No-Press Diplomacy from Scratch

Anton Bakhtin David Wu Adam Lerer Noam Brown
Facebook AI Research
{yolo,dwu,alerer,noambrown}@fb.com

Abstract

Prior AI successes in complex games have largely focused on settings with at most hundreds of actions at each decision point. In contrast, Diplomacy is a game with more than 10^{20} possible actions per turn. Previous attempts to address games with large branching factors, such as Diplomacy, StarCraft, and Dota, used human data to bootstrap the policy or used handcrafted reward shaping. In this paper, we describe an algorithm for action exploration and equilibrium approximation in games with combinatorial action spaces. This algorithm simultaneously performs value iteration while learning a policy proposal network. A double oracle step is used to explore additional actions to add to the policy proposals. At each state, the target state value and policy for the model training are computed via an equilibrium search procedure. Using this algorithm, we train an agent, *DORA*, completely from scratch for a popular two-player variant of Diplomacy and show that it achieves superhuman performance. Additionally, we extend our methods to full-scale no-press Diplomacy and for the first time train an agent from scratch with no human data. We present evidence that this agent plays a strategy that is incompatible with human-data bootstrapped agents. This presents the first strong evidence of multiple equilibria in Diplomacy and suggests that self play alone may be insufficient for achieving superhuman performance in Diplomacy.

1 Introduction

Classic multi-agent AI research domains such as chess, Go, and poker feature action spaces with at most thousands of actions per state, and are therefore possible to explore exhaustively a few steps deep. In contrast, modern multi-agent AI benchmarks such as StarCraft, Dota, and Diplomacy feature combinatorial action spaces that are incredibly large. In StarCraft and Diplomacy, this challenge has so far been addressed by bootstrapping from human data [32, 24, 3, 11]. In Dota, this challenge was addressed through careful expert-designed reward shaping [4].

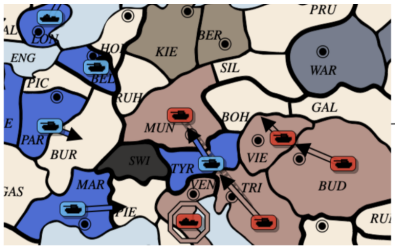

Figure 1: France (blue) would be in a strong position were it not for exactly two actions that Austria (red) can play to dislodge the unit in Tyr. The probability of sampling one of them randomly is roughly 10^{-6}.

In contrast, in this paper we describe an algorithm that trains agents through self play without any human data and can accommodate the large action space of Diplomacy, a long-standing AI benchmark game in which the number of legal actions for a player on a turn is often more than 10^{20} [24].

We propose a form of deep RL (reinforcement learning) where at each visited state the agent approximates a Nash equilibrium for the stage game with 1-step lookahead search, plays the equilibrium policy, and uses the equilibrium value as the training target. Since the full action space is too large to consider, this equilibrium is computed over a smaller set of candidate

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

actions. These actions are sampled from an action proposal network that is trained in unison with the value model. We show that our deep Nash value iteration algorithm, when initialized with a human-bootstrapped model, defeats multiple prior human-derived agents in no-press Diplomacy by a wide margin.

Next, we investigate training an agent completely from scratch with no human data in Diplomacy. In order to improve action exploration when training from scratch, the proposal actions are augmented with exploratory actions chosen through a double-oracle-like process, and if these discovered actions end up as part of the equilibrium then they will be learned by the proposal network. Using this technique, which we call *DORA* (Double Oracle Reinforcement learning for Action exploration), we develop a superhuman agent for a popular two-player variant of Diplomacy. Our agent is trained purely through self-play with no human data and no reward shaping.

We also train a *DORA* agent for the 7-player game of no-press Diplomacy, the first no-press Diplomacy agent to be trained entirely from scratch with no human data. When 6 *DORA* agents play with 1 human-data-based agent, all human-data-based agents perform extremely poorly. However, when one *DORA* agent plays with 6 human-data-based agents, the bot underperforms. These results suggest that self-play in 7-player Diplomacy may converge to equilibria that are incompatible with typical human play. This is in contrast to previous large-scale multi-agent benchmarks like multiplayer poker [9] and validates Diplomacy as a valuable benchmark for learning policies that are compatible with human behavior.

2 Background and Related Work

2.1 Description of Diplomacy

In this section we summarize the rules of Diplomacy. For a more detailed description, see [24]. No-press Diplomacy is a zero-sum board game where 7 players compete for control on a map of Europe. The map contains 75 locations, 34 of which are *supply centers* (SCs). Each player begins with 3-4 units and 3-4 SCs and a player wins by controlling a majority (18) of the SCs. If no player controls a majority and all remaining players agree to a draw, then the victory is divided between the uneliminated players. In case of a draw, we use the **Sum-of-Squares (SoS)** scoring system that defines the score of player i as $C_i^2 / \sum_{i'} C_{i'}^2$, where C_i is the SC count for player i.

Each turn during movement, all players simultaneously choose an action composed of one order for every unit they control. An order may direct a unit to hold stationary, to move to an adjacent location, to support a nearby unit's hold or move, to move across water via convoy, or to act as the convoy for another unit. Because a player may control up to 17 units with an average of 26 valid orders for each unit, the number of possible actions is typically too large to enumerate. Additionally, in Diplomacy, often supports or other critical orders may be indistinguishable from a no-op except in precise combinations with other orders or opponent actions. This makes the action space challenging to learn with black-box RL. Actions are simultaneous, so players typically must play a mixed strategy to avoid being predictable and exploitable.

In the popular *France vs. Austria (FvA)* variant of Diplomacy, the number of players is reduced to two, but the victory condition and all game rules, including the number of locations and centers, remain the same. FvA serves as an effective way to isolate the challenge of an astronomically large imperfect-information action space from the challenge of Diplomacy's multi-agent nature.

2.2 Stochastic Games

Stochastic games [26] are a multi-agent generalization of Markov decision processes. At each state, each player chooses an action and the players' joint action determines the transition to the next state. Formally, at each state $s \in S$ agents simultaneously choose a joint action $\boldsymbol{a} = (a_1, \ldots, a_{\mathcal{N}}) \in A_1 \times \cdots \times A_{\mathcal{N}}$, where $\mathcal{N}$ is the number of players. Agents then receive a reward according to a vector function $\boldsymbol{r}(s, \boldsymbol{a}) \in \mathbb{R}^{\mathcal{N}}$ and transition to a new state $s' = f(s, \boldsymbol{a})$. The game starts at initial state s^0 and continues until a terminal state is reached, tracing a trajectory $\tau = ((s^0, \boldsymbol{a}^0), ..., (s^{t-1}, \boldsymbol{a}^{t-1}))$ where $f(s^{t-1}, \boldsymbol{a}^{t-1})$ is terminal. Each agent i plays according to a policy $\pi_i : S \to \Delta A_i$, where ΔA_i is a set of probability distributions over actions in A_i. Given a policy profile $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_{\mathcal{N}})$, the probability of a trajectory is $p(\tau|\boldsymbol{\pi}) = \prod_{(s,\boldsymbol{a})\in\tau} p(\boldsymbol{a}|\, \boldsymbol{a} \sim \boldsymbol{\pi}(s))$, and we define the vector of

expected values of the game rooted at state s as $\boldsymbol{V}(\boldsymbol{\pi}, s) = \sum_{\tau \in T(s)} p(\tau|\boldsymbol{\pi}) \sum_{(s_t, \boldsymbol{a_t}) \in \tau} \gamma^t \boldsymbol{r}(s_t, \boldsymbol{a_t})$ where $T(s)$ is the set of terminal trajectories beginning at s. We denote by $\boldsymbol{V}(\boldsymbol{\pi}) := \boldsymbol{V}(\boldsymbol{\pi}, s^0)$ the value of a game rooted in the initial state. We denote by $\boldsymbol{\pi}_{-i}$ the policies of all players other than i.

A **best response (BR)** for agent i against policy profile π is any policy that achieves maximal expected value. That is, π_i^* is a BR to π if $V_i(\pi_i^*, \pi_{-i}) = \max_{\pi_i'} V_i(\pi_i', \pi_{-i})$. A policy profile is a **Nash equilibrium (NE)** if all players' policies are BRs to each other. Letting V_i be the ith component of $\boldsymbol{V}$, $\boldsymbol{\pi}$ is a NE if:

$$\forall i \, \forall \pi_i' : V_i(\boldsymbol{\pi}) \geq V_i(\pi_i', \boldsymbol{\pi}_{-i})$$

An **ϵ-NE** is a policy profile in which an agent can only improve its expected reward by at most ϵ by switching to a different policy. If a policy profile is an ϵ-NE, we say its **exploitability** is ϵ.

We may also talk about Nash equilibria for one stage of the game rather than the whole game. For any state s, supposing we have values $Q(s, \boldsymbol{a})$ that represent the expected values of playing joint action $\boldsymbol{a}$ in state s assuming some policy profile is played thereafter, we may define a new 1-step game where instead the game ends immediately after this action and $Q(s, \boldsymbol{a})$ is itself the final reward. For such stage games, notationally we denote NE using the symbol σ rather than π.

2.3 Nash Q-Learning

Value iteration methods such as Nash-Q have been developed that provably converge to Nash equilibria in some classes of stochastic game, including 2p0s (two-player zero-sum) stochastic games [19, 14]. Nash-Q learns joint Q values $\boldsymbol{Q}(s, \boldsymbol{a})$ that aim to converge to the state-action value of $(s, \boldsymbol{a})$ assuming that some NE $\boldsymbol{\pi}$ is played thereafter. This is done by performing 1-step updates on a current estimated function $\boldsymbol{Q}$ as in standard Q-learning, but replacing the max operation with a stage game NE computation.

Formally, suppose that $\boldsymbol{a}$ is played in s leading to s'. A NE σ, $\sigma_i \in \Delta A_i$, is computed for the stage game at s' whose payoffs for each joint action $\boldsymbol{a}'$ are specified by the current values of $\boldsymbol{Q}(s', \boldsymbol{a}')$. The value of this game to each player assuming this NE is then used as the target values of $\boldsymbol{Q}(s, \boldsymbol{a})$. The update rule is thus:

$$\boldsymbol{Q}(s, \boldsymbol{a}) \leftarrow (1 - \alpha)\boldsymbol{Q}(s, \boldsymbol{a}) + \alpha\Big(\boldsymbol{r}(s, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{a}'} \sigma(\boldsymbol{a'})\boldsymbol{Q}(s', \boldsymbol{a}')\Big), \tag{1}$$

where α is the learning rate, γ is the discount factor, and $\sigma(\boldsymbol{a}) := \prod_i \sigma_i(a_i)$ is the probability of joint action $\boldsymbol{a}$.

As presented later in Section 3.1, the foundation of our algorithm is an adaptation of Nash Q-Learning, with several modifications to make it suitable for deep RL and adjusted to use a value function rather than a Q function.

2.4 Double Oracle

Double oracle (DO) is an iterative method for finding a NE over a large set of actions in a matrix game [22]. It leverages the fact that computing a best response is computationally much cheaper than computing a NE. Starting from a small set of candidate actions $A_i^0 \subseteq A_i$ for each player i, on each iteration it computes a NE σ^t in the matrix game restricted to the candidate actions $\boldsymbol{A}^t$. Then, at each iteration t for each player i it finds the best response action $a_i^{t+1} \in A_i$ to the restricted NE σ^t and creates $A_i^{t+1} := A_i^t \bigcup \{a_i^{t+1}\}$. Formally, a_i^{t+1} is an action in the full action set A_i that satisfies

$$a_i^{t+1} = \arg\max_{a' \in A_i} \sum_{\boldsymbol{a} \in \boldsymbol{A}^t} Q_i(s, a_1, \ldots, a_{i-1}, a', a_{i+1}, \ldots, a_{\mathcal{N}}) \prod_{j \neq i} \sigma_j^t(a_j). \tag{2}$$

DO-inspired methods have been used in a variety of other work for handling large or continuous action spaces [15, 7, 18, 1, 21]. Although in the worst case the equilibrium policy may have support over a large fraction of the action space, in practice in many problems a small subset of actions may suffice, and in this case DO-based methods can be computationally efficient due to being able to restrict equilibrium computation to only a tiny fraction of the total number of actions.

As we will present in Section 3.2, we make use of a DO-like procedure both within the RL loop and at inference time, and show it to be an effective way to introduce novel good actions to the policy.

2.5 Regret Matching

Both Nash-Q-Learning and DO require the computation of NEs in matrix games, which we approximate using the iterative **regret matching (RM)** algorithm [5, 13], as has been done in past work [11]. RM is only guaranteed to converge to a NE in 2p0s games and other special classes of games [12]. Nevertheless, past work has found that it produces competitive policies that have low exploitability in important non-2p0s games including six-player poker and no-press Diplomacy [9, 11]. We therefore use RM to approximate a NE σ in restricted matrix games during DO.

In RM, each player i has a **regret** value for each action $a_i \in A_i$. The regret on iteration t is denoted $\text{Regret}_i^t(a_i)$. Initially, all regrets are zero. On each iteration t of RM, $\sigma_i^t(a_i)$ is set according to

$$\sigma_i^t(a_i) = \begin{cases} \frac{\max\{0, \text{Regret}_i^t(a_i)\}}{\sum_{a_i' \in \mathcal{A}_i} \max\{0, \text{Regret}_i^t(a_i')\}} & \text{if } \sum_{a_i' \in \mathcal{A}_i} \max\{0, \text{Regret}_i^t(a_i')\} > 0 \\ \frac{1}{|\mathcal{A}_i|} & \text{otherwise} \end{cases} \tag{3}$$

Next, each player samples an action a_i^* from $\mathcal{A}_i$ according to σ_i^t and all regrets are updated such that

$$\text{Regret}_i^{t+1}(a_i) = \text{Regret}_i^t(a_i) + v_i(a_i, a_{-i}^*) - \sum_{a_i' \in \mathcal{A}_i} \sigma_i^t(a_i') v_i(a_i', a_{-i}^*), \tag{4}$$

where $v_i(\overline{a})$ is the utility of player i in the matrix game. This sampled form of RM guarantees that the *average* policy over all iterations converges to a ϵ-NE in 2p0s games at a rate of at worst $\mathcal{O}(\frac{1}{\epsilon^2})$ iterations with high probability [17]. In order to improve empirical performance, we use linear RM [8], which weighs updates on iteration t by t.[1] We also use optimism [30], in which the most recent iteration is counted twice when computing regret.

3 Algorithms

We now describe DORA in detail. DORA simultaneously learns a state-value function and an action proposal distribution via neural networks trained by bootstrapping on an approximate Nash equilibrium for the stage game each turn. A DO-based process is used to discover actions.

3.1 Deep Nash Value Iteration

The core of the learning procedure is based on Nash Q-Learning as presented in Section 2.3, but simplified to use only a value function, with adaptations for large action spaces and function approximation.

Approximating a Q-function in a game with an action space as large as Diplomacy is challenging. However, in Diplomacy as in many other domains, the reward $\boldsymbol{r}(s, a)$ depends only on the next state $s' = f(s, a)$ and not on how it was reached, and since the transition function is known and can be simulated exactly, we can redefine $\boldsymbol{r}$ to be a function of just the next state s'. In that case, we can rewrite Update Rule 1 (from Section 2.3) in terms of a state value function:

$$\boldsymbol{V}(s) \leftarrow (1-\alpha)\boldsymbol{V}(s) + \alpha(\boldsymbol{r}(s) + \gamma \sum_{\boldsymbol{a'}} \sigma(\boldsymbol{a'})\boldsymbol{V}(f(s, \boldsymbol{a'}))) \tag{5}$$

Since the state space is large, we use a deep neural network $\boldsymbol{V}(s; \theta_v)$ with parameters θ_v to approximate $\boldsymbol{V}(s)$. Since the action space is large, exactly computing the Nash equilibrium σ for the 1-step matrix game at a state is also infeasible. Therefore, we train a policy proposal network $\pi(s; \theta_\pi)$ with parameters θ_π that approximates the distribution of actions under the equilibrium policy at state s. We generate a candidate action set at state s by sampling N_b actions from $\pi(s; \theta_\pi)$ for each player and selecting the $N_c \ll N_b$ actions with highest likelihood. We then approximate a NE σ via RM (described in Section 2.5) in the restricted matrix game in which each player is limited to their N_c actions assuming successor states' values are given by our current value network $\boldsymbol{V}(s; \theta_v)$.

We explore states via self-play to generate data, with both players following the computed NE policy with an ϵ probability of instead exploring a random action from among the N_c proposals. Also, in

[1] In practice, rather than weigh iteration t's updates by t we instead discount prior iterations by $\frac{t}{t+1}$ in order to reduce numerical instability. The two options are mathematically equivalent.

place of θ_v and θ_π, self-play uses slightly-lagged versions of the networks $\hat{\theta}_v$ and $\hat{\theta}_\pi$ to compute σ that are only updated periodically from the latest θ_v and θ_π in training, a normal practice that improves training stability [23]. We then regress the value network towards the computed stage game value under the NE σ using MSE loss (whose gradient step is simply the above update rule), and regress the policy proposal network towards σ using cross entropy loss:

$$\begin{aligned}
\text{ValueLoss}(\theta_v) &= \frac{1}{2}\left(\boldsymbol{V}(s;\theta_v) - \boldsymbol{r}(s) - \gamma \sum_{\boldsymbol{a'}} \sigma(\boldsymbol{a'})\boldsymbol{V}\left(f(s,\boldsymbol{a'});\hat{\theta}_v\right)\right)^2 \\
\text{PolicyLoss}(\theta_\pi) &= -\sum_i \sum_{a_i \in A_i} \sigma_i(a) \log \pi_i(s, a_i; \theta_\pi),
\end{aligned} \tag{6}$$

We show in Appendix B that in 2p0s games, the exact tabular form of the above algorithm with mild assumptions and without the various approximations for deep RL provably converges to a NE. Outside of 2p0s games and other special classes, both Nash Q-learning and RM lack theoretical guarantees; however, prior work on Diplomacy and 6-player poker have shown that similar approaches still often perform well in practice [11, 3, 9], and we also observe that DORA performs well in 7-player no-press Diplomacy.

At a high level our algorithm is similar to previous algorithms such as AlphaZero [28] and ReBeL [6] that update both a policy and value network based on self-play, leveraging a simulator of the environment to perform search. The biggest differences between our algorithm and AlphaZero are that 1-ply RM acts as the search algorithm instead of Monte Carlo tree search (just as in [11]), the value update is uses a 1-step bootstrap instead of the end-of-game-value (which allows the trajectory generation to be off-policy if desired), and the policy network acts as an action proposer for the search but does not regularize the search thereafter. Our algorithm is also similar to a recent method to handle large action spaces for Q-learning [31] and reduces to it in single-agent settings.

3.2 Double-Oracle-Based Action Discovery

The algorithm in Section 3.1 relies on the policy network to generate good candidate actions. However, good actions initially assigned too low of a probability by the network may never be sampled and reinforced. To address this, we use a mechanism based on double oracle, described in Section 2.4, to introduce novel actions. However, vanilla DO requires computing the expected value of all actions against the restricted matrix game NE. Since our action space is too large to enumerate, we generate only a pool of $N_p \gg N_c$ actions to consider.

A natural idea to generate this pool would be to sample actions uniformly at random from all actions. However, our results in Section 5.1 show this approach performs poorly. Intuitively, in a combinatorial action space as large as in Diplomacy often only a minuscule fraction of actions are reasonable, so random sampling is exceedingly unlikely to find useful actions (see Fig 1 for an example).

We instead generate the pool of N_p actions for a player by generating local modifications of the candidate actions already in the restricted matrix game equilibrium. We first sample uniformly among the player's N_d ($d \leq c$) most probable actions in that equilibrium. Next, for each sampled action, we randomly sample a single location on the map and consider all units of the player adjacent to the location. For those units, we randomly choose legal valid new orders. We add the order in this pool that best-responds to the equilibrium as a new candidate action, and DO proceeds as normal.

As shown in Table 3, we found this method of action sampling to outperform simply picking legal and valid combinations of orders uniformly at random. Intuitively, this is likely because locality in Diplomacy, as in a wide variety of other problem domains and algorithms, is a powerful inductive bias and heuristic. Perturbations of good actions are vastly more likely than random actions to also be good, and spatially-nearby actions are more likely to need to covary than spatially-distant ones.

To further improve the scalability of our algorithm, we apply several additional minor approximations, including computing some of the above expected values using sampling and capping the number of DO iterations we perform. For details, see Appendix D.

4 Implementation and Methods

We describe our implementation of the above algorithms and training methods as applied to training in Diplomacy. For details on the exact parameters, see Appendix D. The code and the models are available online [2].

4.1 Data Generation and Training

We implement deep Nash value iteration following the same considerations as deep Q-learning. Multiple data generation workers run self-play games in parallel. The policy and value targets on each turn along with the game state they correspond to are collected in a shared experience replay buffer and sampled by a set of training workers to train the networks. The policy proposal and value networks used by the data generation workers are only periodically updated from those in training.

During self play, each agent plays according to its part of the computed approximate-NE policy σ with probability $1 - \epsilon$, and with probability ϵ plays a uniformly random action from the N_c actions obtained from the policy proposal network. We also use larger values of ϵ on the first two turns to quickly introduce diversity from the opening state, similar to the use of larger early exploration settings in other work, for example AlphaZero [29, 28]. Detailed parameters are in Appendix D.

Algorithm 1 Approximated Double Oracle

function FINDEQUILIBRIUMWITHDO(s, N_c, N_p, N_{iters}, ϵ)
  $\boldsymbol{A} \leftarrow$ SAMPLEACTIONSFROMPROPOSALNET(s, N_c)
  $\boldsymbol{\sigma} \leftarrow$ COMPUTEEQUILIBRIUM(s, $\boldsymbol{A}$)
  for $t = (1..N_{iters})$ **do**
    modified $\leftarrow 0$
    for $i = (1..N_{players})$ **do**
      $\hat{A}_i \leftarrow$ GENERATEACTIONS(s, A_i, N_p) ▷ Uniform sampling or local modification
      $v_i \leftarrow \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\sigma}}\left[V_i(f(s, \boldsymbol{a}); \theta_v)\right]$
      $a_i^*, v_i^* \leftarrow \underset{a_i \in \hat{A}_i}{\arg\max}\, \mathbb{E}_{\boldsymbol{a_{-i}}\sim\boldsymbol{\sigma_{-i}}}\left[V_i(f(s, \boldsymbol{a}); \theta_v)\right]$
      if $v_i^* - v_i > \epsilon$ **then**
        $A_i \leftarrow A_i \bigcup \{a_i^*\}$
        $\boldsymbol{\sigma} \leftarrow$ COMPUTEEQUILIBRIUM(s, $\boldsymbol{A}$)
        modified $\leftarrow 1$
    if !modified **then**
      Break
  return $\boldsymbol{\sigma}$

function GENERATEACTIONSLOCALMODIFICATION(s, A_{base}, N_p)
  $A \leftarrow \{\}$
  while $|A| < N_p$ **do**
    $a \leftarrow$ SAMPLEACTION(A_{base})
    $loc \leftarrow$ SAMPLEMAPLOCATION()
    $clique \leftarrow$ GETNEIGHBOURS(loc)
    $a' \leftarrow$ RANDOMCOORDINATEDMODIFICATION(a, $clique$)
    $A \leftarrow A \bigcup \{a'\}$
  return A

4.2 Model Architecture

Our model closely follows the encoder-decoder architecture of [11]. We use the policy head to model the policy proposal distribution and the value head to predict the expected values of the players.

We found several adjustments to be effective for our use in deep Nash-based RL. Firstly, we removed all dropout layers, as we found this to improve training stability. Secondly, we replaced the graph convolution-based encoder with a simpler transformer encoder, improving model quality and removing the need to hardwire the adjacency matrix of the Diplomacy map into the architecture. Lastly, we also split the network into separate policy and value networks. Since our regret minimization procedure is essentially only a 1-ply search, the states on which we need to evaluate policy

[2] `https://github.com/facebookresearch/diplomacy_searchbot`

(the current state) and value (the states after one turn) are disjoint, so there is no gain to computing both policy and value in one network query, and we found splitting the networks improved the quality and effective capacity of the models. We measure the impact of these choices in Table 2 and describe the architecture in more detail in Appendix A.

4.3 Exploitability Testing

To obtain a metric of the exploitablity of our final agents, we also train additional exploiter agents that rather than attempting to play an equilibrium, instead hold the exploited agent fixed and attempt to converge to a best response. We compare two approaches.

In the first approach, we reimplement the same actor-critic method with sampled entropy regularization previously used in [11] for 7-player no-press Diplomacy and apply it to 2-player FvA. In this approach, the other agent is treated as a black box, reducing the problem to just a standard Markov Decision Process to which we can apply classical RL techniques. Namely, we use Impala [10], an asynchronous version of actor-critic algorithm. Since search agents may take several seconds to act, [11] first trained a policy network to copy the behavior of the search agent and then applied RL to exploit this proxy agent. The proxy agent acts much faster, which enabled generating data faster. In contrast, we optimized the training loop so that it was possible to train to exploit the search agent directly without first generating a proxy policy network agent.

In the second approach, we begin with the policy and value models of the exploited agent and resume deep Nash value iteration, except with a single simple modification, described in Appendix E, that results in learning a best response rather than an equilibrium.

5 Results

Diplomacy features two main challenges for AI research: its large combinatorial action space and its multi-agent (non-2p0s) nature. *DORA* is intended to address the former challenge. To measure its effectiveness on just the action-exploration aspect, we first present results in a 2p0s variant of Diplomacy called France vs. Austria (FvA). We show *DORA* decisively beats top humans in FvA.

Next, we present results for 7-player no-press Diplomacy, which has been a domain for much prior research [24, 3, 11]. We train a *DORA* agent from scratch with no human data and test it against pre-trained models from previous work [24, 11][3]. In the 6v1 setting, (six *DORA* agents playing with one copy of another bot) *DORA* wins overwhelmingly against all other agents. However, in the 1v6 setting (one *DORA* agent playing with six copies of another bot), *DORA* does slightly worse than previous search agents that leveraged human data. Our results suggest that *DORA* converges to strong play within an equilibrium (i.e., a "metagame") that is very different from how humans play. Finally, we present results showing that adding our RL algorithm on top of a human-bootstrapped model clearly beats prior bots in both the 1v6 and 6v1 settings.

5.1 Results in 2-player France vs. Austria

	Austria	**France**	**Total**
Average score (+variance reduction)	89%±5%	66%± 7%	78%±4%
Raw average score	100%±0%	73%±12%	87%±6%
# games	13	13	26

Table 1: Head-to-head score of *DORA* versus top FvA players on `webdiplomacy.net`. The ± shows one standard error. Variance reduction (Appendix C) was used to compute average scores from raw average scores. Note that for raw Austria score, because DORA managed to win *all* 13 games, anomalously the sample standard deviation is technically 0. The true standard deviation is of course larger but cannot be estimated.

In 2-player FvA, we tested our final *DORA* agent against multiple top players of this variant, inviting them to play a series of games against our agent. All players were allowed to play an arbitrary number of practice games against the agent before playing official counted games. Prior to running the games, we also implemented a simple variance reduction method, described in Appendix C.

[3] See Appendix D for the details.

In Table 1 we report both the raw and variance-reduced average scores. DORA defeats top human players by a large and highly significant margin, with an average score of 78% after variance reduction. It also wins significantly more than half of games as France, the side considered much harder to play[4]. These results show DORA achieves a level of play in FvA above that of top humans.

We ran several ablations to loosely estimate the relative importance of the various components contributing to our final agent for France vs Austria.

Architecture	Final Elo	Params
GraphConv [11]	898 ± 13	46.8M
TransformerEnc 5x192	1041 ± 7	9.3M
TransformerEnc 10x224	1061 ± 12	12.4M
TransformerEnc 10x224 (Separate)	1123 ± 13	24.8M

Table 2: Effect in FvA of the architecture on final strength after RL, and number of parameters. GraphConv is the architecture achieving SOTA in supervised learning from [11], and similar to other past work [24, 3]. TransformerEnc b x f replaces the graph-conv encoder with a transformer encoder with b blocks and f feature channels. "Separate" uses separate value and policy networks instead of a shared network with two heads. Elos are based on 100 games against each of several internal baseline agents, including the agents in Table 4.

GraphConv vs Transformers

DO Method During RL Training	Final Elo
No DO	927 ± 13
DO, uniform random sampling	956 ± 13
DO, local modification sampling	1023 ± 13

Table 3: Effect of DO methods during training on final FvA agent strength. Local modification sampling improves results greatly, whereas uniform random sampling is relatively ineffective. Elos $\pm$ one standard error were computed based on 100 games against each of several baseline agents, including the agents in Table 4.

Double Oracle at Training Time See Table 2. Although not a primary focus of this paper, we found that at least for use in RL, changing our model architecture to use a transformer encoder rather than a graph-convolution encoder in Section 4.2 resulted in significant gains over prior neural network architectures in Diplomacy [24, 3, 11], while using fewer parameters and still being similarly fast or faster in performance. Enlarging the model and splitting the network led to further gains.

See Table 3. We demonstrate the effectiveness of our DO method in finding improved actions during training. DO's ability to discover new strong actions for the policy proposal network to learn makes a big difference in the final agent. Local modification sampling is effective, whereas random sampling is relatively ineffective.

Double Oracle at Inference Time See Table 4. We also demonstrate the effectiveness of our DO method in finding improved candidate actions at inference time. Uniform random sampling among legal actions to generate the action pool for finding best responses only improves the weakest of our baseline agent at inference time, due to being ineffective at finding better actions within the huge action space, whereas local modification sampling improves all baselines at inference time.

5.1.1 Exploitability

We implemented and tested both algorithms from Section 4.3 to estimate a lower bound on the exploitability of our agents. We also test a variant where we adapt deep Nash value iteration for exploitation just as in the second method, but disable policy learning - the value network learns to predict states of higher exploitability, but the policy proposal network is fixed and does not learn.

The results against our best agent are in Table 5. We find that despite being of superhuman strength, the agent is still relatively exploitable. In a 2p0s large-action-space setting, this is not necessarily

[4]Between agents of similar strength we often observe France winning only about a third of games vs Austria.

Model used by both players → DO Method used by measured player ↓	Model1	Model2	Model3
None	$0.5_{\pm 0}$	$0.5_{\pm 0}$	$0.5_{\pm 0}$
DO, uniform random sampling	$0.75_{\pm 0.02}$	$0.48_{\pm 0.02}$	$0.47_{\pm 0.02}$
DO, local modification sampling	$0.89_{\pm 0.02}$	$0.59_{\pm 0.02}$	$0.56_{\pm 0.02}$

Table 4: Effect of using DO at inference time in FvA. Each entry is the average score when both players play the column agent, but one player uses a DO variant at inference time. Model1-Model3 are models of increasing quality developed over the course of our research, with the major differences being that Model1 uses a trained value network with a fixed proposal network, Model2 uses GraphConv encoder, and Model3 uses a small Transformer encoder. Stronger agents require better sampling methods to find candidate actions that help.

	vs DORA	**vs DORA w/ inference-time DO**
Actor-Critic PG [11]	$77.0\%_{\pm 0.6\%}$	-
Deep Nash-Derived Exploiter	$90.5\%_{\pm 0.4\%}$	$85.8\%_{\pm 0.5\%}$
Deep Nash-Derived Exploiter, values only	$65.7\%_{\pm 0.6\%}$	$60.5\%_{\pm 0.7\%}$

Table 5: Win rates for exploiter bots trained with different techniques against our best FvA agent with and without inference-time DO. The $\pm$ shows one standard error. Actor-Critic PG vs DORA with inference-time DO was not run due to computational cost.

surprising. For example in Go, another game with a moderately large action space, there exist gaps of more than 1000 Elo (nearly 100% winrate) between different agents, all of which are superhuman, even with no explicit training for exploitation [27]. Moreover, in imperfect-information games such as Diplomacy, exploiting a fixed agent is much easier than training an agent with low exploitability, since the exploiting agent does not need to precisely balance the probabilities of its actions.

Nonetheless, we also observe that our adaptation of the deep Nash value iteration method greatly outperforms the simpler actor-critic method from [11], and that both value learning and DO-based policy learning are important for its performance, with value learning improving over the baseline from 50% to 60.5%, and policy learning improving further to 85.8% winrate over *DORA*.

We also observe that applying DO at inference time reduces the exploitability, from 90.5% using our strongest exploiter agent to 85.8%. Even at inference time, DO continues to find and suggest new effective actions that would otherwise be overlooked by the policy proposal network.

5.2 Results in 7-player No-Press Diplomacy

We also applied our methods to 7-player no-press Diplomacy, training a *DORA* agent entirely from scratch with no human data in this more-challenging domain. For comparison, we also trained an agent via deep Nash value iteration beginning from policy and value networks first trained on human games. Our best results were achieved when we forced the agent to use the human-initialized policy for action proposal during training games, never updating it, and also stopping the training early - see additional ablations in Appendix H). Since we do not update the action proposal policy used in training games, we also do not apply DO for this agent, and so we refer to this agent as *HumanDNVI-NPU* ("*d*eep *N*ash *v*alue *i*teration, *no p*roposal *u*pdate"). The results are reported in Table 6. Note that as these are 7-player games, the expected score if the agents perform equally well is $1/7 \sim 14.3\%$.

Unlike in 2p0s settings, in a competitive multi-agent game, particularly one such as Diplomacy where agents must temporarily ally and coordinate to make progress, there may be multiple different equilibria that are mutually incompatible [20]. An agent playing optimally under one equilibrium may perform very poorly in a pool of agents playing a different equilibrium.

For example in Diplomacy, it may be mutually benefit two agents to ally to attack a third, but in such cases players must often coordinate on a choice, for example on which unit attacks and which unit supports, or on which region to attack. Since an order and a support must exactly match to be effective, each choice may be a stable equilibrium on its own. But an agent that plays one choice in a population of agents that always play another in that situation will fail to match and will do poorly.

1x ↓ vs 6x →	DipNet [24]	SearchBot [11]	*DORA*	*HumanDNVI-NPU*
DipNet [24]	-	$0.8\%_{\pm 0.4\%}$	$0.0\%_{\pm 0.0\%}$	$0.1\%_{\pm 0.0\%}$
SearchBot [11]	$49.4\%_{\pm 2.6\%}$	-	$1.1\%_{\pm 0.4\%}$	$0.5\%_{\pm 0.2\%}$
DORA	$22.8\%_{\pm 2.2\%}$	$11.0\%_{\pm 1.5\%}$	-	$2.2\%_{\pm 0.4\%}$
HumanDNVI-NPU	$45.6\%_{\pm 2.6\%}$	$36.3\%_{\pm 2.4\%}$	$3.2\%_{\pm 0.7\%}$	-
DipNet-Transf	$23.4\%_{\pm 2.2\%}$	$2.1\%_{\pm 0.7\%}$	$0.0\%_{\pm 0.0\%}$	$0.3\%_{\pm 0.1\%}$
SearchBot-Transf	$48.1\%_{\pm 2.6\%}$	$13.9\%_{\pm 1.7\%}$	$0.5\%_{\pm 0.3\%}$	$0.9\%_{\pm 0.2\%}$

Table 6: SoS scores of various agents playing against 6 copies of another agent. The ± shows one standard error. Note that equal performance is 1/7 ≈ 14.3%. The non-human DORA performs very well as a 6x opponent, while performing much worse on the 1x side with 6 human-like opponents. The last 2 rows are equivalent to first 2 rows, but are retrained with the same architecture as self-play agents, i.e., TransformerEnc 5x192, showing the effect of architecture on these baselines separately from our deep Nash RL methods.

1x ↓ vs 6x →	*DORA*	*DORA* (alternate run)
DORA	-	$7.5\%_{\pm 1.0\%}$
DORA (alternate run)	$3.2\%_{\pm 0.7\%}$	-

Table 7: SoS scores between two runs of DORA, 1 agent versus 6 copies of the other. The ± shows one standard error. Note that equal performance would be 1/7 ≈ 14.3%.

Our results are consistent with *DORA* converging roughly to an equilibrium that is very different than the one played by humans, but playing well within that equilibrium. *DORA* achieves only about an 11% winrate in a 1-vs-6 setting against one of the strongest prior human-data-based agents, SearchBot [11] (tying would be a score of 1/7 ≈ 14.3%). However, SearchBot fares even worse with only a 1.1% winrate in a 1-vs-6 setting against *DORA*.

Against a second DORA trained with a different random seed (Table 7), each DORA scores only about 3-8% against the other 1-vs-6. This suggests that different DORA runs may converge to different equilibria from each other.

Our results with *HumanDNVI-NPU* are also consistent with reinforcement learning converging to inconsistent equilibria. *HumanDNVI-NPU* soundly defeats SearchBot, the prior SOTA, winning 36.3% of games 1-vs-6, while SearchBot wins only 0.5% in the reverse matchup. Starting from a human-like policy, deep Nash value iteration massively improves the strength of the agent. Yet *HumanDNVI-NPU* still does extremely poorly vs *DORA*, and vice versa. Our results seem to suggest that in 7-player no-press Diplomacy, deep Nash value iteration converges to an equilibrium in which outside agents cannot compete, even agents trained independently by the same method, but that which equilibrium is found depends on initialization, with some equilibria performing far better versus human-like players than others. See also Appendix H for additional ablations and experiments.

6 Conclusions and Future Work

Diplomacy has been a benchmark for AI research for two main reasons: its massive combinatorial action space and its multi-agent nature. We show that *DORA* achieves superhuman performance in 2-player Diplomacy without any human data or reward shaping, despite the combinatorial action space and imperfect information. *DORA* therefore effectively addresses the first challenge of Diplomacy.

Additionally, we show that *DORA* in full-scale 7-player no-press Diplomacy approximates an equilibrium that differs radically from human-data-based agents. When one human-data-based agent plays against a population of *DORA* agents, the one agent loses by a wide margin. Similarly, one *DORA* agent against a population of human-data-based agents does not exceed an average level of performance. These 7-player results provide evidence that self-play from scratch may be insufficient to achieve superhuman performance in Diplomacy, unlike other multiplayer games such as 6-player poker [9]. Furthermore, the results suggest that there may be a wide space of equilibria, with some differing dramatically from human conventions. *DORA* provides the first competitive technique for exploring this wider space of equilibria without being constrained to human priors.

References

[1] Lukáš Adam, Rostislav Horčík, Tomáš Kasl, and Tomáš Kroupa. Double oracle algorithm for computing equilibria in continuous games. 09 2020.

[2] Thomas Anthony. Ghost-ratings, 2020.

[3] Thomas Anthony, Tom Eccles, Andrea Tacchetti, János Kramár, Ian Gemp, Thomas C Hudson, Nicolas Porcel, Marc Lanctot, Julien Pérolat, Richard Everett, et al. Learning to play no-press diplomacy with best response policy iteration. *arXiv preprint arXiv:2006.04635*, 2020.

[4] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, 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.

[5] David Blackwell et al. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, 1956.

[6] Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. *arXiv preprint arXiv:2007.13544*, 2020.

[7] Noam Brown and Tuomas Sandholm. Simultaneous abstraction and equilibrium finding in games. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

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

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

[10] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

[11] Jonathan Gray, Adam Lerer, Anton Bakhtin, and Noam Brown. Human-level performance in no-press diplomacy via equilibrium search. *arXiv preprint arXiv:2010.02923*, 2020.

[12] James Hannan. Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

[13] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[14] Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.

[15] Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 327–334, 2011.

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

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

[18] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *nternational Conference on Neural Information Processing Systems*, pages 4193—4206, 2017.

[19] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[20] R Duncan Luce and Howard Raiffa. *Games and decisions: Introduction and critical survey*. Courier Corporation, 1989.

[21] Stephen Mcaleer, JB Lanier, Roy Fox, and Pierre Baldi. Pipeline psro: A scalable approach for finding approximate nash equilibria in large games. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20238–20248. Curran Associates, Inc., 2020.

[22] Brendan McMahan, Geoffrey Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *International conference on machine learning*, pages 536—-543, 2003.

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

[24] Philip Paquette, Yuchen Lu, Seton Steven Bocco, Max Smith, O-G Satya, Jonathan K Kummerfeld, Joelle Pineau, Satinder Singh, and Aaron C Courville. No-press diplomacy: Modeling multi-agent gameplay. In *Advances in Neural Information Processing Systems*, pages 4474–4485, 2019.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[26] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

[27] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[28] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

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

[30] Vasilis Syrgkanis, Alekh Agarwal, Haipeng Luo, and Robert E Schapire. Fast convergence of regularized learning in games. In *Advances in Neural Information Processing Systems*, pages 2989–2997, 2015.

[31] Tom Van de Wiele, David Warde-Farley, Andriy Mnih, and Volodymyr Mnih. Q-learning in enormous action spaces via amortized approximate maximization. *arXiv preprint arXiv:2001.08116*, 2020.

[32] 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.

7 Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   [Yes]
   (b) Did you describe the limitations of your work?
   [Yes] Particularly with regards to non-2p0s settings, we discuss limitations interlaced through the discussion of theoretical grounding and through discussion of our results in 7-player and the conclusion.
   (c) Did you discuss any potential negative societal impacts of your work?
   [N/A] We do not expect near-term societal impacts from this work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them?
   [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results?
   [Yes]
   (b) Did you include complete proofs of all theoretical results?
   [Yes] In Appendix B that also refers to prior work that proves the main result we rely on.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)?
   [Yes] We release the code and the models [5].
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)?
   [Yes] Detailed parameters are provided in Appendix D.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)?
   [Yes] Error bars are given on relevant measurements and statistics.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)?
   [Yes] We report the requirements in Appendix D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators?
   [Yes]
   (b) Did you mention the license of the assets?
   [Yes] We specify the license of the model and code we use in appendix D. We plan to release the code and the models under MIT license.
   (c) Did you include any new assets either in the supplemental material or as a URL?
   [Yes] We release the code and the models.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating?
   [Yes] We discuss data used in appendix F.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content?
   [Yes] We do not use any data with personally identifiable information or offensive content (see appendix F).
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable?
   [Yes] We included the instructions given to participants in appendix F.

[5] `https://github.com/facebookresearch/diplomacy_searchbot`

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable?
[N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation?
[N/A]

A Neural Network Architecture

In this appendix, we describe the neural network architecture used for our agents.

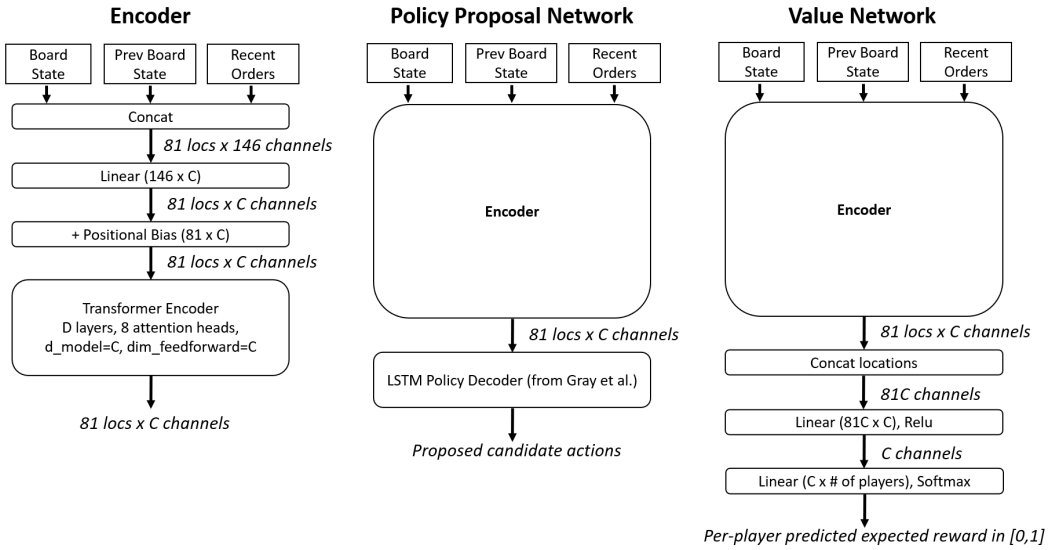

Figure 2: Transformer encoder (left) used in both policy proposal network (center) and value network (right).

Our model architecture is shown in Figure 2. It is essentially identical to the architecture in [11], except that it replaces the specialized graph-convolution-based encoder with a much simpler transformer encoder, removes all dropout layers, and uses separate policy and value networks. Aside from the encoder, the other aspects of the architecture are the same, notably the LSTM policy decoder, which decodes orders through sequential attention over each successive location in the encoder output to produce an action.

The input to our new encoder is also identical to that of [11], consisting of the same representation of the current board state, previous board state, and a recent order embedding. Rather than processing various parts of this input in two parallel trunks before combining them into a shared encoder trunk, we take the simpler approach of concatenating all features together at the start, resulting in 146 feature channels across each of 81 board locations (75 region + 6 coasts). We pass this through a linear layer, add pointwise a learnable per-position per-channel bias, and then pass this to a standard transformer encoder architecture.[6]

In our final agents, we use $(D, C) = (10, 224)$ although in Section 5.1 we also present results for $(D, C) = (5, 192)$ as well as for using a single combined network (i.e. shared encoder) for both policy and value.

B Theoretical Correctness of Nash Value Iteration

In any 2p0s stochastic game with deterministic[7] transition function f, we show the Deep Nash Value Iteration (DNVI) procedure described in Section 3.1 applied over a candidate set of *all* legal actions converges to a value function $\hat{V}(s)$ that is the minimax value of each reachable state s, in the tabular setting. In other words, the use of a value function rather than a Q function does not affect the correctness.

To demonstrate this, we first show an equivalence between our procedure and Nash Q-learning in 2p0s games. At each step of DNVI, we update the value of $\boldsymbol{V}(s)$ using the value computed by RM;

[6]For example, `https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoder.html`

[7]We believe this result holds for games with non-deterministic transitions as well, but that does not allow for a direct reduction from Nash Q learning, so we do not provide a proof here.

since this stage game is 2p0s, RM converges to its unique minimax equilibrium value[13]. For each $(s, \boldsymbol{a})$,

$$\boldsymbol{Q}(s, \boldsymbol{a}) = \boldsymbol{r}(f(s, \boldsymbol{a})) + \boldsymbol{V}(f(s, \boldsymbol{a})),$$

therefore each time $\boldsymbol{V}$ is updated at state s it is equivalent to applying a Nash-Q update at all $(s', \boldsymbol{a})$ for which $f(s', \boldsymbol{a}) = s$.

[14] proves that Nash Q-learning converges to the Q values of a Nash equilibrium for the full game in the tabular setting under the following assumptions:

1. Every one-step game has an adversarial equilibrium and this equilibrium is used in the update rule.
2. All states are visited an infinite number of times.
3. The learning rate α decays such that $\alpha \to 0$ but $\sum \alpha = \infty$.

If these assumptions are satisfied, then $\boldsymbol{V}(s)$ converges to the values of a Nash equilibrium.

Regarding assumption 1, in 2p0s all NEs are automatically adversarial since there is only one opponent (*adversarial* equilibria are NEs that are additionally robust to joint deviations by opponents [14]).

Assumption 2 is satisfied as long as $\forall a, P(a|s) > 0$, which DNVI achieves via ϵ-Nash exploration over the candidate set of all legal actions, which plays a random action with probability $\epsilon > 0$. Note that the modification we use in practice explores only actions that are likely under the current policy proposal net.

Assumption 3 concerns the learning rate schedule, which can be set arbitrarily. In the Deep RL setting we find in practice a fixed learning rate with the Adam [16] optimizer works well.

Thus, DNVI converges to a NE in a 2p0s stochastic game. In a 2p0s game all equilibria have identical (minimax) values. Therefore playing a minimax policy in each stage game based on these values is optimal.

We remind the reader that in Diplomacy the action space is too large to perform exact RM over the entire action space, so in practice we perform DNVI on a small subset of candidate actions proposed by the policy network and rely on Double Oracle to discover actions that should be added to this set.

C Variance Reduction in Diplomacy

We apply a simple form of variance reduction for our experiments playing with humans to help compensate for the relatively low sample size.

As in many simultaneous-action games, not uncommonly in Diplomacy there are situations that behave like a "matching pennies" subgame, where if player A plays action a_1, player B prefers to match it with an action b_1, and if A plays action a_2, player B prefers to match it with action b_2, whereas player A's preferences are the other way around, preferring the outcomes (a_1, b_2) and (a_2, b_1). Rock-paper-scissors-like or other kinds of simple subgames may also occur. In any of these cases, both players should randomize their strategies to reduce exploitability. See Table 8 for an example of a matching-pennies-like subgame.

	b_1	b_2
a_1	-1, 1	2,-2
a_2	1,-1	-3, 3

Table 8: Example payoff matrix for a matching-pennies-like game with nonuniform payoffs. In each cell, the payoff to the row player is listed first.

In such a subgame, once a player commits to playing a particular mixed strategy, then upon seeing the opponent's action, even without knowing the opponent's probability of playing that action, one can identify a significant component of variance due to the luck of randomization in one's *own* mixed strategy that can be subtracted out. For example, in the example in Table 8 above if player A commits to a policy 60% a_1 and 40% a_2, then supposing player B is revealed to have chosen b_2,

player A knows at that point that their expected value is $60\% \cdot 2$ (the payoff for a_1) $+40\% \cdot -3$ (the payoff for a_2). The deviation from the expected value due to whether player A *actually* samples a_1 or a_2 on that round is now a matter of pure luck, i.e. just zero-mean variance that can be subtracted out.

In Diplomacy, we can apply this idea as well. When playing a game as agent i, on each turn we record the full distribution of our approximated NE σ_i from which we sample the action a_i that we submit that turn. Upon observing all other agents' actions a_{-i}, and letting Q be the learned approximate Q-value function of our agent, we compute:

$$\delta(a_i) = Q(s, (a_i, a_{-i})) - E_{a'_i \sim \pi_i} Q(s, (a'_i, a_{-i}))$$

$\delta(a_i)$ measures how lucky we are to have chosen the specific a_i that we did on that turn relative to the expected value over all actions a'_i that we could have sampled from policy σ_i, given the observed actions of the other players. On each turn, we subtract this quantity from the final game win/loss reward of 1 or 0 that we ultimately observe. Since $E_{a_i} \delta(a_i) = 0$, doing this introduces no bias in expectation. So letting δ_t be the $\delta(a_i)$ value computed for turn t, and R the total reward of the game, our final variance-adjusted result is

$$R - \sum_t \delta_t$$

In the event that Q is a good estimate of the true value of the state given the players' policies, then the δ_t values should be correlated with the final game outcome, so subtracting them should reduce the variance of the outcome, particularly in a 2-player setting such as FvA Diplomacy. In practice, we observe roughly a factor of 2 reduction in variance in informal test matches. As part of our internal practice, we also implemented and tested and committed to the use of this form of variance reduction prior to assembling the final human results.

D Implementation details

We report optimizations for Double Oracle we use in practice, details of the pretraining procedure, modifications for 7p, and the hyper parameters used.

D.1 Double Oracle optimizations

We apply several modifications and approximations to DO to make it run faster in practice:

- We find a best response (line 8 of the algorithm 1) for only one player at a time. When N_p is large, evaluating N_p potential best responses is more expensive than computing a NE among the N_c candidate actions ($N_p \times N_c$ value function calls versus $N_c \times N_c$).[8] Recomputing the NE cheaply ensures each best response takes into account the newest action added by the opponent. Note, that for 7p caching is not as efficient due to large joint action space (N_c^7 instead of N_c^2), but computing BR still dominates the computations cost.
- When finding a best response, we truncate the opponent's equilibrium policy, which spans up to N_p actions, to only its k highest-probability actions for some small k and renormalize. This allows us to compute a best response with only $N_c * k$ value function calls rather than $N_c * N_p$ calls, while only introducing minimal error.
- We cap the number of iterations of DO we execute.

D.2 Pretraining

We perform a short phase of pretraining before switching to deep Nash value iteration that improves the speed and stability of training. This phase differs from the main training phase in the following ways:

[8] When querying the value network is the dominating cost, the number of iterations needed to compute a NE is largely irrelevant because the value network results can be cached. For this reason, we consider the cost to be roughly $N_c \times N_c$ rather than $T \times N_c \times N_c$.

- Rather than query the policy proposal network and/or apply DO to obtain actions, we select N_c candidate actions for each player uniformly at random among all legal actions.
- Rather than training the value network to predict the 1-step value based on the computed equilibrium over the N_c actions, we train it to directly predict the final game outcome.

By directly training on the final game outcome, this pre-training phase initializes the value network to a good starting point for the main training more quickly than bootstrapped value iteration. This phase also initializes the policy proposal network, which is trained to predict the output of regret matching on the random sampled actions, to a good high-entropy starting point - e.g. to predict a wide range of actions, mildly biased towards actions with high value. If instead a randomly initialized (untrained) policy proposal network were used to select actions, it may select a highly non-uniform initial distribution of candidate actions and require a lot of training for excluded actions to be rediscovered via DO.

D.3 Extending to 7p

We found a small number of additional implementation details were needed to handle 7-player no-press Diplomacy that were not needed for 2-player FvA.

Firstly, exactly computing the value loss in equation 6 is not feasible because the number of possible joint actions $\boldsymbol{a}'$ scales with the power of the number of players, which for 7 players is too large. So we instead approximate the 1-step value via sampling.

Similarly, during DO, when computing the expected value of potential best responses, given a certain number of actions per opponent, computing the value exactly scales with the power of the number of opponents, which for 6 opponents is too large. So again, we approximate by sampling.

Finally, in line with [11] we found that using Monte Carlo rollouts to compute state values improved the play compared to using a value function alone. Therefore, we use rollouts of depth 2 in all experiments at inference time. We do not use rollouts during training due to the computational cost.

D.4 Hyper-parameters

D.4.1 Double Oracle

We use slightly different parameters at training and inference time to speed up the data generation for training. See table 9.

	Training	Inference
Pool size (N_p)	1,000	10,000
Max opponent action (k)	8	20
Min value difference (ϵ)	0.04	0.01
Max iterations (N_{iters})	6	16
Pool recomputed after each iteration	No	Yes

Table 9: Hyper-parameter values used for Double Oracle for *DORA*.

D.4.2 DORA training

Details for training of FvA *DORA* bot are provided in table 10. We use a few additional heuristics to facilate training that are explained below.

The training is bottlenecked by the data generation pipeline, and so we use only a few GPUs for training, but an order of magnitude more for data generation. To make sure the training does not overfit when the generation speed is not enough, we throttle the training when the training to generation speeds ratio is above a threshold. This number could be interpreted as a number of epochs over a fixed buffer.

Our final FvA *DORA* model is trained using 192 Nvidia V100 GPUs on an internal cluster, 4 used for the training and the rest is used for data generation. Both stages take around a week to complete.

We made a number of changes for 7p training compared to FvA for computational efficiency reasons. Each 7p game is 4-5 times longer and each equilibrium computation requires 3 times more operations due to the increased number of players. Moreover, the cost of double oracle also increases many fold as the probability of finding a deviation for at least one player during a loop over players increases. Therefore, for 7p *DORA* we add an additional pre-training stage where we train as normal, but without DO. For speed reasons, we use a smaller transformer model for both the value and the policy proposal nets.

	FvA	7p
Learning rate	10^{-4}	
Gradient clipping	0.5	
Warmup updates	10k	
Batch size	1024	
Buffer size	1,280,000	
Max train/generation ratio	6	
Regret matching iterations	256	
Number of candidate actions (N_c)	50	
Max candidate actions per unit	-	6
Number of sampled actions (N_b)	250	
Nash explore (ε)	0.1	0.1
Nash explore, S1901M	0.8	0.3
Nash explore, F1901M	0.5	0.2

Table 10: Hyper-parameter values used to train *DORA* agents.

D.4.3 Evaluation details

We describe parameters used for agent evaluation in table 6.

To compare against DipNet [24] we use the original model checkpoint[9] and we sample from the policy with temperature 0.5. Similarly, to compare against SearchBot [11] agent we use the released checkpoint[10] and agent configuration[11]. To make the comparison more fair, we used the same search parameters for *DORA* and *HumanDNVI-NPU* as for SearchBot (see table 11).

Number candidate actions (N_c)	50
Max candidate actions per unit	3.5
Number CFR iterations	256
Policy sampling temperature for rollouts	0.75
Policy sampling top-p	0.95
Rollout length, move phases	2

Table 11: Parameters used for all 7p agents with search in table 6 in the main text.

D.4.4 Distributed training and data generation

We use PyTorch [25] for distributed data parallel training and a custom framework for distributed data generation. We run several Python processes in parallel across multiple machines:

- **Training processes.** All training processes run on one machine. Each is assigned a separate GPU and has a separate replay buffer from which it computes gradients, which are then broadcast and synchronized and across processes. One training process is also responsible for publishing new checkpoints for data generation workers to use and for collecting training statistics.

[9] DipNet SL from `https://github.com/diplomacy/research`. MIT License

[10] blueprint from `https://github.com/facebookresearch/diplomacy_searchbot/releases/tag/1.0`. MIT License

[11] `https://github.com/facebookresearch/diplomacy_searchbot/blob/master/conf/common/agents/searchbot_02_fastbot.prototxt`

- **Data collection and rollout processes.** Each machine other than the training machine has a data collection process and many rollout processes. The data collection process gathers rollouts from rollout processes on the same machine via shared memory and sends them to the replay buffers via RPC. Rollout processes run the data generation loop. Each iteration of the loop consists of reading a new model checkpoint if available, stepping the game till the end, and sending results to the collection process. We run up to 8 rollout processes per GPU. Each process has its own copy of both the model and the environment.
- **Evaluation processes** run one-vs-six games of the current model checkpoint versus some fixed agent and collect running winrates for monitoring the run.
- **The metric collection process** receives metrics from other processes via RPC, aggregates and saves them.

Our design was optimized to work on 8-GPU machines with 10 CPUs per GPU, such as Nvidia DGX-1 machines. Our main DORA run used 192 GPUs total, across 24 machines. In such a setting our setup achieves over 90% average utilization for both GPU and CPU.

E Adapting Deep Nash Value Iteration to Learn Best Responses

As mentioned in Section 4.3, our best exploiter agents resulted from adapting our main deep Nash value iteration to learn a best response instead of an equilibrium.

To do so, we begin with the policy and value models of the exploited agent and resume deep Nash value iteration, except with a simple modification. On each turn of the game we first invoke the exploited agent to precompute its policy. Then the exploiting agent, when performing Nash learning, only performs RM for itself while the opponent samples only the precomputed policy. All other aspects of the architecture are identical.

Whereas normal RM approximates an NE, one-sided RM instead approximates a best response. With this one change, deep Nash value iteration, rather than training the networks to learn equilibrium policy and values, trains them to learn best-response exploitation and expected state-values assuming future best-response exploitation.

This second approach resembles the Sampled Best Response (SBR) exploiter from [3], in that both methods use a policy model to sample candidate actions for the exploiter, and then approximate a 1-ply best response by directly maximizing against the exploited-agent's known policy with respect to a value model. However, unlike SBR, which uses policy and value models obtained via other methods such as supervised learning that may not be optimized for exploitation, our approach directly trains the policy to sample better exploitative candidate actions, and trains the value to also prefer states where exploitability on future turns will be high.

At test time, our exploiter agent takes the average of 3 samples of the exploited agent's reported average policy instead of only one to get a better estimate, since due to sampling noise or multiple equilibria, regret matching will sometimes return different final average policies. The exploited agent in test itself acts according to a 4th entirely independent sample, to ensure the exploiter can only optimize against the average and not the exact seed.

F Human data

To conduct human experiments we informally reached out to top FvA Diplomacy players with a proposal to help with our research by playing against our agent.

The message contained the following instructions:

```
Hi [Person],

I'm AUTHORNAME, one of the developers of BOTNAME. We've put a new version
of the bot online recently that we think might be superhuman, and we're
hoping to do some testing against top humans to measure whether that's the
case.
```

```
Would you be interested in playing  10 games against the bot?  You'd be
able to play at your own pace.  You're also welcome to play against the bot
as many times as you'd like as practice before starting the "real" matches.
```

```
For the real matches, you could call them something like "[Person] vs
BOTNAME 1", or something along those lines.  You can launch them just like
a normal game.  We just ask that you not cancel any games and only put in a
draw if it's clearly a stalemate.
```

```
These results (aggregated with other top players) would go into an academic
paper.  We could either use your real name or your username, or just report
your FvA GhostRating, whichever you prefer.
```

Among the players that we contacted, 5 played matches against our bot. Those players were ranked 1st, 8th, 16th, 22nd, and 29th in FvA among all players on `webdiplomacy.net` according to the Ghost-Ratings [2] ranking system. None of the individual players had a positive win rate against the bot, though no single player played enough games against the bot to measure individual performance with statistical significance.

We did not collect any information regarding the games besides player's actions in the games played. Thus, the data does not contain any personally identifiable information or offensive content.

G Action Exploration Example

We go into more detail regarding the example provided in Figure 1 that motivated our double oracle approach. This is an example of a situation from a real game played by our agent (France, blue) vs a human (Austria, red). The agent has an army in Tyr next to the SCs of the opponent. None of the actions that the agent considers for red could dislodge the army *and* block its retreat further east. Therefore, the estimated state value for blue in this position is high and one of the possible expected outcome is shown in Figure 3.

However, human players were quick to realize that there are two actions that can dislodge the blue army and force it to retreat to Pie (Figure 4, left). Moreover, as the agent blocked its own retreat by moving Mar to Pie, the agent had to disband its army in Tyr (Figure 4, right). Once the optimal action for red is added to the set of candidate actions, the probability of blue moving Mar to Pie and the expected score for blue in this state go down.

There are more than 4 million valid actions[12] or around 400,000 valid coordinated actions. While it is still computationally feasible to run exact DO at inference time for this number of actions, adding even one more unit makes DO search infeasible. Moreover, running DO at inference time only is not sufficient, as during optimal play an agent should know that this state has low value and so the actions leading to this state should be avoided. Since the number of actions one can realistically evaluate during training is around 1,000, we have less than a 1% chance of finding a BR action at training time. Our approximate local modification approach allows us to dramatically reduce the search size and find good approximate BRs with relatively few actions.

Our final *DORA* agent finds red's action from Figure 4 without using inference time DO, as the action is learned by the policy proposal network.

[12] Product of possible orders per location: MUN (17), SEV (6), VEN (8), VIE (18), TRI (18), BUD (16). Unit in SEV is not shown on the map.

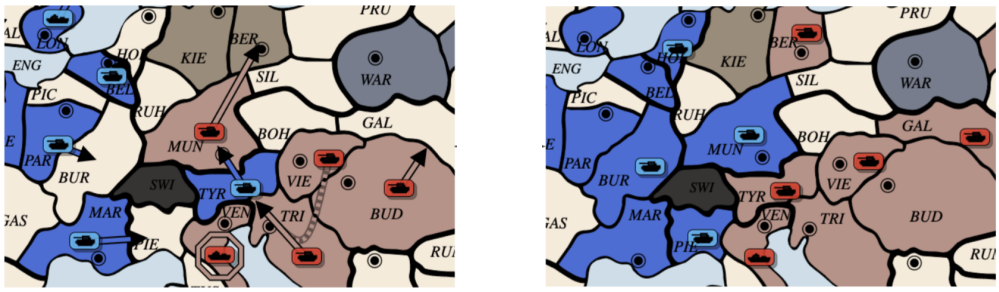

Figure 3: The most probable actions for both players as predicted by the agent. The left figure shows the actions and the right figure shows the state after the actions are executed. France's army escapes to MUN.

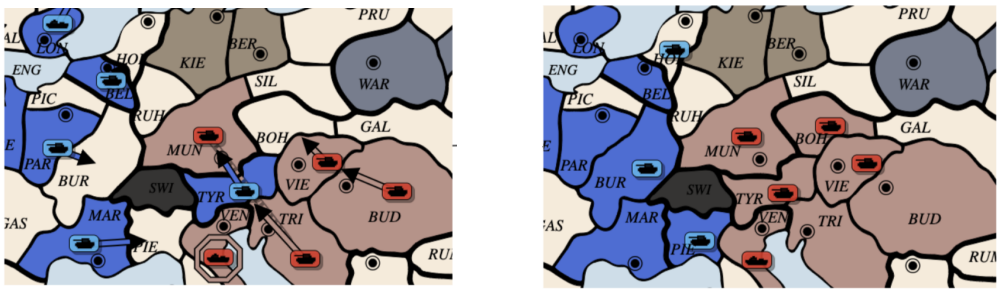

Figure 4: The most probable action for France (blue) as predicted by the agent vs the best action for Austria (red) that the policy proposal network failed to find on its own. The left figure shows the actions and the right figure shows the state after the actions are executed. France's army is crushed and disbanded.

H Additional Results for 7-player No-Press Diplomacy

In this section we include more preliminary results exploring how initialization and training affect convergence to different equilibria for *HumanDNVI-NPU*.

Namely, we show in Table 12 how performance versus human-like agents changes as self-play training progresses as well as the effect of freezing the action proposal network ("no policy update", "NPU"). As a reminder, *HumanDNVI-NPU* uses a training procedure identical to *DORA*, except a fixed blueprint from supervised learning on human games is used to propose actions throughout the training and hence no DO is used. The policy proposal net is still trained, but is only used at test time. *HumanDNVI* is one step closer to *DORA*: it does update the policy proposal network during training on rollout workers, but does not use DO. In general, both early stopping and NPU help the agent converge towards a strategy more effective against human-like models.

Additionally, we evaluated *HumanDNVI-NPU*-BP-Policy, an agent that uses the value function from *HumanDNVI-NPU*, but the human blueprint policy proposal net during both training and test time. At all times this model performs worse than *HumanDNVI-NPU*. This suggests that a trained policy does increase the strength of the agent, even though using that policy within the RL loop makes the training too-easily diverge away from human-compatible strategies or equilibria. More research in the future may find better techniques to get the best of both worlds.

Finally, Table 13 provides additional data on the impact of DO for 7-player Diplomacy. We trained an agent from scratch for 300k updates without DO and then trained for 60k updates more with DO. This finetuning doubles the score versus SearchBot and also greatly increases it versus Dipnet.

1x Agent	# training updates	vs 6x DipNet [24]	vs 6x SearchBot [11]
DipNet [24]	-	-	$0.8\%_{\pm 0.4\%}$
Transf	-	$23.4\%_{\pm 2.2\%}$	$2.1\%_{\pm 0.7\%}$
SearchBot [11]	-	$49.4\%_{\pm 2.6\%}$	-
Transf+Search	-	$48.1\%_{\pm 2.6\%}$	$13.9\%_{\pm 1.7\%}$
DORA	600k	$22.8\%_{\pm 2.2\%}$	$11.0\%_{\pm 1.5\%}$
HumanDNVI	100k	$30.6\%_{\pm 2.4\%}$	$20.5\%_{\pm 2.0\%}$
HumanDNVI	300k	$25.3\%_{\pm 2.3\%}$	$18.6\%_{\pm 2.0\%}$
HumanDNVI-NPU	50k	$45.6\%_{\pm 2.6\%}$	$36.3\%_{\pm 2.4\%}$
HumanDNVI-NPU	100k	$41.4\%_{\pm 2.5\%}$	$34.3\%_{\pm 2.3\%}$
HumanDNVI-NPU	300k	$35.9\%_{\pm 2.6\%}$	$28.4\%_{\pm 2.3\%}$
HumanDNVI-NPU-BP-Policy	50k	-	$25.7\%_{\pm 2.1\%}$
HumanDNVI-NPU-BP-Policy	100k	-	$25.0\%_{\pm 2.1\%}$
HumanDNVI-NPU-BP-Policy	300k	-	$22.9\%_{\pm 2.0\%}$

Table 12: SoS scores of various agents playing against 6 copies of another agent. The $\pm$ shows one standard error. Note that equal performance is 1/7 $\approx$ 14.3%. All our agents are based on TransformerEnc 5x192.

1x Agent	# training updates	vs 6x DipNet [24]	vs 6x SearchBot [11]
ScratchDNVI	300k	$13.6\%_{\pm 1.8\%}$	$3.5\%_{\pm 0.9\%}$
+finetune with DO	+60k	$21.9\%_{\pm 2.2\%}$	$8.0\%_{\pm 1.4\%}$

Table 13: Effect of using DO for training from scratch. Even training for a small fraction of time with DO increases the strength of an agent by a significant margin.