DIFFNASH: FINDING NASH EQUILIBRIUM IN HIGH-DIMENSIONAL CONTINUOUS ACTION SPACES WITH DIFFUSION POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Computing equilibria in high-dimensional continuous games requires learning and representing rich, multi-modal strategy distributions, something conventional neural policies struggle to achieve. Diffusion models offer a natural solution, but standard diffusion training requires direct samples from a fixed target distribution, which are generally unavailable under iterative game-solving updates. We propose a diffusion-based framework for equilibrium computation that enables diffusion-based variants of Online Mirror Descent (OMD) and Fictitious Play (FP). Our approach derives a reweighted score-matching objective that trains diffusion policies without requiring direct samples from each updated policy. Our main method, **DiffNash**, combines the fast convergence of OMD with the sample efficiency of FP by reusing past actions through a replay buffer. To correct the resulting distribution shift from sample reuse, we derive an unbiased estimator of the difference between current and past policies' Evidence Lower Bounds, whose exponential provides a tractable proxy for the corresponding importance weight without evaluating diffusion likelihoods. We evaluate our approach on forest protection games and continuous Colonel Blotto games, reducing the Nash gap over the strongest baseline by **35.6%** and **24.7%**, respectively, with minimal hyperparameter tuning.

1 INTRODUCTION

In real-world decision making, many choices are strategic: a party's outcome depends not only on its own decisions, but also on those of adversaries with competing objectives. Game theory provides a framework for modeling these interactions. In many applications, however, strategic decisions are not naturally enumerable and instead lie in continuous action spaces (Stein et al., 2008; Ganzfried, 2021; Kroer & Sandholm, 2015). In spatial security games, for example, a defender allocates resources while an adversary chooses where to attack (Kamra et al., 2018; Gan et al., 2017; Alpern et al., 2022); these decisions are naturally represented by locations or trajectories rather than discrete actions. Continuous decisions also arise in economic competition, where firms choose production quantities (Shi & Zhang, 2020), and in resource-allocation games, where players divide continuous budgets across competing objectives (Roberson, 2006). Such settings are difficult to capture with finite strategy spaces and require reasoning over distributions on large continuous action spaces.

A Nash equilibrium is a strategy profile in which neither player can improve its outcome by unilaterally changing its strategy. A mixed strategy is a probability distribution over a player's possible actions; in continuous games, this distribution may be defined over an infinite action space (Glicksberg, 1952). As the dimensionality and complexity of the action space grow, representing the mixed strategy itself becomes challenging, since equilibrium play may require complex, multimodal distributions over high-dimensional continuous actions (Perkins et al., 2015).

Continuous-action games, even in the more well-understood two player zero-sum (2p0s) setting, present unique computation challenges. While efficient algorithms (Dantzig, 1951) and practical solvers (Jain et al., 2011; Cerny et al., 2024) exists for 2p0s games with finite action set, they cannot directly handle infinitely many actions in continuous-action games. Therefore, prior works (Kamra et al., 2019; Martin & Sandholm, 2023) parameterize player strategies with neural networks and optimizes them through iterative algorithms such as Fictitious Play (FP) or Online Mirror Descent (OMD) to approximate Nash equilibria. However, these neural policies can suffer from mode collapse

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

in high-dimensional action spaces, failing to capture the multimodal distributions that naturally arise in continuous-action games. Since diffusion model is capable of modeling multimodality in continuous vectors (Ho et al., 2020; Wang et al., 2023; Li et al., 2024), we ask

Can we parameterize player strategy with diffusion policies in iterative game-solving algorithms?

In this work we propose **DiffNash**, a method for introducing diffusion models into equilibrium computation for games with continuous action. We parameterize each player's mixed strategy with a diffusion model and use Online Mirror Descent (OMD) Blair (1985) to iteratively update these strategies towards equilibrium. At each iteration, we sample candidate actions from the player's current diffusion policy and evaluate how useful those actions would be against the opponent's current policy. Standard diffusion training assumes access to samples from the target distribution, while the OMD update defines this distribution only implicitly. Hence, we adapt the reweighting technique of Ma et al. (2025) to modify the diffusion training loss, provably recovering the correct target update. To reduce the cost of repeatedly sampling news actions from diffusion and stabilize training along OMD iterations, we reuse samples from past policies through a replay buffer. We correct the resulting distribution shift with importance weighting and approximate the importance weights using Evidence Lower Bounds (ELBOs).

Our main contributions are: (1) We develop a diffusion-policy framework for equilibrium computation in continuous action games using utility-weighted score matching to represent complex, potentially multimodal mixed strategies (Section 3.1) (2) We incorporate a diffusion update into OMD, with an empirical comparison showing substantially faster convergence than a corresponding diffusion-FP approach (Section 3.2) (3) We construct an ELBO-difference based importance weighting that allows past actions to be reused for opponent evaluation (Section 3.3) and stabilizing training (Section 3.4) (4) We evaluate our approach on two continuous-action games, reducing the Nash gap over the strongest baseline by **35.6%** and **24.7%**, respectively, with minimal hyperparameter tuning (Section 5).

2 Background

2.1 Problem Formulation

We consider a two-player normal-form game $\mathcal{G} = (\mathcal{A}^1, \mathcal{A}^2, u^1, u^2)$, where $\mathcal{A}^i \subseteq \mathbb{R}^{d^i}$ is a non-empty compact action space of player i and $u^i : \mathcal{A}^1 \times \mathcal{A}^2 \to \mathbb{R}$ be its bounded and measurable utility function [1]. We consider high-dimensional continuous action set, which may also subject to global feasibility constraints, such as a fixed resource budget or a bounded spatial domain. A mixed strategy $\pi^i \in \Delta(\mathcal{A}^i)$ is a probability distribution over $\mathcal{A}^i$. Given a strategy profile $\pi = (\pi^1, \pi^2)$, the expected payoff to player 1 is $U^i(\pi^1, \pi^2) = \mathbb{E}_{\substack{\mathbf{a}^1 \sim \pi^1 \\ \mathbf{a}^2 \sim \pi^2}} \left[u^i(\mathbf{a}^1, \mathbf{a}^2) \right]$. Our goal is to find the Nash equilibrium of the game, where neither player has the incentive to deviate from their chosen strategy. Formally,

Definition 1 (Nash Equilibrium). $\pi^\star = (\pi^{1,\star}, \pi^{2,\star})$ is a Nash equilibrium if, for $i \in \{1, 2\}$, $U^i(\pi^{i,\star}, \pi^{-i,\star}) \geq U^i(\pi^i, \pi^{-i,\star}) \quad$ for all $\pi^i \in \Delta(\mathcal{A}^i)$.

We focus primarily on two-player zero-sum (2p0s) games, where $u^1(\mathbf{a}^1, \mathbf{a}^2) = -u^2(\mathbf{a}^1, \mathbf{a}^2)$. 2p0s games with finite action can be solved by linear programming (Dantzig, 1951). For games with large combinatorial action spaces, such as network security games, practical methods exploit problem structure and propose practical solvers (Jain et al., 2011; Cerny et al., 2024). Continuous-action games require representing and optimizing distributions over an infinite action space. Recent approaches therefore use neural parameterizations of mixed strategies within iterative game-solving algorithms (Kamra et al., 2019; Martin & Sandholm, 2023). Our work follows this latter setting, using diffusion models as expressive representations of continuous mixed strategies.

We primarily study the 2p0s setting, where equilibrium objective and associated learning dynamics are well understood. Methods with strong convergence guarantees in 2p0s games are also commonly studied and applied in general-sum settings, although convergence to Nash equilibrium is no longer guaranteed (Anagnostides et al., 2022). We also include general-sum games, where player's objectives are not necessarily opposed and equilibrium computation is generally more challenging. We measure

[1] We use superscripts for game-theoretic notation, such as player and equilibrium-solver iteration indices, and subscripts for diffusion model notation.

solution quality using the *Nash gap*, defined as the total improvement available to players through unilateral best responses (Lanctot et al., 2017). It is non-negative and equals zero at Nash equilibrium, measuring how exploitable the strategy profile is.

Definition 2 (Nash Gap). $\text{Gap}(\pi_1, \pi_2) = \max_{\pi_1' \in \Delta(\mathcal{A}_1)} U(\pi_1', \pi_2) - \min_{\pi_2' \in \Delta(\mathcal{A}_2)} U(\pi_1, \pi_2')$.

A low Nash gap limits gains from opponent's unilateral deviations. We seek such strategies in high-dimensional action spaces, where conventional neural policies struggle (Kamra et al., 2019).

2.2 Diffusion Models

Denoising diffusion probabilistic models (DDPMs) (Sohl-Dickstein et al., 2015; Ho et al., 2020) is a generative model consisting of two stochastic processes: one forward process which progressively adds noise to the data, and a reverse (denoising) process which learns to gradually remove noise. Formally, given a clean data from the target distribution $\mathbf{z}_0 \sim p_0$, the forward process gradually perturbs the data by $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) = \mathcal{N}\big(\mathbf{z}_t \mid \sqrt{1-\beta_t}\mathbf{z}_{t-1}, \beta_t \mathbf{I}\big)$, where $\mathbf{z}_t$ is the perturbed data at time t and β_t is the noise variance at time step $t = 1, \cdots, T$. The corresponding joint distribution of forward trajectory is $q_{0:T}(\mathbf{z}_{0:T}) = p_0(\mathbf{z}_0) \prod_{t=1}^{T} q_{t|t-1}(\mathbf{z}_t|\mathbf{z}_{t-1})$. Since each step adds independent Gaussian noise, we can sample $\mathbf{z}_t$ by $\mathbf{z}_t \sim p_t(\mathbf{z}_t) = \int p_0(\mathbf{z}_0)\, q_{t|0}(\mathbf{z}_t \mid \mathbf{z}_0)\, \mathrm{d}\mathbf{z}_0$, where $q_{t|0}(\mathbf{z}_t \mid \mathbf{z}_0) := \mathcal{N}(\mathbf{z}_t; \sqrt{\bar{\alpha}_t}\,\mathbf{z}_0,\ (1-\bar{\alpha}_t)\mathbf{I})$, $\bar{\alpha}_t := \prod_{\ell=1}^{t} \alpha_\ell$, $\alpha_t := 1 - \beta_t$. [2]

The backward process recovers the data distribution from a noise distribution p_T with a iterative process. When β_t is small, one can approximate the denoising process by the score function $s(\mathbf{z}_t) := \nabla_{\mathbf{z}_t} \log p(\mathbf{z}_t)$. Specifically, $q(\mathbf{z}_{t-1}|\mathbf{z}_t) \approx N(\mathbf{z}_{t-1}; \frac{1}{\alpha_t}(x_t + \beta_t s(x_t)), \sigma_t^2 \mathbf{I})$, where $\sigma_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. Therefore, one can follow the following iterative procedure to draw samples from p_0:

$$\mathbf{z}_{t-1} = \frac{1}{\alpha_t}(\mathbf{z}_t + \beta_t s(\mathbf{z}_t)) + \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \epsilon_t \tag{1}$$

for t from T to 1 and $\epsilon_t \sim \mathcal{N}(0, 1)$. However, we do not know $s(\mathbf{z}_t)$ in closed form, so we train a score network $s_\theta(\mathbf{z}_t, t)$ to approximate it, with the following tractable loss function (Ho et al., 2020):

$$L(\theta) = \mathbb{E}_{t\in[T]}(1-\bar{\alpha}_t)\mathbb{E}_{\mathbf{z}_0\sim p_0, \mathbf{z}_t \sim q_{t|0}}[||s_\theta(\mathbf{z}_t; t) - \nabla_{\mathbf{z}_t} \log q_{t|0}(\mathbf{z}_t|\mathbf{z}_0)||^2] \tag{2}$$

Ma et al. (2025) shows that fix a time t, Equation (3) has the same global minimizer as Equation (2):

$$L(\theta; t) = \mathbb{E}_{\mathbf{z}_t \sim p_t}[||s_\theta(\mathbf{z}_t; t) - \nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t)] \tag{3}$$

Though Equation (3) is intractable because we do not know $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t)$, it's important for the reweighting technique used in Section 3.1.

2.3 Equilibrium Solvers: Online Mirror Descent and Fictitious Play

Let $u^i(\mathbf{a}^i; \pi^{-i}) := \mathbb{E}_{\mathbf{a}^{-i}\sim\pi^{-i}}[u^i(\mathbf{a}^i, \mathbf{a}^{-i})]$ denote player i's expected utility for action $\mathbf{a}^i$ against opponent strategy π^{-i}. We introduce two major iterative algorithms used to compute Nash equilibrium for continuous action in ML literature: Online Mirror Descent (OMD) and Fictitious Play (FP).

Online mirror descent. In OMD (Kivinen & Warmuth, 1997; Beck & Teboulle, 2003), each player computes a KL-regularized best response to the opponent's strategy from the previous iteration. At iteration k, player i computes

$$\pi^{i,k+1}(\mathbf{a}^i) = \pi^{i,k}(\mathbf{a}^i) \exp\left(\frac{1}{\eta} u_i(\mathbf{a}^i; \pi^{-i,k})\right) / Z^{i,k}_{\mathbf{OMD}}\,, \tag{4}$$

where $\frac{1}{\eta}$ is called temperature and $Z^{i,k}_{\mathbf{OMD}} = \int_{\mathcal{A}^i} \pi^{i,k}(\mathbf{a}^i) \exp\left(\frac{1}{\eta}\, u_i(\mathbf{a}^i; \pi^{-i,k})\right)\, da^i$ is an intractable unknown constant. We define empirical average of OMD responses as $\Pi^{i,k} = \frac{1}{k}\sum_{h=1}^{k} \pi^{i,h}_{\mathbf{OMD}}$. Under standard regularity conditions and an appropriate learning rate schedule, OMD is a no-regret algorithm on continuous action space (Krichene et al., 2015). When both players play such algorithms, their empirical strategies converge asymptotically to the Nash equilibrium set (Balandat et al., 2016).

Fictitious play. While in OMD player computes a KL-regularized best response to the opponent's current strategy, in FP (Brown, 1951) player best responds to the empirical distribution of the

[2] Both p and q are density function. We use q to denote tractable distributions like Gaussian corruption. When there is no ambiguity, we use p and p_0 interchangably.

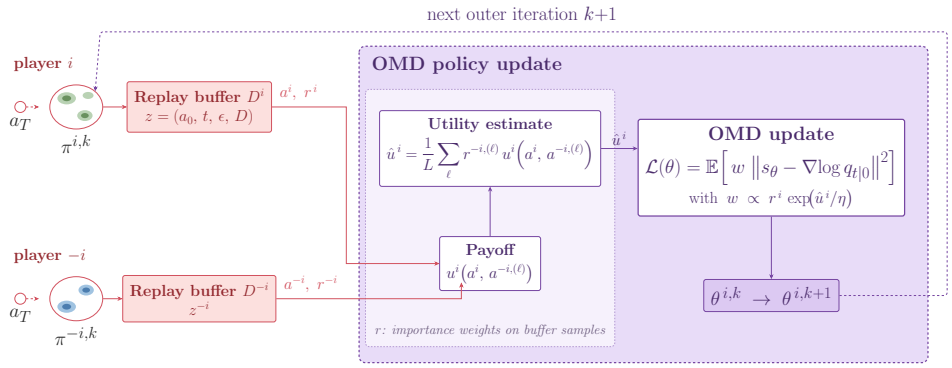

Figure 1: **Overview of DiffNash.** Each player samples actions from a diffusion policy into a replay buffer. Both lanes pass through a shared replay weight, obtained from a single-call ELBO difference (Proposition 2), and scored against the opponent's buffered actions by importance sampling. The OMD weight then reweights the denoising loss. Player $-i$ is updated symmetrically.

opponent's past play (Brown, 1951). FP is known to converge to Nash equilibrium with continuous action space and continuous utility function (Danskin, 1954). We consider a special variant called Generalised Weakened Fictitious Play (GWFP) (Leslie & Collins, 2006). Let $\pi^{i,k}_{\mathbf{GWFP}}$ denote player i's best response at iteration k, and let $\Pi^{i,k} := \frac{1}{k}\sum_{h=1}^{k} \pi^{i,h}_{\mathbf{GWFP}}$ denote the empirical average of its past responses. Under GWFP, player i computes

$$\pi^{i,k+1}_{\mathbf{GWFP}} = \exp\left(\frac{1}{\eta^k} u^i(\mathbf{a}^i, \Pi^{-i,k})\right)/Z^{i,k}\,, \tag{5}$$

where η^k is called temperature and $Z^{i,k} = \int_{\mathcal{A}^i} \exp\left(\frac{u^i(a^i,\Pi^{-i,k})}{\eta_k}\right) \mathrm{d}\mathbf{a}^i$ is an intractable normalization constant. Under vanishing temperature condition, $(\Pi^{1,k}, \Pi^{2,k})$ converges to the set of Nash equilibria asymptotically under GWFP (Leslie & Collins, 2006).

To represent multimodal mixed strategies, we represent player i's latent action with a diffusion model p_{θ^i}. A game-specific map $f^{\mathcal{G}}$ will convert the latent sample to a feasible player action, i.e., $\mathbf{a}^i = f^{\mathcal{G}}(\mathbf{z}), \mathbf{z} \sim p_{\theta^i}$. The resulting mixed strategy is then $\pi_{\theta^i} = f^{\mathcal{G}}_{\#} p_{\theta^i}$. We call π_{θ^i} player i's diffusion policy following Ma et al. (2025)'s notation, abstracting away latent space details.

Understanding Convergence Dynamics. Computing the Nash gap requires solving an intractable best response problem, and approximating it at every iteration is prohibitively expensive. To study convergence dynamics, we use a multi-market Cournot game (Kamra et al., 2019) whose unique Nash equilibrium is an analytically derivable singleton, which we denote as $(\mathbf{a}^{1,*}, \mathbf{a}^{2,*})$. In Cournot game, two firms compete by choosing production level across d independent markets to maximizes total profits. We defer Cournot game's details to Appendix D.2. We define the distance of a strategy profile Π to the equilibrium by $\frac{1}{2}\sum_{i=1}^{2} \mathbb{E}_{a^i\sim\pi^i} ||a^i - a^{i,\star}||_2$. Because this distance is easy to calculate, we compute it at each iteration in the Cournot game to study convergence dynamics.

3 Proposed Method

3.1 Enabling Diffusion-based GWFP and OMD via Reweighted Score Matching

The game-solving updates in GWFP Equation (5) and OMD Equation (4) both involve unknown normalization constants. Such unnormalized distributions can be viewed as energy-based models, which are generally difficult to train and sample from (Song & Kingma, 2021). Moreover, standard diffusion training requires samples from the target distribution, which are not directly available for the GWFP and OMD updates. Ma et al. (2025) show that, instead of integrating the squared score loss in Equation (3) with respect to p_t, the objective can be reweighted by any strictly positive function $g(\mathbf{a}_t)$ without changing its global minimizer:

$$L(\theta; t) = \int g(\mathbf{a}_t) ||s_\theta(\mathbf{a}_t; t) - \nabla_{\mathbf{a}_t} \log p_t(\mathbf{a}_t)||^2 d\mathbf{a}_t \tag{6}$$

By choosing appropriate $g(\mathbf{a}_t)$, they derive tractable loss to learn the score function of diffusion policy in online RL. We use $\pi^{i,k+1}_{t,\text{solver}}(\mathbf{a}_t)$ to denote noise corrupted target update at time t, i.e., $\pi^{i,k+1}_{t,\text{solver}}(\mathbf{a}_t) := \int \pi^{i,k+1}_{\text{solver}}(\mathbf{a}_0)\, q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0)\, \mathrm{d}\mathbf{a}_0$, where $\text{solver} \in \{\text{GWFP}, \text{OMD}\}$. Adapting the reweighting technique to our setting, with proper choice of $g(\mathbf{a}_t)$, the resulting objectives can recover the score functions of corrupted updated in both OMD and GWFP.

Proposition 1 (OMD and GWFP Loss)**.** *By choosing an appropriate $g_{\text{OMD}}(\mathbf{a}_t)$ and substitute it into Equation (6), we can equivalently write their loss functions as*

$$\mathcal{L}_{\text{OMD}}(\theta; t) = \mathbb{E}_{\mathbf{a}_0 \sim \pi^{i,k}} \left[\mathbb{E}_{\mathbf{a}_t \sim q_{t|0}(\cdot|\mathbf{a}_0)} \left[\exp\left(\frac{1}{\eta} u^i(\mathbf{a}_0; \pi^{-i,k}) \right) \left\| s_{\theta^i}(\mathbf{a}_t, t) - \nabla_{\mathbf{a}_t} \log q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) \right\|_2^2 \right] \right]. \tag{7}$$

For every $t \in (0, T]$, the minimizer of Equation (7) over s is the score function of $\pi^{i,k+1}_{t,\text{OMD}}(\mathbf{a}_t)$. In Appendix E.1, we show with a proper $g_{\text{GWFP}}(\mathbf{a}_t)$, we can recover the score function of $\pi^{i,k+1}_{t,\text{GWFP}}(\mathbf{a}_t)$.

We defer Proposition 1's proof to Appendix E.1. The intuition of choosing $g_{\mathbf{a}}$ is to ensure we can sample from all distributions appear in the resulting loss. For example, at iteration $k+1$, we have sampling access to $\pi^{i,k}$ and $q_{t|0}$, while $u^i(\tilde{\mathbf{a}}_0; \pi^{-i,k})$ can be unbiasedly estimated with samples from opponent's current strategy $\pi^{-i,k}$. Thus, the objectives in Proposition 1 are tractable, and optimizing them recover the corresponding score function $\nabla_{\mathbf{a}_t} \log \pi^{i,k+1}_{t,\text{solver}}(\mathbf{a}_t)$. Recall we only need the score function to perform the iterative diffusion sampling process specified in Equation (1). Hence, these objectives suffice to implement diffusion-based OMD and GWFP updates.

3.2 Comparison Between FP and OMD

In this section, we examine the complementary strengths of FP and OMD to guide the design of an algorithm that combines their advantages. (Perolat et al., 2021; Lauriere et al., 2022) report faster convergence for OMD, partly because best responding to stale historical strategies can slow FP's convergence. We confirm this finding in Figure 2: we run diffusion-based GWFP and OMD for 600 iterations on the Cournot game. OMD converges substantially faster in both dimensionalities. With action dimension $d = 30$, GWFP's distance to equilibrium decreases only marginally, highlighting the difficulty of convergence in high-dimensional action spaces. While prior works are often based on FP Kamra et al. (2019); Karthikeyan & Pant (2025), this result motivates using OMD as our equilibrium solver's backbone.

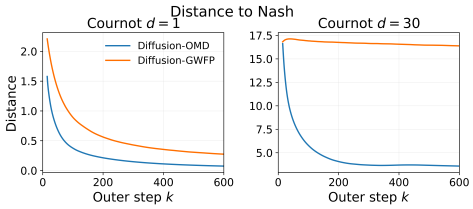

Figure 2: We run diffusion-based GWFP and OMD on the Cournot game for 600 iterations across three random seeds and report their mean distances to the Nash equilibrium.

Evaluating $\mathcal{L}_{\text{OMD}}(\theta; t)$ in Equation (7) requires estimating the expected utility against the current opponent policy. Using L freshly generated opponent actions, this expectation can be estimated as $\widehat{u}^i(\mathbf{a}_0; \pi^{-i,k}) = \frac{1}{L} \sum_{\ell=1}^{L} u^i\left(\mathbf{a}_0, \mathbf{a}^{-i,k}_{(\ell)}\right), \mathbf{a}^{-i,k}_{(\ell)} \sim \pi^{-i,k}$. However, since diffusion generation involves an iterative denoising process, repeatedly sampling fresh actions from the current opponent policy can be computationally expensive. FP largely avoids this cost because each player best-responds to the opponent's historical mixture $\Pi^{-i,k}$. Following Kamra et al. (2019), samples from this mixture can be maintained in a replay buffer, making them cheap to reuse. To improve the sample efficiency of our OMD solver, we similarly maintain a replay buffer for each player and use importance sampling (IS) to correct the distribution shift between the policy that generates a sample and $\pi^{-i,k}$. Formally, let $k^{(\ell)}$ denote the iteration at which the ℓ-th sampled action was generated, so that $\mathbf{a}^{-i,k^{(\ell)}}_{(\ell)} \sim \pi^{-i,k^{(\ell)}}$. We then have

$$\widehat{u}^i(\mathbf{a}_0; \pi^{-i,k}) = \frac{1}{L} \sum_{\ell=1}^{L} r^{-i,k,k^{(\ell)}} u^i\left(\mathbf{a}_0, \mathbf{a}^{-i,k^{(\ell)}}_{(\ell)}\right), \text{ where } r^{-i,k,k^{(\ell)}} = \frac{\pi_{\theta^{-i,k}}\left(\mathbf{a}^{-i,k^{(\ell)}}_{(\ell)}\right)}{\pi_{\theta^{-i,k_\ell}}\left(\mathbf{a}^{-i,k^{(\ell)}}_{(\ell)}\right)}, \tag{8}$$

$r^{-i,k,k^{(\ell)}}$ is sample l's importance weight[3]. However, computing $\pi_\theta(\cdot)$ in $r^{-i,k,k^{(\ell)}}$ exactly remains expensive for diffusion models (Song et al., 2020), motivating us to develop a principled proxy.

3.3 Enable Importance Sampling

In this section, we fix a player and suppress the player index. Consider a diffusion sample $\mathbf{a}_0$.

Lemma 1. $r^{k,k^{(\ell)}} = \exp\big(\log \pi_{\theta^k}(\mathbf{a}_0)) - \log \pi_{\theta^{k^{(\ell)}}}(\mathbf{a}_0)\big)$.

Lemma 1 is proved by direct calculation. By Lemma 1, we only need to approximate $\log \pi_\theta(\mathbf{a}_0)$ to enable efficient importance sampling. We achieve this by using evidence lower bounds (ELBO).

Definition 3. The ELBO of a sample $\mathbf{a}_0$ is $\mathrm{ELBO}_\theta(\mathbf{a}_0) \coloneqq \mathbb{E}_{q(\mathbf{a}_{1:T}|\mathbf{a}_0)}\left[\log \frac{p_\theta(\mathbf{a}_{0:T})}{q(\mathbf{a}_{1:T}|\mathbf{a}_0)}\right]$.

ELBO lower bounds the log-likelihood, i.e., $\mathrm{ELBO}_\theta(\mathbf{a}_0) \le \log \pi_\theta(\mathbf{a}_0)$ (Ho et al., 2020). Since $\log \pi_\theta(\mathbf{a}_0)$ is intractable for diffusion, the ELBO is commonly used as a tractable surrogate for training diffusion models. McAllister et al. (2026) also uses ELBO to approximate IS weight, though their method is for flow matching policy. We develop new method for diffusion policy. Substituting log likelihood with ELBO in Lemma 1, it remains to compute $\mathrm{ELBO}_{\theta^k}(\mathbf{a}_0) - \mathrm{ELBO}_{\theta^{k^{(\ell)}}}(\mathbf{a}_0)$ to obtain IS weights. Although evaluating each ELBO is still expensive (Ho et al., 2020), Proposition 2 shows that their difference admits an unbiased estimator requiring only one score network evaluation.

Proposition 2. *Suppose π^k and $\pi^{k'}$ share the same forward-noising schedule $\{\beta_t\}_{t=1}^T$. Denote $D(\theta; \mathbf{a}_0, t, \epsilon) = C_t \|s_\theta(\mathbf{a}_t, t) + \frac{\epsilon}{\sqrt{1-\bar\alpha_t}}\|_2^2$, where C_t a constant and $\mathbf{a}_t = \sqrt{\bar\alpha_t}\mathbf{a}_0 + \sqrt{1-\bar\alpha_t}\epsilon$. If $t \sim \mathrm{Unif}\{1,\dots,T\}$ and $\epsilon \sim \mathcal{N}(0, I)$, then $\widehat{\Delta}^{k',k} := T\big(D(\theta^k; \mathbf{a}_0, t, \epsilon) - D(\theta^{k'}; \mathbf{a}_0, t, \epsilon)\big)$ is ELBO difference's unbiased estimator: $\mathbb{E}_{t,\epsilon}[\widehat{\Delta}^{k',k}] = \mathrm{ELBO}^{k'}(\mathbf{a}_0) - \mathrm{ELBO}^{k}(\mathbf{a}_0)$.*

The proof is deferred to Section E.2. In iteration k, when we sample $\mathbf{a}_0 \sim \pi^k$, we record $\big[\mathbf{a}_0, t, \epsilon, D(\theta^k; \mathbf{a}_0, t, \epsilon)\big]$ in a buffer, so that we only need to compute $D(\theta^{k'}; \mathbf{a}_0, t, \epsilon)$ in future iteration k' to construct $\widehat{\Delta}^{k',k}$. Drawing sample from $\pi_{\theta^{k'}}$ requires T iterative score network calls by Equation (1), while computing $D(\theta^{k'}; \mathbf{a}_0, t, \epsilon)$ only requires one. Hence, our method reduces much sampling cost. We plug $\exp\left(\widehat{\Delta}^{k,k^{(\ell)}}\right)$ back to Equation (8) to enable reusing opponent buffers $\Pi^{-i,k}$.

3.4 Self-Action Importance Sampling Stabilizes Training

Recall evaluating the OMD loss Equation (7) requires samples from a player's self action $\pi^{i,k}$. Since we are maintaining replay buffers for both players, we can also use replay samples for a player's own action, not just opponent action. Denote a player's self action batch $\mathrm{B}_{\mathrm{self}}$. Instead of always getting fresh samples from $\pi^{i,k}$, we use $\lceil \zeta |B_{\mathrm{self}}| \rceil$ IS samples with the rest from $\pi^{i,k}$. Similar to opponent buffer reuse, we use the IS weight to correct distribution shift induced by replay samples. We defer the full algorithm details to Appendix C.

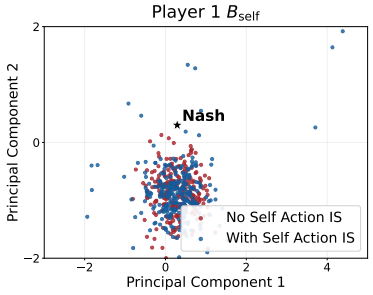

Figure 3: We use PCA to visualize the self-action buffer at iteration 101 for Cournot $d = 30$.

We find that adding importance-weighted replay stabilizes training and prevents premature plateaus. The policy starts concentrate on a narrow set of high-utility actions as it approaches equilibrium, limiting the coverage of fresh samples. Replaying diverse past actions broadens the training distribution and prevents the policy from overfitting to a narrow region. We visualize the impact of replay mechanism in Figure 3, where self action IS significantly improves coverage. We introduce an additional hyperparameter called trigger κ, and we initialize the self-action IS once standard deviation of $\mathrm{B}_{\mathrm{self}}$ falls below a fraction κ of its initial value. We analyze the impact of the self action IS mechanism in Section 5.3.

[3] In practice, we use $\min\left\{r^{-i,k,k^{(\ell)}}, C\right\}$ as the weight in Equation (8), where C caps the weight to prevent excessively large values (Schulman et al., 2017).

4 Related Work

Equilibrium Computation. We provide a comprehensive review in Section B and focus here on learning-based approaches for continuous action spaces. Prior work learns generative approximate best responses (Kamra et al., 2019), trains randomized policy networks with zeroth-order equilibrium dynamics (Martin & Sandholm, 2023), or minimizes exploitability using learned best responses (Martin & Sandholm, 2025). In contrast, we parameterize each player's evolving mixed strategy with a diffusion model to capture more expressive distributions.

Diffusion Models And Its Application in Online RL Diffusion models learn complex, multimodal distributions through iterative denoising and score matching (Ho et al., 2020; Song et al., 2021). Their ability to model complex distributions motivated their use as policy parameterizations in RL (Wang et al., 2024; Psenka et al., 2024; Ren et al., 2025). Most closely related to ours, Ma et al. (2025) show that exponential reweighting of samples from the current diffusion policy is sufficient to implement a KL-regularized policy-improvement step. We adapt the reweighting technique and address it in the equilibrium computation setting. We give a comprehensive review of the field in Section B.

Diffusion Applications in Strategic Environment To our knowledge, the only direct application of diffusion to our setting is Karthikeyan & Pant (2025), which we include as a baseline and discuss in Section 5. We discuss diffusion model applications in other related strategic settings in Section B.

5 Experiments

Beyond the Cournot game, we evaluate **DiffNash** on a forest security game and a continuous variant of Colonel Blotto, two widely studied games that also admit intuitive visualizations.

Metrics and Baselines: We use Nash Gap (Definition 2) as our main evaluation metric. The best response optimization required in Nash Gap cannot be computed exactly. Following Kamra et al. (2019), we use an evolutionary search that repeatedly retains high-utility candidates and perturbs them with Gaussian noise to approximate the Nash Gap. We defer computation details to Section D.1.

We compare against three baselines. **DeepFP** (Kamra et al., 2019) represents each player's strategy in FP with a neural generator $\mathcal{T}_\theta(\mathbf{z})$, where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. **Neural OMD** adapts the mirror descent policy optimization framework of Tomar et al. (2022) to equilibrium computation, using a Gaussian neural policy for each player. **DiffFP** (Karthikeyan & Pant, 2025) parameterizes player strategies with diffusion models within FP. Since FP updates do not directly provide samples from the target strategy, DiffFP instead refines sampled actions $\mathbf{a}^i$ toward higher-utility regions using $\nabla_{\mathbf{a}^i} u^i(\mathbf{a}^i; \Pi^{-i,k})$ and trains the diffusion policy on the refined actions. These baselines help isolate the benefits of diffusion policy and principled policy updates. For fair comparison, we train each player's policy for $40K$ gradient updates across experiments. We use default hyperparameters for baselines.

5.1 Forest Security Game

Forest security game (Johnson et al., 2012) originated from forest economics (Albers, 2010). We consider the continuous version studied by Kamra et al. (2018) and **DeepFP** (Kamra et al., 2019). A defender selects M locations to patrol, while an attacker selects N locations to target in an endangered circular forest. Each location is parameterized by its radial distance and radian angle. Hence, defender and attacker's action dimensionality are $2M$ and $2N$, respectively. As M and N increase, the resulting strategy space becomes high-dimensional, a setting where Kamra et al. (2019) reports degraded performance. An attacker can collect trees within R_A of its location, and a defender can stop any attacker within R_D of them. The attacker's utility is the value of the harvested trees but receives no value and incurs a penalty if caught by any defender. The defender's utility is the negative of the attacker's utility. Across experiments, we keep the interaction and payoff structure fixed and vary only the spatial distribution of trees along with the number of defenders and attackers. Changing the geometry creates qualitatively different optimal strategies by altering which attack locations are valuable, which regions can be protected together, and how many spatially separated patches the strategy must cover. In this game, we aim to find a robust patrol strategy for defender, so besides Nash gap, we also compute $\mathrm{Exploit}_\mathrm{D}$, measured as the attacker's best-response reward against the learned average defender. We seek strategies profiles with low $\mathrm{Exploit}_\mathrm{D}$ and Low Nash Gap GAP.

Experimental results. Table 1 shows that DiffNash achieves the best mean performance across all six forests, reducing Nash gap by an average of **35.6%** relative to the strongest baseline. The

Table 1: **Forest security game results.** We report the attacker best-response reward Exploit_D against the learned average defender and the two-sided Nash gap G as mean $\pm$ 95% Student-t confidence intervals over 5 random seeds; lower is better for both. F1 and F2 are taken from Kamra et al. (2019).

		DeepFP		Neural OMD		DiffFP		DiffNash (Ours)	
Forest	$m = n$	Exploit_D	Gap	Exploit_D	Gap	Exploit_D	Gap	Exploit_D	Gap
F1	1	244.5 ± 164.2	245.8 ± 160.6	212.6 ± 160.4	216.6 ± 160.2	490.2 ± 3.4	493.0 ± 3.6	$\mathbf{96.7 \pm 21.2}$	$\mathbf{79.5 \pm 28.8}$
F1	3	370.5 ± 309.2	373.3 ± 297.8	307.8 ± 287.6	314.5 ± 290.0	735.8 ± 40.8	741.3 ± 41.0	$\mathbf{130.4 \pm 9.3}$	$\mathbf{122.1 \pm 12.5}$
F2	1	338.6 ± 6.5	330.3 ± 5.8	309.7 ± 28.3	313.7 ± 28.4	354.9 ± 0.9	357.8 ± 0.7	$\mathbf{300.8 \pm 5.2}$	$\mathbf{264.4 \pm 95.5}$
F2	3	702.1 ± 84.3	687.4 ± 78.5	501.0 ± 179.6	500.1 ± 177.2	817.3 ± 101.5	821.6 ± 101.3	$\mathbf{322.5 \pm 110.1}$	$\mathbf{300.7 \pm 126.4}$
F3	5	578.8 ± 109.4	565.7 ± 119.7	767.1 ± 278.4	761.1 ± 271.1	818.5 ± 123.7	800.9 ± 121.3	$\mathbf{559.7 \pm 76.4}$	$\mathbf{528.2 \pm 142.7}$
F4	5	804.4 ± 118.5	776.6 ± 117.4	1145.9 ± 191.6	1140.9 ± 187.3	836.4 ± 114.7	822.7 ± 112.8	$\mathbf{660.4 \pm 39.8}$	$\mathbf{622.6 \pm 52.3}$

gap is most pronounced in heterogeneous, multi-resource settings, where equilibrium strategies must distribute probability mass across several strategically distinct regions. DiffNash also achieves broader effective coverage. In F3 of Figure 4, for example, **DiffNash**'s guards (blue) concentrate around all four tree clusters (green), whereas DeepFP and DiffFP learn more diffuse patrol strategies and Neural OMD places substantial mass in low-tree-density regions. Importantly, diffuse coverage is not necessarily effective coverage: allocating guard probability to regions with few or no trees provides little protection value. We analyze the failure modes of all baselines:

- **DeepFP/NeuralOMD**: Both exhibit mode collapse. In F2 of Figure 4, **Neural OMD** concentrates on a single forest patch. For **DeepFP**, Figure 4 and additional Appendix F.1 visualizations show both players' strategies forming narrow circular bands. Since actions are represented in polar coordinates, this pattern indicates collapse along the radial dimension.
- **DiffFP**: In large forests, small changes in guard locations are unlikely to change whether attackers are captured. Consequently, the defender utility gradient, i.e., $\nabla_{\mathbf{a}^i} u^i(\mathbf{a}^i; \Pi^{-i,k})$, is nearly zero for most locations, causing **DiffFP**'s defender strategy to get stuck, as shown by Figure 4's F2. In contrast, **DiffNash** does not rely on local utility gradients. We update the entire defender policy in a principled way, yielding substantially better performance.

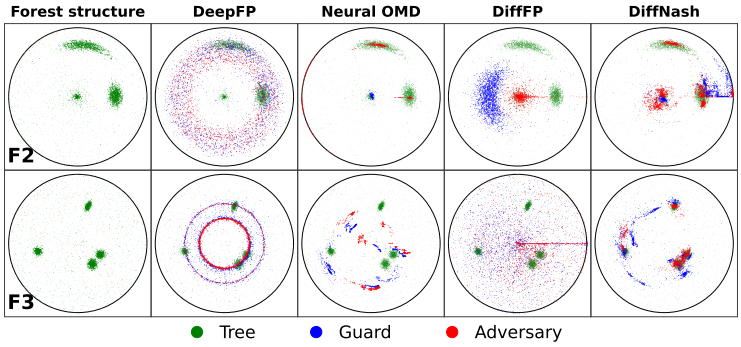

Figure 4: **Empirical strategies in the forest security game** *Top:* F2 ($m = n = 3$). *Bottom:* F3 ($m = n = 5$). Green points denote trees. Each blue/red points denote a sample from defender/attacker policy, respectively. For each method and environment, the displayed strategy is taken from 5 random seeds whose Nash gap is the median. Quantitative results are in Table 1.

5.2 Continuous Colonel Blotto

We evaluate **DiffNash** on the continuous Colonel Blotto game (Adam et al., 2021) where two players simultaneously allocate a fixed budget across n weighted battlefields. Player's utility depends on how much they and opponent invest in each battlefield. Because allocations are coupled by the budget constraint, equilibrium play generally requires nontrivial randomization over a continuous action space. We consider games with $n = 6$ and $n = 10$. Adam et al. (2021) provides a discrete algorithm which approximates the target equilibrium well [4], visualized in the first column of Figure 5. We vary player utilty function to get distinct target equilibrium, one with symmetric and another with radial shape. Full game details are provided in Appendix D.4.

[4] Such oracle-based methods do not directly extend to general continuous-action games without a tractable best-response oracle; see Appendix D.4.2.

Experimental results. Overall **DiffNash** reduces the Nash Gap by **24.7%** relative to the strongest baseline. In the symmetric setting of Table 2, diffusion-based policies achieve substantially lower Nash gaps than the neural baselines, highlighting their ability to represent multimodal strategies. In the radial-band setting, **DiffNash** consistently outperforms **DiffFP**. As in the forest game, when high-utility actions lie within a narrow region (the radial band), **DiffFP** can receive little local gradient signal from low-utility initializations. In Figure 5, this causes its policy to remain concentrated near the center, outside the support of the target equilibrium, whereas **DiffNash** recovers the correct equilibrium geometry. We visualize all settings in Appendix Figure 8.

Table 2: **Continuous Colonel Blotto results.** Nash gap G on symmetric and radial-band Blotto with $n \in \{6, 10\}$. Values are mean $\pm$ 95% Student-t confidence intervals over five seeds; lower is better.

n	Equilibrium	DeepFP	Neural OMD	DiffFP	DiffNash (Ours)
6	Symmetric	6.854 ± 1.366	6.807 ± 1.832	$\mathbf{0.919 \pm 0.065}$	0.942 ± 0.168
6	Radial band	0.688 ± 0.001	0.836 ± 0.236	1.038 ± 0.056	$\mathbf{0.501 \pm 0.139}$
10	Symmetric	9.525 ± 1.707	7.233 ± 1.304	$\mathbf{1.506 \pm 0.056}$	1.724 ± 0.346
10	Radial band	0.696 ± 0.002	0.793 ± 0.318	1.446 ± 0.037	$\mathbf{0.531 \pm 0.116}$

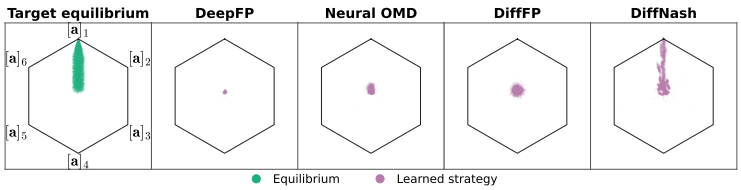

Figure 5: Continuous Colonel Blotto (radial-band variant,$n = 6$). Green denotes samples from the target equilibrium, while purple denotes samples from the learned strategy. Each point represents an allocation, where proximity to $[\mathbf{a}]_j$ indicates a larger allocation to the j-th battlefield. The point cloud visualizes the distribution of allocations induced by each strategy.

Robustness Across experiments in Section 5.1 and Section 5.2 we perform minimal hyperparameter tuning: we fix self action ratio $\zeta = 1\%$ and only choose IS trigger $\kappa = \{0, 10\%\}$. **DiffNash** shows strong robustness to hyperparameters.

5.3 Ablation and Hyperparameter Study

Ablation We ablate the two importance-sampling components of **DiffNash** on Cournot ($d = 30$). In Figure 6, the green and blue curves almost overlap, indicating that opponent IS has little effect on convergence. Its primary benefit is computational: in this experiment, opponent IS reduces the number of score-network calls to roughly 66% of that required without opponent IS. Moreover, **DiffNash** (orange) achieves the lowest Nash Gap, showing that self action IS prevents the early plateau observed without self action IS and reduces variance across runs. We observe a similar pattern in the forest ablation in Appendix F.4. These results support using IS for both opponent and self-action replay.

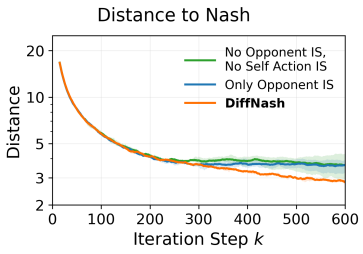

Figure 6: Performance under different IS configurations. Solid lines denote mean, and shades denote the min to max range across runs.

We additionally show our method's performance in a general sum version of the Forest game in Section F.3.

Hyperparameter Study We provide hyperparameter study for self action IS ratio ζ, trigger κ, self action buffer size and opponent action buffer size in Appendix F.5.

6 Conclusion

We develop **DiffNash**, a framework that parameterizes player strategies with diffusion models within OMD to compute Nash equilibria in high-dimensional continuous action spaces. We derive a reweighted score-matching objective that trains the diffusion model to realize OMD updates. To improve sample efficiency and training stability, we reuse past actions through a replay buffer and correct the resulting distribution shift with an ELBO-based importance-sampling strategy. On forest security and continuous Colonel Blotto games, **DiffNash** better recovers equilibrium distributions and reduces Nash gap by **35.6%** and **24.7%** compared to strongest baselines, respectively.

AI USAGE DISCLOSURE

In this work, we used generative AI tools (GPT-5.6 Sol, GPT-6 Sol, and Claude Opus 5) for polishing and drafting sections of the manuscript, flowcharts, and assists with coding and proofs. Generating synthetic data is not applicable to this work. Additionally, we used generative AI tools for brainstorming research ideas and searching related work. At least one coauthor has reviewed AI-assisted work. We manually check the generated code and proof. The main technical contributions, including reweighted score matching objective and importance sampling strategy, are the authors' original ideas. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies algorithms for computing Nash equilibria in simulated continuous-action games and does not involve human subjects, personal or sensitive data, or real-world interventions. While equilibrium-computation methods may be applicable to domains such as security and resource allocation, all experiments in this work use synthetic benchmark environments and do not involve deployed decision-making systems. We are not aware of additional ethical concerns arising from this work.

REPRODUCIBILITY STATEMENT

We provide complete pseudocode for DiffNash and describe the training procedure, replay mechanism, game definitions, environment parameters, reference solutions, hyperparameters, best-response and Nash-gap evaluation procedures, ablations, and computing infrastructure in the appendix. These details are intended to make our method and reported experiments reproducible from the paper. Some baseline components and environment code, including the original Forest Security Game and DeepFP implementations, were not publicly available and were obtained directly from the original authors. Reported results are averaged over multiple random seeds with confidence intervals, and proofs of all theoretical results are included.

REFERENCES

Lukáš Adam, Rostislav Horčík, Tomáš Kasl, and Tomáš Kroupa. Double oracle algorithm for computing equilibria in continuous games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 5070–5077, 2021.

Heidi J Albers. Spatial modeling of extraction and enforcement in developing country protected areas. *Resource and Energy Economics*, 32(2):165–179, 2010.

Steve Alpern, Thuy Bui, Thomas Lidbetter, and Katerina Papadaki. Continuous patrolling games. *Operations Research*, 70(6):3076–3089, 2022.

Ioannis Anagnostides, Gabriele Farina, Ioannis Panageas, and Tuomas Sandholm. Optimistic mirror descent either converges to nash or to strong coarse correlated equilibria in bimatrix games. *Advances in Neural Information Processing Systems*, 35:16439–16454, 2022.

Maximilian Balandat, Walid Krichene, Claire Tomlin, and Alexandre Bayen. Minimizing regret on reflexive banach spaces and nash equilibria in continuous zero-sum games. *Advances in Neural Information Processing Systems*, 29, 2016.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Charles Blair. Problem complexity and method efficiency in optimization (as nemirovsky and db yudin). *Siam Review*, 27(2):264, 1985.

George W Brown. Iterative solution of games by fictitious play. *Act. Anal. Prod Allocation*, 13(1): 374, 1951.

Jakub Cerny, Chun Kai Ling, Christian Kroer, and Garud Iyengar. Layered graph security games. In *Proceedings of the Thirty-ThirdInternational Joint Conference on Artificial Intelligence*, pp. 2695–2703. International Joint Conferences on Artificial Intelligence Organization, 2024. doi: 10.24963/ijcai.2024/298. URL https://doi.org/10.24963/ijcai.2024/298.

Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player nash equilibria. *Journal of the ACM (JACM)*, 56(3):1–57, 2009.

JM Danskin. Fictitious play for continuous games. *Naval Research Logistics Quarterly*, 1(4): 313–320, 1954.

George B Dantzig. A proof of the equivalence of the programming problem and the game problem, in "activity analysis of production and allocation"(ed. tc koopmans), cowles commission monograph, no. 13, 1951.

Jiarui Gan, Bo An, Yevgeniy Vorobeychik, and Brian Gauch. Security games on a plane. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Sam Ganzfried. Algorithm for computing approximate nash equilibrium in continuous games with application to continuous blotto. *Games*, 12(2):47, 2021.

Irving L Glicksberg. A further generalization of the kakutani fixed theorem, with application to nash equilibrium points. *Proceedings of the American Mathematical Society*, 3(1):170–174, 1952.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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*, pp. 327–334, 2011.

Matthew Johnson, Fei Fang, and Milind Tambe. Patrol strategies to maximize pristine forest area. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pp. 295–301, 2012.

Nitin Kamra, Umang Gupta, Fei Fang, Yan Liu, and Milind Tambe. Policy learning for continuous space security games using neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Nitin Kamra, Umang Gupta, Kai Wang, Fei Fang, Yan Liu, and Milind Tambe. Deep fictitious play for games with continuous action spaces. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2042–2044, 2019.

Akash Karthikeyan and Yash Vardhan Pant. Difffp: Learning behaviors from scratch via diffusion-based fictitious play. *arXiv preprint arXiv:2511.13186*, 2025.

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

Jyrki Kivinen and Manfred K Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and computation*, 132(1):1–63, 1997.

Lingkai Kong, Haichuan Wang, Yuqi Pan, Cheol Woo Kim, Mingxiao Song, Alayna Nguyen, Tonghan Wang, Haifeng Xu, and Milind Tambe. Robust optimization with diffusion models for green security. In *The 41st Conference on Uncertainty in Artificial Intelligence*, 2025. URL https://openreview.net/forum?id=lLdal3hZtz.

Lingkai Kong, Anagha Satish, Hezi Jiang, Akseli Kangaslahti, Andrew Ma, Wenbo Chen, Mingxiao Song, Lily Xu, and Milind Tambe. Latent spherical flow policy for reinforcement learning with combinatorial actions. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=07wwDFdi3k.

Lingkai Kong, Haichuan Wang, Charles A Emogor, Vincent Böersch-Supan, Lily Xu, and Milind Tambe. Generative ai against poaching: Latent composite flow matching for poaching prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 40279–40286, 2026b.

Walid Krichene, Maximilian Balandat, Claire Tomlin, and Alexandre Bayen. The hedge algorithm on a continuum. In *International Conference on Machine Learning*, pp. 824–832. PMLR, 2015.

Christian Kroer and Tuomas Sandholm. Discretization of continuous action spaces in extensive-form games. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, pp. 47–56, 2015.

Tomáš Kroupa and Tomáš Votroubek. Multiple oracle algorithm to solve continuous games. In *International Conference on Decision and Game Theory for Security*, pp. 149–167. Springer, 2022.

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. *Advances in neural information processing systems*, 30, 2017.

Mathieu Lauriere, Sarah Perrin, Sertan Girgin, Paul Muller, Ayush Jain, Theophile Cabannes, Georgios Piliouras, Julien Pérolat, Romuald Elie, Olivier Pietquin, et al. Scalable deep reinforcement learning algorithms for mean field games. In *International conference on machine learning*, pp. 12078–12095. PMLR, 2022.

David S Leslie and Edmund J Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006.

Steven Li, Rickmer Krohn, Tao Chen, Anurag Ajay, Pulkit Agrawal, and Georgia Chalvatzaki. Learning multimodal behaviors from scratch with diffusion policy gradient. *Advances in Neural Information Processing Systems*, 37:38456–38479, 2024.

Zhuoran Li, Hai Zhong, Xun Wang, Qingxin Xia, Lihua Zhang, and Longbo Huang. Diffusing to coordinate: Efficient online multi-agent diffusion policies. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=tHY0G4PhCR.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=6Anv3KB9lz.

Carlos Martin and Tuomas Sandholm. Finding mixed-strategy equilibria of continuous-action games without gradients using randomized policy networks. In Edith Elkind (ed.), *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 2844–2852. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/317. URL https://doi.org/10.24963/ijcai.2023/317. Main Track.

Carlos Martin and Tuomas Sandholm. Approxed: Approximate exploitability descent via learned best responses. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1454–1463. IEEE Computer Society, 2025. doi: 10.65109/PNRR9301. URL https://doi.org/10.65109/pnrr9301.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=eoEmoKoQpJ.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KVf2SFL1pi.

Steven Perkins and David S Leslie. Stochastic fictitious play with continuous action sets. *Journal of Economic Theory*, 152:179–213, 2014.

Steven Perkins, Panayotis Mertikopoulos, and David S Leslie. Mixed-strategy learning with continuous action sets. *IEEE Transactions on Automatic Control*, 62(1):379–384, 2015.

Julien Perolat, Sarah Perrin, Romuald Elie, Mathieu Laurière, Georgios Piliouras, Matthieu Geist, Karl Tuyls, and Olivier Pietquin. Scaling up mean field games with online mirror descent. *arXiv preprint arXiv:2103.00623*, 2021.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=35ahHydjXo.

Sasha Rakhlin and Karthik Sridharan. Optimization, learning, and games with predictable sequences. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/f0dd4a99fba6075a9494772b58f95280-Paper.pdf.

Allen Z. Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mEpqHvbD2h.

Brian Roberson. The colonel blotto game. *Economic Theory*, 29(1):1–24, 2006.

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

Yuanyuan Shi and Baosen Zhang. Multi-agent reinforcement learning in cournot games. In *2020 59th ieee conference on decision and control (cdc)*, pp. 3561–3566. IEEE, 2020.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Noah D Stein, Asuman Ozdaglar, and Pablo A Parrilo. Separable and low-rank continuous games. *International Journal of Game Theory*, 37(4):475–504, 2008.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=aBO5SvgSt1.

Tonghan Wang, Yanchen Jiang, and David Parkes. Bundleflow: Deep menus for combinatorial auctions by diffusion-based optimization. *Advances in Neural Information Processing Systems*, 38: 16853–16876, 2026.

Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang WU, Jingliang Duan, and Shengbo Eben Li. Diffusion actor-critic with entropy regulator. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=l0c1j4QvTq.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ACagRwCCqu.

Appendix for DiffNash: Find Nash Equilibrium in High Dimensional Continuous Action Spaces with Diffusion Policies

A Limitations **14**

B Additional Related Work **14**

C Training Algorithm **15**

D Details about Experiment Setup **16**

D.1 Nash Gap Computation 16

D.2 Experimental Details of Cournot Game 16

D.3 Experimental Details of Forest Protection Game 19

D.4 Experimental Details of Continuous Colonel Blotto 21

E Missing Proofs **25**

E.1 Proof of Proposition 1 25

E.2 Proof of Proposition 2 27

F Additional Experimental Results **29**

F.1 Additional Forest Game Visualizations 29

F.2 10-dimensional Continuous Colonel Blotto 30

F.3 General-Sum Forest Protection 30

F.4 Importance-sampling Forest ablation 31

F.5 Forest Game Hyperparameter Sweep 32

G Computing Infrastructure **33**

A Limitations

Our results suggest several directions for future work. Although replay substantially reduces the need to generate fresh actions, diffusion sampling still requires iterative denoising. Alternative expressive generative policies, such as flow-based models could further reduce the computational cost of equilibrium computation.

We also focus on two-player normal-form games. Extending DiffNash to multiplayer or sequential settings, is a natural direction for future work.

B Additional Related Work

Equilibrium Computation Finite two-player zero-sum games admit linear-programming formulations (Dantzig, 1951), while equilibrium computation in finite general-sum games is PPAD-complete (Chen et al., 2009). Continuous action games remain challenging because mixed strategies are distributions over infinite action spaces and best-response computation may require difficult continuous optimizaition. Prior work establishes continuous-action variants of fictitious-play (Perkins & Leslie, 2014) and develops strategy-generaion methods for zero-sum and general-sum continuous games (Adam et al., 2021; Kroupa & Votroubek, 2022). Learning-based approaches include DeepFP, which learns generative approximate best responses (Kamra et al., 2019), randomized policy networks trained with zeroth-order equilibrium dynamics (Martin & Sandholm, 2023), and exploitability

minimization with learned best responses (Martin & Sandholm, 2025) . Our approach instead parameterizes each player's evolving mixed strategy with a diffusion model and trains it using FP and OMD based equilibrium updates (Rakhlin & Sridharan, 2013).

Diffusion and Flow-Based Models for Online RL Diffusion and flow-based policies have been widely studied for RL (Wang et al., 2024; Psenka et al., 2024; Ma et al., 2025; Ren et al., 2025). Most closely related to our policy update, Ma et al. (2025) show that exponential reweighting of samples from the current diffusion policy implements a KL-regularized policy-improvement step, yielding Diffusion Policy Mirror Descent.

Related work also applies expressive diffusion policies in cooperative multi-agent RL (Li et al., 2026), while flow-matching policies have been developed for online continuous and combinatorial control (Park et al., 2025; Zhang et al., 2025; McAllister et al., 2026; Kong et al., 2026a). These methods optimize return or coordination rather than equilibrium against strategic opponents. In our setting, policy-improvement weights depend on utility against an evolving opponent. We additionally use ELBO differences as approximate density ratios to reuse samples across strategic iterates.

Diffusion Applications in Strategic Environment Diffusion models and their variants, such as flow matching (Lipman et al., 2023), have been applied to strategic settings adjacent to equilibrium computation. Wang et al. (2026) uses flow matching to represent bundle distributions in combinatorial auctions, achieving higher revenue while scaling to larger numbers of items. When opponent data is available, diffusion models can also be used to learn opponent behavior. Kong et al. (2025) uses diffusion models to learn poacher behavior in green security games and design robust patrol strategies, while (Kong et al., 2026b) uses flow matching to model poacher behavior from real-world data. Unlike these works, we assume no prior knowledge of opponent behavior and instead seek a Nash strategy, which provides a worst-case guarantee in 2p0s games.

C Training Algorithm

We provide the full pseudocode for DiffNash in Algorithm 3. Algorithm 1 constructs replay entries, Algorithm 2 computes their ELBO-based importance weights, and Algorithm 3 gives the complete OMD training procedure. Self-action replay is activated only if the specified trigger fires; otherwise, training continues using fresh self-actions.

Algorithm 1 PrepareSample

Input: Action a_0, behavior parameters θ^b, and diffusion schedule $\{\alpha_t, \bar{\alpha}_t, \beta_t, \sigma_t^2\}_{t=1}^T$.
Output: Replay entry $z = (a_0, t, \epsilon, D_b)$.

1: Sample
$$t \sim \mathrm{Unif}\{1, \dots, T\}, \qquad \epsilon \sim \mathcal{N}(0, I).$$
2: Construct
$$a_t \leftarrow \sqrt{\bar{\alpha}_t} a_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon.$$
3: Compute
$$D_b \leftarrow \frac{\beta_t^2}{2\sigma_t^2 \alpha_t} \left\| s_{\theta^b}(a_t, t) + \frac{\epsilon}{\sqrt{1 - \bar{\alpha}_t}} \right\|_2^2.$$
4: **return** $z \leftarrow (a_0, t, \epsilon, D_b)$.

Algorithm 2 COMPUTEWEIGHT

Input: Current parameters θ^k, replay entry $z = (a_0, t, \epsilon, D_b)$ generated at iteration b, diffusion schedule, and maximum log-weight $C > 0$.
Output: Clipped importance weight $r^{k,b} \le e^C$.

1: Reconstruct
$$a_t \leftarrow \sqrt{\bar{\alpha}_t} a_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon.$$
2: Compute
$$D_k \leftarrow \frac{\beta_t^2}{2\sigma_t^2 \alpha_t} \left\| s_{\theta^k}(a_t, t) + \frac{\epsilon}{\sqrt{1 - \bar{\alpha}_t}} \right\|_2^2.$$
3: Compute
$$\widehat{\Delta}^{k,b} \leftarrow T(D_b - D_k).$$
4: **return**
$$r^{k,b} \leftarrow \exp\Big(\min\{\widehat{\Delta}^{k,b}, C\}\Big).$$

D DETAILS ABOUT EXPERIMENT SETUP

D.1 NASH GAP COMPUTATION

For a learned strategy profile $\pi = (\pi^1, \pi^2)$, we measure the total gain available from unilateral deviations. For player i, define

$$\epsilon^i(\pi) = \max_{a^i \in \mathcal{A}^i} u^i(a^i; \pi^{-i}) - U^i(\pi^i, \pi^{-i}), \tag{9}$$

and compute the Nash gap as

$$G(\pi^1, \pi^2) = \epsilon^1(\pi) + \epsilon^2(\pi). \tag{10}$$

Because expected utility is linear in the deviating strategy, it is sufficient to optimize over pure best responses. In the two-player zero-sum setting, writing U for player 1's payoff, this simplifies to

$$G(\pi^1, \pi^2) = \max_{a^1 \in \mathcal{A}^1} U(a^1, \pi^2) - \min_{a^2 \in \mathcal{A}^2} U(\pi^1, a^2). \tag{11}$$

For continuous action spaces, the best-response optimization is computed with a game-specific evaluation oracle. For the forest game, following Kamra et al. (2019), we use evolutionary search with a population of 50 candidates for 20 iterations. At each iteration, the top 10 candidates are retained and four Gaussian-perturbed mutations are generated from each survivor. Candidate utilities are estimated against 100 opponent samples during the search, and the resulting best response is reevaluated against 1,000 samples. When estimating the individual unilateral gains in (10), the utility of the learned profile is estimated from 2,000 sampled action pairs.

For symmetric Colonel Blotto, we follow the discretized best-response procedure of Adam et al. (2021). Final Nash gaps use a simplex grid with spacing $1/1024$ and an exact knapsack dynamic program, with 20,000 samples from each learned policy. For the radial-band variant, we approximate each best response from a pool of approximately 11,400 candidates, refine the best 64 candidates for 400 gradient steps, and estimate final payoffs using 8,000 policy samples per player. The same evaluation procedure is used for all methods within each game.

D.2 EXPERIMENTAL DETAILS OF COURNOT GAME

We consider a two-player multi-market Cournot competition game with d independent markets, where $A > 0$ is the demand intercept and $b > 0$ controls the slope of inverse demand. Player $i \in \{1, 2\}$ chooses production quantities

$$a^i = (a_1^i, \ldots, a_d^i) \in [0, q_{\max}]^d.$$

In each market j, the inverse demand function is

$$p_j(a_j^1, a_j^2) = A - b(a_j^1 + a_j^2),$$

Algorithm 3 Replay-Buffer Diffusion Online Mirror Descent (DiffNash)

Input: Outer iterations K, inner steps H, self-action batch size B_{self}, opponent batch size M, initial buffer size N_0, opponent-buffer capacity N_{buf}, refresh size R, OMD parameter η, self-action replay fraction ζ, trigger threshold κ, and maximum log-weight C.

1: Initialize $\pi_{\theta^{i,1}} = \text{Unif}(\mathcal{A}^i)$, $i \in \{1,2\}$; initialize each player's opponent replay buffer and self-action history with N_0 samples using PREPARESAMPLE with behavior parameters $\theta^{i,1}$, and record the initial self-action spread.

2: **for** $k = 1, \dots, K$ **do**

3: **for** $i \in \{1,2\}$ **do**

4: Sample opponent entries $\widetilde{z}^{-i,(1:M)}$ with behavior iterations $b_1, \dots, b_M$; let $\widetilde{a}_0^{-i,(m)}$ denote the clean action in $\widetilde{z}^{-i,(m)}$, and compute

$$\widetilde{r}^{-i,k,b_m} \leftarrow \text{COMPUTEWEIGHT}(\theta^{-i,k}, \widetilde{z}^{-i,(m)}, C), \qquad \widetilde{w}^{-i,(m)} \leftarrow \frac{M\widetilde{r}^{-i,k,b_m}}{\sum_{j=1}^{M} \widetilde{r}^{-i,k,b_j}}, \quad m \in [M].$$

5: Set $b_{\text{IS}} \leftarrow \lceil \zeta B_{\text{self}} \rceil$ if the current self-action spread is below κ times its initial value, and $b_{\text{IS}} \leftarrow 0$ otherwise; set $\theta^{i,k,0} \leftarrow \theta^{i,k}$.

6: **for** $h = 1, \dots, H$ **do**

7: Sample

$$a_0^{i,(b_{\text{IS}}+1:B_{\text{self}})} \overset{\text{i.i.d.}}{\sim} \pi_{\theta^{i,k}}, \qquad z^{i,(1:b_{\text{IS}})} \sim \text{self-action history}.$$

8: For $l \in [b_{\text{IS}}]$, let $a_0^{i,(l)}$ be the clean action in $z^{i,(l)}$, with behavior iteration b_l, and compute

$$r^{i,k,b_l} \leftarrow \text{COMPUTEWEIGHT}(\theta^{i,k}, z^{i,(l)}, C), \qquad w^{i,(l)} \leftarrow \frac{b_{\text{IS}} r^{i,k,b_l}}{\sum_{j=1}^{b_{\text{IS}}} r^{i,k,b_j}}.$$

9: For $l \in [B_{\text{self}}]$, estimate

$$\widehat{u}^i\left(a_0^{i,(l)}; \pi_{\theta^{-i,k}}\right) \leftarrow \frac{1}{M} \sum_{m=1}^{M} \widetilde{w}^{-i,(m)} u^i\left(a_0^{i,(l)}, \widetilde{a}_0^{-i,(m)}\right).$$

10: For $l \in [B_{\text{self}}]$, sample $t^{i,(l)} \sim \text{Unif}\{1, \dots, T\}$ and $\epsilon^{i,(l)} \sim \mathcal{N}(0, I)$, and set

$$a_{t^{i,(l)}}^{i,(l)} \leftarrow \sqrt{\bar{\alpha}_{t^{i,(l)}}}\, a_0^{i,(l)} + \sqrt{1 - \bar{\alpha}_{t^{i,(l)}}}\, \epsilon^{i,(l)}.$$

11: Take one optimizer step from $\theta^{i,k,h-1}$ on

$$\mathcal{L}^{i,k,h}(\theta) = \frac{1}{B_{\text{self}}} \left[\sum_{l=1}^{b_{\text{IS}}} w^{i,(l)} \exp\left(\frac{\widehat{u}^i(a_0^{i,(l)}; \pi_{\theta^{-i,k}})}{\eta} \right) (1 - \bar{\alpha}_{t^{i,(l)}}) \left\| s_\theta(a_{t^{i,(l)}}^{i,(l)}, t^{i,(l)}) + \frac{\epsilon^{i,(l)}}{\sqrt{1 - \bar{\alpha}_{t^{i,(l)}}}} \right\|_2^2 \right.$$
$$\left. + \sum_{l=b_{\text{IS}}+1}^{B_{\text{self}}} \exp\left(\frac{\widehat{u}^i(a_0^{i,(l)}; \pi_{\theta^{-i,k}})}{\eta} \right) (1 - \bar{\alpha}_{t^{i,(l)}}) \left\| s_\theta(a_{t^{i,(l)}}^{i,(l)}, t^{i,(l)}) + \frac{\epsilon^{i,(l)}}{\sqrt{1 - \bar{\alpha}_{t^{i,(l)}}}} \right\|_2^2 \right],$$

obtaining $\theta^{i,k,h}$.

12: **end for**

13: Set $\theta^{i,k+1} \leftarrow \theta^{i,k,H}$.

14: Draw R actions from $\pi_{\theta^{i,k+1}}$ and apply PREPARESAMPLE with behavior parameters $\theta^{i,k+1}$; append the resulting entries to both buffers and retain the most recent N_{buf} opponent entries.

15: **end for**

16: **end for**

17: **return** $\Pi^{i,K} = \frac{1}{K} \sum_{k=1}^{K} \pi_{\theta^{i,k}}, \quad i \in \{1,2\}.$

and both firms have constant marginal cost c. Player i's total utility is

$$u^i(a^i, a^{-i}) = \sum_{j=1}^{d} \left(A - b(a_j^i + a_j^{-i}) - c\right) a_j^i. \tag{12}$$

We use $A = 10$, $b = 1$, $c = 1$, and $q_{\max} = (A - c)/b = 9$, with $d \in \{1, 30\}$ in the experiments.

In the implementation, the policy produces a normalized action $z^i \in [-1, 1]^d$, which is mapped to the feasible Cournot action by

$$a^i = \frac{z^i + 1}{2} q_{\max}.$$

During optimization, we use the scaled utility u^i/d so that its magnitude does not grow with the number of markets.

Analytical equilibrium. The separability and quadratic structure of the Cournot game allow its Nash equilibrium to be characterized exactly.

Proposition 3. *The Cournot game in* (12) *has a unique Nash equilibrium. It is pure, and for every market* $j = 1, \ldots, d$*,*

$$a_j^{1,\star} = a_j^{2,\star} = \frac{A - c}{3b}.$$

Under our parameters, $a_j^{1,\star} = a_j^{2,\star} = 3$ *for every market.*

Proof. Consider an arbitrary mixed strategy π^{-i} of player $-i$, and let

$$\mu_j^{-i} = \mathbb{E}_{a^{-i} \sim \pi^{-i}}[a_j^{-i}]$$

denote its mean production in market j. Since the payoff is separable across markets and linear in the opponent's production, the expected utility from a pure action a^i is

$$\mathbb{E}_{a^{-i} \sim \pi^{-i}} \left[u^i(a^i, a^{-i})\right] = \sum_{j=1}^{d} \left[(A - c - b\mu_j^{-i})a_j^i - b(a_j^i)^2\right].$$

For $b > 0$, this expression is strictly concave in a^i, so player i has a unique best response against any mixed opponent strategy. In each market,

$$\mathrm{BR}_j^i(\pi^{-i}) = \frac{A - c - b\mu_j^{-i}}{2b}.$$

Since

$$0 \le \mu_j^{-i} \le q_{\max} = \frac{A - c}{b},$$

this quantity lies in $[0, q_{\max}]$, so the action constraints do not alter the best response.

A player can randomize at a Nash equilibrium only over best responses. Since the best response is unique, every Nash equilibrium must therefore be pure. For each market j, the equilibrium quantities satisfy

$$2ba_j^{1,\star} + ba_j^{2,\star} = A - c, \qquad ba_j^{1,\star} + 2ba_j^{2,\star} = A - c.$$

Subtracting the two equations gives $a_j^{1,\star} = a_j^{2,\star}$, and substitution yields

$$a_j^{1,\star} = a_j^{2,\star} = \frac{A - c}{3b}.$$

Because the markets are independent, this holds for every $j = 1, \ldots, d$. With $A = 10$, $b = 1$, and $c = 1$, the equilibrium quantity is 3 in every market. □

D.3 Experimental Details of Forest Protection Game

D.3.1 Environment Details

We use the continuous forest security game introduced by Kamra et al. (2018) and subsequently adopted as the forest protection benchmark in DeepFP (Kamra et al., 2019). Our environment and parameter settings follow the latter. The game is played between a defender D and an adversary A in a circular forest containing a fixed set of tree locations $\mathcal{T} \subset \mathbb{R}^2$. The adversary controls n adversaries, while the defender controls m stationary guards.

Each adversary enters at the boundary of the forest, travels along a radial path toward the center, stops at a chosen location, and exits along the same path. Its stopping location therefore determines both its trajectory and its harvesting region. The adversary's action is

$$a_A = (\ell_1, \ldots, \ell_n) \in \mathcal{F}^n, \tag{13}$$

where $\mathcal{F}$ is the unit disk and ℓ_k is the stopping location of the adversary k. The defender selects the guard locations

$$a_D = (g_1, \ldots, g_m) \in \mathcal{F}^m. \tag{14}$$

where g_j is the location of guard j. Locations are represented using normalized polar coordinates (r, θ), so the adversary and defender action dimensions are $2n$ and $2m$, respectively.

An adversary is captured when its path passes within guard radius R_g of at least one defender resource. Let $P_k(\ell_k)$ denote the radial path followed by the adversary k. Its capture indicator is

$$C_k(a_D, a_A) = \mathbb{I}\left[\min_{j \in \{1,\ldots,m\}} \min_{p \in P_k(\ell_k)} \|p - g_j\|_2 \le R_g\right]. \tag{15}$$

An uncaptured adversary harvests trees lying within chopping radius R_ℓ of its stopping location. Define

$$H_k(\ell_k) = \{t \in \mathcal{T} : \|t - \ell_k\|_2 \le R_\ell\}. \tag{16}$$

Let $N_{\text{stolen}}(a_D, a_A)$ denote the number of trees successfully removed by the uncaptured adversaries, with overlapping harvest regions combined according to the game simulator. A captured adversary loses its harvest and incurs a penalty $r_{\text{pen}} > 0$. The adversary's reward is

$$u_A(a_D, a_A) = N_{\text{stolen}}(a_D, a_A) - r_{\text{pen}} \sum_{k=1}^{n} C_k(a_D, a_A), \tag{17}$$

and the defender receives the negative reward, $u_D(a_D, a_A) = -u_A(a_D, a_A)$. Following the DeepFP experimental setup (Kamra et al., 2019), we use

$$R_g = 0.1, \qquad R_\ell = 0.04, \qquad r_{\text{pen}} = 4.0. \tag{18}$$

The reward landscape is highly nonconvex and contains large flat regions. In particular, moving a guard without changing which trajectories it intercepts does not change the defender's reward. Heterogeneous tree densities also create spatially separated high-value regions, so effective mixed strategies may place probability mass on several distinct guard or adversary configurations. The game therefore tests both the expressiveness of the learned strategy distribution and its ability to improve when informative utility gradients are sparse.

Forest instances. We use four forest layouts, denoted F1–F4 throughout the paper. F1 and F2 contain 4,000 and 6,000 trees, respectively, and are evaluated with $m = n \in \{1, 3\}$. F3 and F4 are denser layouts containing 6,200 and 7,100 trees and are evaluated with $m = n = 5$. Thus, the six configurations reported in the main experiments are

$$(\mathrm{F1}, 1),\ (\mathrm{F1}, 3),\ (\mathrm{F2}, 1),\ (\mathrm{F2}, 3),\ (\mathrm{F3}, 5),\ (\mathrm{F4}, 5).$$

The interaction and payoff structure is held fixed across instances; only the spatial tree layout and number of resources change. This changes which attack locations are valuable and how many spatially separated regions must be represented by the equilibrium strategy.

Scaling the action space. Each guard and adversary is represented by a polar coordinate (r, θ), so the defender and attacker action spaces have dimensions $2m$ and $2n$, respectively. Latent actions in $[-1, 1]$ are mapped to $r \in [0, 1]$ and $\theta \in [0, 2\pi]$. Consequently, the headline experiments increase each player's action dimension from 2 at $m = n = 1$, to 6 at $m = n = 3$, and 10 at $m = n = 5$. Although resources of the same type are interchangeable in the payoff, we use the direct ordered representation and do not explicitly enforce permutation symmetry.

D.3.2 Implementation Details

Compared methods. We compare DiffNash with DeepFP (Kamra et al., 2019), DiffFP (Karthikeyan & Pant, 2025), and Neural OMD. DeepFP uses neural fictitious play with a learned attacker and geometric defender best response. DiffFP replaces the neural policy with a diffusion best response while retaining fictitious-play updates. Neural OMD uses the same OMD training loop as DiffNash but represents the policy with a squashed diagonal Gaussian, isolating the contribution of the diffusion policy. DiffNash combines a diffusion policy with OMD, opponent replay, and self-action importance sampling.

Training details. All headline methods are trained for 40,000 gradient updates per player. DiffNash and Neural OMD use 2,000 outer OMD iterations with 20 inner updates per iteration. DiffFP uses 100 fictitious-play iterations with 400 best-response updates per iteration. Unless otherwise noted, we run five seeds. OMD methods use batches of 256 player actions and 128 opponent replay samples. DiffNash maintains a FIFO opponent replay buffer of size 1,024 and draws 1% of each player batch from its full self-action history.

DiffNash and Neural OMD use Adam Kingma & Ba (2014) with learning rate 3×10^{-4}, decayed to 3×10^{-5} during training. The diffusion policies use $T = 20$ denoising steps. All experiments are run on CPU using 8 cores and 16 GB of memory.

Evaluation. We report defender exploitability, defined as the best attacker reward against the learned time-averaged defender, together with the two sided Nash gap. Best responses are approximated using evolutionary search on the reference simulator. The search uses a population of 50 candidates for 20 iterations, retaining the top 10 and generating four mutations per survivor. Each candidate is evaluated against 100 sampled opponent actions, followed by a final evaluation against 1,000 samples. Self-play utility for the Nash gap is estimated from 2,000 sampled action pairs.

We evaluate only the final checkpoint and use the complete time-averaged strategy accumulated during training. Reported uncertainty is the 95% Student-t confidence interval across seeds.

Hyperparameters. DiffNash uses a DACER-style denoiser with three 256-unit hidden layers and Mish activations. The diffusion-time embedding is a 16-dimensional sinusoidal encoding followed by a two-layer MLP. We use OMD step size $\eta = 1$, normalize utility-derived energies within each batch, clip their logits to $[-3, 3]$, and renormalize the resulting weights to mean one. Opponent importance ratios are upper-clipped at 10. Self-action replay constitutes 1% of the training batch and is activated once the standard deviation of the current action samples falls below 10% of its initial value.

D.4 Experimental Details of Continuous Colonel Blotto

D.4.1 Environment Details

We follow the continuous Colonel Blotto formulation of Adam et al. (2021). Two players simultaneously allocate a unit budget across n battlefields. Both players have the action space

$$\mathcal{A}^1 = \mathcal{A}^2 = \mathcal{A} = \left\{ a = (a_1, \ldots, a_n) \in \mathbb{R}_+^n : \sum_{j=1}^{n} a_j = 1 \right\}. \tag{19}$$

Let $a^1, a^2 \in \mathcal{A}$ denote the allocations selected by players 1 and 2, respectively. Each battlefield j has a positive weight ω_j. Player 1's utility is

$$u^1(a^1, a^2) = \sum_{j=1}^{n} \omega_j \, \ell_c(a_j^1 - a_j^2), \tag{20}$$

and player 2 receives

$$u^2(a^1, a^2) = -u^1(a^1, a^2).$$

The battlefield contest function is

$$\ell_c(z) = \begin{cases} -1, & z \le -c, \\ z/c, & -c < z < c, \\ 1, & z \ge c, \end{cases} \tag{21}$$

where $c > 0$ controls the allocation advantage required to obtain the full value of a battlefield. Thus, player 1 receives ω_j when it allocates at least c more than player 2 to battlefield j, receives $-\omega_j$ when it allocates at least c less, and receives a payoff that varies linearly with the allocation difference inside this margin. As $c \to 0$, ℓ_c approaches the sign function used in the classical continuous Colonel Blotto game.

Standard six-battlefield instance. Our main Blotto experiment uses the symmetric six-battlefield setting

$$n = 6, \qquad \omega_j = 1 \text{ for all } j, \qquad c = \frac{1}{16}. \tag{22}$$

The game is symmetric and zero-sum, so its equilibrium value is zero. Because the budget constraint couples all six coordinates, increasing the allocation to one battlefield reduces the allocation available to the others. Equilibrium play therefore requires coordinated allocations rather than independent decisions at individual battlefields. This makes the game a useful test of whether a policy class can represent a broad mixed strategy.

Radial-band instances. We construct two synthetic continuous allocation games with $n \in \{6, 10\}$ battlefields. Each player chooses an allocation $a \in \Delta^{n-1}$. Let

$$\bar{a} = \frac{1}{n}\mathbf{1}, \qquad v = e_1 - \bar{a}, \qquad m(r) = \bar{a} + rv,$$

so that $m(r)$ traces the radial line from the uniform allocation toward the first simplex vertex. We define a tapered corridor around this line with width

$$w(r) = \min\{0.06,\ 0.20(1 - r)\}.$$

For $n = 6$, the corridor spans $r \in [0.12, 0.995]$, whereas for $n = 10$, it spans $r \in [0.15, 0.995]$. Writing

$$r_{\min} = \begin{cases} 0.12, & n = 6, \\ 0.15, & n = 10, \end{cases}$$

we define

$$\hat{r}(a) = \min\left\{0.995,\ \max\left\{r_{\min},\ \frac{\langle a - \bar{a}, v\rangle}{\|v\|_2^2}\right\}\right\},$$

and the corridor potential

$$V(a) = -15\left[\|a - m(\hat{r}(a))\|_2 - w(\hat{r}(a))\right]_+ .$$

Thus, $V(a) = 0$ for actions inside the prescribed corridor and $V(a) < 0$ for actions sufficiently far from it.

Radial-band equilibrium. For each $n \in \{6, 10\}$, define a distribution $\mu_n^\star$ supported on the radial corridor as follows. First sample

$$R \sim \begin{cases} \mathrm{Unif}[r_{\min}, 0.80), & \text{with probability } 0.6, \\ \mathrm{Unif}[0.80, 0.995], & \text{with probability } 0.4. \end{cases}$$

Then set

$$A = m(R) + \xi,$$

where

$$\mathbf{1}^\top \xi = 0, \qquad v^\top \xi = 0, \qquad \|\xi\|_2 \le w(R), \qquad m(R) + \xi \in \Delta^{n-1}.$$

Such points always exist, since $\xi = 0$ is feasible for every $R \in [r_{\min}, 0.995]$. Thus, $\mu_n^\star$ is a well-defined probability distribution supported entirely on the zero-potential corridor, with 60% of its mass on the interior radial region and 40% near the extreme allocation.

We next construct the payoff so that this radial distribution is an exact equilibrium. Let

$$k(a^1, a^2) = \exp\left(-\frac{\|a^1 - a^2\|_2^2}{2\sigma^2}\right), \qquad \sigma = 0.08,$$

and define

$$h_n(a) = \mathbb{E}_{A\sim\mu_n^\star}[k(a, A)], \qquad \kappa_n = \mathbb{E}_{A,A'\sim\mu_n^\star}[k(A, A')].$$

The centered kernel is

$$\bar{k}_n(a^1, a^2) = k(a^1, a^2) - h_n(a^1) - h_n(a^2) + \kappa_n.$$

We define the row player's utility as

$$u_n^1(a^1, a^2) = V(a^1) - V(a^2) + \bar{k}_n(a^1, a^2),$$

with

$$u_n^2(a^1, a^2) = -u_n^1(a^1, a^2).$$

To see that a radial equilibrium exists, observe that for every $a^1, a^2 \in \Delta^{n-1}$,

$$\mathbb{E}_{A^2\sim\mu_n^\star}\left[\bar{k}_n(a^1, A^2)\right] = 0, \qquad \mathbb{E}_{A^1\sim\mu_n^\star}\left[\bar{k}_n(A^1, a^2)\right] = 0,$$

by the definition of the centered kernel. Since $\mu_n^\star$ is supported inside the corridor, $V(A) = 0$ almost surely for $A \sim \mu_n^\star$, while $V(a) \le 0$ for every feasible action. Therefore, for any pure deviations a^1 and a^2,

$$U_n^1(a^1, \mu_n^\star) = V(a^1) \le 0,$$

and

$$U_n^1(\mu_n^\star, a^2) = -V(a^2) \ge 0.$$

Moreover,

$$U_n^1(\mu_n^\star, \mu_n^\star) = 0.$$

Thus,

$$U_n^1(a^1, \mu_n^\star) \le U_n^1(\mu_n^\star, \mu_n^\star) \le U_n^1(\mu_n^\star, a^2)$$

for all $a^1, a^2 \in \Delta^{n-1}$. Hence $(\mu_n^\star, \mu_n^\star)$ is an exact Nash equilibrium with value zero for both $n = 6$ and $n = 10$.

In implementation, we replace $\mu_n^\star$ by a fixed empirical distribution and compute all kernel-centering terms from the same bank. The same argument then applies exactly to the resulting finite game.

Visualization. For visualization, six-dimensional allocations are projected onto a regular hexagon using barycentric coordinates. Let $b_1, \dots, b_6 \in \mathbb{R}^2$ denote the hexagon vertices. An allocation a is plotted at

$$p(a) = \sum_{j=1}^{6} a_j b_j. \tag{23}$$

Under this projection, the uniform allocation appears at the hexagon center. Allocating the entire budget to battlefield j would map directly to vertex b_j. The same projection is used for both the standard Blotto and radial-band games, which allows their learned strategy distributions to be compared directly.

D.4.2 Implementation Details

Reference solutions. For the symmetric Colonel Blotto instances, we compute reference equilibria with the grid-based double-oracle (DO) solver proposed in Adam et al. (2021). For both $n = 6$ and $n = 10$, we discretize allocations with grid spacing $1/256$ and run DO to tolerance 10^{-6}, using an exact knapsack dynamic program to compute best responses over the discretized action space. For $n = 6$, DO terminates after 21 iterations with a 42-action support and Nash gap 4.31×10^{-14}; for $n = 10$, it terminates after 12 iterations with a 50-action support and gap 1.33×10^{-14}. Both instances use $c = 1/16$. This solver relies on a game-specific best-response oracle: discretization makes exact best responses tractable for these Blotto instances, but such an oracle is generally unavailable in high-dimensional continuous-action games, where best-response optimization may itself be difficult and uniform discretization scales poorly with dimension. We therefore use DO only to construct reference equilibria for these structured symmetric instances, rather than as a general-purpose baseline.

For the radial-band instances, the reference equilibrium is known by construction. We instantiate it using a fixed empirical distribution $\widehat{\mu}_n^\star$ with $K = 1200$ anchors, consisting of 720 samples from the interior portion of the corridor and 480 from the near-extreme portion. The anchor bank is sampled once for each game instance and then frozen, and all kernel-centering terms are computed from the same bank. Thus, $\widehat{\mu}_n^\star$ is an exact equilibrium of the implemented finite-anchor game. We use $r_{\min} = 0.12$ for $n = 6$ and $r_{\min} = 0.15$ for $n = 10$, with $r_{\text{split}} = 0.80$ and $r_{\max} = 0.995$. The corresponding reference Nash gap is exactly zero. We additionally report the finite-sample evaluation floors, 0.014213 for $n = 6$ and 0.024741 for $n = 10$.

Compared Methods. We compare DiffNash against **DeepFP**, **DiffFP**, and **Neural OMD**. For DeepFP, we use the original implementation provided by the authors; their codebase is not publicly released. For DiffFP, no implementation is available, so we reimplement the method from the paper, using diffusion policies within the fictitious-play update and its Q-guided action refinement procedure.

We additionally introduce Neural OMD as a controlled baseline to separate the effect of the OMD update from the expressive diffusion used by DiffNash. Neural OMD uses the same OMD training structure and optimization budget as DiffNash, but replaces the diffusion policy with a squashed diagonal-Gaussian policy parameterized by a two-layer MLP with hidden dimensions $(256, 256)$ and `tanh` activations. Because this policy admits a tractable density, the mirror-descent regularizer is implemented using the analytic KL divergence between consecutive policies, with coefficient 1.0. Neural OMD is trained on-policy using fresh samples at each outer iteration and doe not use a replay buffer for importance weighting.

For a given seed, all methods are evaluated on the same game instance and frozen anchor bank. The target distribution, kernel centering terms, and payoff parameters are therefore identical across methods.

Training details. All four experiments use five random seeds 0-4.

DiffNash and Neural OMD each use 2000 outer iterations with 20 gradient updates per iteration, for a total of $40,000$ updates per player. Both use Adam with a learning rate annealed from 3×10^{-4} to 3×10^{-5}, a policy batch size of 256, and 128 opponent samples. DiffNash parameterizes each policy with a 20-step diffusion model with three hidden layers of width 256 and a linear noise schedule scaled by 0.8. Its replay buffer if FIFO with capacity 1024, with 32 new samples added per outer iteration. Opponent samples are corrected using importance weights capped at 10. For own-policy correction, 3 of the 256 samples in each batch are importance weighted, corresponding to approximately 1.17% of the batch.

DiffFP uses a 20-step diffusion model with hidden dimensions $(256, 256)$ and a 32-dimensional time embedding. We train it for 100 fictitious-play iterations, with 400 best-response updates per iteration, again giving $40{,}000$ gradient updates. It uses Adam (Kingma & Ba, 2014) with learning rate 3×10^{-4}, gradient-norm clipping at 2.0, training batches of 256, and rollout batches of 32. Its replay buffer has capacity $200{,}000$, with a pool of 100 historical policies, and the best-response policy is reinitialized at each fictitious-play iteration. DiffFP does not use importance sampling.

DeepFP uses the authors' TensorFlow 1 implementation, with a two-layer $(128, 128)$ policy network followed by a softmax output. We use Adam with learning rate 10^{-3} and train for $400{,}000$ games, applying a policy update every 10 games.

Evaluation We measure convergence using the two-player zero-sum Nash gap

$$G(p, q) = \max_x U(x, q) - \min_y U(p, y),$$

which is the sum of the two players' unilateral improvement incentives and is zero at equilibrium.

For symmetric Blotto, we follow the double-oracle evaluation of Adam et al. (2021), computing best responses exactly on a discretized simplex with a knapsack dynamic program. Reference solutions use grid spacing $1/256$, while final Nash gaps use a finer $1/1024$ grid and $20{,}000$ policy samples per player. For the radial-band games, we approximate each best response using a pool of approximately $11{,}400$ candidates followed by 400 gradient-refinement steps on the best 64, and estimate payoffs using $8{,}000$ samples per player. We additionally report recovery of the known radial equilibrium, including near-extreme mass (0.40 at equilibrium) and active versus inactive radial mass. All metrics are aggregated over five seeds using the same evaluator for every method within each game.

E MISSING PROOFS

E.1 PROOF OF PROPOSITION 1

Proposition 1 (GWFP and OMD Loss). *By choosing appropriate $g_{\mathrm{OMD}}(\mathbf{a}_t)$ and $g_{\mathrm{GWFP}}(\mathbf{a}_t)$ and substituting them into Equation (6), we can equivalently write their loss functions as*

$$\mathcal{L}_{\mathrm{OMD}}(\theta; t) = \mathbb{E}_{\mathbf{a}_0 \sim \pi^{i,k}} \Bigg[\underbrace{\mathbb{E}_{\mathbf{a}_t \sim q_{t|0}(\cdot|\mathbf{a}_0)} \left[\exp\left(\frac{1}{\eta} u^i(\mathbf{a}_0; \pi^{-i,k}) \right) \left\| s_{\theta^i}(\mathbf{a}_t, t) - \nabla_{\mathbf{a}_t} \log q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) \right\|_2^2 \right]}_{G_{\theta^i}(\mathbf{a}_0, t)} \Bigg]. \tag{24}$$

$$\textit{and} \quad \mathcal{L}_{\mathrm{GWFP}}(\theta; t) = \mathbb{E}_{\substack{\mathbf{a}_t \sim h_t, \\ \tilde{\mathbf{a}}_0 \sim \phi_{0|t}}} \left[\exp\left(\frac{1}{\eta^k} u^i(\tilde{\mathbf{a}}_0; \Pi^{-i,k}) \right) \left\| s_{\theta^i}(\mathbf{a}_t, t) - \nabla_{\mathbf{a}_t} \log \phi_{0|t}(\tilde{\mathbf{a}}_0 \mid \mathbf{a}_t) \right\|_2^2 \right], \tag{25}$$

where h_t is any distribution with full support and $\phi_{0|t}(\tilde{\mathbf{a}}_0 \mid \mathbf{a}_t) := \mathcal{N}\left(\tilde{\mathbf{a}}_0; \frac{\mathbf{a}_t}{\sqrt{\bar{\alpha}_t}}, \frac{1-\bar{\alpha}_t}{\bar{\alpha}_t} I\right)$. For every $t \in (0, T]$, the minimizers of Equations (7) and (25) over s are the score functions of $\pi^{i,k+1}_{t,\mathrm{GWFP}}(\mathbf{a}_t)$ and $\pi^{i,k+1}_{t,\mathrm{OMD}}(\mathbf{a}_t)$, respectively.

Proof. Fix player i, equilibrium-solving iteration k, and diffusion time $t \in (0, T]$. Recall that

$$q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) = \mathcal{N}\left(\mathbf{a}_t; \sqrt{\bar{\alpha}_t}\mathbf{a}_0, (1 - \bar{\alpha}_t) I\right).$$

For either solver $\in \{\mathrm{GWFP}, \mathrm{OMD}\}$, define

$$\pi^{i,k+1}_{t,\mathrm{solver}}(\mathbf{a}_t) = \int \pi^{i,k+1}_{\mathrm{solver}}(\mathbf{a}_0) q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0)\, d\mathbf{a}_0.$$

We first recall the denoising score-matching identity. For any target density ρ_0 and its noised marginal

$$\rho_t(\mathbf{a}_t) = \int \rho_0(\mathbf{a}_0) q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0)\, d\mathbf{a}_0,$$

differentiating under the integral gives

$$\begin{aligned} \nabla_{\mathbf{a}_t} \log \rho_t(\mathbf{a}_t) &= \frac{\int \rho_0(\mathbf{a}_0) q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) \nabla_{\mathbf{a}_t} \log q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0)\, d\mathbf{a}_0}{\rho_t(\mathbf{a}_t)} \\ &= \mathbb{E}_{\mathbf{a}_0 \sim \rho_{0|t}(\cdot|\mathbf{a}_t)} \left[\nabla_{\mathbf{a}_t} \log q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) \right]. \end{aligned} \tag{26}$$

For any nonnegative function $w_t(\mathbf{a}_t)$ for which the expectations are finite,

$$\begin{aligned} &\int w_t(\mathbf{a}_t) \rho_t(\mathbf{a}_t) \left\| s_{\theta^i}(\mathbf{a}_t, t) - \nabla_{\mathbf{a}_t} \log \rho_t(\mathbf{a}_t) \right\|_2^2 d\mathbf{a}_t \\ &\quad = \iint w_t(\mathbf{a}_t) \rho_0(\mathbf{a}_0) q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) \left\| s_{\theta^i}(\mathbf{a}_t, t) - \nabla_{\mathbf{a}_t} \log q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) \right\|_2^2 d\mathbf{a}_0\, d\mathbf{a}_t + C_t, \end{aligned} \tag{27}$$

where C_t does not depend on θ^i. This follows directly from Equation (26) by expanding the squared norms.

OMD. From Equation (4), the OMD update is

$$\pi^{i,k+1}_{\mathrm{OMD}}(\mathbf{a}_0) = \frac{1}{Z^{i,k}_{\mathrm{OMD}}} \pi^{i,k}(\mathbf{a}_0) \exp\left(\frac{1}{\eta} u^i(\mathbf{a}_0; \pi^{-i,k}) \right). \tag{28}$$

Choose

$$g_{\mathrm{OMD}}(\mathbf{a}_t) = Z^{i,k}_{\mathrm{OMD}} \pi^{i,k+1}_{t,\mathrm{OMD}}(\mathbf{a}_t). \tag{29}$$

Substituting Equation (29) into Equation (6) and applying Equation (27) gives

$$\begin{aligned}\mathcal{L}^{g_{\mathrm{OMD}}}(\theta^i; t) &= \iint \pi^{i,k}(\mathbf{a}_0) q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) \exp\left(\frac{1}{\eta} u^i(\mathbf{a}_0; \pi^{-i,k})\right) \\ &\qquad \times \left\| s_{\theta^i}(\mathbf{a}_t, t) - \nabla_{\mathbf{a}_t} \log q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) \right\|_2^2 d\mathbf{a}_t \, d\mathbf{a}_0 + C_t^{\mathrm{OMD}} \\ &= \mathbb{E}_{\mathbf{a}_0 \sim \pi^{i,k}} \left[\mathbb{E}_{\mathbf{a}_t \sim q_{t|0}(\cdot|\mathbf{a}_0)} \left[\exp\left(\frac{1}{\eta} u^i(\mathbf{a}_0; \pi^{-i,k})\right) \right.\right. \\ &\qquad \left.\left. \times \left\| s_{\theta^i}(\mathbf{a}_t, t) - \nabla_{\mathbf{a}_t} \log q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) \right\|_2^2 \right] \right] + C_t^{\mathrm{OMD}}. \end{aligned} \tag{30}$$

which is Equation (7) up to an additive constant independent of θ^i.

GWFP. From Equation (5), the GWFP target is

$$\pi^{i,k+1}_{\mathrm{GWFP}}(\mathbf{a}_0) = \frac{1}{Z^{i,k}_{\mathrm{GWFP}}} \exp\left(\frac{1}{\eta^k} u^i(\mathbf{a}_0; \Pi^{-i,k})\right), \tag{31}$$

where $Z^{i,k}_{\mathrm{GWFP}}$ is the corresponding normalization constant. Choose

$$g_{\mathrm{GWFP}}(\mathbf{a}_t) = \bar{\alpha}_t^{d_i/2} Z^{i,k}_{\mathrm{GWFP}} h_t(\mathbf{a}_t) \pi^{i,k+1}_{t,\mathrm{GWFP}}(\mathbf{a}_t). \tag{32}$$

Substituting Equation (32) into Equation (6) and applying Equation (27) gives

$$\begin{aligned}\mathcal{L}^{g_{\mathrm{GWFP}}}(\theta^i; t) = \bar{\alpha}_t^{d_i/2} \iint h_t(\mathbf{a}_t) \exp\left(\frac{1}{\eta^k} u^i(\mathbf{a}_0; \Pi^{-i,k})\right) q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) \\ \times \left\| s_{\theta^i}(\mathbf{a}_t, t) - \nabla_{\mathbf{a}_t} \log q_{t|0}(\mathbf{a}_T \mid \mathbf{a}_0) \right\|_2^2 d\mathbf{a}_0 \, d\mathbf{a}_t + C_t^{\mathrm{GWFP}}. \end{aligned} \tag{33}$$

Now define

$$\phi_{0|t}(\mathbf{a}_0 \mid \mathbf{a}_t) = \mathcal{N}\left(\mathbf{a}_0; \frac{\mathbf{a}_t}{\sqrt{\bar{\alpha}_t}}, \frac{1-\bar{\alpha}_t}{\bar{\alpha}_t} I\right).$$

Comparing the Gaussian densities gives

$$\phi_{0|t}(\mathbf{a}_0 \mid \mathbf{a}_t) = \bar{\alpha}_t^{d_i/2} q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0), \tag{34}$$

and therefore

$$\nabla_{\mathbf{a}_t} \log \phi_{0|t}(\mathbf{a}_0 \mid \mathbf{a}_t) = \nabla_{\mathbf{a}_t} \log q_{t|0}(\mathbf{a}_t \mid \mathbf{a}_0) = -\frac{\mathbf{a}_t - \sqrt{\bar{\alpha}_t}\mathbf{a}_0}{1-\bar{\alpha}_t}. \tag{35}$$

Applying Equations (34) and (35) to Equation (33) yields

$$\begin{aligned}\mathcal{L}^{g_{\mathrm{GWFP}}}(\theta^i; t) = \mathbb{E}_{\substack{\mathbf{a}_t \sim h_t \\ \tilde{\mathbf{a}}_0 \sim \phi_{0|t}(\cdot|\mathbf{a}_t)}} \left[\exp\left(\frac{1}{\eta^k} u^i(\tilde{\mathbf{a}}_0; \Pi^{-i,k})\right) \right. \\ \left. \times \left\| s_{\theta^i}(\mathbf{a}_t, t) - \nabla_{\mathbf{a}_t} \log \phi_{0|t}(\tilde{\mathbf{a}}_0 \mid \mathbf{a}_t) \right\|_2^2 \right] + C_t^{\mathrm{GWFP}}, \end{aligned} \tag{36}$$

which is the GWFP loss up to an additive constant independent of θ^i.

□

Finally, Gaussian corruption implies $\pi^{i,k+1}_{t,\mathrm{GWFP}}(\mathbf{a}_t) > 0$ and $\pi^{i,k+1}_{t,\mathrm{OMD}}(\mathbf{a}_t) > 0$ for $t > 0$. Because h_t is strictly positive on its support, both g_{GWFP} and g_{OMD} are positive almost everywhere. Equation (6) is therefore globally minimized precisely when

$$s_{\theta^i}(\mathbf{a}_t, t) = \nabla_{\mathbf{a}_t} \log \pi^{i,k+1}_{t,\mathrm{solver}}(\mathbf{a}_t)$$

almost everywhere. This proves the result for both solver $\in \{\mathrm{GWFP}, \mathrm{OMD}\}$.

E.2 Proof of Proposition 2

Proposition 2. *Suppose π^k and $\pi^{k'}$ share the same forward-noising schedule $\{\beta_t\}_{t=1}^T$. Denote $D(\theta; \mathbf{a}_0, t, \epsilon) = C_t||s_\theta(\mathbf{a}_t, t) + \frac{\epsilon}{\sqrt{1-\bar\alpha_t}}||_2^2$, where C_t a constant and $\mathbf{a}_t = \sqrt{\bar\alpha_t}\mathbf{a}_0 + \sqrt{1-\bar\alpha_t}\epsilon$. If $t \sim \mathrm{Unif}\{1, \dots, T\}$ and $\epsilon \sim \mathcal{N}(0, I)$, then $\widehat{\Delta}^{k',k} := T\big(D(\theta^k; \mathbf{a}_0, t, \epsilon) - D(\theta^{k'}; \mathbf{a}_0, t, \epsilon)\big)$ is ELBO difference's unbiased estimator: $\mathbb{E}_{t,\epsilon}[\widehat{\Delta}^{k',k}] = \mathrm{ELBO}^{k'}(\mathbf{a}_0) - \mathrm{ELBO}^{k}(\mathbf{a}_0)$.*

Proof. Fix $\mathbf{a}_0$. By the ELBO decomposition in Ho et al. (2020),

$$\begin{aligned}\mathrm{ELBO}_\theta(\mathbf{a}_0) = \mathbb{E}_{q(\mathbf{a}_1|\mathbf{a}_0)}\big[\log p_\theta(\mathbf{a}_0 \mid \mathbf{a}_1)\big] \\ - \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{a}_t|\mathbf{a}_0)}\left[D_{\mathrm{KL}}\left(q(\mathbf{a}_{t-1} \mid \mathbf{a}_t, \mathbf{a}_0) \,\|\, p_\theta(\mathbf{a}_{t-1} \mid \mathbf{a}_t)\right)\right] \\ - \underbrace{D_{\mathrm{KL}}(q(a_T \mid a_0) \,\|\, p(a_T))}_{\text{Terminal prior}}.\end{aligned} \tag{37}$$

Under the forward process,

$$\mathbf{a}_t = \sqrt{\bar\alpha_t}\,\mathbf{a}_0 + \sqrt{1-\bar\alpha_t}\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I).$$

For $t \geq 2$, the forward posterior is Gaussian with mean

$$\widetilde{\mu}_t(\mathbf{a}_t, \mathbf{a}_0) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{a}_t - \frac{\beta_t}{\sqrt{1-\bar\alpha_t}}\,\epsilon\right).$$

whereas the reverse model has mean

$$\mu_\theta(\mathbf{a}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{a}_t + \beta_t s_\theta(\mathbf{a}_t, t)\right).$$

Hence

$$\widetilde{\mu}_t(\mathbf{a}_t, \mathbf{a}_0) - \mu_\theta(\mathbf{a}_t, t) = -\frac{\beta_t}{\sqrt{\alpha_t}}\left(s_\theta(\mathbf{a}_t, t) + \frac{\epsilon}{\sqrt{1-\bar\alpha_t}}\right).$$

Since the covariance of $p_\theta(\mathbf{a}_{t-1} \mid \mathbf{a}_t)$ is the fixed $\sigma_t^2 I$, the Gaussian KL can be written as

$$D_{\mathrm{KL}}\left(q(\mathbf{a}_{t-1} \mid \mathbf{a}_t, \mathbf{a}_0) \,\|\, p_\theta(\mathbf{a}_{t-1} \mid \mathbf{a}_t)\right) = \mathrm{const}_t(\mathbf{a}_0) + D(\theta; \mathbf{a}_0, t, \epsilon),$$

where $\mathrm{const}_t(\mathbf{a}_0)$ is independent of θ.

For $t = 1$, since $\bar\alpha_1 = \alpha_1$ and $1 - \bar\alpha_1 = \beta_1$, we have

$$\mathbf{a}_0 - \mu_\theta(\mathbf{a}_1, 1) = -\frac{\beta_1}{\sqrt{\alpha_1}}\left(s_\theta(\mathbf{a}_1, 1) + \frac{\epsilon}{\sqrt{1-\bar\alpha_1}}\right).$$

Therefore, since $p_\theta(\mathbf{a}_0 \mid \mathbf{a}_1) = \mathcal{N}(\mu_\theta(\mathbf{a}_1, 1), \sigma_1^2 I)$,

$$\log p_\theta(\mathbf{a}_0 \mid \mathbf{a}_1) = \mathrm{const}_1(\mathbf{a}_0) - D(\theta; \mathbf{a}_0, 1, \epsilon),$$

with $\mathrm{const}_1(\mathbf{a}_0)$ independent of θ.

The terminal-prior term in (37) is also independent of θ. Thus,

$$\mathrm{ELBO}_\theta(\mathbf{a}_0) = \mathrm{const}(\mathbf{a}_0) - \sum_{t=1}^{T} \mathbb{E}_\epsilon\left[D(\theta; \mathbf{a}_0, t, \epsilon)\right],$$

where $\mathrm{const}(\mathbf{a}_0)$ is independent of θ.

Because the two policy iterates share the same forward schedule, reverse-variance schedule, and terminal prior, the same $\mathrm{const}(\mathbf{a}_0)$ applies to both. Consequently,

$$\mathrm{ELBO}^{k'}(\mathbf{a}_0) - \mathrm{ELBO}^{k}(\mathbf{a}_0) = \sum_{t=1}^{T} \mathbb{E}_{\epsilon}\left[D(\theta^{k}; \mathbf{a}_0, t, \epsilon) - D(\theta^{k'}; \mathbf{a}_0, t, \epsilon)\right]. \tag{38}$$

Finally, because $t \sim \mathrm{Unif}\{1, \ldots, T\}$,

$$\begin{aligned}
\mathbb{E}_{t,\epsilon}[\widehat{\Delta}^{k',k}] &= T\,\mathbb{E}_{t,\epsilon}\left[D(\theta^{k}; \mathbf{a}_0, t, \epsilon) - D(\theta^{k'}; \mathbf{a}_0, t, \epsilon)\right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\epsilon}\left[D(\theta^{k}; \mathbf{a}_0, t, \epsilon) - D(\theta^{k'}; \mathbf{a}_0, t, \epsilon)\right] \\
&= \mathrm{ELBO}^{k'}(\mathbf{a}_0) - \mathrm{ELBO}^{k}(\mathbf{a}_0).
\end{aligned}$$

Thus, $\widehat{\Delta}^{k',k}$ is an unbiased estimator of the ELBO difference. □

F Additional Experimental Results

F.1 Additional Forest Game Visualizations

Figure 7 provides the complete strategy visualization for all six forest configurations reported in Table 1. Although the payoff rule is unchanged, each tree layout induces a different spatial equilibrium.

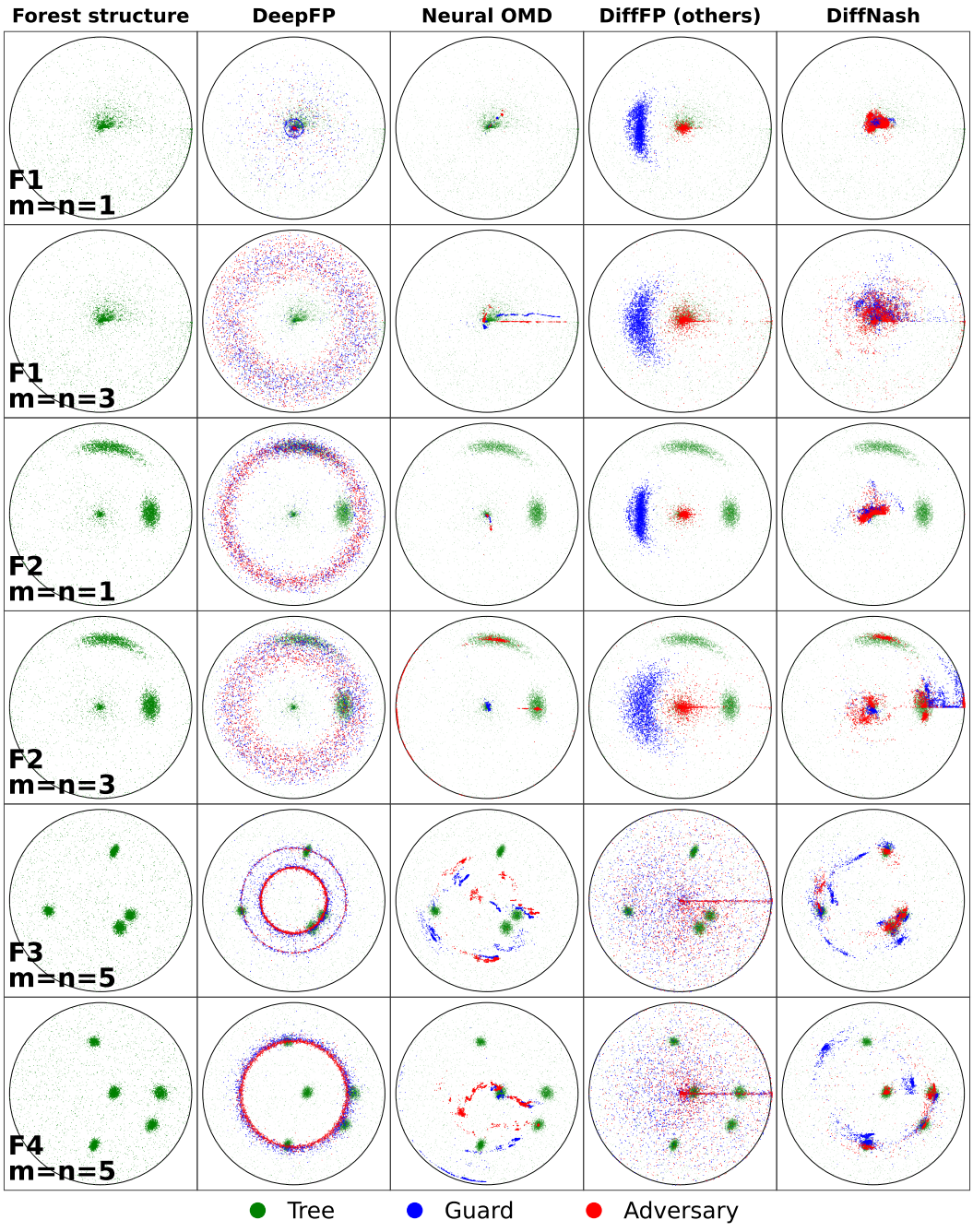

Figure 7: **Learned strategies across all forest security games.** Green points denote trees, blue points defender actions, and red points attacker actions. Rows vary both the forest geometry and the number of resources $m = n$. DiffNash obtains the lowest mean Nash gap in all six configurations; Figure 4 highlights two representative multi-resource cases in the main text.

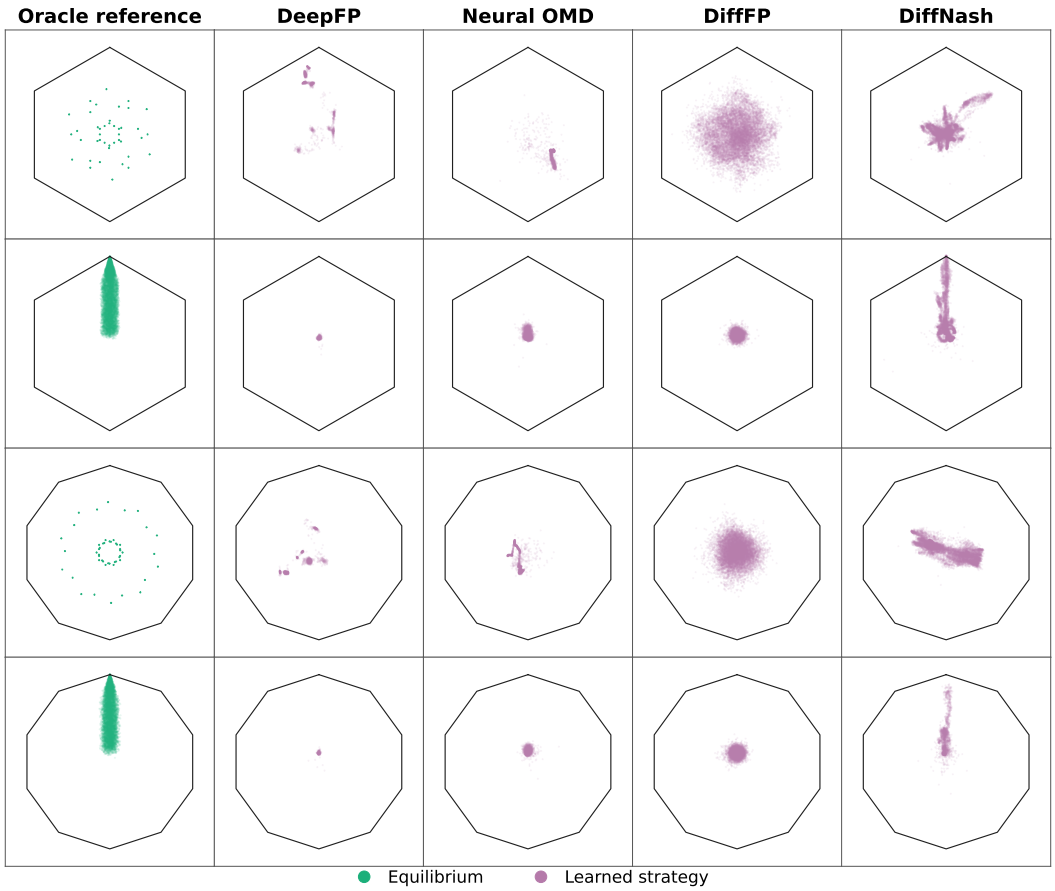

Figure 8: **Continuous Colonel Blotto strategy distributions.** Representative learned strategies projected into the polygonal action-space visualization. Columns show the oracle reference and each method; rows correspond to symmetric and radial-band Blotto for $n = 6$ and $n = 10$, respectively. Green denotes equilibrium samples and purple the learned strategy.

F.2 10-DIMENSIONAL CONTINUOUS COLONEL BLOTTO

Figure 8 shows the learned strategy distributions for the $n = 6$ and $n = 10$ Blotto experiments reported in Table 2.

F.3 GENERAL-SUM FOREST PROTECTION

We also consider a general-sum variant of the F2 forest protection game found in Kamra et al. (2019). We retain the original F2 geometry and interaction mechanics, but assign each tree separate ecological and economic values. This creates different objectives for the defender and poacher without changing the continuous action space.

Let v_t^D and v_t^P denote the ecological and economic values of tree t, respectively. For each adversary logger l, let $a_l = 1$ indicate that it is captured by at least one guard and let $s_{t,l}$ denote its share of tree t when multiple adversary target the same tree. The payoffs are

$$u^D = \sum_l \left[-(1 - a_l) \sum_t s_{t,l} v_t^D + 4a_l \right], \tag{39}$$

$$u^P = \sum_l \left[(1 - a_l) \sum_t s_{t,l} v_t^P - 4a_l \right]. \tag{40}$$

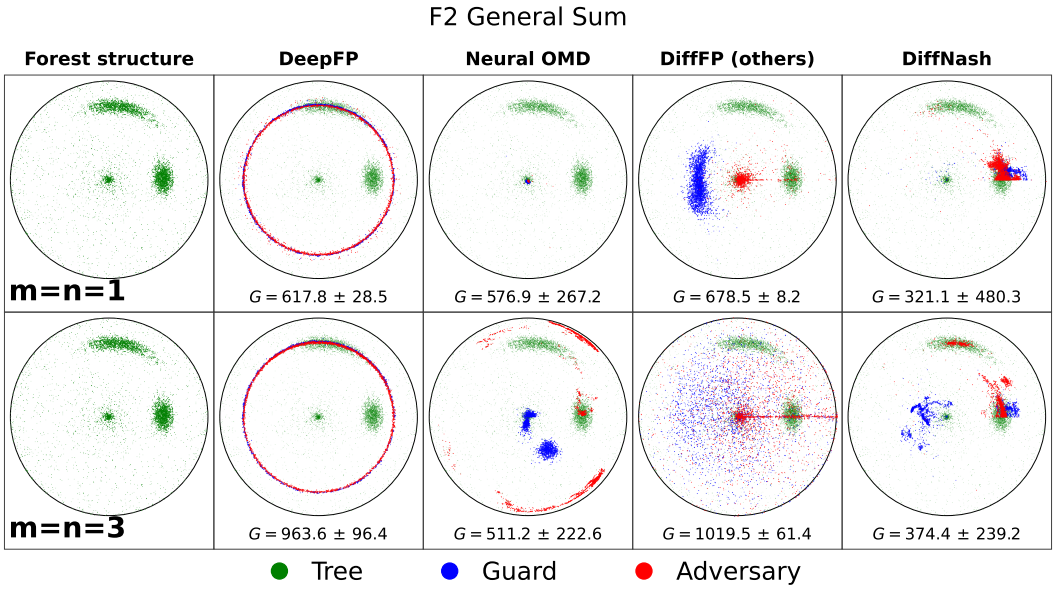

Figure 9: **General-sum Forest Game.** Learned strategies on the general-sum F2 forest game for $m = n \in \{1, 3\}$. Metrics are mean $\pm$ 95% confidence intervals over three seeds.

where a_l indicates whether adversary l is captured and $s_{(t,l}$ is its share of tree t. The original zero-sum game is recovered when $v_t^D = v_t^P = 1$.

We construct the two value fields to favor different parts of the forest. For a tree t, let r_t denote its radial distance from the center and let k_t be the number of other trees within Euclidean distance 0.05. We define

$$\tilde{v}_t^D = \frac{\exp(-r_t/0.3)}{1 + k_t}, \qquad \tilde{v}_t^P = \exp\left(-\frac{1 - r_t}{0.3}\right)(1 + k_t). \tag{41}$$

and independently rescale and clip each field to $[0.25, 4]$ while preserving mean 1, so that the overall payoff scale is comparable to the original game. Thus, the defender places greater value on isolated interior trees, while the poacher focuses more on dense patches near the boundary. On F2, the two value fields have correlation -0.52.

Figure 9 compares DeepFP, Neural OMD, DiffFP, and DiffNash for $m = n \in \{1, 3\}$. DiffNash attains the lowest mean Nash gap, while DeepFP and DiffFP remain more exploitable. Neural OMD performs better than the FP-based methods, but exhibits higher variance than DiffNash. The visualized seed also shows qualitatively different spatial strategies across methods.

F.4 Importance-sampling Forest ablation

We use F2, where $m = n = 3$, to isolate the roles of the two importance-sampling corrections in DiffNash. In particular, we separate the correction applied to replayed samples from the player's own past policies (*own IS*) from the correction used when evaluating samples from past opponent policies (*opponent IS*). Figure 10 shows that the two corrections have different effects. Removing own IS substantially degrades performance, increasing the Nash gap from 300.7 ± 126.4 for the full DiffNash method to 797.0 ± 153.4 with opponent IS alone. In contrast, opponent IS has little effect on convergence when own IS is omitted: the Nash gap is 787.1 ± 134.6 without either correction and 797.0 ± 153.4 with opponent IS alone. This is consistent with the Cournot ablation in the main paper (Figure 6, where opponent IS likewise has little effect on convergence performance, but reduces score-network calls to roughly 66% of those required without opponent IS. Together, these results suggest that own IS provides the primary optimization benefit by correcting the distribution shift induced by reusing the player's historical diffusion samples, while opponent IS mainly improves computational efficiency.

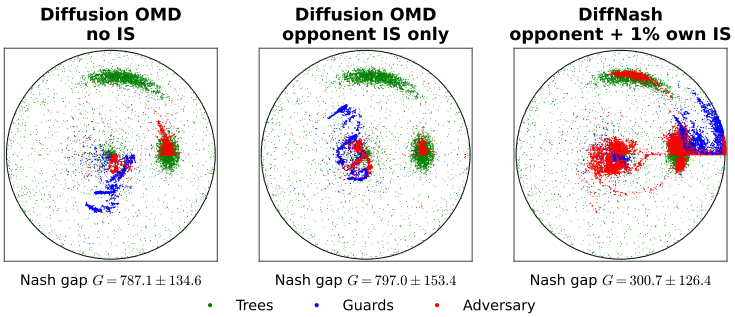

Figure 10: **Importance-sampling ablation on F2.** Removing own-policy importance sampling substantially degrades DiffNash, whereas opponent IS has little effect on convergence when own IS is omitted.

Table 3: **Hyperparameter sensitivity on F2 (m=n=3) and F3 (m=n=5).** Each row changes one parameter from the default DiffNash configuration (own batch 256, opponent batch 128, $\zeta = 1\%$, $\kappa = 10\%$). We report defender exploitability $\mathrm{Exploit_D}$ and Nash gap G; lower is better. We add 95% confidence interval to each value.

		F2		F3	
Setting	Value	$\mathrm{Exploit_D}$	Gap	$\mathrm{Exploit_D}$	Gap
Default	–	**322.5** $\pm$ 110.1	**300.7** $\pm$ 126.4	559.7 $\pm$ 76.4	**528.2** $\pm$ 142.7
Own batch	128	388.9 $\pm$ 62.0	358.7 $\pm$ 88.2	553.5 $\pm$ 250.5	510.5 $\pm$ 271.8
	512	441.5 $\pm$ 53.0	431.5 $\pm$ 66.7	669.2 $\pm$ 87.8	631.4 $\pm$ 75.8
Opponent batch	64	442.0 $\pm$ 179.8	419.5 $\pm$ 200.5	679.8 $\pm$ 289.7	667.2 $\pm$ 292.1
	256	397.9 $\pm$ 129.4	385.5 $\pm$ 141.3	576.7 $\pm$ 209.6	485.0 $\pm$ 246.6
Trigger κ	5%	372.2 $\pm$ 39.9	361.9 $\pm$ 51.6	576.7 $\pm$ 94.7	531.8 $\pm$ 126.4
	15%	373.0 $\pm$ 93.5	366.2 $\pm$ 81.2	570.3 $\pm$ 189.1	**502.0** $\pm$ 245.7

F.5 Forest Game Hyperparameter Sweep

We fix other hyperparamters and change self action rate ζ in Figure 11. Figure 11 shows that lower self-action IS rates generally yield smaller mean Nash gaps. Because importance weights are clipped, including too many samples from past iterations may bias the OMD loss in Equation (7). Empirically, a 1% self-action IS rate suffices to stabilize training.

In Table 3, we study sensitivity to the own-action batch size, opponent batch size, and trigger threshold κ on F2 and F3 over five seeds.

Trigger κ : The default $\kappa = 10\%$ achieves a slightly lower Nash gap on F2 than the other trigger values; otherwise, no trigger value consistently outperforms the others, demonstrating that **DiffNash** is relatively robust to this hyperparameter.

Own Buffer Size : Increasing the own-action batch size to 512 degrades exploitability by 25.9% and the Nash gap by 28.2% relative to the default, suggesting that a moderate own-action batch size combined with self action importance sampling is more robust. In F2, the default batch size modestly outperforms a batch size of 128.

Opponent Buffer Size : The default opponent batch size wins in all but one setting in terms of defender exploitability and Nash gap.

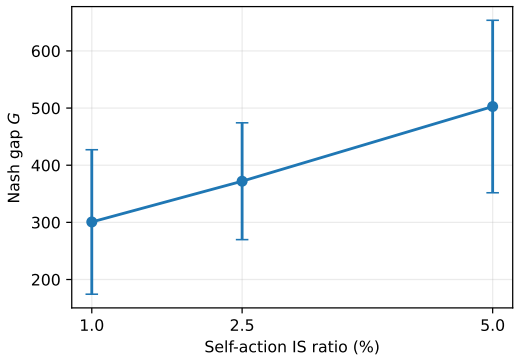

Figure 11: **Self-action IS ratio sensitivity.** Nash gap on F2 (m=n=3) as the self-action IS ratio varies. Lower is better.

G Computing Infrastructure

Experiments were run entirely on CPU hardware, using Intel Xeon Platinum 8268 nodes (48 cores, 192 GB RAM) and 32-vCPU cloud instances.