Beyond Local Nash Equilibria for Adversarial Networks

Frans A. Oliehoek[1], Rahul Savani[2], Jose Gallego[3], Elise van der Pol[3], and Roderich Groß[4]

[1] Delft University of Technology, Delft, The Netherlands
[2] University of Liverpool, Liverpool, UK
[3] University of Amsterdam
[4] The University of Sheffield

Abstract. Save for some special cases, current training methods for Generative Adversarial Networks (GANs) are at best guaranteed to converge to a 'local Nash equilibrium' (LNE). Such LNEs, however, can be arbitrarily far from an actual Nash equilibrium (NE), which implies that there are no guarantees on the quality of the found generator or classifier. This paper proposes to model GANs explicitly as finite games in mixed strategies, thereby ensuring that every LNE is an NE. We use the Parallel Nash Memory as a solution method, which is proven to monotonically converge to a resource-bounded Nash equilibrium. We empirically demonstrate that our method is less prone to typical GAN problems such as mode collapse and produces solutions that are less exploitable than those produced by GANs and MGANs.

1 Introduction

Generative Adversarial Networks (GANs) [14] are a framework in which two neural networks compete with each other: the *generator (G)* tries to trick the *classifier (C)* into classifying its generated fake data as true. GANs hold great promise for the development of accurate generative models for complex distributions without relying on distance metrics [21]. However, GANs are difficult to train [36, 1, 2]. A typical problem is *mode collapse*, which can take the form of *mode omission*, where G does not produce any points from certain modes, or *mode degeneration*, in which G only partially covers some of the modes. In fact, except for special cases (cf. Section 6), current training methods [16, 36] can only guarantee to converge to a *local* Nash equilibrium (LNE) [31]. However, an LNE can be arbitrarily far from an NE, and the corresponding generator might be exploitable by an opponent. due to suffering from problems such as mode collapse. Moreover, adding computational resources alone may not offer a way to escape these local equilibria: the problem does not lie in the lack of computational resources, but is inherently the result of only allowing small steps in strategy space using gradient-based training.

We introduce an approach that does not get trapped in LNEs by formulating adversarial networks as *finite* zero-sum games. The solutions that we try to find are saddle points in *mixed strategies*. This approach is motivated by the observation that, in the space of mixed strategies, any LNE is an NE. We employ Parallel Nash Memory (PNM) [26], to search for approximate mixed equilibria with small support.

PNM has been shown to monotonically converge to a NE, provided that in its iterations it has non-zero probability to find better responses [26]. However, due to the extremely large number of pure strategies that result for sensible choices of neural network classes, we cannot expect to find exact best responses. Therefore, we introduce resource-bounded best-responses (RBBRs), and show that our PNM approach monotonically converges to the corresponding resource-bounded Nash equilibrium (RB-NE).

Key features of our approach are that: 1) It is based on finite zero-sum games, and as such it enables the use of existing game-theoretic methods. In this paper we focus on one such method, Parallel Nash Memory (PNM) [26]. 2) It will not get trapped in LNEs: we prove that it monotonically converges to an RB-NE, which means that more computation can improve solution quality. 3) It works for any network architecture In particular, future improvements in classifiers/generator networks can be exploited directly.

We investigate empirically the effectiveness of PNM and show that it can indeed deal well with typical GAN problems. We show that the found solutions closely match the theoretical predictions made by [14] about the conditions at a Nash equilibrium, and are much less susceptible to being exploited by an adversary than those produced by GANs and MGANs [17].

2 Background

We defer a more detailed treatment of related work until Section 6. Here, we introduce some basic game theory.

A *two-player strategic game*, which we will simply call 'game', is a tuple $\langle \mathcal{D}, \{\mathcal{S}_i\}_{i\in\mathcal{D}}, \{u_i\}_{i\in\mathcal{D}} \rangle$, where $\mathcal{D} = \{1, 2\}$ is the set of *players*, $\mathcal{S}_i$ is the set of *pure strategies* (actions) for player i, and $u_i : \mathcal{S} \to \mathbb{R}$ is $i's$ payoff function defined on the set of pure strategy profiles $\mathcal{S} := \mathcal{S}_1 \times \mathcal{S}_2$. When the action sets are finite, the game is *finite*. We also write s_i and s_{-i} for the strategy of agent i and its opponent respectively. A fundamental concept is the Nash equilibrium (NE), which is a strategy profile $s = \langle s_i, s_{-i} \rangle$ such that no player can unilaterally deviate and improve his payoff: $u_i(s) \geq u_i(\langle s'_i, s_{-i} \rangle)$ for all players i and $s'_i \in \mathcal{S}_i$.

A finite game may not possess a pure NE. A *mixed strategy* μ_i of player i is a probability distribution over i's pure strategies $\mathcal{S}_i$. The payoff of a player under a profile of mixed strategies $\mu = \langle \mu_1, \mu_2 \rangle$ is defined as the expectation: $u_i(\mu) := \sum_{s\in\mathcal{S}} [\prod_{j\in\mathcal{D}} \mu_j(s_j)] \cdot u_i(s)$. Then an NE in mixed strategies is defined as follows. A $\mu = \langle \mu_i, \mu_{-i} \rangle$ is an NE if and only if $u_i(\mu) \geq u_i(\langle s'_i, \mu_{-i} \rangle)$ for all players i and potential unilateral deviations $s'_i \in \mathcal{S}_i$. Every finite game has at least one NE in mixed strategies [23]. In this paper we deal with two-player

zero-sum games, where $u_1(s_1, s_2) = -u_2(s_1, s_2)$ for all $s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2$. The equilibria of zero-sum games, also called *saddle points*, [5] have several important properties, as stated in Von Neuman's Minmax theorem [24]:

Theorem 1 *In a finite zero-sum game, v^* is the* value *of the game that satisfies:* $\min_{\mu_2} \max_{\mu_1} u_1(\mu) = \max_{\mu_1} \min_{\mu_2} u_1(\mu) = v^*$.

All equilibria have payoff v^* and equilibrium strategies are interchangeable: if $\langle \mu_1, \mu_2 \rangle$ and $\langle \mu_1', \mu_2' \rangle$ are equilibria, then so are $\langle \mu_1', \mu_2 \rangle$ and $\langle \mu_1, \mu_2' \rangle$ [28]. This means that in zero-sum games we do not need to worry about equilibrium selection. Moreover, the convex combination of two equilibria is an equilibrium, meaning that the game either has one or infinitely many equilibria. We also employ the standard, additive notion of *approximate equilibrium:* A pair of strategies (μ_i, μ_{-i}) is an ϵ-NE if $\forall i \quad u_i(\mu_i, \mu_{-i}) \geq \max_{\mu_i'} u_i(\mu_i', \mu_{-i}) - \epsilon$.

In the literature, GANs have not typically been considered as finite games. The natural interpretation of the standard setup of GANs is of an infinite game where payoffs are defined over all possible weight parameters for the respective neural networks. With this view we do not obtain existence of saddle points *in the space of parameters*, nor the desirable properties that follow from Theorem 1.[6] This is why the notion of *local Nash equilibrium (LNE)* has arisen in the literature [31, 36]. Roughly, an LNE is a strategy profile where neither player can improve in a small neighborhood of the profile. In finite games every LNE is an NE, as, whenever there is a global deviation (i.e., a better response), one can always deviate locally in the space of mixed strategies towards a pure best response (by playing that better response with ϵ higher probability).

3 GANGs

In order to capitalize on the insight that we can escape local equilibria by switching to mixed strategy space for a finite game, we formalize adversarial networks in a finite games setting.

We consider a standard adversarial network setup: $\mathcal{M} = \langle p_d, \langle G, p_z \rangle, C, \phi \rangle$ where $p_d(x)$ is the distribution over ('true' or 'real') data points $x \in \mathbb{R}^d$. G is a neural network class with d outputs, parametrized by a parameter vector $\theta_G \in \Theta_G$, such that $G(z; \theta_G) \in \mathbb{R}^d$ denotes the ('fake' or 'generated') output of G on a random vector z drawn from some distribution $z \sim p_z$. C is a neural network class with a single output, parametrized by a parameter vector $\theta_C \in \Theta_C$, such that the output $C(x; \theta_C) \in [0, 1]$ indicates the 'realness' of x according to

[5] Note that in game theory the term 'saddle point' is used to denote a 'global' saddle point which corresponds to a Nash equilibrium: there is no profitable deviation near or far away from the current point. In contrast, in machine learning, the term 'saddle point' typically denotes a 'local' saddle point: no player can improve its payoff by making a small step from the current joint strategy.

[6] Some results on the existence of saddle points in infinite action games are known, but they require properties like convexity and concavity of utility functions [5], which we cannot apply as they would need to hold w.r.t. the neural network parameters.

C. $\phi : [0, 1] \to \mathbb{R}$ is a *measuring function* [4]—e.g., log for GANs, the identity mapping for WGANs—used to specify game payoffs, explained next.

We call $\mathcal{M}$ a *Generative Adversarial Network Game (GANG)*, since it induces a zero-sum game $\langle \mathcal{D} = \{G, C\}, \{\mathcal{S}_G, \mathcal{S}_C\}, \{u_G, u_C\}\rangle$ with: $\mathcal{S}_G = \{G(\cdot; \theta_G) \mid \theta_G \in \Theta_G\}$ the set of strategies s_G; $\mathcal{S}_C = \{C(\cdot; \theta_C) \mid \theta_C \in \Theta_C\}$ the set of strategies s_C; $u_C(s_G, s_C) = \mathbf{E}_{x\sim p_d}[\phi(s_C(x))] - \mathbf{E}_{z\sim p_z}[\phi(s_C(s_G(z)))]$. I.e., the score of C is the expected 'measured realness' of the real data minus that of the fake data; $u_G(s_G, s_C) = -u_C(s_G, s_C)$. As such, when using $\phi = \log$, GANGs employ a payoff function for G that use [14]'s trick to enforce strong gradients early in the training process (but it applies this transformation to u_C too, in order to retain the zero-sum property). Correctness of these transformations is shown in [27].

In practice, GANs are represented using floating point numbers, of which, for a given setup, there is only a finite (albeit large) number. From now on, we will focus on finite GANGs, which have finite parameter sets and a finite set of neural network architectures.

We emphasize this finiteness, because this is exactly what enables us to obtain the desirable properties mentioned in Section 2: existence of (one or infinitely many) mixed NEs with the same value, as well as the guarantee that any LNE is an NE. Moreover, these properties hold for the GANG in its original formulation—not for a theoretical abstraction in terms of (infinite capacity) densities—which means that we can truly expect solution methods (that operate in the parametric domain [34]) to exploit these properties. However, since we do not impose any additional constraints or discretization[7], the number of strategies (all possible unique instantiations of the network class with floating point numbers) is *huge*. Therefore, we think that finding (near-) equilibria with small supports is one of the most important challenges for making principled advances in the field of adversarial networks. As a first step towards addressing this challenge, we propose to make use of the *Parallel Nash Memory (PNM)* [26], which can be seen as a generalization (to non-exact best responses) of the *double oracle method* [22, 6].

4 Solving GANGs

Treating GANGs as finite games in mixed strategies permits building on existing tools and algorithms for these classes of games [11, 29, 10]. In this section, we describe how to use Parallel Nash Memory (PNM) [26], which is particularly tailored to find approximate NEs with small support, and monotonically[8] converges to such an equilibrium.

[7] Therefore, our finite GANGs have the same representational capacity as normal GANs that are implemented using floating point arithmetic.

[8] For an explanation of the precise meaning of monotonic here, we refer to [26]. Roughly, we will be 'secure' against more strategies of the other agent with each iteration. This does not imply that the worst case payoff for an agent also improves monotonically. The latter property, while desirable, is not possible with an approach that incrementally constructs sub-games of the full game, as considered here: there might always be a part of the game we have not seen yet, but which we might dis-

Algorithm 1 PARALLEL NASH MEMORY WITH DETERMINISTIC RBBRS

1: $\langle s_G, s_C \rangle \leftarrow$ INITIALSTRATEGIES()
2: $\langle \mu_G, \mu_C \rangle \leftarrow \langle \{s_G\}, \{s_C\} \rangle$ ▷ set initial mixtures
3: **while** True **do**
4: $s_G \leftarrow \text{RBBR}(\mu_C)$ ▷ get new bounded best resp.
5: $s_C \leftarrow \text{RBBR}(\mu_G)$
6: // Expected payoffs of these 'tests' against mixture:
7: $u_{BRs} \leftarrow u_G(s_G, \mu_C) + u_C(\mu_G, s_C)$
8: **if** $u_{BRs} \leq 0$ **then**
9: **break**
10: **end if**
11: $SG \leftarrow$ AUGMENTGAME(SG, s_G, s_C)
12: $\langle \mu_G, \mu_C \rangle \leftarrow$ SOLVEGAME(SG)
13: **end while**
14: **return** $\langle \mu_G, \mu_C \rangle$ ▷ found an RB-NE

Parallel Nash Memory for GANGs The basic idea of PNM is that we iteratively find new strategies which are good candidates for improvement of an approximate mixed strategy NE $\langle \mu_G, \mu_C \rangle$. Previous works (such as the original PNM paper [26], and before that the double-oracle method [22]) have considered the use of exact best response (BR) functions to deliver such new candidates. In GANGs, however, computing such an exact BR is intractable, and we typically use gradient descent, or another way to compute an approximate best response. In phrasing our algorithm, we abstract away from the actual implementation of *how* it is computed, but we acknowledge the fact that the quality we can expect is bounded by computational resources. As such we will use the term 'resource-bounded best response' (RBBR) to denote an approximate best response function which computes the best possible answer it can given some amount of computational resources. For ease of explanation, we focus on the setting with deterministic best responses, but the approach can easily be extended to non-deterministic RBBR functions[9] and our empirical evaluation makes use of such non-deterministic RBBR functions (due to random initializations).

Algorithm 1 details our approach. PNM incrementally grows a strategic game SG over a number of iterations using the AUGMENTGAME function. It uses SOLVEGAME to compute (via linear programming, see, e.g., [33]) a mixed strategy NE $\langle \mu_G, \mu_C \rangle$ *of this smaller game* at the end of each iteration. At the beginning of each iteration the algorithm uses the RBBR functions to deliver new promising strategies (s_G, s_C). Then we test if they 'beat' the current $\langle \mu_G, \mu_C \rangle$. If they do, $u_{BRs} > 0$, and the game is augmented with these and solved again to find a new NE of the sub-game SG. If they do not, $u_{BRs} \leq 0$, and the algorithm stops.

cover in the future that will lead to a very poor worst case payoff for one of the agents.

[9] By changing the termination criterion of line 8 in Algorithm 1 into a criterion for including the newly found strategies. See the formulation in [26] for more details.

AugmentGame evaluates (by simulation) each newly found strategy for each player against all of the existing strategies of the other player, thus constructing a new row and column for the maintained payoff matrix. In order to implement the best response functions, we have used standard stochastic gradient descent, which means that any existing neural network architectures can be used. However, we need to compute RBBRs against *mixtures* of networks of the other player. For C this is trivial: we can simply generate a batch of fake data from the mixture μ_G. Implementing an RBBR for G against μ_C is slightly more involved, as we need to back-propagate the gradient from all the different $s_C \in \mu_C$ to G. Intuitively, one can think of a combined network consisting of the G network with its outputs connected to every $s_C \in \mu_C$. The predictions $\hat{y}_{s_C}$ of these components $s_C \in \mu_C$ are combined in a single linear output node $\hat{y} = \sum_{s_C \in \mu_C} \mu_C(s_C) \cdot \hat{y}_{s_C}$. This allows us to evaluate and backpropagate through the entire network. A practical implementation loops through each component $s_C \in \mu_C$ and does the evaluation of the weighted prediction $\mu_C(s_C) \cdot \hat{y}_{s_C}$ and subsequent backpropagation per component.

Analysis Given that we do not compute exact BRs, we cannot get convergence to an NE. Instead, using RBBRs, we define an intuitive specialization of NE:

Definition 1. $\mu = \langle \mu_i, \mu_j \rangle$ *is a* resource-bounded NE (RB-NE) *if and only if* $\forall i \quad u_i(\mu_i, \mu_j) \geq u_i(RBBR_i(\mu_j), \mu_j)$.

That is, an RB-NE can be thought of as follows: we present μ to each player i and it gets the chance to switch to another strategy, for which it can apply its bounded resources (i.e., use $RBBR_i$) exactly once. After this application, the player's resources are exhausted and if the found $RBBR_i(\mu_j)$ does not lead to a higher payoff it will not have an incentive to deviate.[10]

Intuitively, it is clear that PNM converges to an RB-NE, which we now state formally. All proofs can be found in [27].

Theorem 2 *If PNM terminates, it has found an RB-NE.*

Moroever, making use of the finiteness of the game, it can be easily shown that Algorithm 1 terminates and *monotonically* converges to an equilibrium. Finally, an RB-NE can be linked to the familiar notion of ϵ-NE by making assumptions on the power of the best response computation.

Theorem 3 *If both players are powerful enough to compute ϵ-best responses, then an RB-NE is an ϵ-NE.*

The PNM algorithm for GANGs is parameter free, but we mention two adaptations that are helpful: Interleaved training of best responses and regularization of classifier best responses. Details can be found in [27].

[10] During training the RBBR functions will be used many times. However, the goal of the RB-NE is to provide a characterization of the *end point* of training.

5 Experiments

Here we report on experiments that aim to test if searching in mixed strategies with PNM-GANG can help in reducing problems with training GANs, and if the found solutions (near-RB-NEs) provide better generative models and are potentially closer to true Nash equilibria than those found by GANs (near-LNEs). Since our goal is to produce better generative models, we refrain from evaluating these methods on complex data like images: image quality and log likelihood are not aligned as for instance shown by [35]. Moreover there is debate about whether GANs are overfitting and assessing this from samples is difficult; some methods have been proposed e.g., [3, 32, 25, 20], but most provide merely a measure of variability, not over-fitting. As such, we choose to focus on irrefutable results on mixture of Gaussian (MoG) tasks, for which the distributions can readily be visualized.

Experimental setup. We compare our PNM approach ('PNM-GANG') to a vanilla GAN implementation and state-of-the-art MGAN [17]. Table 1 summarizes the settings for GAN and PNM training. RBBR models were taken to be as small as possible while still achieving good results. As suggested by [8], we use leaky ReLU as inner activation for our GAN implementation to avoid sparse gradients. Generators have linear output layers. Classifiers use sigmoids for the final layer. Both classifiers and generators are multi-layer perceptrons with 3 hidden layers. We do not use techniques such as Dropout or Batch Normalization, as they did not yield significant improvements in the quality of our experimental results. The MGAN configuration is identical to that of Table 3 in Appendix C1 of [17].

	GAN	**RBBR**
Learning Rate	$3 \cdot 10^{-4}$	$5 \cdot 10^{-3}$
Batch Size	128	128
Dimension of z	40	5
Hidden nodes	50	5
Iterations	20000	750
Generator Parameters	4902	92
Classifier Parameters	2751	61
Inner Activation	Leaky ReLU	Leaky ReLU
Measuring Function	log	10^{-5}-bounded log

Table 1: Settings used to train GANs and RBBRs.

We test on 3 MoG tasks: 'round', 'grid' and 'random' (cf. Figure 1). For each we create test cases with 9 and 16 components. In our plots, black points are real data, green points are generated data. Blue indicates areas that are classified as 'realistic' while red indicates a 'fake' classification by C.

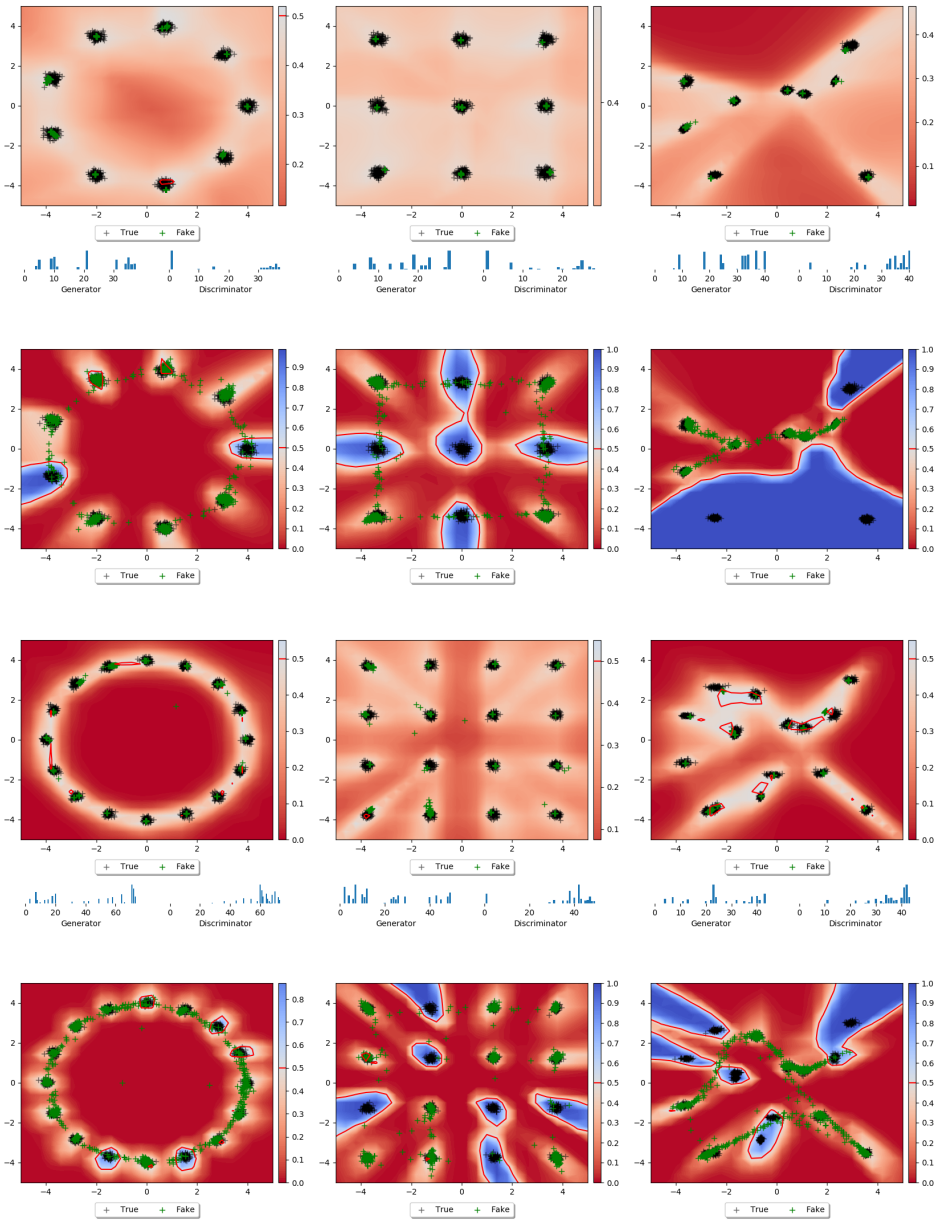

Fig. 1: Results for mixtures of Gaussians with 9 and 16 modes. Odd rows: PNM-GANG, Even rows: GAN. The histograms represent the probabilities in the mixed strategy of each player. True data is shown in black, while fake data is green. The classification boundary (where the classifier outputs 0.5) is indicated with a red line. Best seen in color.

Found solutions compared to normal GANs. The results produced by regular GANs and PNM-GANGs are shown in Figure 1 and clearly convey three main points: 1) The PNM-GANG mixed classifier has a much flatter surface than the classifier found by the GAN. Around the true data, it outputs around 0.5 indicating indifference, which is *in line with the theoretical predictions* about the equilibrium [14]. 2) This flatter surface is not coming at the cost of inaccurate samples. In contrast: nearly all samples shown are hitting one of the modes and thus *the PNM-GANG solutions are highly accurate*, much more so than the GAN solutions. 3) Finally, the PNM-GANGs, unlike GANs, *do not suffer from mode omission.*We also note that PNM-GANG typically uses fewer total parameters than the regular GAN, e.g., 1463 vs. 7653 for the random 9 task in Figure 1. This shows that, qualitatively, the use of multiple generators seems to lead to good results. However, not all modes are *fully* covered. This can be controlled by varying the learning rate [27].

Found solutions compared to MGANs. Here we compare the solutions found above for PNM-GANGs to a state-of-the-art GAN variant: MGAN [17] proposes a setup with a mixture of k generators, a classifier, and a discriminator. In an MGAN, the generator mixture aims to create samples which match the training data distribution, while the discriminator distinguishes real and generated samples, and the classifier tries to determine which generator a sample comes from. We use MGAN as a state-of-the art baseline that was explicitly designed to overcome the problem of mode collapse.

Figure 2 shows the results of MGAN on the mixture of Gaussian tasks. We see that MGAN results do seem qualitatively quite good. Comparing them to the PNG-GANG results from Figure 1, we see that MGAN may even have less mode degeneration. However, we also see that in the MGAN results there is one missed mode (and thus also one mode covered by 2 generators) on the randomly located components task (right column). In contrast, the PNM-GANGs results did not fail to capture any mode.

We point out that MGAN results were obtained with an architecture and hyperparameters which exactly match those proposed by [17] for a similar task. This means that the MGAN models shown use many more parameters (approx. 310,000) than the GAN and GANG models (approx. 2,000). MGAN requires the number of generators to be chosen upfront as a hyperparameter of the method. We chose this to be equal to the number of mixture components, so that MGAN could cover all modes with one generator per mode. We note that PNM does not require such a hyperparameter to be set, nor does PNM require the related "diversity" hyperparameter of the MGAN method (called β in the MGAN paper).

Overall, these results show that the quality of the solutions found by PNM-GANGs is competitive to that of the state-of-the-art MGAN, while using much fewer parameters.

Exploitability of solutions. Finally, to complement the above qualitative analysis, we also provide a quantitative analysis of the solutions found by GANs, MGANs and PNM-GANGs. We investigate to what extent they are exploitable

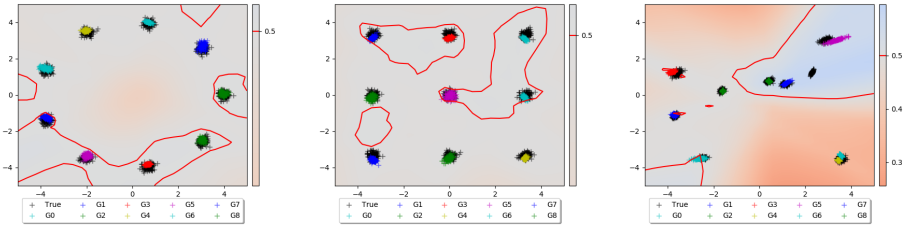

Fig. 2: Results for MGAN on several mixture of Gaussian tasks with 9 modes. Markers correspond to samples created by each generator.

by newly introduced adversaries with some fixed computational power (as modeled by the complexity of the networks we use to attack the found solution). Intuitively, since PNM-GANGs are trained by (against) more powerful attack than GANs, we expect them to be more robust against new attacks of any kind. Specifically, for a given solution $\tilde{\mu} = (\tilde{\mu_G}, \tilde{\mu_C})$ we use the following measure of exploitability:

$$expl^{RB}(\tilde{\mu_G}, \tilde{\mu_C}) \triangleq \text{RBmax}_{\mu_G} u_G(\mu_G, \tilde{\mu_C}) + \text{RBmax}_{\mu_C} u_C(\tilde{\mu_G}, \mu_C),$$

where 'RBmax' denotes an approximate maximization performed by an adversary of some fixed complexity.[11]

Figure 3 shows our exploitability results for all three tasks with nine modes. We observe roughly the same trend across the three tasks. First, we investigate the exploitability of solutions delivered by GANs, MGANs and GANGs of different complexities (in terms of *total* number of parameters used). For this, we compute 'attacks' to these solutions using attackers of fixed complexity (a total of 453 parameters for the attacking G and C together). These results are shown in Figure 3 (left and middle column). The left column shows the exploitability of PNM-GANG after different numbers of iterations, as well as the number of parameters used in the solutions found in those iterations (a sum over all the networks in the support of the mixture). Error bars indicate standard deviation over 15 trials. It is apparent that PNM-GANG solutions with more parameters typically are less exploitable. Also shown is that the variance of exploitability depends heavily on the solution that we happen to attack.

The middle column shows how exploitable GAN, MGAN and PNM-GANG models of different complexities are: the x-axis indicates the total number of parameters, while the y-axis shows the exploitability. The PNM results are the

[11] That is, the 'RBmax' functions are analogous to the RBBR functions employed in PNM, but the computational resources of 'RBmax' could be different. Our exploitability is closely related to the use of GAN training metrics [18], but additionally includes the exploitability of the classifier. This is important: when only testing the exploitability of the generator, this does give a way to compare generators, *but it does not give a way to assess how far from equilibirum we might be*. For a further motivation of this measure of exploitability, please see [27].

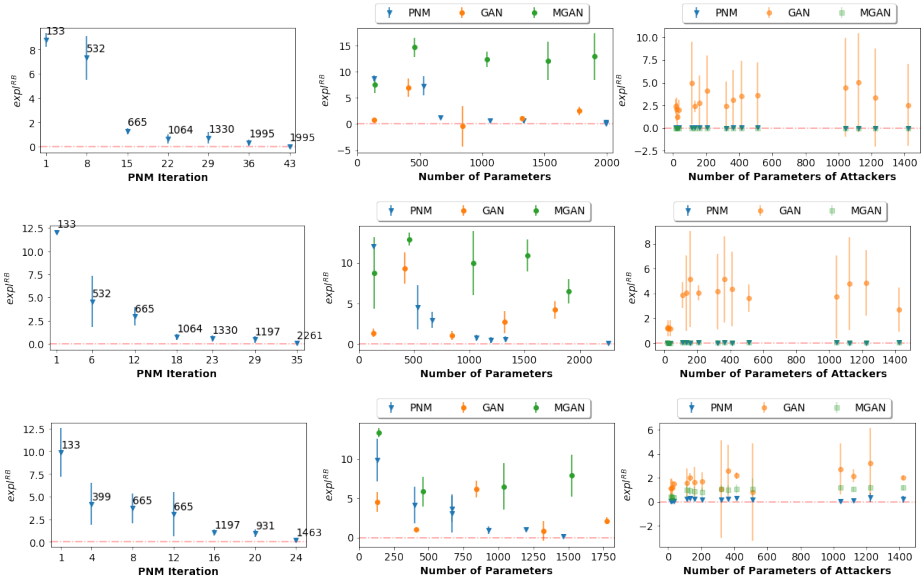

Fig. 3: Exploitability results all 9 mode tasks. Top to bottom: round, grid, random.

same points also shown in the left column, but repositioned at the appropriate place on the x-axis. All data points are exploitability of models that were trained until convergence. Note that here the x-axis shows the complexity in terms of total parameters. The figure shows an approximately monotonic decrease in exploitability for GANGs, while GANs and MGANs with higher complexity are still very exploitable in many cases. In contrast to GANGs, more complex architectures for GANs or MGANs are thus not necessarily a way to guarantee a better solution.

We also we investigate what happens for the converged GAN / PNM-GANG solution of Figure 1, which have comparable complexities, when attacked with varying complexity attackers. We also employ an MGAN which has a significantly larger number of parameters (approx. 310,000) than the GAN and GANG models (approx. 2,000). These results are shown in Figure 3 (right). Clearly shown is that the PNM-GANG is robust with near-zero exploitability even when attacked with high complexity attackers. The MGAN solution has a non-zero level of exploitability, roughly constant for several attacker complexities. In stark contrast, we see that the converged GAN solution is exploitable already for low-complexity attackers, again suggesting that the GAN was stuck in an LNE far away from a NE.

Additionally, we investigate the exploitability of the trained models presented in Figure 1 and Figure 2 when attacked by neural networks of varying complexity. These results are shown in the right column of Figure 3. Clearly shown is that the PNM-GANG is robust with near-zero exploitability even when attacked with high-complexity attackers. The MGAN models also have low exploitability,

but recall that these models are *much* more complex (GANG and GAN models have approximately 2,000 parameters, while the MGAN model involves approximately 310,000 parameters). Even with such a complex model, in the 'random' task, the MGAN solution has a non-zero level of exploitability, roughly constant for several attacker complexities. This is related to the missed mode and the fact that two of the MGAN generators collapsed to the same lower-right mode in Figure 1. In stark contrast to both PNM-GANGs and MGAN, we see that the converged GAN solution is exploitable already for low-complexity attackers, again suggesting that the GAN was stuck in an Local Nash Equilibrium far away from a (global) Nash Equilibrium.

Overall, these results demonstrate that GANGs can provide more robust solutions than GANs/MGANs with the same number of parameters, suggesting that they are closer to a Nash equilibrium and provide better generative models.

6 Related work

Due to limited space, we only describe the most relevant related papers here. For a discussion of recent progress on solving in zero-sum games, more general GAN improvements, and bounded rationality, please see [27].

Recently, more researchers have investigated the idea of (more or less) explicitly representing a set or mixture of strategies for the players. For instance, [19] retains sets of networks that are trained by randomly pairing up with a network for the other player thus forming a GAN. This, like PNM, can be interpreted as a coevolutionary approach, but unlike PNM, it does not have any convergence guarantees. MAD-GAN [13] uses k generators, but one discriminator. MGAN [17] proposes mixtures of k generators, a classifier and a discriminator with weight sharing; and presents a theoretical analysis similar to [14] assuming infinite capacity densities. None of these approaches have convergence guarantees.

Generally, explicit mixtures can bring advantages in two ways: *(1) Representation*: intuitively, a mixture of k neural networks could better represent a complex distribution than a single neural network of the same size, and would be roughly on par with a single network that is k times as big. [4] show how to create such a bigger network that is particularly suitable for dealing with multiple modes using a 'multi-way selector'. In our experiments we observed mixtures of simpler networks leading to better performance than a single larger network of the same total complexity (in terms of number of parameters). *(2) Training*: Arora et al. [4] use an architecture that is tailored to representing a mixture of components and train a single such network. We, in contrast, explicitly represent the mixture; given the observation that good solutions will take the form of a mixture. This is a form of domain knowledge that facilitates learning and convergence guarantees.

A closely related paper is the work by [15], which also builds upon game-theoretic tools to give certain convergence guarantees. The main differences are as follows: 1) We provide a more general form of convergence (to an RB-NE) that is applicable to *all* architectures, that only depends on the power to compute best

responses, and show that PNM-GANG converges in this sense. We also show that if agents can compute an ϵ-best response, then the procedure converges to an ϵ-NE. 2) [15] show that for a quite specific GAN architecture their first algorithm converges to an ϵ-NE. On the one hand, this result is an instantiation of our more general theory: they assume they can compute exact (for G) and ϵ-approximate (for C) best responses; for such powerful players our Theorem 3 provides that guarantee. On the other hand, their formulation works without discretizing the spaces of strategies. 3) The practical implementation of the algorithm in [15] does not provide guarantees.

Finally, [12] propose a method similar to ours that uses *fictitious play* [7, 11] rather than PNM. Fictitious play does not explicitly model mixed strategies for the agents, but interprets the opponent's historical behavior as such a mixed strategy. The average strategy played by the 'Fictitious GAN' approach converges to a Nash equilibrium *assuming that "the discriminator and the generator are updated according to the best-response strategy at each iteration"*, which follow from the result by [9] which states that fictitious play converges in continuous zero-sum games. Intuitively, fictitious play, like PNM, in each iteration only ever touches a finite subset of strategies, and one can show that the value of such subgames converges. While this result gives some theoretical underpinning to Fictitious GAN, of course in practice the assumption is hard to satisfy and the notion of RB-NE that we propose may apply to analyze their approach too. Also, in their empirical results they limit the history of actions (played neural networks in previous iterations) to 5 to improve scalability at the cost of convergence guarantees. The Fictitious GAN is not explicitly shown to be more robust than normal GANs, as we show in this paper, but it is demonstrated to produce high quality images, thus showing the potential of game theoretical approaches to GANs to scale.

7 Conclusions

We introduce finite GANGs—Generative Adversarial Network Games—a novel framework for representing adversarial networks by formulating them as finite zero-sum games. By tackling them with techniques working in mixed strategies we can avoid getting stuck in local Nash equilibria (LNE). As finite GANGs have extremely large strategy spaces we cannot expect to exactly (or ϵ-approximately) solve them. Therefore, we introduced the resource-bounded Nash equilibrium (RB-NE). This notion is richer than LNE in that it captures not only failures of escaping local optima of gradient descent, but applies to any approximate best-response computations, including methods with random restarts. Additionally, GANGs can draw on a rich set of methods for solving zero-sum games [11, 30, 10, 26]. In this paper, we build on PNM and prove that the resulting method monotonically converges to an RB-NE. We empirically demonstrate that the resulting method does not suffer from typical GAN problems such as mode collapse and forgetting. We also show that the GANG-PNM solutions are closer to theoretical predictions, and are less exploitable than normal GANs: by using

PNM we can train models that are more robust than GANs of the same total complexity, indicating they are closer to a Nash equilibrium and yield better generative performance.

We presented a framework that can have many instantiations and modifications. For example, one direction is to employ different learning algorithms. Another direction could focus on modifications of PNM, such as to allow discarding "stale" pure strategies, which would allow the process to run for longer without being inhibited by the size of the resulting zero-sum "subgame" that must be maintained and repeatedly solved.

Acknowledgments

This research made use of a GPU donated by NVIDIA. F.A.O. is funded by EPSRC First Grant EP/R001227/1, and ERC Starting Grant #758824—INFLUENCE.

References

1. Arjovsky, M., Bottou, L.: Towards principled methods for training generative adversarial networks. In: ICLR (2017)
2. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: ICML (2017)
3. Arora, S., Zhang, Y.: Do GANs actually learn the distribution? An empirical study. ArXiv e-prints (Jun 2017)
4. Arora, S., Ge, R., Liang, Y., Ma, T., Zhang, Y.: Generalization and equilibrium in generative adversarial nets (GANs). In: ICML (2017)
5. Aubin, J.P.: Optima and equilibria: an introduction to nonlinear analysis, vol. 140. Springer Science & Business Media (1998)
6. Bosanský, B., Kiekintveld, C., Lisý, V., Pechoucek, M.: An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. Journal of AI Research **51** (2014)
7. Brown, G.W.: Iterative solution of games by fictitious play. Activity analysis of production and allocation **13**(1), 374–376 (1951)
8. Chintala, S.: How to train a GAN? tips and tricks to make GANs work. `https://github.com/soumith/ganhacks` (2016), retrieved on 2018-02-08.
9. Danskin, J.M.: Fictitious play for continuous games revisited. International Journal of Game Theory **10**(3), 147–154 (Sep 1981). https://doi.org/10.1007/BF01755961, `https://doi.org/10.1007/BF01755961`
10. Foster, D.J., Li, Z., Lykouris, T., Sridharan, K., Tardos, E.: Learning in games: Robustness of fast convergence. In: NIPS 29 (2016)
11. Fudenberg, D., Levine, D.K.: The theory of learning in games. MIT press (1998)
12. Ge, H., Xia, Y., Chen, X., Berry, R., Wu, Y.: Fictitious GAN: Training GANs with Historical Models. ArXiv e-prints (Mar 2018)
13. Ghosh, A., Kulharia, V., Namboodiri, V.P., Torr, P.H.S., Dokania, P.K.: Multi-agent diverse generative adversarial networks. CoRR **abs/1704.02906** (2017)
14. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS 27 (2014)
15. Grnarova, P., Levy, K.Y., Lucchi, A., Hofmann, T., Krause, A.: An Online Learning Approach to Generative Adversarial Networks. ArXiv e-prints (Jun 2017)

16. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: NIPS 30 (2017)
17. Hoang, Q., Nguyen, T.D., Le, T., Phung, D.Q.: Multi-generator generative adversarial nets. In: ICLR (2018)
18. Im, D.J., Ma, A.H., Taylor, G.W., Branson, K.: Quantitatively evaluating GANs with divergences proposed for training. In: ICLR (2018)
19. Jiwoong Im, D., Ma, H., Dongjoo Kim, C., Taylor, G.: Generative Adversarial Parallelization. ArXiv e-prints (Dec 2016)
20. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: ICLR (2018)
21. Li, W., Gauci, M., Groß, R.: Turing learning: a metric-free approach to inferring behavior and its application to swarms. Swarm Intelligence **10**(3) (2016)
22. McMahan, H.B., Gordon, G.J., Blum, A.: Planning in the presence of cost functions controlled by an adversary. In: Proceedings of the 20th International Conference on Machine Learning (ICML-03). pp. 536–543 (2003)
23. Nash, J.F.: Equilibrium points in N-person games. Proceedings of the National Academy of Sciences of the United States of America **36** (1950)
24. von Neumann, J.: Zur Theorie der Gesellschaftsspiele. Mathematische Annalen **100**(1) (1928)
25. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: ICML (2017)
26. Oliehoek, F.A., de Jong, E.D., Vlassis, N.: The parallel Nash memory for asymmetric games. In: Proc. Genetic and Evolutionary Computation (GECCO) (2006)
27. Oliehoek, F.A., Savani, R., Gallego-Posada, J., van der Pol, E., Gross, R.: Beyond local Nash equilibria for adversarial networks. ArXiv e-prints (2018)
28. Osborne, M.J., Rubinstein, A.: A Course in Game Theory, chap. Nash Equilibrium. The MIT Press (Jul 1994)
29. Rakhlin, A., Sridharan, K.: Online learning with predictable sequences. In: Proc. of the Conference on Learning Theory (2013)
30. Rakhlin, A., Sridharan, K.: Optimization, learning, and games with predictable sequences. In: NIPS 26 (2013)
31. Ratliff, L.J., Burden, S.A., Sastry, S.S.: Characterization and computation of local Nash equilibria in continuous games. In: Communication, Control, and Computing, 2013 51st Annual Allerton Conference on. IEEE (2013)
32. Salimans, T., Goodfellow, I.J., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. In: NIPS 29 (2016)
33. Shoham, Y., Leyton-Brown, K.: Multi-Agent Systems: Algorithmic, game-theoretic and logical foundations. Cambridge University Press (2008)
34. Sinn, M., Rawat, A.: Non-parametric estimation of jensen-shannon divergence in generative adversarial network training. In: AISTATS (2018)
35. Theis, L., van den Oord, A., Bethge, M.: A note on the evaluation of generative models. In: ICLR (2016)
36. Unterthiner, T., Nessler, B., Klambauer, G., Heusel, M., Ramsauer, H., Hochreiter, S.: Coulomb GANs: Provably optimal Nash equilibria via potential fields. In: ICLR (2018)