{s_0^d, s_1^d, \ldots, s_T^d\}$. For each training episode, we uniformly sample starting states from the sub-sequence $\{s_{T-k}^d, s_{T-k+1}^d, \ldots, s_{T-j}^d\}$ for some window $[j, k]$. Note that this training regime requires the ability to reset the environment to any state. As training continues, we 'advance' the window according to a curriculum by increasing the values of $j$ and $k$ until we are training on the initial state in every episode ($j = k = T$). In this manner, our hyperparameters for Backplay are the windows and the training epochs at which we advance them.

### 3.1 Intuition

We first provide intuition regarding the conditions under which Backplay can improve sample-efficiency or lead to a better policy than that of the demonstrator. Figure 2 contains three toy grid worlds. In each of them, the agent begins at $s_0$, receives a reward of 1 iff it navigates to the goal $s_*$,



(a) Backplay Helps (b) Backplay Helps (c) Backplay Hinders

Figure 2. Three environments illustrating when Backplay can help or hinder learning an optimal policy. Backplay is expected to learn faster than standard RL on the first and second mazes, but perform worse on the third maze.

and incurs a per step cost. Thus, the optimal strategy is to reach the goal in as few steps as possible.

The left grid world shows a sparse reward environment in which Backplay can decrease the requisite training time compared to vanilla RL. Using the sub-optimal demonstration will position the Backplay agent close to high value states. In addition, the agent will likely surpass the expert policy because, unlike in behavioral cloning approaches, Backplay does not encourage the agent to imitate expert actions. Rather, the curriculum forces the agent to first explore states with large associated value and consequently, estimating the value function suffers less from the curse of dimensionality.

The middle grid illustrates a maze with bottlenecks. Backplay will vastly decrease the exploration time because the agent will be less prone to exploring the errant right side chamber where it can get lost if it traverses to the right instead of going up to the goal.

On the right side grid, while Backplay is still likely to surpass its demonstrator, it will have trouble doing better than vanilla RL because the latter will always start in $s_0$ and consequently will be more likely to stumble upon the optimal solution of going up and then right. In contrast, Backplay will spend the dominant amount of its early training starting in states in the sub-optimal demonstration.

## 3.2 Analysis

Next, we formally explore the conditions under which Backplay can improve the sample-efficiency of RL training. More specifically, we will derive an estimate for the sample complexity of a Q-learning agent trained with Backplay in a sparse reward navigation environment and we will show that it is significantly better than that of standard RL. Note that we simplify to a setup using a single starting state for each training stage instead of sampling from a window. In the notation from Section 3, $k = j$. One can verify that the general setup where $k > j$ is a more favorable regime, and therefore our approach provides an upper bound on the sample complexity.

Consider a connected, undirected graph $G = (V, E)$. An agent is placed at a fixed node $v_0 \in V$ and moves to neighboring nodes at a negative reward of $-1$ for each step. Its goal is to reach a target node $v_* \in V$, terminating the episode. This corresponds to an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R)$ with state space $\mathcal{S} \sim V$, action space $\mathcal{A} \sim E$, deterministic transition kernel corresponding to $P(s_{t+1} = j \mid s_t = i, a_t = (l,k)) = \delta(j = k)\delta(l = i) + \delta(j = i)\delta(l \neq i)$ and uniform reward $R(s_t, a_t) = -1$ for $s_{t+1} \neq v_*$. Assume that $\pi$ is a fixed policy on $\mathcal{M}$, such that the underlying Markov chain is irreducible and aperiodic:

$$K_\pi(s', s) = \sum_{a \in \mathcal{A}} P(s' = j \mid s_0 = s, a_0 = a)\pi(a|s_0 = s)$$

$K_\pi$ has a single absorbing state at $s = s_* := v_*$. Our goal is to estimate the value function of this policy, equivalent to the expected first-passage time of the Markov chain $K_\pi$ from $s_0 = s$ to $s_*$ - $V_\pi(s) = \mathbb{E}\tau(s, s_*)$, or

$$\mathbb{E} \min\{j \geq 0 \; ; s.t. \, s_j = s_*, s_0 = s, s_{i+1} \sim K_\pi(\cdot, s_i)\} \; .$$

We make a simplifying assumption that the function approximation is completely determined by projecting this chain into the process $z_t = \mathrm{dist}_G(s_t, s_*) \in \{0, 1, \ldots, M\}$, with $M = \mathrm{diam}(G)$. That is, we consider a latent variable at each state indicating its distance to the target. Since $z_t$'s transition probabilities are not a function of only $z_t$, it is in general not Markovian. Its Markov approximation $\bar{z}_t$ is defined as the Markov chain $\overline{K}$ given by the expected transition probabilities

$$\Pr(\bar{z}_{t+1} = l - 1 | \bar{z}_t = l) := \Pr_\mu(z_{t+1} = l - 1 \mid z_t = l)$$
$$\Pr(\bar{z}_{t+1} = l | \bar{z}_t = l) := \Pr_\mu(z_{t+1} = l \mid z_t = l)$$

Define $\alpha_l = \Pr_\mu(z_{t+1} = l - 1 \mid z_t = l)$ and $\beta_l = \Pr_\mu(z_{t+1} = l \mid z_t = l) \to \Pr(\bar{z}_{t+1} = l + 1 | \bar{z}_t = l) = 1 - \alpha_l - \beta_l = \Pr_\mu(z_{t+1} = l + 1 \mid z_t = l)$ under the stationary distribution $\mu$ of $K_\pi$. In other words, we create an equivalence class given by the level sets of the optimal value function $V_{opt}(s)$, the shortest-path distance in $G$.

The analysis of Backplay in the 1D chain $\overline{K}$ is now tractable. Given a demonstration $\mathbf{d} = (d_0 = s_0, \ldots, d_L = s_*)$, $d_l \in \mathcal{S}$, we sample it using a step size $m > 0$ (such that $j = 0 \bmod m$, where $j$ is defined in Section 3) to obtain $\bar{d}_l := \mathrm{dist}_G(d_{L-ml}, s_*)$, $l = 0, 1, \ldots, \frac{L}{m}$, which satisfies $\bar{d}_l \leq lm$ for all $l$. For fixed $l$, we initialize the chain $\overline{K}$ at $\bar{d}_l$: $\bar{z}_0 = \bar{d}_l$. Since $Pr(\bar{z}_m = 0) \geq \prod_{j=0}^{m-1} \alpha_j := \gamma_{0,m}$, after $O(\gamma_{0,m}^{-1})$ trials of length $\leq M$, we will reach the absorbing state and finally have a signal-carrying update for the Q-function at the originating state.

We can consequently merge that state into the absorbing state and reduce the length of the chain by one. Repeat the argument $m$ times so that after $O(\sum_{j=0}^{m} \gamma_{j,m}^{-1}) = O(m\gamma_{0,m}^{-1})$ trials, the Q-function is updated at $\bar{z}_0$. Repeat at Backplay steps $lm$, $l = 1, \dots \frac{M}{m}$, and we reach a sample complexity of

$$T_m = \sum_{k=0}^{\frac{M}{m}-1} O\left( M \sum_{j=0}^{m} \gamma_{km,(k+1)m-j}^{-1} \right) .$$

In the case where $\alpha_l = \alpha$ for all $l$, we obtain $\gamma_{km,(k+1)m-j}^{-1} = \gamma_{0,m-j} = \alpha^{-m+j}$ and therefore $T_m = O\left(\frac{M^2(1-\alpha^{m+1})}{m(1-\alpha)}\alpha^{-m}\right)$, where **the important term is the rate $\frac{M^2}{m}\alpha^{-m}$.**

On the other hand, (Hong and Zhou 2017) shows that the first-passage time $\tau(0, M)$ in a skip-free finite Markov chain of $M$ states with a single absorbing state is a random variable whose moment-generating function $\varphi(s) = \mathbb{E}s^{\tau(s,s_*)}$ is given by

$$\varphi(s) = \prod_{j=1}^{M} \frac{(1-\lambda_j)s}{1-\lambda_j s} , \tag{1}$$

where $\lambda_1, \dots, \lambda_M$ are the non-unit eigenvalues of the transition probability matrix. It follows that $\mathbb{E}\tau(0, M) = \varphi'(1) = \sum_{j=1}^{M} \frac{1}{1-\lambda_j} \approx (1-\lambda_1)^{-1}$, which corresponds to the reciprocal spectral gap. (Chen and Saloff-Coste 2013) further shows that this reciprocal spectral gap is $\Omega(\alpha^{-M/2})$ in our case, and therefore the model without Backplay will on average take $T_M = \Omega(\alpha^{-M/2})$ trials to reach the absorbing state and receive information.

**Hence, in this simple birth-death scenario, the sample complexity gains are exponential in the diameter of the graph - $O(\frac{M^2}{m}\alpha^{-m})$ vs $\Omega(M\alpha^{-M/2})$.**

We can analyze the uniform strategy similarly. The probability that a trajectory initialized at one of the uniform samples will reach the absorbing state is lower bounded by

$$\sum_{j=1}^{M} \alpha^j P(\bar{z}_0 = j) = \frac{1}{M}\sum_{j=1}^{M} \alpha^j = \frac{\alpha - \alpha^{M+1}}{M(1-\alpha)} , \tag{2}$$

which is approximately $\frac{\alpha}{M}$ when $\alpha$ is small, leading to a sample complexity of $O(M^2\alpha^{-1})$ to update the value function at the originating state, and $O(M^3\alpha^{-1})$ at the starting state. Comparing this rate to Backplay with $m = 1$, observe that the uniform strategy is slower by a factor of $M$ (and one can verify that the same is true for generic step size $m$ by imagining that we first sampled a window of size $m$ and then sub-sampled our state from that window), suggesting that it loses efficiency on environments with large diameter.

This analysis relies on the projection into the 1D skip-free process given by the distance to the target and makes two strong simplifying assumptions. First, we assume that the level sets of our estimated value functions correspond to the level sets of the true value function, ie, for all $\theta$, $V_\theta(s) = V_\theta(s')$ whenever $\text{dist}_G(s', s_*) = \text{dist}_G(s, s_*)$. Second, we assume that the projected process is well described by its Markovian approximation. The first assumption is related to the ability of our function approximator to generalize from visited states $s$ to unvisited states $s'$ such that $\text{dist}_G(s', s_*) = \text{dist}_G(s, s_*)$, and is generally violated unless one leverages prior information. The second assumption has an impact on the derived bounds and could be relaxed by replacing conditional probabilities with infima over the level sets.

As Figure 2 suggests, Backplay is not a universal strategy to improve sample complexity. Even in the navigation setting, if the task randomizes the initial state $s_0$, a single demonstration trajectory does not generally improve the coverage of the state-space outside an exponentially small region around said trajectory. For example, imagine a binary tree and a navigation task that starts at a random leaf and needs to reach the root. A single expert trajectory will be disjoint from half of the state space (because the root is absorbing), thus providing no sample complexity gains on average.

The preceding analysis suggests that a full characterization of Backplay is a fruitful direction for reinforcement and imitation learning theory, albeit beyond the scope of this paper.

## 4 Experiments

We now move to evaluating Backplay empirically in two environments: a grid world maze and a four-player free-for-all game. The questions we study across both environments are the following:

- Is Backplay more efficient than training an RL agent from scratch?
- How does the quality of the given demonstration affect the effectiveness of Backplay?
- Can Backplay agents surpass the demonstrator when it is non-optimal?
- Can Backplay agents generalize?

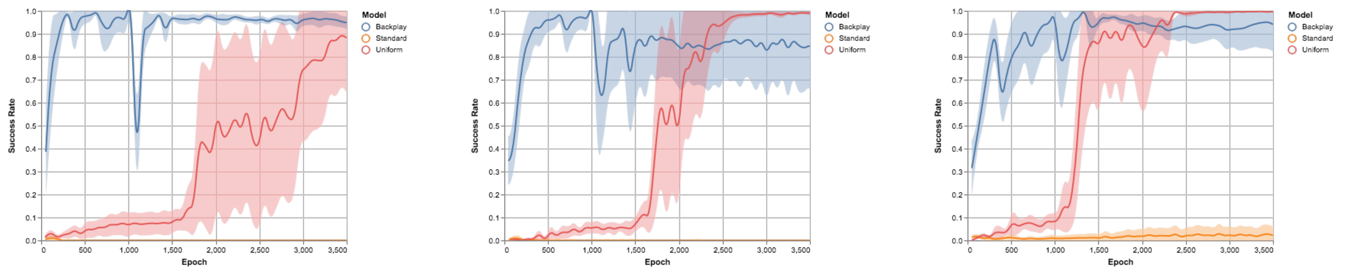
### 4.1 Training Details

We compare several training regimes. The first is **Backplay**, which uses the Backplay algorithm corresponding to a particular sequence of windows and epochs as specified in A.1. The second, **Standard** is vanilla model-free RL with the agent always starting at the initial state $s_0$. The last, **Uniform**, is an ablation that considers how important the curriculum aspect of Backplay is, by sampling initial states randomly from the entire demonstration.

In all these regimes, we use Proximal Policy Optimization (PPO, (Schulman et al. 2017)) to train an agent with policy and value functions parameterized by convolutional neural networks. Training details and network architectures for all environments can be found in A.3 and A.5, while A.8 contains empirical observations for using Backplay in practice. We report the mean and standard deviation of the success rate across five random seeds.

| Algorithm | Demonstrator | % Optimal | % 0-5 Optimal | Avg Suboptimality | Std Suboptimality |
|---|---|---|---|---|---|
| Standard | None | 0 | 0 | N/A | N/A |
| Uniform | Optimal | 27 | 91 | 8.26 | 32.92 |
| Uniform | 5-Optimal | 51 | 98 | 2.04 | 17.39 |
| Uniform | 10-Optimal | 49 | 98 | 2.04 | 16.79 |
| Backplay | Optimal | **31** | **100** | **0.64** | **4.96** |
| Backplay | 5-Optimal | 37 | 94 | 7.94 | 33.35 |
| Backplay | 10-Optimal | **54** | 99 | **0.37** | 3.49 |

Table 1: Results after 2000 epochs on 100 mazes. N-optimal uses demonstrations N steps longer than the shortest path. From left to right, the table shows: the percentage of mazes on which the agent optimally reaches the goal, percentage on which it reaches in at most five steps more than optimal, and the average and standard deviation of extra steps over optimal. Standard has failed to learn, while both Backplay and Uniform succeed on almost all mazes and, importantly, can outperform the experts' demonstrations. Results were consistent across all seeds. For more details, see A.2.



(a) Optimal Demonstrations (b) Demos Five > Optimal (c) Demos Ten > Optimal

Figure 3. Maze learning curves when training with optimal demonstrations, demonstrations that are five longer than optimal, and demonstrations that are ten longer than optimal. In all cases, the Backplay models first encounter the initial state at epoch 1400 and it becomes the default starting state at epoch 1750. Observe that in all circumstances Backplay significantly decreases the time to learn a strong policy.

## 4.2 Maze

We generated mazes of size $24 \times 24$ with 120 randomly placed walls, a random start position, and a random goal position. We then used A* to generate trajectories. These included both Optimal demonstrations (true shortest path) and N-Optimal demonstrations (N steps longer than the shortest path). More details on this setup are given in A.2.

Our model receives as input four $24 \times 24$ binary maps. They contain ones at the positions of, respectively, the agent, the goal, passages, and walls. It outputs one of five options: Pass, Up, Down, Left, or Right. The game ends when the agent has reached its goal or after a maximum of 200 steps, whereupon the agent receives reward of +1 if it reaches the goal and a per step penalty of -0.03.

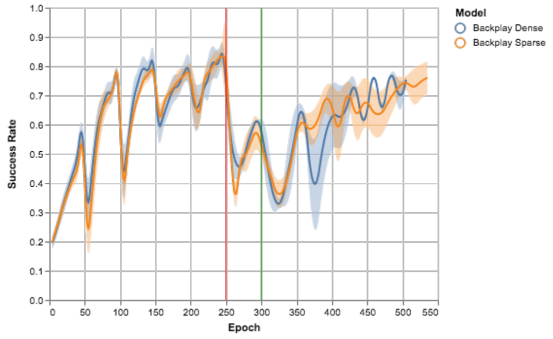
Table 1 shows that Standard has immense trouble learning in this sparse reward environment while both Backplay and Uniform find an optimal path approximately 30-50% of the time and a path within five of the optimal path almost always. Thus, in this environment, demonstrations of even sub-optimal experts are extremely useful, while the curriculum created by Backplay is not necessary. That curriculum does, however, aid convergence speed (3). We will see in the next section that the curriculum becomes vital as the environment increases in complexity.

Finally, we evaluated the degree to which Backplay generalizes to unseen environment configurations by testing the agent on ten new mazes and found that none of our training regimes were successfully able to navigate them. We also tried generating new mazes for each episode of training and found that the models failed to learn at all even in training. These experiments suggest that in this environment, Backplay does not aid generalization.

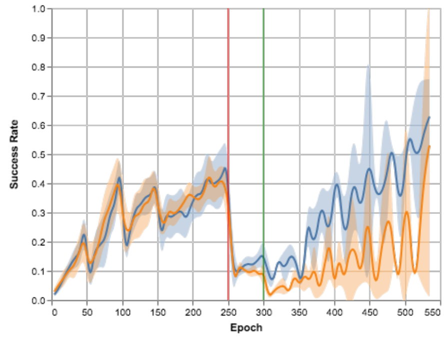
## 4.3 Pommerman

Pommerman is a stochastic environment (Resnick et al. 2018) based on the classic console game Bomberman and will be a competition at NIPS 2018. It is played on an 11x11 grid where on every turn, each of four agents either move in a cardinal direction, pass, or lay a bomb. The agents begin fenced in their own area by two different types of walls - rigid and wooden. The former are indestructible while bombs destroy the latter. Upon blowing up wooden walls, there is a uniform chance at yielding one of three power-ups: an extra bomb, an extra unit of range in the agent's bombs, or the ability to kick bombs. The maps are designed randomly, albeit there is always a guaranteed path between any two agents. For a visual aid of the start state, see Figure 6 in A.4.

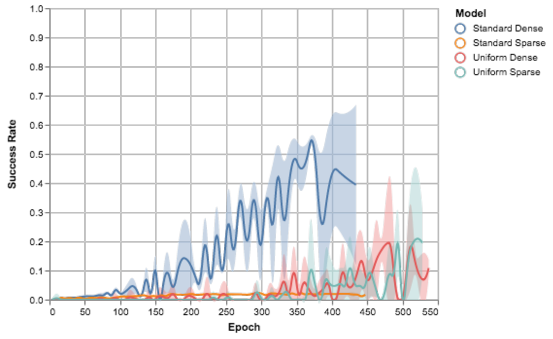
In our experiments, we use the purely adversarial Free-For-All (FFA) environment. This environment is introduced in (MultiAgentLearning 2018; Resnick et al. 2018) and we point the reader there for more details. The winner of the game is the last agent standing. It is played from the perspective of one agent whose starting position is uniformly picked among the four. The three opponents are copies of the winner of the June 3rd 2018 FFA competition, a stochas-



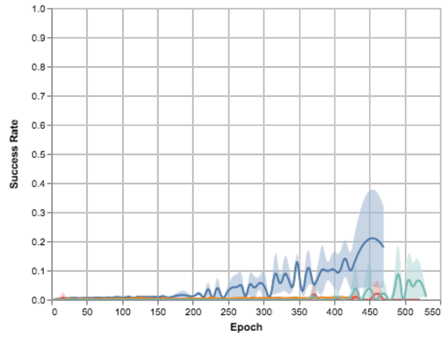
(a) Backplay Winner



(b) Backplay Runner Up



(c) Standard & Uniform Winner



(d) Standard & Uniform Runner Up

Figure 4. Pommerman results when training on four maps. **a** and **c** are trained from the perspective of the winning agent, while **b** and **d** are from that of the runner up. Dense is with reward shaping and Sparse is without. Performance is shown for the training episodes. The Backplay models first encounter the initial state at epoch 250 (vertical red line) starts training only from the initial state from epoch 300 onward (vertical green line). While Standard achieves notable results only in the dense setting, Backplay achieves strong performance in both sparse and dense setups, and Uniform fails to learn a worthwhile policy in either. Finally, there is a clear difference in performance when following the winner versus the runner up, although the maps are the same.

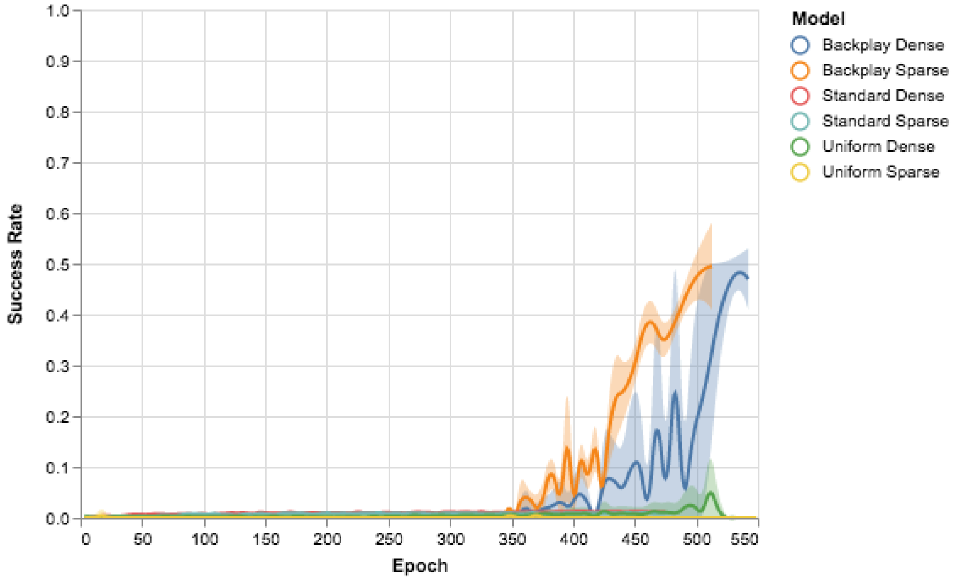


Figure 5. Pommerman results when training on a hundred maps from the winner's perspective. Performance is shown on episodes with the agent always starting in the initial state. Backplay first encounters the initial state at epoch 340 and defaults to it for half of the environments at epoch 425. Observe that Backplay performs significantly better than Standard and Uniform in both Dense and Sparse settings.

tic agent using a Finite State Machine Tree-Search approach (FSMTS, (Zhou et al. 2018)). We also make use of the FSMTS agent as the 'expert' in the Backplay demonstrations.

The observation state is represented by 19 11x11 maps, and we feed the concatenated last two states to our agent as input. A detailed description of this mapping is given in A.4. The game ends either when the learning agent wins or dies, or when 800 steps have passed. Upon game end, the agent receives $+1$ for winning and $-1$ otherwise (**Sparse**). We also run experiments where the agent additionally receives $+0.1$ whenever it collects an item (**Dense**).

Our first two scenarios draw maps from four fixed Pommerman board configurations. We independently consider two Backplay trajectories, one following the winner and one following the runner up. Figure 4 shows that **Backplay can defeat the FSMTS agents in both sparse and dense settings, while Standard requires dense rewards to win**. Visit this link for an example gif of our trained Backplay agent (top left - red). Note that the agent learned to 'throw' bombs, a unique playing style that no prior Pommerman competitor had exhibited, including the FSMTS demonstrator.

We then consider a harder setup where the board is drawn from a hundred maps. Figure 5 shows that **Backplay achieves high success rates while Standard and Uniform fail to learn anything of substance**. Moreover, Backplay agents win half of the games at rates $> 80\%$ (Table 5 in A.7).

Unlike on the maze, Backplay can, to some extent, generalize to unseen Pommerman maps. **Backplay wins 416 / 1000 games on a held out test set of ten new maps**, with the following success rates on each of the ten map: 85.3%, 84.1%, 81.6%, 79.5%, 52.4%, 47.4%, 38.1%, 22.6%, 20%, and 18.3%.

## 5 Conclusion

We have introduced and analyzed Backplay, a technique which improves the sample efficiency of model-free RL by constructing a curriculum around a demonstration. We showed that Backplay agents can learn in complex environments where standard model-free RL fails, and that they can outperform the 'expert' whose trajectories they use while training. We also presented a theoretical analysis of its sample complexity in a simplified setting.

An important future direction is combining Backplay with more complex and complementary methods such as Monte Carlo Tree Search (MCTS, (Browne et al. 2012; Vodopivec, Samothrakis, and Šter 2017)). There are many potential ways to do so, for example by using Backplay to warm-start MCTS.

Another direction is to use Backplay to accelerate self-play learning in zero-sum games. However, special care needs to be taken to avoid policy correlation during training (Lanctot et al. 2017) and thus to make sure that learned strategies are safe and not exploitable (Brown and Sandholm 2017).

A third direction is towards non-zero sum games. It is well known that standard independent multi-agent learning does not produce agents that are able to cooperate in social dilemmas (Leibo et al. 2017; Lerer and Peysakhovich 2017; Peysakhovich and Lerer 2017; Foerster et al. 2017) or risky coordination games (Yoshida, Dolan, and Friston 2008; Peysakhovich and Lerer 2018). In contrast, humans are much better at finding these coordinating and cooperating equilibria (Bó 2005; Kleiman-Weiner et al. 2016). Thus, we conjecture that human demonstrations can be combined with Backplay to construct agents that perform well in such situations.

Other future priorities are to gain a better understanding of when Backplay works well, when it fails, and how we can make the procedure more efficient. Could we speed up Backplay by ascertaining confidence estimates of state values? Do the gains in sample complexity come from value estimation like our analysis suggests, from policy iteration, or from both? Is there an ideal rate for advancing the curriculum window and is there a better approach than a hand-tuned schedule?

## References

[Aytar et al. 2018] Aytar, Y.; Pfaff, T.; Budden, D.; Paine, T. L.; Wang, Z.; and de Freitas, N. 2018. Playing hard exploration games by watching youtube. *arXiv preprint arXiv:1805.11592*.

[Bengio et al. 2009] Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, 41–48. ACM.

[Bó 2005] Bó, P. D. 2005. Cooperation under the shadow of the future: experimental evidence from infinitely repeated games. *American economic review* 95(5):1591–1604.

[Brown and Sandholm 2017] Brown, N., and Sandholm, T. 2017. Safe and nested subgame solving for imperfect-information games. In *Advances in Neural Information Processing Systems*, 689–699.

[Browne et al. 2012] Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4(1):1–43.

[Chen and Saloff-Coste 2013] Chen, G.-Y., and Saloff-Coste, L. 2013. On the mixing time and spectral gap for birth and death chains. *arXiv preprint arXiv:1304.4346*.

[Clark and Amodei 2016] Clark, J., and Amodei, D. 2016. Faulty reward functions in the wild. `https://blog.openai.com/faulty-reward-functions/`.

[Das et al. 2017a] Das, A.; Datta, S.; Gkioxari, G.; Lee, S.; Parikh, D.; and Batra, D. 2017a. Embodied question answering. *CoRR* abs/1711.11543.

[Das et al. 2017b] Das, A.; Kottur, S.; Moura, J. M. F.; Lee, S.; and Batra, D. 2017b. Learning cooperative visual dialog agents with deep reinforcement learning. *CoRR* abs/1703.06585.

[Daumé, Langford, and Marcu 2009] Daumé, H.; Langford, J.; and Marcu, D. 2009. Search-based structured prediction. *Machine learning* 75(3):297–325.

[Edwards, Downs, and Davidson 2018] Edwards, A. D.; Downs, L.; and Davidson, J. C. 2018. Forward-backward reinforcement learning. *arXiv preprint arXiv:1803.10227*.

[Florensa et al. 2017] Florensa, C.; Held, D.; Wulfmeier, M.; and Abbeel, P. 2017. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*.

[Foerster et al. 2017] Foerster, J. N.; Chen, R. Y.; Al-Shedivat, M.; Whiteson, S.; Abbeel, P.; and Mordatch, I. 2017. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*.

[Gao et al. 2018] Gao, Y.; Lin, J.; Yu, F.; Levine, S.; Darrell, T.; et al. 2018. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*.

[Goyal et al. 2018] Goyal, A.; Brakel, P.; Fedus, W.; Lillicrap, T.; Levine, S.; Larochelle, H.; and Bengio, Y. 2018. Recall traces: Backtracking models for efficient reinforcement learning. *arXiv preprint arXiv:1804.00379*.

[Hester et al. 2017] Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Horgan, D.; Quan, J.; Sendonaris, A.; Dulac-Arnold, G.; et al. 2017. Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*.

[Ho and Ermon 2016] Ho, J., and Ermon, S. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 4565–4573.

[Hong and Zhou 2017] Hong, W., and Zhou, K. 2017. A note on the passage time of finite-state markov chains. *Communications in Statistics-Theory and Methods* 46(1):438–445.

[Hosu and Rebedea 2016] Hosu, I.-A., and Rebedea, T. 2016. Playing atari games with deep reinforcement learning and human checkpoint replay. *arXiv preprint arXiv:1607.05077*.

[Ivanovic et al. 2018] Ivanovic, B.; Harrison, J.; Sharma, A.; Chen, M.; and Pavone, M. 2018. Barc: Backward reachability curriculum for robotic reinforcement learning. *arXiv preprint arXiv:1806.06161*.

[Kakade and Langford 2002] Kakade, S., and Langford, J. 2002. Approximately optimal approximate reinforcement learning. In Sammut, C., and Hoffmann, A. G., eds., *ICML*, 267–274. Morgan Kaufmann.

[Kleiman-Weiner et al. 2016] Kleiman-Weiner, M.; Ho, M. K.; Austerweil, J. L.; Littman, M. L.; and Tenenbaum, J. B. 2016. Coordinate to cooperate or compete: abstract goals and joint intentions in social interaction. In *COGSCI*.

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

[Laskey et al. 2016] Laskey, M.; Staszak, S.; Hsieh, W. Y.-S.; Mahler, J.; Pokorny, F. T.; Dragan, A. D.; and Goldberg, K. 2016. Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 462–469. IEEE.

[Leibo et al. 2017] Leibo, J. Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; and Graepel, T. 2017. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 464–473. International Foundation for Autonomous Agents and Multiagent Systems.

[Lerer and Peysakhovich 2017] Lerer, A., and Peysakhovich, A. 2017. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *arXiv preprint arXiv:1707.01068*.

[Lerer and Peysakhovich 2018] Lerer, A., and Peysakhovich, A. 2018. Learning social conventions in markov games. *arXiv preprint arXiv:1806.10071*.

[Levine et al. 2016] Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17(39):1–40.

[Li et al. 2016] Li, J.; Monroe, W.; Ritter, A.; Galley, M.; Gao, J.; and Jurafsky, D. 2016. Deep reinforcement learning for dialogue generation. *CoRR* abs/1606.01541.

[McAleer et al. 2018] McAleer, S.; Agostinelli, F.; Shmakov, A.; and Baldi, P. 2018. Solving the rubik's cube without human knowledge. *arXiv preprint arXiv:1805.07470*.

[Moravcík et al. 2017] Moravcík, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. H. 2017. Deepstack: Expert-level artificial intelligence in no-limit poker. *CoRR* abs/1701.01724.

[MultiAgentLearning 2018] MultiAgentLearning. 2018. Pommerman. `https://github.com/MultiAgentLearning/playground`.

[Nair et al. 2017] Nair, A.; McGrew, B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2017. Overcoming exploration in reinforcement learning with demonstrations. *arXiv preprint arXiv:1709.10089*.

[Ng, Harada, and Russell 1999] Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

[Peng et al. 2018] Peng, X. B.; Abbeel, P.; Levine, S.; and van de Panne, M. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *arXiv preprint arXiv:1804.02717*.

[Peysakhovich and Lerer 2017] Peysakhovich, A., and Lerer, A. 2017. Consequentialist conditional cooperation in social dilemmas with imperfect information. *arXiv preprint arXiv:1710.06975*.

[Peysakhovich and Lerer 2018] Peysakhovich, A., and Lerer, A. 2018. Prosocial learning agents solve generalized stag hunts better than selfish ones. *Proceedings of the 17th Conference on Autonomous Agents and MultiAgent Systems*.

[Ranzato et al. 2015] Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2015. Sequence level training with recurrent neural networks. *CoRR* abs/1511.06732.

[Resnick et al. 2018] Resnick, C.; Eldridge, W.; Ha, D.; Britz, D.; Foerster, J.; Togelius, J.; Cho, K.; and Bruna, J. 2018. Pommerman: A multi-agent playground.

[Ross, Gordon, and Bagnell 2011] Ross, S.; Gordon, G.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 627–635.

[Salimans and Chen 2018] Salimans, T., and Chen, R. 2018. Learning montezuma's revenge from a single demonstration. `https://blog.openai.com/learning-montezumas-revenge-from-a-single-demonstration/`. Accessed: 2018-07-12.

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

[Silver et al. 2017] Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T. P.; Simonyan, K.; and Hassabis, D. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR* abs/1712.01815.

[Vodopivec, Samothrakis, and Šter 2017] Vodopivec, T.; Samothrakis, S.; and Šter, B. 2017. On monte carlo tree search and reinforcement learning. *J. Artif. Int. Res.* 60(1):881–936.

[Yoshida, Dolan, and Friston 2008] Yoshida, W.; Dolan, R. J.; and Friston, K. J. 2008. Game theory of mind. *PLoS computational biology* 4(12):e1000254.

[Zhang and Cho 2016] Zhang, J., and Cho, K. 2016. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*.

[Zhou et al. 2018] Zhou, H.; Gong, Y.; Mugrai, L.; Khalifa, A.; Nealen, A.; and Togelius, J. 2018. A hybrid search agent in pommerman. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, FDG '18, 46:1–46:4. New York, NY, USA: ACM.

[Zhu et al. 2018] Zhu, Y.; Wang, Z.; Merel, J.; Rusu, A. A.; Erez, T.; Cabi, S.; Tunyasuvunakool, S.; Kramár, J.; Hadsell, R.; de Freitas, N.; and Heess, N. 2018. Reinforcement and imitation learning for diverse visuomotor skills. *CoRR* abs/1802.09564.

## A Appendix

### A.1 Backplay Hyperparameters

As mentioned in Section 3, the Backplay hyperparameters are the window bounds and the frequency with which they are shifted. When we get to a training epoch represented in the sequence of epochs, we advance to the corresponding value in the sequence of windows. For example, consider training an agent with Backplay in the Maze environment (Table 2) and assume we are at epoch 1000. We will select a maze at random, an $N \in [16, 32)$, and start the agent in that game $N$ steps from the end. Whenever a pair is chosen such that the game's length is smaller than $N$, we use the initial state.

There isn't any downside to having the model continue training in a window for too long, albeit the ideal is that this method increases the speed of training. There is however a downside to advancing the window too quickly. A scenario common to effective training is improving success curves punctured by step drops whenever the window advances.

| Starting at training epoch | Uniform window |
|---|---|
| 0 | [0, 4) |
| 350 | [4, 8) |
| 700 | [8, 16) |
| 1050 | [16, 32) |
| 1400 | [32, 64) |
| 1750 | [64, 64) |

Table 2: Backplay hyperparameters for Maze.

| Starting at training epoch | Uniform window |
|---|---|
| 0 | [0, 32) |
| 50 | [24, 64) |
| 100 | [56, 128) |
| 150 | [120, 256) |
| 200 | [248, 512) |
| 250 | [504, 800) |
| 300 | [800, 800] |

Table 3: Backplay hyperparameters for Pommerman 4 maps.

| Starting at training epoch | Uniform window |
|---|---|
| 0 | [0, 32) |
| 85 | [24, 64) |
| 170 | [56, 128) |
| 255 | [120, 256) |
| 340 | [248, 512) |
| 425 | [504, 800) |
| 510 | [800, 800] |

Table 4: Backplay hyperparameters for Pommerman 100 maps.

### A.2 Maze: Demonstration Details

For N-Optimal demonstrations, we used a noisy A* where at each step, we follow A* with probability $p$ or choose a random action otherwise. We considered $N \in \{5, 10\}$. In all scenarios, we only selected maps in which there exists at least a path from the the initial state to the goal state, we filtered any path that was less than 35 in length and stopped when we found a hundred valid training games. Note that we held the demonstration length invariant rather than the optimal length (i.e. all N-optimal paths have the same length regardless of N, which means that the length of the optimal path of a N-optimal demonstration decreases with N). This could explain why the results in Table 1 (column 1) show that Backplay's performance increases with N (since the larger the N, the smaller the true optimal path, so the easier it is to learn an optimal policy for that maze configuration).

## A.3 Maze: Network Architecture and Training Parameters

We use a standard deep RL setup for our agents. The agent's policy and value functions are parameterized by a convolutional neural network with 2 layers each of 32 output channels, followed by two linear layers with 128 dimensions. Each of the layers are followed by ReLU activations. This body then feeds two heads, a scalar value function and a softmax policy function over the five actions. All of the CNN kernels are 3x3 with stride and padding of one.

We train our agent using Proximal Policy Optimization (PPO, (Schulman et al. 2017)) with $\gamma = 0.99$, learning rate $1 \times 10^{-3}$, batch size 102400, 60 parallel workers, clipping parameter 0.2, generalized advantage estimation with $\tau = 0.95$, entropy coefficient 0.01, value loss coefficient 0.5, mini-batch size 5120, horizon 1707, and 4 PPO updates at each iteration. The number of interactions per epoch is equal to the batch size (102400).

## A.4 Pommerman: Observation State

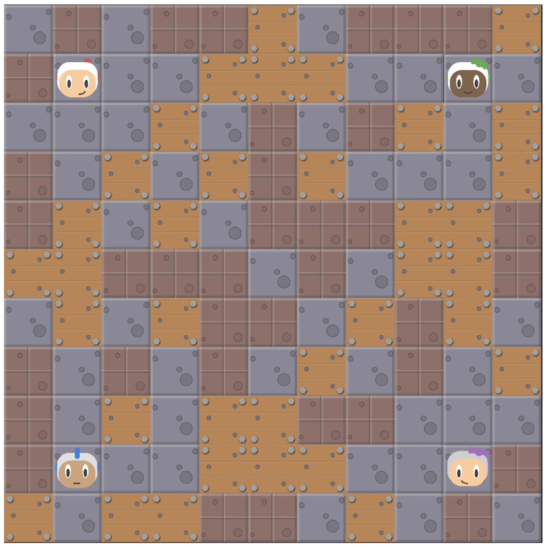


Figure 6. Pommerman start state. Each agent begins in one of four positions. Yellow squares are wood, brown are rigid, and gray are passages.

There are 19 feature maps that encompass each observation. They consist of the following: the agents' identities and locations, the locations of the walls, power-ups, and bombs, the bombs' blast strengths and remaining life counts, and the current time step.

The first map contains the integer values of each bomb's blast strength at the location of that bomb. The second map is similar but the integer value is the bomb's remaining life. At all other positions, the first two maps are zero. The next map is binary and contains a single one at the agent's location. If the agent is dead, this map is zero everywhere. The following two maps are similar. One is full with the agent's integer current bomb count, the other with its blast radius. We then have a full binary map that is one if the agent can kick and zero otherwise.

The next maps deal with the other agents. The first contains only ones if the agent has a teammate and zeros otherwise. This is useful for building agents that can play both team and solo matches. If the agent has a teammate, the next map is binary with a one at the teammate's location (and zero if she is not alive). Otherwise, the agent has three enemies, so the next map contains the position of the enemy that started in the diagonally opposed corner from the agent. The following two maps contain the positions of the other two enemies, which are present in both solo and team games.

We then include eight feature maps representing the respective locations of passages, rigid walls, wooden walls, flames, extra-bomb power-ups, increase-blast-strength power-ups, and kicking-ability power-ups. All are binary with ones at the corresponding locations.

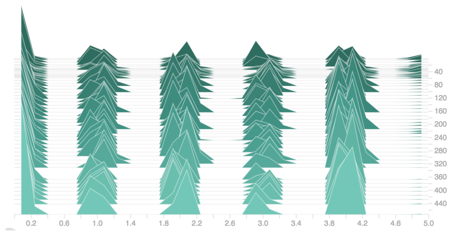
Finally, we include a full map with the float ratio of the current step to the total number of steps. This information is useful for distinguishing among observation states that are seemingly very similar, but in reality are very different because the game has a fixed ending step where the agent receives negative reward for not winning.

## A.5 Pommerman: Network Architecture and Training Parameters

We use a similar setup to that used in the Maze game. The architecture differences are that we have an additional two convolutional layers at the beginning, use 256 output channels, and have output dimensions of 1024 and 512, respectively, for the linear layers. This architecture was not tuned at all during the course of our experiments. Further hyperparameter differences are that we used a learning rate of $3 \times 10^{-4}$ and a gamma of 1.0. These models trained for 72 hours, which is $\sim$50M frames. [1]

[1]In our early experiments, we also used a batch-size of 5120, which meant that the sampled transitions were much more correlated. While Backplay would train without any issues in this setup, Standard would only marginally learn. In an effort to improve our baselines, we did not explore this further.

## A.6 Pommerman: Action Distribution Analysis


(a) Standard Sparse

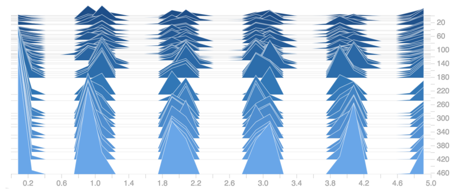

(b) Backplay Sparse

Figure 7. Typical histograms for how the Pommerman action selections change over time. From left to right are the concatenated counts of the actions (Pass, Up, Down, Left, Right, Bomb), delineated on the y-axis by the epoch. Note how the Standard agent learns to not use bombs.

Pommerman can be difficult for reinforcement learning agents. The agent must learn to effectively wield the bomb action in order to win against competent opponents. However, bombs destroy agents indiscriminately, so placing one without knowing how to retreat often results in negative reward for that agent. Since agents begin in an isolated area, they are prone to converging to policies which do not use the bomb action (as seen in the histograms in Figure 7), which leads them to sub-optimal policies in the long-term.

## A.7 Pommerman: Win Rates

Here we show the per-map win rates obtained by the agent trained with Backlpay on the 100 Pommerman maps.

| Win | Maps |
|---|---|
| 90% | 24 |
| 80% | 26 |
| 70% | 6 |
| 60% | 5 |
| 50% | 5 |

Table 5: Aggregate per-map win rates of the model trained with Backplay on 100 Pommerman maps. The model was run over 5000 times in total, with at least 32 times on each of the 100 maps. The *Maps* column shows the number of maps on which the Backplay agent had a success rate of at least the percent in the *Win* column. Note that this model has a win rate of $> 80\%$ on more than half of the maps and a win rate of $> 50\%$ on all maps.

## A.8 Practical Findings

We trained a large number of models through our research into Backplay. Though these findings are tangential to our main points (and are mainly qualitative), we list some observations here that may be helpful for other researchers working with Backplay.

First, we found that Backplay does not perform well when the curriculum is advanced too quickly, however it does not fail when the curriculum is advanced 'too slowly.' Thus, researchers interested in using Backplay should err on the side of advancing the window too slowly rather than too quickly.

Second, we found that Backplay does not need to hit a high success rate before advancing the starting state window. Initially, we tried using adaptive approaches that advanced the window when the agent reached a certain success threshold. This worked but was too slow. Our hypothesis is that what is more important is that the agent gets sufficiently exposed to enough states to attain a reasonable barometer of their value rather than that the agent learns a perfectly optimal policy for a particular set of starting states.

Third, Backplay can still recover if success goes to zero. This surprising and infrequent result occurred at the juncture where the window moved back to the initial state and even if the policy's entropy over actions became maximal. We are unsure what differentiates these models from the ones that did not recover.

We also explored using DAgger ((Ross, Gordon, and Bagnell 2011)) for training our agent, but found that it achieved approximately the same win rate ($\sim 20\%$) as what we would expect when four FSMTS agents played each other (given that there are also ties).