BRExIt: On Opponent Modelling in Expert Iteration

Daniel Hernandez[1,2] , **Hendrik Baier**[3] , **Michael Kaisers**[4]
[1]Sony AI
[2]University of York, UK
[3]Eindhoven University of Technology, The Netherlands
[4]Centrum Wiskunde & Informatica, The Netherlands

Abstract

Finding a best response policy is a central objective in game theory and multi-agent learning, with modern population-based training approaches employing reinforcement learning algorithms as best-response oracles to improve play against candidate opponents (typically previously learnt policies). We propose Best Response Expert Iteration (BRExIt), which accelerates learning in games by incorporating opponent models into the state-of-the-art learning algorithm Expert Iteration (ExIt). BRExIt aims to (1) improve feature shaping in the apprentice, with a policy head predicting opponent policies as an auxiliary task, and (2) bias opponent moves in planning towards the given or learnt opponent model, to generate apprentice targets that better approximate a best response. In an empirical ablation on BRExIt's algorithmic variants against a set of fixed test agents, we provide statistical evidence that BRExIt learns better performing policies than ExIt. Code available at: *https://github.com/Danielhp95/on-opponent-modelling-in-expert-iteration-code*. Supplementary material available at *https://arxiv.org/abs/2206.00113*.

1 Introduction

Reinforcement learning has been successfully applied in increasingly challenging settings, with multi-agent reinforcement learning being one of the frontiers that pose open problems [Hernandez-Leal *et al.*, 2017; Albrecht and Stone, 2018; Nashed and Zilberstein, 2022]. Finding strong policies for multi-agent interactions (games) requires techniques to selectively explore the space of best response policies, and techniques to learn such best response policies from gameplay. We here contribute a method that speeds up the learning of approximate best responses in games.

State-of-the-art training schemes such as population based training with centralized control [Lanctot *et al.*, 2017; Liu *et al.*, 2021] employ a combination of outer-loop training schemes and inner-loop learning agents in a nested loop fashion [Hernandez *et al.*, 2019]. In the outer loop, a combination of fixed policies is chosen via game theoretical analysis to act as static opponents. Within the inner loop, a reinforcement learning (RL) agent repeatedly plays against these static opponents, in order to find an approximate best response policy against them. In theory, any arbitrary RL algorithm could be used as such a **policy improvement operator/best response oracle** in the inner loop. The best response policy found is then added to the training scheme's population, and the outer loop continues, constructing a population of increasingly stronger policies over time. This technique has been successfully applied in highly complex environments such as Dota 2 [Berner *et al.*, 2019] and Starcraft II [Vinyals *et al.*, 2019], with PPO [Schulman *et al.*, 2017] and IMPALA [Espeholt *et al.*, 2018] respectively used as policy improvement operators.

The large number of training episodes required by modern deep RL (DRL) algorithms as policy improvement operators makes the inner loop of such training schemes a computational bottleneck. In this paper we accelerate approximating best responses by introducing ***Best Response Expert Iteration (BRExIt)***. We extend Expert Iteration (ExIt) [Anthony *et al.*, 2017], famously used in AlphaGo [Silver *et al.*, 2016], by introducing opponent models (OMs) in both (1) the **apprentice** (a deep neural network), with the aim of feature shaping and learning a surrogate model, and (2) the **expert** (Monte Carlo Tree Search), to bias its search towards approximate best responses to the OMs, yielding better targets for the apprentice. BRExIt thus better exploits OMs that are available in centralized training approaches, or also in Bayesian settings which assume a given set of opponents [Oliehoek and Amato, 2014].

For the case of given OMs that are computationally too demanding for search, but which can be used to generate training games, we also test a variant of BRExIt that uses learned surrogate OMs instead. In the game of Connect4, we find BRExIt learns significantly stronger policies than ExIt with the same computation time, alleviating the computational bottleneck in training towards a best response.

2 Related work

BRExIt stands at the confluence of two streams of literature: improving the sample efficiency of the ExIt framework (Section 2.1), and incorporating opponent models within deep reinforcement learning (Section 2.2).

2.1 Expert Iteration and improvements

Expert Iteration [Anthony *et al.*, 2017] combines planning and learning during training, with the goal of finding a parameterized policy $\pi_{\boldsymbol{\theta}}$ (where $\boldsymbol{\theta} \in \mathbb{R}^n$) that maximizes a reward signal. The trained policy can either be used within a planning algorithm or act as a standalone module during deployment without needing environment models.

ExIt's two main components are (1) the **expert**, traditionally an MCTS procedure [Browne *et al.*, 2012], and (2) the **apprentice**, a parameterized policy $\pi_{\boldsymbol{\theta}}$, usually a neural network. In short, the expert takes actions in an environment using MCTS, generating a dataset of good quality moves. The apprentice updates its parameters $\boldsymbol{\theta}$ to better predict both the expert's actions and future rewards. The expert in turn uses the apprentice inside MCTS to bias its search. Updates to the apprentice yield higher quality expert searches, yielding new and better targets for the apprentice to learn. This iterative improvement constitutes the main ExIt training loop, depicted at the top of Figure 1.

Significant effort has been aimed at improving ExIt, e.g. exploring alternate value targets [Willemsen *et al.*, 2021] or incorporating prioritized experience replay [Schaul *et al.*, 2015], informed exploratory measures [Soemers *et al.*, 2020] or domain specific auxiliary tasks for the apprentice [Wu, 2019]. BRExIt improves upon ExIt by deeply integrating opponent models within it.

2.2 Opponent modelling in DRL

Policy reconstruction methods predict agent policies from environment observations via OMs [Albrecht and Stone, 2018], which has been shown to be beneficial in collaborative [Carroll *et al.*, 2019], competitive [Nashed and Zilberstein, 2022] and mixed settings [Hong *et al.*, 2018]. Deep Reinforcement Opponent Modelling (DRON) was one of the first works combining DRL with opponent modelling [He *et al.*, 2016]. The authors used two networks, one that learns Q-values using Deep Q-Network (DQN) [Mnih *et al.*, 2013], and another that learns an opponent's policy by observing hand-crafted opponent features. Their key innovation is to combine the output of both networks to compute a Q-function that is **conditioned** on (a latent encoding of) the approximated opponent's policy. This accounts for a given agent's Q-value dependency on the other agents' policies. Deep Policy Inference Q-Network (DPIQN) [Hong *et al.*, 2018] brings two further innovations: (1) merging both modules into a single neural network, and (2) baking the aforementioned Q function conditioning into the neural network architecture by reusing parameters from the OM module in the Q function.

OMs within MCTS have been shown to improve search [Timbers *et al.*, 2022] when assuming access to ground truth opponent models, or partially correct models [Goodman and Lucas, 2020]. As auxiliary tasks for the apprentice, OMs have been used in ExIt to predict the follow-up move an opponent would play in sequential games [Wu, 2019]. BRExIt both learns opponent models as an auxiliary task, and uses them inside of MCTS.

3 Background

Section 3.1 introduces relevant RL and game theory constructs, followed by the approach to opponent modelling used in BRExIt in Section 3.2 and the inner workings of ExIt in Section 3.3.

3.1 Multiagent Reinforcement Learning

Let E represent a fully observable stochastic game with n agents, state space S, shared action space A and shared policy space Π. Policies are stochastic mappings from states to actions, with $\pi_i : S \times A \rightarrow [0, 1]$ denoting the ith agent's policy, and $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_n]$ the joint policy vector, which can be regarded as a distribution over the joint action space. $T : S \times A \times S \rightarrow [0, 1]$ is the transition model (the environment dynamics), determining how an environment state s changes to a new state s' given a joint action $\boldsymbol{a}$. $R_i : S \times A \times S \rightarrow \mathbb{R}$ is agent i's reward function. $G^i_t \in \mathbb{R}$ is the return from time t for agent i, the accumulated reward obtained by agent i from time t until episode termination; $\gamma \in (0, 1]$ is the environment's discount factor. When considering the viewpoint of a specific agent i, we decompose a joint action vector $\boldsymbol{a} = (a_i, \boldsymbol{a_{-i}})$ and joint policy vector $\boldsymbol{\pi} = (\pi_i, \boldsymbol{\pi_{-i}})$ into the individual action or policy for agent i, and the other agents denoted by $-i$.

The state-value function $V^{\boldsymbol{\pi}}_i : S \rightarrow \mathbb{R}$ denotes agent i's expected cumulative reward from state s onwards assuming all agents act as prescribed by $\boldsymbol{\pi}$.

$$V^{\boldsymbol{\pi}}_i(s) = \sum_{\boldsymbol{a} \in A} \boldsymbol{\pi}(\boldsymbol{a}|s) \sum_{s' \in S} T(s'|s, a_i, \boldsymbol{a_{-i}})[R_i(s, a_i, \boldsymbol{a_{-i}}, s') + \gamma V^{\boldsymbol{\pi}}_i(s')]$$

Agent i's optimal policy depends on the other policies:

$$\pi^*_i(\cdot|s) = \arg\max_{\pi_i} V^{(\pi_i, \boldsymbol{\pi_{-i}})}_i(s) \tag{1}$$

Assuming $\boldsymbol{\pi_{-i}}$ to be stationary, agent i's optimal policy π^*_i is also called a **best response** $\pi^*_i \in BR(\boldsymbol{\pi_{-i}})$, where $BR(\boldsymbol{\pi_{-i}})$ denotes the set of all best responses against $\boldsymbol{\pi_{-i}}$. Our goal is to train a system to (1) predict and encode what is knowable about the opponents' policies $\boldsymbol{\pi_{-i}}$ and (2) compute a best response to that.

3.2 Opponent modelling in DRL

We follow the opponent modelling approach popularized by DPIQN [Hong *et al.*, 2018], which uses a neural network to both learn opponent models and an optimal Q-function. The latter is learnt by minimising the loss function $\mathcal{L}_Q$ of DQN [Mnih *et al.*, 2013]. Opponent modelling is an auxiliary task, trained by minimising the cross-entropy between one-hot encoded observed action for each agent j, $\boldsymbol{a}^j$, and their corresponding predicted opponent policies at state s, $\hat{\pi}_j(\cdot|s)$, defined as the **policy inference** loss $\mathcal{L}_{PI}$ in Equation 2. These two losses are combined into $\mathcal{L}_{DPIQN}$ with an

adaptive weight to improve learning stability.

$$\mathcal{L}_{PI} = -\frac{1}{N}\sum_{j=0}^{N} \boldsymbol{a_j} log(\hat{\pi}_j(\cdot|s)) \tag{2a}$$

$$\mathcal{L}_{DPIQN} = \frac{1}{\sqrt{\mathcal{L}_{PI}}}\mathcal{L}_Q + \mathcal{L}_{PI} \tag{2b}$$

BRExIt makes use of both the policy inference loss for its OMs and the adaptive learning weight to regularize its critic loss. However, instead of learning a Q-function as a critic, BRExIt learns a state-value function V, as suggested by previous work [Hernandez-Leal *et al.*, 2019].

3.3 Expert Iteration

We use an open-loop MCTS implementation [Silver *et al.*, 2018] as the expert; tree nodes represent environment states s, and edges (s, a) represent taking action a at state s. Each edge stores a set of statistics:

$$\{N(s,a), Q(s,a), P(s,a), i, A_n\} \tag{3}$$

$N(s,a)$ is the number of visits to edge (s,a). $Q(s,a)$ is the mean action-value for (s,a), aggregated from all simulations that have traversed (s,a). $P(s,a)$ is the prior probability of following edge (s,a). ExIt uses the apprentice policy to compute priors, $P(s,a) = \pi_{\boldsymbol{\theta}}(a|s)$. One of our contributions is adding opponent-awareness to the computation of these priors. Finally, index i denotes the player to act in the node and A_n indicates the available actions.

For every state s encountered by an ExIt agent during a training episode, the expert takes an action a computed by running MCTS from state s and selecting the action of the root node's most visited edge (s,a). During search, the tree is traversed using the same selection strategy as AlphaZero [Silver *et al.*, 2018]. Edges (s,a) are traversed following the most promising action according to the PUCT formula:

$$\arg\max_a \; Q(s,a) + C_{PUCT}\frac{P(s,a)\sqrt{\sum_{a'} N(s,a')}}{1+N(s,a)}, \tag{4}$$

where C_{PUCT} is a tunable constant. The apprentice $\pi_{\boldsymbol{\theta}}$ is a distillation of previous MCTS searches, and provides $P(s,a)$, thus biasing MCTS towards actions that were previously computed to be promising. Upon reaching a leaf node with state s', we backpropagate a value given by a learnt state value function $V_{\boldsymbol{\phi}}(s')$ with parameters $\boldsymbol{\phi} \in \mathbb{R}^n$ trained to regress against observed returns.

After completing search from a root node representing state s, the policy $\pi_{MCTS}(\cdot|s)$ can be extracted from the statistics stored on the root node's edges. This policy is stored as a training target for the apprentice's policy head to imitate:

$$\pi_{MCTS}(a|s) = \frac{N(s,a)}{\sum_{a'} N(s,a')} \tag{5}$$

As shown in the top right corner of Figure 1 ExIt builds a dataset containing a datapoint for each timestep t:

$$\{s_t, \pi_{MCTS}(\cdot|s_t), G_t^i\} \tag{6}$$

The top left corner of Figure 1 shows an actor-critic architecture, used as the apprentice, with a policy head $\pi_{\boldsymbol{\theta}}$, and a value head $V_{\boldsymbol{\phi}}$. A cross-entropy loss is used to train the actor towards imitating the expert's moves, and a mean-square error loss is used to update the critic's state value function towards observed returns G_t^i.

4 BRExIt: Opponent modelling in ExIt

We present Best Response Expert Iteration (BRExIt), an extension on ExIt that uses opponent modelling for two purposes. First, to enhance the apprentice's architecture with opponent modelling heads, acting as feature shaping mechanisms – see Subsection 4.1. Second, to allow the MCTS expert to approximate a best response against a set of opponents: $\pi_{MCTS} \in BR(\boldsymbol{\pi_{-i}})$ – see Subsection 4.3. We visually compare BRExIt and ExIt in Figure 1.

4.1 Learning opponent models in sequential games

Previous approaches to learn opponent models used observed actions as learning targets, coming from a game theoretical tradition where individual actions can be observed but not the policy that generated them [Brown, 1951]. However, the centralized population-based training schemes which motivate our research already require access during training to all policies in the environment, both for the training agent and the opponents' policies. We further exploit this assumption by separately testing two options for the learning targets in the policy inference loss from Equation 2: the one-hot encoded observed opponent actions, or the full distributions over actions computed by the opponents during play. As a clarifying example, imagine a uniform random policy for the game of Rock-Paper-Scissors where it plays *rock* for a given round. It's on-hot encoded action will be $[Rock : 1, Paper : 0, Scirros : 0]$ and the corresponding full distribution encoding would place equal weight over all actions: $[Rock : \frac{1}{3}, Paper : \frac{1}{3}, Scirros : \frac{1}{3}]$.

Prior work has typically focused on fully observable *simultaneous* games, where a shared environment state is used by all agents to compute an action at every timestep. Thus, if agent i wanted to learn models of its opponents' policies, storing (a) each of i's observed shared states and (b) opponent actions, was sufficient to learn opponent models. We extend this to sequential games, where agents take turns acting based on individual states, in the following way. BRExIt augments the dataset collected by ExIt, specified in Equation 6, by adding (1) the state which each opponent agent $j \neq i$ observed and (2) either only the observed opponent actions or the full ground truth action distributions given by the opponent policies in their corresponding turn. The data collection process for sequential environments is described in Algorithm 1. Formally, for the BRExIt agent acting at timestep t and its n_o opponents acting at timesteps $t+1$ to $t+n_o$, BRExIt adds to its dataset either the observed action of every agent j, or their policy π_j evaluated at the state of their turn.

$$\{s_t, \pi_{MCTS}(\cdot|s_t), \{s_{t+j}, \pi_j(\cdot|s_{t+j})\}_{j=1}^{n_o}, G_t^i\} \tag{7}$$

4.2 Apprentice Representation

BRExIt's three headed apprentice architecture extends ExIt's actor-critic representation with opponent models as per DPIQN's design [Hernandez-Leal *et al.*, 2019], depicted on

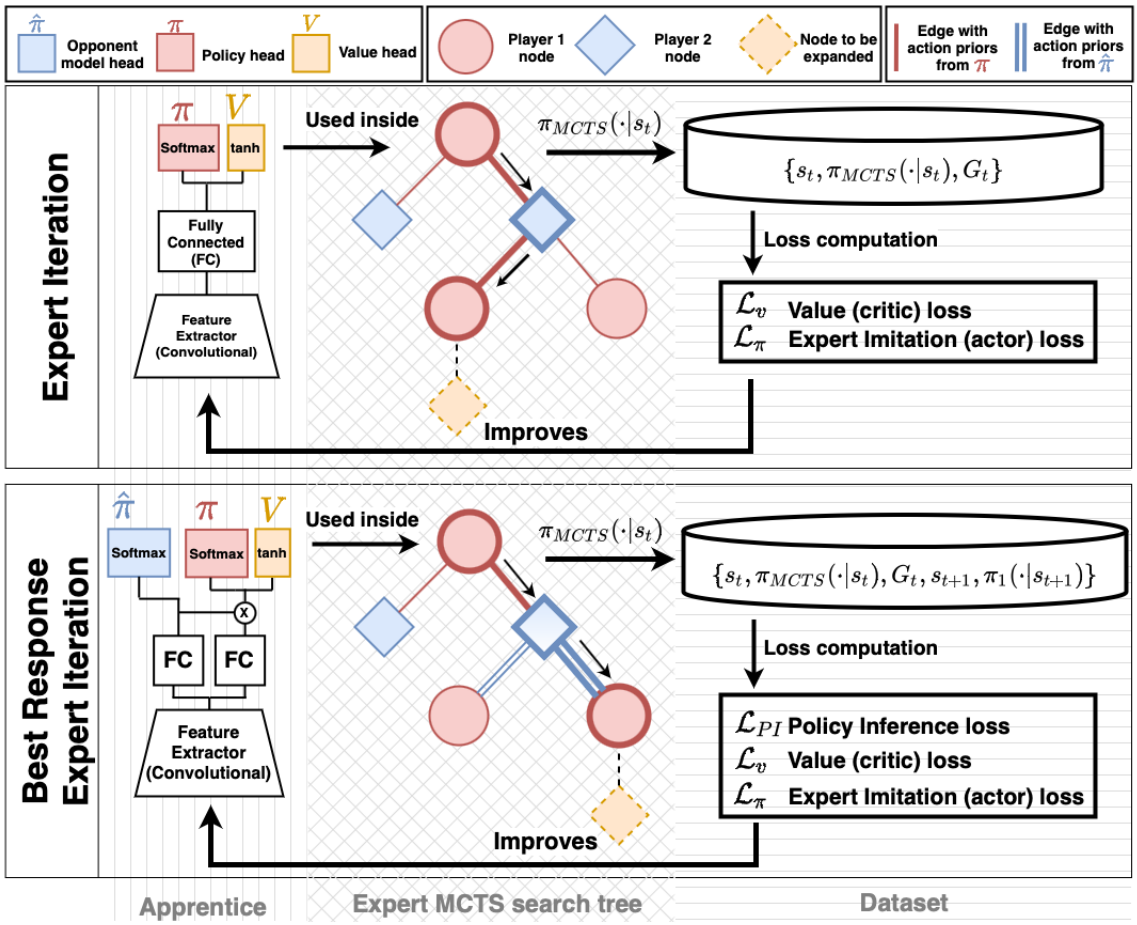


Figure 1: Illustration of ExIt (top half) and BRExIt (bottom half) for a 2-player game. BRExIt bases the decision on how to compute edge priors based on whose turn a node corresponds to. For opponent nodes, during training, these action priors can come from the true opponent's policy or from the apprentice's opponent model heads $\hat{\pi}$. BRExIt adds an opponent modelling loss by also gathering the states observed by the opponent s_{t+1} and either the output of their policy $\pi_1(\cdot|s_{t+1})$, or a corresponding one-hot encoded action.

the bottom left of Figure 1 for a single opponent. This architecture reuses parameters from the OM as a feature shaping mechanism for both the actor and the critic. It takes as input an environment state $s_t \in \mathcal{S}$ for a timestep t, which can correspond to the observed state of any agent. It features 3 outputs: (1) the apprentice's actor policy $\pi_{\boldsymbol{\theta}}(s_t)$ (2) the state-value critic $V_{\boldsymbol{\phi}}^{\pi_{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}}_{\boldsymbol{\Psi}}}(s_t)$ (3) the opponent models $\hat{\pi}^{\psi_j}(s_t) \in \hat{\boldsymbol{\pi}}^{\boldsymbol{\Psi}}$, where $\boldsymbol{\Psi}$ contains the parameters for all opponent models and $\psi_j \subset \boldsymbol{\Psi}$ the parameters for opponent model head j.

On certain sequential games the distribution of states encountered by each agent might differ, and so the actor-critic head and each of the OM heads could each be trained on different state distributions. In practice this means that the output of one of these heads might only be usable if the input state s_t for a timestep t comes from the distribution it was trained on; however, the OM can in any case help via feature shaping. This does not apply to simultaneous games where all agents observe the same state.

4.3 Opponent modelling inside MCTS

BRExIt follows Equation 8 to compute the action priors $P(s, a)$ from Equation 4 for a node with player index j, notably using opponent models in nodes within the search tree that correspond to opponents' turns. This process is exemplified in the lower middle half of Figure 1. Such opponent models can be either the ground truth opponent policies (the real opponent policies $\boldsymbol{\pi}_{-i}$) by exploiting centralized training scheme assumptions, or otherwise the apprentice's learnt opponent models $\hat{\boldsymbol{\pi}}^{\boldsymbol{\Psi}}$. This is a key difference from ExIt, which always uses the apprentice's $\pi_{\boldsymbol{\theta}}$ to compute $P(s, a)$.

$$P(s,a) = \begin{cases} \pi_{\boldsymbol{\theta}}\ (a|s) & j == \text{BRExIt player index} \\ \pi^j_{-i}(a|s) & \text{For ground truth models} \\ \hat{\pi}^{\psi_j}(a|s) & \text{For learnt opponent models} \end{cases} \quad (8)$$

By using either the ground truth or learnt opponent models, we initially bias the search towards a best response against the actual policies in the environment. However, note that initial $P(s, a)$ values will be overridden by the aggregated simulation returns $Q(s, a)$, as with infinite compute MCTS

converges to best response against a perfectly rational player, whose actions may deviate from the underlying opponent's policy. Thus, BRExIt's search maintains the asymptotic behaviour of MCTS while simultaneously priming the construction of the tree towards areas which are likely to be explored by the policies in the environment.

In contrast to BRExIt, ExIt biases its expert search towards a best response against the apprentice's own policy, by using the apprentice $\pi_{\boldsymbol{\theta}}$ to compute $P(s,a)$ at *every* node in the tree. MCTS here assumes that all agents follow the same policy as the apprentice, whereas in reality agents might follow any arbitrary policy. Not trying to exploit the opponents it is trained against, ExIt generates more conservative searches. Recent studies show that this conservativeness can be detrimental in terms of finding diverse sets of policies throughout training for population-based training schemes [Balduzzi *et al.*, 2019; Liu *et al.*, 2021]. Instead, they advocate for a more direct computation of best responses against known opponents as a means to discover a wider area of the policy space. This allows the higher level training scheme to better decide on which opponents to use as targets to guide future exploration. We argue that BRExIt has this property built-in, by actively biasing its search towards a best response against ground truth opponent policies. Future work could investigate this claim.

We could have designed BRExIt to be even more exploitative towards opponent models, by masking actions sampled from these models as part of the environment dynamics. While this would yield actions very specifically targeted to respond to the modelled policies, it makes such best reponses very brittle, and can be problematic especially for imperfect OMs. In contrast, we propose using opponent models as priors in BRExIt, such that planning can still improve upon the opponent policies; this results in more robust learning targets.

Algorithms 1, 2 and 3 depict BRExIt's data collection, model update logic and overarching training loop respectively for a sequential environment. Coloured lines represent our contributions w.r.t ExIt.

5 Experiments & Discussion

We are trying to answer the following two questions: Primarily, is BRExIt more performant than ExIt at distilling a competitive policy against fixed opponents into its apprentice? Secondarily, are full distribution targets for learning opponent models preferable over one-hot action encodings?

The environment: We conducted our experiments in the fully observable, sequential two-player game of Connect4, which is computational amenable and possesses a high degree of skill transitivity [Czarnecki *et al.*, 2020]. We decided on using a single environment in order to obtain statistically significant results through a larger number of runs over granular algorithmic ablations. We acknowledge the limitations of using a single test domain.

Test opponents: We generated two test agents π_{weak}, π_{strong} by freezing copies of a PPO [Schulman *et al.*, 2017] agent trained under $\delta = 0$-Uniform self-play [Hernandez *et al.*, 2019] after 200k and 600k episodes. Motivated by population-based training schemes, we also used an additional opponent policy π_{mixed}, which randomly

Algorithm 1: BRExIt data collection

Input: (apprentice $\pi_{\boldsymbol{\theta}}$, opp. models $\hat{\boldsymbol{\pi}}^{\boldsymbol{\psi}}_{-i}$, critic $V_{\boldsymbol{\phi}}$)
Input: *Opponent policies*: $\boldsymbol{\pi_{-i}}$
Input: Environment: $E = (\mathcal{P}, \rho_0)$
Initialize dataset: $D = [\,]$;
Initialize time $t \leftarrow 0$;
Sample initial state $s_0 \sim \rho_0$;
while s_t *is not terminal* **do**
  Search: $a_t, tree = MCTS(s_t, \pi_{\boldsymbol{\theta}}, \boldsymbol{\pi_{-i}}, V_{\boldsymbol{\psi}})$;
  Act in the game $s_{t+1}, r_t \sim \mathcal{P}(s_t, a_t)$;
  Get from $tree$: $\pi_{MCTS}(s_t, a) = \frac{N_{root}(s_t,a)}{\sum_{a'} N_{root}(s_t,a')}$;
  for $j = 1, \ldots, |\boldsymbol{\pi_{-i}}|$ **do**
    Sample opp. action: $a_{t+j} \sim \boldsymbol{\pi}^j_{\boldsymbol{-i}}(s_{t+j})$;
    Act in the game $s_{t+j,\_} \sim \mathcal{P}(s_{t+j}, a_{t+j})$
  end
  $D \cup \{s_t, \pi_{mcts}(s_t), \{s_{t+j}, \boldsymbol{\pi}^j_{\boldsymbol{-i}}(s_{t+j})\}_{j=1}^{|\boldsymbol{\pi}_o|}, r_i\}$;
  $t \leftarrow t + |\boldsymbol{\pi_{-i}}|$;
end
return D;

Algorithm 2: BRExIt model update

Input: *Three head network*: $NN = (\pi_{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}}^{\boldsymbol{\psi}}_{-i}, V_{\boldsymbol{\phi}})$
Input: *Dataset*: D
for $t = 0, 1, 2, \ldots$ **do**
  Sample n datapoints from D:
  $(s_t, r_t, \pi_{MCTS}(s_t, \cdot), r_i, \{s_{t+j}, \boldsymbol{\pi}^j_{\boldsymbol{-i}}(\cdot|s_{t+1})\}_{j=1}^{|\hat{\boldsymbol{\pi}}^{\boldsymbol{\psi}}_{-i}|})_{1,\ldots,n}$;
  MSE value loss: $\mathcal{L}_v = (v - V_{\boldsymbol{\phi}}(s_t))^2$;
  CE policy loss $\mathcal{L}_\pi = \pi_{MCTS}(s_t) \log(\pi_{\boldsymbol{\theta}}(s_t))$;
  CE policy inference loss:
  $\mathcal{L}_{PI} = \frac{1}{|\boldsymbol{\pi}_{-i}|} \sum_{j=1}^{|\boldsymbol{\pi}_{-i}|} \pi^j_{-i}(s_{t+j}) \log(\hat{\pi}^{\boldsymbol{\psi}_j}_{-i}(s_{t+j}))$;
  Policy inference weight: $\lambda = \frac{1}{\sqrt{\mathcal{L}_{PI}}}$;
  Weighted final loss $\mathcal{L}_{total} = \lambda(\mathcal{L}_v + \mathcal{L}_\pi) + \mathcal{L}_{PI}$;
  Backpropagate $\nabla \mathcal{L}_{total}$ through $\boldsymbol{\theta}, \boldsymbol{\psi}, \boldsymbol{\phi}$;
end

Algorithm 3: BRExIt training loop

Input: *Three head network*: $NN = (\pi_{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}}^{\boldsymbol{\psi}}_{-i}, V_{\boldsymbol{\phi}})$
Input: *Opponent policies*: $\boldsymbol{\pi_{-i}}$
for *training iteration* $= 0, 1, 2, \ldots$ **do**
  Algo. 1: $D = DatasetCollection(NN, \boldsymbol{\pi_{-i}})$;
  Algo. 2: $NN = UpdateApprentice(NN, D)$;
end
return NN

selects one of the test agents every episode.

Trained agents: We independently trained 7 types of agents for 48 wall-clock hours each, performing an additive construction from ExIt to BRExIt. **ExIt** is the original algorithm, **ExIt-OMFS** denotes ExIt using OMs *only* for feature shaping. **BRExIt-OMS** additionally uses learnt OMs during search and **BRExIt** uses the ground truth OMs during search. For the agents using OMs, we trained both a version using full action distributions as action targets and another with one-hot encoded action targets. Each algorithm was independently trained 10 times against the 3 test opponents, yielding a total of 280 training runs. Following statistical practices [Agarwal *et al.*, 2021] we use Inter Quartile Metrics (IQM) for all results, discarding the worst and best performing 25% runs to obtain performance metrics less susceptible to outliers.

5.1 On BRExIt's performance vs. ExIt

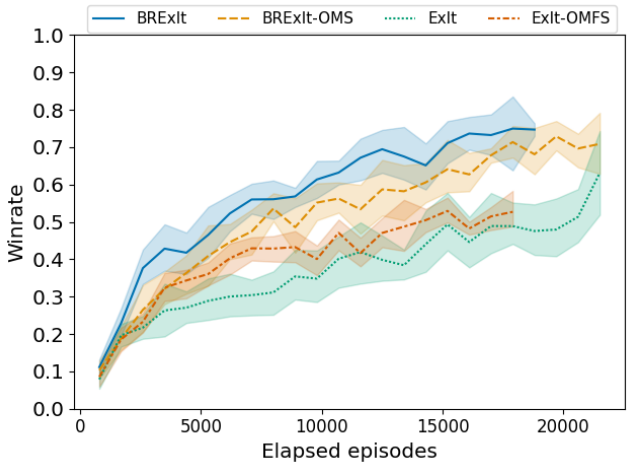
To answer our first question, Figure 2 shows the evolution of the winrate of each of the ablation's apprentice policies throughout training. A datapoint was computed every policy update (i.e every 800 episodes) by evaluating the winrate of the apprentice policy against the opponent over 100 episodes. (Note that the difference in number of episodes between all ablations depends on the average episode length over the 48h of training time, which can vary as a function of both players involved.) Figure 3 analyzes these results and shows the probability of improvement (PoI) that one ablation has over another, defined as the probability that algorithm X would yield an apprentice policy which has a higher winrate against its training opponent that algorithm Y [Agarwal *et al.*, 2021]. All agents are using full distribution OM targets here.

Figure 2 shows that BRExIt style agents consistently achieve a higher winrate than ExIt agents. BRExIt regularly outperforms BRExIt-OMS (77% PoI), successfully exploiting the centralized assumption of having opponent policies available during search for extra performance. If opponent policies can only be sampled for training games but not during search, using learnt OMs during search is still beneficial, as we see that BRExIt-OMS consistently outperforms ExIt (90 % PoI). Surprisingly, ExIt-OMFS performs worse than ExIt by a significant margin (the latter has a 80% PoI against the former), providing empirical evidence that OMs with static opponents can be *detrimental* for ExIt if OMs are not exploited within MCTS. This goes against previous results [Wu, 2019], which explored OMs within ExIt merely as a feature shaping mechanism and claimed modest improvements when predicting the opponent's follow-up move. Differences may be attributed to the fact that we model the opponent policy on the current state, instead of the next state.

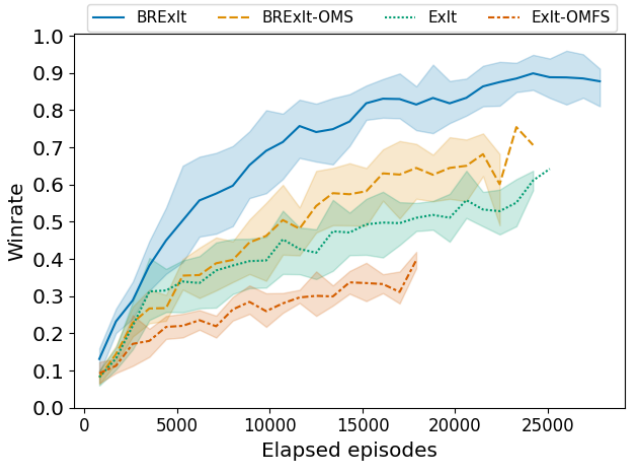
In summary, with BRExIt (using ground truth OMs) and BRExIt-OMS (using learnt OMs) featuring a $> 97\%$ and $> 91\%$ PoI respectively against vanilla ExIt, our empirical results warrant the use of our novel algorithmic variants instead of ExIt whenever opponent policies are available for training.

5.2 On full distribution VS one-hot targets

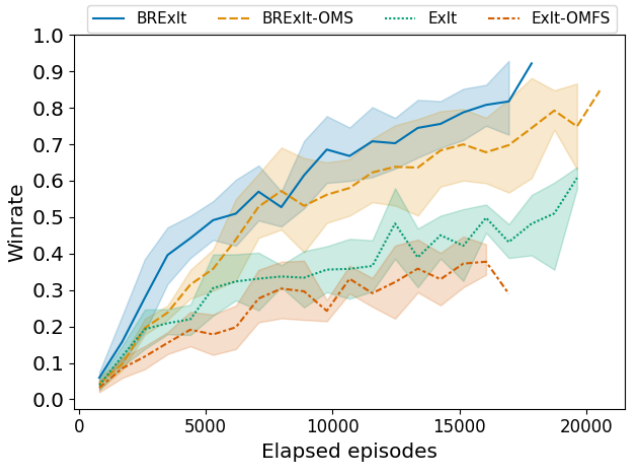
To answer our second question, we conducted Kolmogorov-Smirnov tests comparing full action distribution targets to



(a) VS Weak agent



(b) VS Strong



(c) VS Mixed

Figure 2: The evolution of winrates for each ablation during training vs fixed opponents for 48h wall-clock time. Lines represent the mean value the of final apprentice's winrate over all runs. Higher is better; shaded areas show 95% bootstrap confidence intervals.

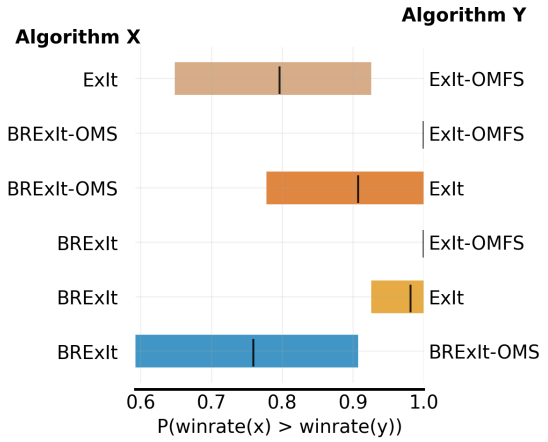


Figure 3: Each row shows the probability (vertical marker) that the algorithm X (left) trains its apprentice's policy to reach a higher winrate than algorithm Y (right) after the allotted 48h. Colored bars indicate 95% bootstrap confidence intervals. Note that every training run for both BRExIt and BRExIt-OMS yielded higher performing policies than any run from ExIt-OMFS, and thus their comparisons have a 100% probability of improvement over ExIt-OMFS.

one-hot encoded action targets for OMs. The samples we compared were the sets of winrates at the end of training, one datapoint per training run. Table 1 shows the results: There is no statistical difference between agents trained with algorithms using one-hot encoded OM targets when compared to using full action distributions. We obtain $p \gg 0.05$ for each algorithmic combination, so we cannot reject the null hypotheses that both data samples come from the same distribution; the only exception being ExIt-OMFS, which shows a statistically significant decrease in performance when using one-hot encoded targets.

These results run contrary to the intuition that richer targets for opponent models will in turn improve the quality of the apprentice's policy. However, we observed that full distributional targets do yield OMs with better prediction capabilities, as indicated by lower loss of the OM during training. This hints at the possibility that OM's usefulness increases only up to a certain degree of accuracy, echoing recent findings [Goodman and Lucas, 2020]. Hence, while BRExIt does require access to ground truth policies during search, search in BRExIt-OMS achieves similar performance with opponent models trained on action observations, which is promising for transferring BRExIt-OMS into practical applications.

6 Conclusion

We investigated the use of opponent modelling within the ExIt framework, introducing the BRExIt algorithm. BRExIt augments ExIt by introducing opponent models both within the **expert planning phase**, biasing its search towards a best response against the opponent, and within the **apprentice's model**, to use opponent modelling as an auxiliary task. In Connect4, we demonstrate BRExIt's improved performance compared to ExIt when training policies against fixed agents.

There are multiple avenues for future work. At the level of population based training schemes (such as self-play), future work can focus on measuring whether the quality of populations generated by different training schemes using BRExIt surpasses that of populations where ExIt is used as a policy improvement operator. In addition, search methods can struggle with complex games due to their high branching factor – simultaneous move games for example have combinatorial action spaces – which could be alleviated by using BRExIt's opponent models to narrow down the search space.

Table 1: Testing potential improvements of full distribution targets to one-hot encoded action targets. p-values are from two-sample Kolmogorov-Smirnov tests.

Base Algorithm	Opponent	p-value	Distr. targets signif. better?
BRExIt	Weak	0.930	No
BRExIt-OMS	Weak	0.931	No
ExIt-OMFS	Weak	0.930	No
BRExIt	Strong	0.930	No
BRExIt-OMS	Strong	0.999	No
ExIt-OMFS	Strong	0.930	No
BRExIt	Mixed	0.142	No
BRExIt-OMS	Mixed	0.930	No
ExIt-OMFS	Mixed	0.025	Yes

References

[Agarwal *et al.*, 2021] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

[Albrecht and Stone, 2018] Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

[Anthony *et al.*, 2017] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *NIPS*, 2017.

[Balduzzi *et al.*, 2019] David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In *International Conference on Machine Learning*, pages 434–443. PMLR, 2019.

[Berner *et al.*, 2019] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[Brown, 1951] George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

[Browne *et al.*, 2012] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of

Monte Carlo Tree Search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[Carroll *et al.*, 2019] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. In *Advances in Neural Information Processing Systems*, pages 5175–5186, 2019.

[Czarnecki *et al.*, 2020] Wojciech M Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. Real world games look like spinning tops. *Advances in Neural Information Processing Systems*, 33:17443–17454, 2020.

[Espeholt *et al.*, 2018] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. *CoRR*, abs/1802.01561, 2018.

[Goodman and Lucas, 2020] James Goodman and Simon Lucas. Does it matter how well I know what you're thinking? opponent modelling in an RTS game. In *IEEE Congress on Evolutionary Computation, CEC 2020*, pages 1–8. IEEE, 2020.

[He *et al.*, 2016] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning*, pages 1804–1813, 2016.

[Hernandez *et al.*, 2019] Daniel Hernandez, Kevin Denamganaï, Yuan Gao, Peter York, Sam Devlin, Spyridon Samothrakis, and James Alfred Walker. A generalized framework for self-play training. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2019.

[Hernandez-Leal *et al.*, 2017] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. arxiv 2017. *arXiv preprint arXiv:1707.09183*, 2017.

[Hernandez-Leal *et al.*, 2019] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. Agent modeling as auxiliary task for deep reinforcement learning. In Gillian Smith and Levi Lelis, editors, *Proceedings of the Fifteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2019*, pages 31–37. AAAI Press, 2019.

[Hong *et al.*, 2018] Zhang-Wei Hong, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, and Chun-Yi Lee. A deep policy inference q-network for multi-agent systems. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1388–1396. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[Lanctot *et al.*, 2017] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in Neural Information Processing Systems*, 30:4190–4203, 2017.

[Liu *et al.*, 2021] Xiangyu Liu, Hangtian Jia, Ying Wen, Yaodong Yang, Yujing Hu, Yingfeng Chen, Changjie Fan, and Zhipeng Hu. Towards unifying behavioral and response diversity for open-ended learning in zero-sum games. *Advances in Neural Information Processing Systems*, 34, 2021.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5602, 2013.

[Nashed and Zilberstein, 2022] Samer Nashed and Shlomo Zilberstein. A survey of opponent modeling in adversarial domains. *Journal of Artificial Intelligence Research*, 73:277–327, 2022.

[Oliehoek and Amato, 2014] Frans A Oliehoek and Christopher Amato. Best response bayesian reinforcement learning for multiagent systems with state uncertainty. In *Proceedings of the Ninth AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, 2014.

[Schaul *et al.*, 2015] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. *CoRR*, abs/1511.05952, 2015.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Silver *et al.*, 2018] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *Science*, 362:1140–1144, 2018.

[Soemers *et al.*, 2020] Dennis JNJ Soemers, Éric Piette, Matthew Stephenson, and Cameron Browne. Manipulating the distributions of experience used for self-play learning in expert iteration. *arXiv preprint arXiv:2006.00283*, 2020.

[Timbers *et al.*, 2022] Finbarr Timbers, Nolan Bard, Edward Lockhart, Marc Lanctot, Martin Schmid, Neil Burch, Julian Schrittwieser, Thomas Hubert, and Michael Bowling.

Approximate exploitability: Learning a best response. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3487–3493, 2022.

[Vinyals *et al.*, 2019] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[Willemsen *et al.*, 2021] Daniel Willemsen, Hendrik Baier, and Michael Kaisers. Value targets in off-policy alphazero: a new greedy backup. *Neural Computing and Applications*, pages 1–14, 2021.

[Wu, 2019] David J Wu. Accelerating self-play learning in go. *arXiv preprint arXiv:1902.10565*, 2019.