DO-GAN: A Double Oracle Framework for Generative Adversarial Networks

Aye Phyu Phyu Aung[1,2*], Xinrun Wang[1*†], Runsheng Yu[1*], Bo An[1†]
Senthilnath Jayavelu[2], Xiaoli Li[1,2,3†]
[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2]Institute for Infocomm Research, A*STAR, Singapore
[3]A*STAR Centre for Frontier AI Research, Singapore
{ayep0001,xinrun.wang, boan}@ntu.edu.sg, runshengyu@gmail.com,
{j_senthilnath, xlli}@i2r.a-star.edu.sg

Abstract

In this paper, we propose a new approach to train Generative Adversarial Networks (GANs) where we deploy a double-oracle framework using the generator and discriminator oracles. GAN is essentially a two-player zero-sum game between the generator and the discriminator. Training GANs is challenging as a pure Nash equilibrium may not exist and even finding the mixed Nash equilibrium is difficult as GANs have a large-scale strategy space. In DO-GAN, we extend the double oracle framework to GANs. We first generalize the players' strategies as the trained models of generator and discriminator from the best response oracles. We then compute the meta-strategies using a linear program. For scalability of the framework where multiple generators and discriminator best responses are stored in the memory, we propose two solutions: 1) pruning the weakly-dominated players' strategies to keep the oracles from becoming intractable; 2) applying continual learning to retain the previous knowledge of the networks. We apply our framework to established GAN architectures such as vanilla GAN, Deep Convolutional GAN, Spectral Normalization GAN and Stacked GAN. Finally, we conduct experiments on MNIST, CIFAR-10 and CelebA datasets and show that DO-GAN variants have significant improvements in both subjective qualitative evaluation and quantitative metrics, compared with their respective GAN architectures.

1. Introduction

Generative Adversarial Networks (GANs) [8] have been applied in various domains such as image and video generation, text-to-image synthesis and equipment condition monitoring [18, 29–31]. Various architectures are proposed to generate more realistic samples [23, 27, 28] as well as regularization techniques [1, 25]. From the game-theoretic perspective, GANs can be viewed as a two-player game where the generator samples the data and the discriminator classifies the data as real or generated. They are alternately trained to maximize their respective utilities till convergence corresponding to a pure Nash Equilibrium (NE).

Figure 1. Training images with fixed noise for SGAN and DO-SGAN/P (pruning) until termination. Both during training and after convergence, DO-SGAN/P can generate better quality images with FID score of 6.32 against SGAN's FID score of 6.98.

However, pure NE cannot be reliably reached by existing algorithms as pure NE may not exist [7, 21]. This also leads to unstable training in GANs depending on the data and the hyperparameters. Therefore, mixed NE is a more suitable solution concept [11]. Several recent works propose mixture architectures with multiple generators and discriminators that consider mixed NE such as MIX+GAN [2] and MGAN [10] but they cannot guarantee to converge to mixed NE. Mirror-GAN [11] computes the mixed NE by sampling over the infinite-dimensional strategy space and proposes provably convergent proximal methods. However, the sampling approach may not be efficient as mixed NE may only have a few strategies in the support set.

*Equal contribution
†Corresponding author

Double Oracle (DO) algorithm [20] is a powerful framework to compute mixed NE in large-scale games. The algorithm starts with a restricted game that is initialized with a small set of actions and solves it to get the NE strategies of the restricted game. The algorithm then computes players' best-responses using oracles to the NE strategies and add them into the restricted game for the next iteration. DO framework has been applied in various disciplines [4, 13], as well as Multi-agent Reinforcement Learning (MARL) [15].

Inspired by successful applications of DO framework, we, for the first time, propose a Double Oracle Framework for Generative Adversarial Networks (DO-GAN). This paper presents four key contributions. First, we treat the generator and the discriminator as players and obtain the best responses from their oracles and add the utilities to a meta-matrix. Second, we propose a linear program to obtain the probability distributions of the players' pure strategies (meta-strategies) for the respective oracles. The linear program computes an exact mixed NE of the meta-matrix game in polynomial time. Third, since multiple generators and discriminator from the best responses oracles are stored in the memory, the algorithm may be memory-inefficient for problems to train GAN with large-scaled real-world datasets. Thus, we propose two solutions for the scalable double oracle framework: 1) a pruning method for reducing the support set of best response strategies to prevent the oracles from becoming intractable as there is a risk of the meta-matrix growing very large with each iteration of oracle training; 2) applying continual learning to retain the previous knowledge of the networks for the best responses from the generator and discriminator oracles in the multi-task learning setup. We also address the problems in continual learning such as catastrophic forgetting. Finally, we provide comprehensive evaluation on the performance of DO-GAN with different GAN architectures using both synthetic and real-world datasets. Experiment results show that DO-GAN variants have significant improvements in terms of both subjective qualitative evaluation and quantitative metrics such as inception score and FID score.

2. Related Works

In this section, we briefly introduce existing GAN architectures, double oracle algorithm and its applications such as policy-state response oracles that are related to our work.

GAN Architectures. Various GAN architectures have been proposed to improve the performance of GANs. Deep Convolutional GAN (DCGAN) [28] replaces fully-connected layers in the generator and the discriminator with deconvolution layer of Convolutional Neural Networks (CNN). Weight normalization techniques such as Spectral Normalization GAN (SNGAN) [24] stabilize the training of the discriminator and reduce the intensive hyperparameters tuning. There are also multi-model architectures such as Stacked Generative Adversarial Networks (SGAN) [12] that consist of a top-down stack of generators and a bottom-up discriminator network. Each generator is trained to generate lower-level representations conditioned on higher-level representations that can fool the corresponding representation discriminator. Training GANs is very hard and unstable as pure NE for GANs might not exist and cannot be reliably reached by the existing approaches [21]. Considering mixed NE, MIX+GAN [2] maintains a mixture of generators and discriminators with the same network architecture but have their own trainable parameters. However, training a mixture of networks without parameter sharing makes the algorithm computationally expensive. Mixture Generative Adversarial Nets (MGAN) [10] propose to capture diverse data modes by formulating GAN as a game between a classifier, a discriminator and multiple generators with parameter sharing. However, MIX+GAN and MGAN cannot converge to mixed NE. Mirror-GAN [11] finds the mixed NE by sampling over the infinite-dimensional strategy space and proposes provably convergent proximal methods. The sampling approach may be inefficient to compute mixed NE as the mixed NE may only have a few strategies with positive probabilities in the infinite strategy space.

Double Oracle Algorithm. Double Oracle (DO) algorithm starts with a small restricted game between two players and solves it to get the players' strategies at Nash Equilibrium (NE) of the restricted game. The algorithm then exploits the respective best response oracles for additional strategies of the players. The DO algorithm terminates when the best response utilities are not higher than the equilibrium utility of the current restricted game, hence, finding the NE of the game without enumerating the entire strategy space. Moreover, in two-player zero-sum games, DO converges to a min-max equilibrium [20]. DO framework is used to solve large-scale normal-form and extensive-form games such as security games [13, 38], poker games [40] and search games [5]. DO framework is also used in MARL settings [15, 26]. Policy-Space Response Oracles (PSRO) generalize the double oracle algorithm in a multi-agent reinforcement learning setting [15]. PSRO treats the players' policies as the best responses from the agents' oracles, builds the meta-matrix game and computes the mixed NE but it uses Projected Replicator Dynamics (PRD) that updates the changes in the probability of each player's policy at each iteration. Since PRD needs to simulate the update for several iterations, the use of PRD takes a longer time to compute the meta-strategies and does not guarantee to compute an exact NE of the meta-matrix game. However, in DO-GAN, we can use a linear program to compute the players' meta-strategies in polynomial time since GAN is a two-player zero-sum game [35]. We present the corresponding terminologies between GAN and game theory in Appendix A.

Continual Learning and Catastrophic Forgetting. Continual learning in GANs has been ongoing research to combine a network's knowledge through time or knowledge of multiple networks to a single network. Continual Learning in GANs [36] employed Elastic Weight Consolidation (EWC) to remedy the catastrophic forgetting in GANs continual training. MGAN [10] and GMAN [6] have employed continual learning to multiple generators and multiple discriminators respectively. Our work is closely related to Bayesian GAN [33] which assigns a posterior over the multiple networks of generator and discriminator. However, we cannot directly adapt the work as it only assigns a distribution to multiple generators and discriminators with a Bayesian formula without a single continual network while we assign the distributions to the generator/discriminator tasks of a continual learning architecture by solving a meta-game.

3. Preliminaries

In this section, we mathematically explain the preliminary works to effectively our DO-GAN approach.

3.1. Generative Adversarial Networks

Generative Adversarial Networks (GANs) [8] have become one of the dominant methods for fitting generative models to complicated real-life data. GANs are deep neural net architectures comprised of two neural networks trained in an adversarial manner to generate data that resembles a distribution. The first neural network, a generator G, is given some random distribution $p_{\mathbf{z}}(\mathbf{z})$ on the input noise $\mathbf{z}$ and a real data distribution $p_{data}(\mathbf{x})$ on training data $\mathbf{x}$. The generator is supposed to generate as close as possible to $p_{data}(\mathbf{x})$. The second neural network, a discriminator D, is to discriminate between two different classes of data (real or fake) from the generator.

Let the generator's differentiable function be denoted as $G(\mathbf{z}, \uparrow_g)$ and similarly $D(\mathbf{x}, \uparrow_d)$ for the discriminator, where G and D are two neural networks with parameters $\uparrow_g$ and $\uparrow_d$. Thus, $D(\mathbf{x})$ represents the probability that $\mathbf{x}$ comes from the real data. The generator loss L_G and the discriminator loss L_D are defined as:

$$L_D = \mathbb{E}_{\mathbf{x} \leftarrow p_{data}(\mathbf{x})}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \leftarrow p_{\mathbf{z}}(\mathbf{z})}[-\log(1 - D(G(\mathbf{z}))], \quad (1)$$

$$L_G = \mathbb{E}_{\mathbf{z} \leftarrow p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}))]. \quad (2)$$

GAN is then set up as a two-player zero-sum game between G and D as follows:

$$\min_G \max_D \; \mathbb{E}_{\mathbf{x} \leftarrow p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \leftarrow p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}))]. \quad (3)$$

During training, the parameters of G and D are updated alternately until we reach the global optimal solution $D(G(\mathbf{z})) = 0.5$. Next, we let $\Uparrow_g$ and $\Uparrow_d$ be the set of parameters for G and D, considering the probability distributions σ_g and σ_d, the mixed strategy formulation [11] is:

$$\min_g \max_d \; \mathbb{E}_{\uparrow_d \leftarrow \sigma_d} \mathbb{E}_{\mathbf{x} \leftarrow p_{data}(\mathbf{x})}[\log D(\mathbf{x}, \uparrow_d)] + \mathbb{E}_{\uparrow_d \leftarrow \sigma_d} \mathbb{E}_{\uparrow_g \leftarrow \sigma_g} \mathbb{E}_{\mathbf{z} \leftarrow p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}, \uparrow_g), \uparrow_d)]. \quad (4)$$

Similarly to GANs, DCGAN, SNGAN and SGAN can also be viewed as two-player zero-sum games with mixed strategies of the players. DCGAN modifies the vanilla GAN by replacing fully-connected layers with the convolutional layers. SGAN trains multiple generators and discriminators using the loss as a linear combination of 3 loss terms: adversarial loss, conditional loss and entropy loss.

3.2. Double Oracle Algorithm

A normal-form game is a tuple $(\Uparrow, U, n)$ where n is the number of players, $\Uparrow = (\Uparrow_1, \ldots, \Uparrow_n)$ is the set of strategies for each player $i \in N$, where $N = \{1, \ldots, n\}$ and $U : \Uparrow \rightarrow \mathbb{R}^n$ is a payoff table of utilities R for each joint policy played by all players. Each player chooses the strategy to maximize own expected utility from $\Uparrow_i$, or by sampling from a distribution over the set of strategies $\sigma_i \in \Delta(\Uparrow_i)$. We can use linear programming, fictitious play [3] or regret minimization [32] to compute the probability distribution over players' strategies.

In the Double Oracle (DO) algorithm [20], there are two best response oracles for the row and column player respectively. The algorithm creates restricted games from a subset of strategies at the point of each iteration t for row and column players, i.e., $\Uparrow^t_r \subset \Uparrow_r$ and $\Uparrow^t_c \subset \Uparrow_c$ as well as a meta-matrix U^t at the t^{th} iteration. We then solve the meta-matrix to get the probability distributions on $\Uparrow^t_r$ and $\Uparrow^t_c$. Given a probability distribution σ_c of the column player strategies, $BR_r(\sigma_c)$ gives the row player's best response to σ_c. Similarly, given probability distribution σ_r of the row player's strategies, $BR_c(\sigma_r)$ is the column player's best response to σ_r. The best responses are added to the restricted game for the next iteration. The algorithm terminates when the best response utilities are not higher than the equilibrium utility of current restricted game. Although in the worst-case, the entire strategy space may be added to the restricted game, DO is guaranteed to converge to mixed NE in two-player zero-sum games. DO is also extended to the multi-agent reinforcement learning in PSRO [15] to approximate the best responses to the mixtures of agents' policies, and compute the meta-strategies for the policy selection.

4. DO-GAN

As discussed in previous sections, computing mixed NE for GANs is challenging as there is an extremely large number of pure strategies, i.e., possible parameter settings of the generator and discriminator networks. Thus, we propose

a double oracle framework for GANs (DO-GAN) to compute the mixed NE efficiently. DO-GAN builds a restricted meta-matrix game between the two players and computes the mixed NE of the meta-matrix game, then DO-GAN iteratively adds more generators and discriminators into the meta-matrix game until termination.

4.1. General Framework of DO-GAN

GAN can be translated as a two-player zero-sum game between the generator player g and the discriminator player d. To compute the mixed NE of GANs, at iteration t, DO-GAN creates a restricted meta-matrix game U^t with the trained generators and discriminators as strategies of the two players, where the generators and discriminators are parameterized by $\uparrow_g \in G$ and $\uparrow_d \in D$. We use $U^t(\uparrow_g, \uparrow_d)$ to denote the generator player's payoff when playing $\uparrow_g$ against $\uparrow_d$, which is defined as L_D. Since GAN is zero-sum, the discriminator player's payoff is $-U^t(\uparrow_g, \uparrow_d)$. We define $0'^t_g$ and $0'^t_d$ as the mixed strategies of generator player and discriminator player, respectively. With a slight abuse of notation, we define the generator player's expected utility of mixed strategies $\langle 0'^t_g, 0'^t_d \rangle$ as $U^t(0'^t_g, 0'^t_d) = \sum_{\uparrow_g \in G} \sum_{\uparrow_d \in D} 0'^t_g(\uparrow_g) \cdot 0'^t_d(\uparrow_d) \cdot U^t(\uparrow_g, \uparrow_d)$. We use $\langle 0'^{t\leftarrow}_g, 0'^{t\leftarrow}_d \rangle$ to denote mixed NE of the restricted meta-matrix game U^t. We solve U^t to obtain the mixed NE, compute best responses and add them into U^t for next iteration. Figure 2 presents an illustration of DO-GAN and Algorithm 1 describes the overview of the framework.

Our algorithm starts by initializing two arrays G and D to store multiple generators and discriminators (line 1). We train the first $\uparrow_g$ and $\uparrow_d$ with the canonical training procedure of GANs (line 2). We store the parameters of trained models in G and D (line 3), compute the adversarial loss L_D and add it to the meta-matrix U^0 (line 4). We initialize the meta-strategies $0'^{0\leftarrow}_g = [1]$ and $0'^{0\leftarrow}_d = [1]$ since there is only one pair of generator and discriminator available (line 5). For each epoch, we use `generatorOracle()` and `discriminatorOracle()` to obtain best responses $\uparrow^0_g$ and $\uparrow^0_d$ to $0'^{t\leftarrow}_d$ and $0'^{t\leftarrow}_g$ via Adam Optimizer, respectively, then add them into G and D (lines 7-10). We then augment U^{t-1} by adding $\uparrow^0_g$ and $\uparrow^0_d$ and calculating $U^t(\uparrow^0_g, \uparrow^0_d)$ to obtain U^t and compute the missing entries (line 11). We compute the missing payoff entries $U^t(\uparrow^0_g, \uparrow_d), \forall \uparrow_d \in D$ and $U^t(\uparrow_g, \uparrow^0_d), \forall \uparrow_g \in G$ by sampling a few batches of training data. After that, we compute the mixed NE $\langle 0'^{t\leftarrow}_g, 0'^{t\leftarrow}_d \rangle$ for U^t with linear programming (line 12). The algorithm terminates if the criteria described in Algorithm 2 is satisfied (line 13).

In `generatorOracle()`, we train $\uparrow^0_g$ to obtain the best response against $0'^{t\leftarrow}_d$, i.e., $U^t(\uparrow^0_g, 0'^{t\leftarrow}_d) \geq U^t(\uparrow_g, 0'^{t\leftarrow}_d), \forall \uparrow_g \in \Uparrow_g$. Similarly, in `discriminatorOracle()`, we train $\uparrow_d$ to obtain the best response against $0'^{t\leftarrow}_g$, i.e., $U^t(0'^{t\leftarrow}_g, \uparrow^0_d) \geq U^t(0'^{t\leftarrow}_g, \uparrow_d), \forall \uparrow_d \in \Uparrow_d$. Full details of generator oracle and discriminator oracle can be found in Appendix B.

Algorithm 1: DO-GAN()

1 Initialize generator and discriminator arrays $G = \emptyset$; and $D = \emptyset$;
2 Train generator & discriminator to get the first $\uparrow_g$ and $\uparrow_d$;
3 $G \leftarrow G \frown \{\uparrow_g\}$; $D \leftarrow D \frown \{\uparrow_d\}$;
4 Compute the adversarial loss L_D and add it to meta-matrix U^0;
5 Initialize $0'^{0\leftarrow}_g = [1]$ and $0'^{0\leftarrow}_d = [1]$;
6 **for** *epoch* $t \in \{1, 2, \ldots\}$ **do**
7 $\quad \uparrow^0_g \leftarrow$ `generatorOracle`$(0'^{t\leftarrow}_d, D)$;
8 $\quad G \leftarrow G \frown \{\uparrow^0_g\}$;
9 $\quad \uparrow^0_d \leftarrow$ `discriminatorOracle`$(0'^{t\leftarrow}_g, G)$;
10 $\quad D \leftarrow D \frown \{\uparrow^0_d\}$;
11 $\quad$ Augment U^{t-1} with $\uparrow^0_g$ and $\uparrow^0_d$ to obtain U^t and compute missing entries;
12 $\quad$ Compute mixed NE $\langle 0'^{t\leftarrow}_g, 0'^{t\leftarrow}_d \rangle$ for U^t with linear program; // Section 4.2
13 $\quad$ **if** `TerminationCheck`$(U^t, 0'^{t\leftarrow}_g, 0'^{t\leftarrow}_d)$ **then** **break**; // Section 4.3

4.2. Linear Program for Meta-matrix Game

Since the current restricted meta-matrix game U^t is a zero-sum game, we can use a linear program to compute the mixed NE in polynomial time [35]. Given the generator player g's mixed strategy $0'^t_g$, the discriminator player d will play strategies that minimize the expected utility of g. Thus, the mixed NE strategy for the generator player $0'^{t\leftarrow}_g$ is to maximize the worst-case expected utility, which is obtained by solving the following linear program:

$$0'^{t\leftarrow}_g = \arg\max_{0'^t_g} \{v : 0'^t_g \in \mathbb{0}, \sum_{i \in G} 0'^t_g(i) = 1, U^t(0'^t_g, \uparrow_d) \geq v, \forall \uparrow_d \in D\}. \quad (5)$$

Similarly, we can obtain the mixed NE strategy for the discriminator $0'^{t\leftarrow}_d$ by solving a linear program that maximizes the worst-case expected utility of the discriminator player. Therefore, we obtain the mixed NE $\langle 0'^{t\leftarrow}_g, 0'^{t\leftarrow}_d \rangle$ of the restricted meta-matrix game U^t.

4.3. Termination Check

DO terminates the training by checking whether the best response $\uparrow^0_g$ (or $\uparrow^0_d$) is in the support set G (or D) [13], but we cannot apply this approach to DO-GAN as GAN has infinite-dimensional strategy space [11]. Hence, we terminate the training if the best responses cannot bring a higher utility to the two players than the entries of the current support sets, as discussed in [15, 26]. Specifically, we first compute $U^t(0'^{t\leftarrow}_g, 0'^{t\leftarrow}_d)$ and the expected utilities for new generator and discriminator $U^t(G[m], 0'^{t\leftarrow}_d)$, $U^t(0'^{t\leftarrow}_g, D[n])$ (line 1-3). Then, we calculate the utility increment (lines 4-5) and returns **True** if both $U^t(G[m], 0'^{t\leftarrow}_d)$ and $U^t(0'^{t\leftarrow}_g, D[n])$ cannot bring a higher utility than $U^t(0'^{t\leftarrow}_g, 0'^{t\leftarrow}_d)$ by ϵ (lines 6-8).

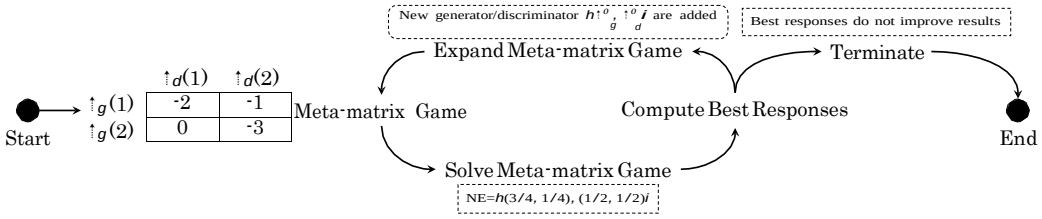

Figure 2. An illustration of DO-GAN. Figure adapted from [15].

Algorithm 2: TerminationCheck($U^t, \sigma_g^{t-}, \sigma_d^{t-}$)

// U^t is of size $m \times n$
// $|G| = m$, $|D| = n$
Compute $U^t(\sigma_g^{t-}, \sigma_d^{t-})$;
Compute $U^t(\sigma_g^{t-}, D[n])$;
Compute $U^t(G[m], \sigma_d^{t-})$;
$genInc = U^t(G[m], \sigma_d^{t-}) - U^t(\sigma_g^{t-}, \sigma_d^{t-})$;
$disInc = -U^t(\sigma_g^{t-}, D[n]) - (-U^t(\sigma_g^{t-}, \sigma_d^{t-}))$;
if $genInc < \epsilon$ && $-disInc < \epsilon$ **then**
  return True
else return False ;

Algorithm 3: DO-GAN/P

// I stores indices to be pruned from G and D
// G stores models to be pruned from G and D
DO-GAN();
$I_g = \{\}$; $I_d = \{\}$
$K_g = \{\}$; $K_d = \{\}$ **if** $|G| > s$ **then**
  for $i \in \{0, \ldots, |G| - 1\}$ **do**
    if $\sigma_g^{t-}(G[i]) == \min \sigma_g^{t-}$ **then** $I_g \leftarrow I_g \cup \{i\}$;
    $K_g \leftarrow K_g \cup \{G[i]\}$;
if $|D| > s$ **then**
  for $j \in \{0, \ldots, |D| - 1\}$ **do**
    if $\sigma_d^{t-}(D[j]) == \min \sigma_d^{t-}$ **then** $I_d \leftarrow I_d \cup \{j\}$;
    $K_d \leftarrow K_d \cup \{D[j]\}$;
$G \leftarrow G \setminus K_g$; $D_t \leftarrow D_T \setminus K_d$;
$U \leftarrow J_{I_g,m} \cdot U \cdot J_{I_d,n}$

Algorithm 4: DO-GAN/C

Initialize generator and discriminator task arrays $G = \{\}$ and $D = \{\}$;
Train generator & discriminator to get with the first task to get σ_g^0 and σ_d^0;
$G \leftarrow G \cup \{\sigma_g^0\}$; $D \leftarrow D \cup \{\sigma_d^0\}$;
Compute the adversarial loss L_D and add it to meta-matrix U^0;
Initialize $\sigma_g^{0-} = [1]$ and $\sigma_d^{0-} = [1]$;
for *epoch* $t \in \{1, 2, \ldots\}_t$ **do**
  Create new tasks σ_g^t and σ_d^t;
  $\sigma_g^t \leftarrow$ generatorOracle(σ_d^{t-}, D);
  $G \leftarrow G \cup \{\sigma_g^t\}$;
  $\sigma_d^t \leftarrow$ discriminatorOracle(σ_g^{t-}, G);
  $D \leftarrow D \cup \{\sigma_d^t\}$;
  if $t > 2$ **then**
    $G \leftarrow G \setminus \{\sigma_g^{t-2}\}$ and $D \leftarrow D \setminus \{\sigma_d^{t-2}\}$;
  Create U^t with σ_g^{t-1}, σ_g^t and σ_d^{t-1}, σ_d^t;
  Compute mixed NE $\langle \sigma_g^{t-}, \sigma_d^{t-} \rangle$ for U^t with linear program; // Section 4.2
  if TerminationCheck($U^t, \sigma_g^{t-}, \sigma_d^{t-}$) **then**
    break; // Section 4.3

5. Practical Implementations

As the number of epochs grows during the training of DO, the number of networks and the size of the meta-matrix also grows. Hence, there is a risk that the support strategy set becomes very large and G and D become intractable. To make the algorithm practical and scalable, we propose two methods: DO-GAN with meta-matrix pruning (DO-GAN/P) and DO-GAN with continual learning (DO-GAN/C).

5.1. Meta-matrix Pruning (DO-GAN/P)

The first method is to prune the meta-matrix. Here, we adapt the greedy pruning algorithm, as depicted in Algorithm 3. When either $|G|$ or $|D|$ is greater than the limit of the support set size s, we prune at least one strategy with the least probability, which is the strategy that contributes the least to the player's winning. Specifically, we define $J_{I,b}$ where I is the set of row numbers to be removed, b is the total rows of a matrix. To remove the 2^{nd} row of a matrix having 3 rows, we define $I = \{1\}$, $b = 3$ and $J_{\{1\},3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

If $|G| > s$, at least one strategy with minimum probability is pruned from G, similarly for D (lines 4-10). Finally, we prune the meta-matrix using matrix multiplication (line 11).

5.2. Continual Learning (DO-GAN/C)

Our ablation studies show that we still need a support set of at least $s = 10$ for DO-GAN/P to converge and the time complexity grows as s increases. Thus, we further reduce both time and space complexity by making the network retain the knowledge of previous networks so that the algorithm will converge with even smaller support set. Hence, we propose to adapt continual learning to consolidate the

knowledge of multiple networks to a single network while setting $s = 2$ to reduce the space complexity as much as possible. We treat each network as a task to train the adaptive continual learning network while having a distribution over the tasks to represent the player's strategies.

To remedy the catastrophic forgetting i.e., all the generator tasks focus only towards fooling the newest discriminator, we adapt Elastic Weight Consolidation (EWC) method [17]. We change the generator loss function from non-saturating to saturating and add a penalty function accordingly. Let G be trained on task $t-1$ to have optimal parameters θ_g^{t-1}, the Fisher information F is:

$$F = \mathbb{E}_{z \sim p(z)}\left[\nabla_{\theta_g^{t-1}} \log D(G(\mathbf{z})|\theta_g^{t-1})^2\right] \quad (6)$$

After obtaining the Fisher information, we directly use F as a regularization loss to penalize the weight change during the training. Hence, G's loss function is augmented as:

$$L_G = \mathbb{E}_{z \sim p_z(z)}[-\log D(G(\mathbf{z})) + \lambda \cdot \theta_g^{t-1} \cdot \sum_i F_i(\theta_i - \theta_g^{t-1,i})^2 \quad (7)$$

where θ_g^{t-1} represents the parameters learned for task $t-1$, i is the index of each parameter of the generator model, and λ is the regularization weight.

Algorithm 4 describes the changes to $DO-GAN$ algorithm with continual learning. Instead of arrays G and D in $DO-GAN$, we initialize tasks arrays for generator and discriminator (line 1). At every epoch, we create new tasks and train the generator and discriminator networks to outperform the previous optimal parameters with the distribution at NE θ_d^{t-1} and θ_g^{t-1} respectively (lines 7-11). Then, we keep the previously and currently trained optimal parameters (line 13) to create meta-matrix, solve it to compute mixed-NE (lines 14-15). Finally, we perform the termination check (line 16).

Complexity. *Given the same architectures, the space complexity of DO-GAN is $O(t)$ where t is the number of epochs until convergence. In contrast, the space complexity of DO-GAN/P is $O(s)$ where s is the size of the support set. DO-GAN/C has the minimum storage complexity which is $O(1)$.*

In DO-GAN, we add a pair of generator and discriminator for every epoch of training. Thus, the space complexity of DO-GAN is $O(t)$, making the algorithm memory-inefficient to train with real-world datasets where it needs a large number of epochs to converge. The space complexity of DO-GAN/P is $O(s)$ since we prune the meta-matrix and the players' strategies if the $|G| > s$ or $|D| > s$ where s is the limit of the support set size. In DO-GAN/C, we train a single adaptive network storing the optimal strategies only for the tasks created at $(t-1)^{th}$ and t^{th} epochs. Thus, the space complexity of DO-GAN/C is kept at $O(1)$.

6. Experiments

We conduct our experiments on a machine with Xeon(R) CPU E5-2683 v3@2.00GHz and 4→ Tesla v100-PCIE-16GB running Ubuntu operating system. We evaluate DO-framework for established GAN architectures such as vanilla GAN [8], DCGAN [28], SNGAN [24] and SGAN [12]. We adopt the parameter settings and criterion of the GAN architectures as published. We set $s = 10$ unless mentioned otherwise. We compute the mixed NE of the meta game with Nashpy. According to the ablation studies by [17,36], we set λ as 1000 for MNIST, 5000 for CIFAR-10 and $5 \rightarrow 10^8$ for CelebA. The evaluation details are shown in Appendix C.

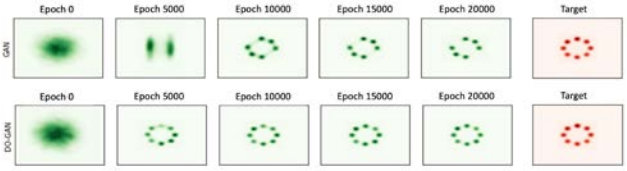

Figure 3. Comparison of GAN and DO-GAN/P on 2D synthetic Gaussian Mixture Dataset

6.1. Evaluation on 2D Gaussian Mixture Dataset

To illustrate the effectiveness of the architecture, we train a double oracle framework with the simple vanilla GAN architecture on a 2D mixture of 8 Gaussian mixture components with cluster standard deviation 0.1 which follows the experiment by [22]. Figure 3 shows the evolution of 512 samples generated by GAN and DO-GAN/P through 20000 epochs. The goal of GAN and DO-GAN/P is to correctly generate samples at 8 modes as shown in the target. The results show that GAN can only identify 6 out of 8 modes of the synthetic Gaussian data distribution, while the DO-GAN/P can obtain all the 8 modes of the distribution. Furthermore, DO-GAN/P takes shorter time (less than 5000 epochs) to identify all 8 modes of the data distribution. We present a more detailed evolution of data samples through the training process on 2D Gaussian Mixtures in Appendix D.

Ablations. We also varied the limit of support set size for DO-GAN/P with $s = 5, 10, 15$ and recorded the computation time as discussed in Appendix E. We found that the training cannot converge when $s = 5$ and takes a long time when $s = 15$. Thus, we chose $s = 10$ for the training.

6.2. Evaluation on Real-world Datasets

We evaluate the performance of the double oracle framework which takes several established GAN architectures as the backbone as discussed in Appendix H, i.e., GAN, DCGAN and SGAN with convolutional layers for deep neural networks of GAN as well as SNGAN which uses normalization techniques. We run experiments on MNIST [16],

CIFAR-10 [14] and CelebA [19]. MNIST contains 60,000 samples of handwritten digits with images of $28 \rightarrowtail 28$. CIFAR-10 contains $50,000$ training images of $32 \rightarrowtail 32$ of 10 classes. CelebA is a large-scaled face dataset with more than 200K images of size $128 \rightarrowtail 128$.

6.2.1 Qualitative Evaluation

We choose the CelebA dataset for the qualitative evaluation since the training images contain noticeable artifacts (aliasing, compression, blur) that make the generator difficult to produce perfect and faithful images. We compare performances of DO-DCGAN/P, DO-SNGAN/P and DO-SGAN/P with their counterparts. SNGAN which is trained for 40 epochs with termination ✏ of $5 \rightarrowtail 10^{-5}$ for DO-SNGAN/P where other architectures are trained for 25 epochs with termination ✏ of $5 \rightarrowtail 10^{-5}$ for DO variants. The generated CelebA images of DCGAN and DO-DCGAN/P are shown in Figure 4, where we find that DCGAN suffers mode-collapse, while DO-DCGAN/P does not. We also present the generated images of SNGAN vs DO-SNGAN/P using fixed noise at different training epochs in Figure 5. From the results, we can see that SNGAN, SGAN, DO-SNGAN/P and DO-SGAN/P are able to generate various faces, i.e., no mode-collapse. Judging from subjective visual quality, we find that DO-SNGAN/P and DO-SGAN/P are able to generate plausible images faster than SNGAN and SGAN during training, i.e., 17 epochs for DO-SGAN/P and 20 epochs for SGAN. More experimental results on CIFAR-10 and DO-GAN/C variants. which can produce competitive results, can be found in Appendix F.

Figure 4. Training images with fixed noise for DCGAN and DO-DCGAN/P until termination.

6.2.2 Quantitative Evaluation

Inception Score. We first leverage the Inception Score (IS) [34] by using Inception_v3 [37] as the inception model. To compute the inception score, we first compute the Kullback-Leibler (KL) divergence for all generated images and use the equation $\text{IS} = \exp(\mathsf{E}_{\mathbf{x}}[\text{KL}(D(p(y|\mathbf{x}) \,\mathsf{k}\, p(y)))])$ where $p(y)$ is the conditional label distributions for the images in the split and $p(y|\mathbf{x})$ is that of the image $\mathbf{x}$ estimated by the reference inception model. Inception score evaluates the quality and diversity of all generated images rather than the similarity to the real data from the test set.

FID Score. Fréchet Inception Distance (FID) measures the distance between the feature vectors of real and generated images using Inception_v3 model [9]. Here, we let p and q be the distributions of the representations obtained by projecting real and generated samples to the last hidden layer of Inception model. Assuming that p and q are the multivariate Gaussian distributions, FID measures the 2-Wasserstein distance between the two distributions. Hence, FID Score can capture the similarity of generated images to real ones better than inception score.

Figure 5. Training images with fixed noise for SNGAN and DO-SNGAN/P until termination.

Results. The results are shown in Table 1. In CIFAR-10 dataset, the pruning method i.e., DO-GAN/P, DO-DCGAN/P and DO-SNGAN/P obtain much better results (7.2 ± 0.16, 7.86 ± 0.14 and 8.55 ± 0.08) than GAN, DCGAN and SNGAN ($3.84{\pm}0.09$, $6.32{\pm}0.05$ and $7.58{\pm}0.12$). However, we do not see a significant improvement in DO-SGAN/P compared to SGAN $8.62{\pm}0.12$ and $8.69{\pm}0.10$ since SGAN already can generate diverse images. We did not include IS for CelebA dataset as IS cannot reflect the real image quality for CelebA, as observed in [9]. In CIFAR-10 dataset, DO-GAN/P, DO-DCGAN/P, DO-SNGAN/P and DO-SGAN/P obtain much lower FID scores (31.44, 22.25, 16.56, 18.20) respectively. The trend follows in CelebA obtaining 7.11 for DO-DCGAN/P while 10.92 for DCGAN, 7.62 for SNGAN while 6.92 for DO-SNGAN/P, 6.98 for SGAN and 6.32 for DO-SGAN/P respectively. Although we see a significant improvement in the quality of DO-SGAN/P images, FID score for DO-SGAN/P is affected by distortions.

We observe the competitive results for continual learning method with the pruning method: DO-GAN/C, DO-DCGAN/C, DO-SNGAN/C and DO-SGAN/C obtain inception scores of 7.32 ± 0.30, 8.04 ± 0.22, 8.54 ± 0.16 and 9.78 ± 0.11 respectively as well as FID scores of 26.93, 21.50, 19.57 and 16.07 respectively for CIFAR-10 dataset. Moreover, 7.16, 6.74 and 6.30 respectively for CelebA

Table 1. Inception scores (higher is better) and FID scores (lower is better) of DO-GAN with pruning (DO-GAN/P) and continual learning (DO-GAN/C). The mean and standard deviation are drawn from running 10 splits on 10000 generated images.

	Inception Score		FID Score	
	MNIST	CIFAR-10	CIFAR-10	CelebA
GAN	1.04±0.05	3.84±0.09	71.44	-
DCGAN	1.26±0.05	6.32±0.05	37.66	10.92
SNGAN	1.35±0.11	7.58±0.12	25.50	7.62
SGAN	1.39±0.09	8.62±0.12	24.83	6.98
MIX+DCGAN	-	7.72±0.09	-	-
MGAN	-	8.33±0.10	26.70	-
DCGAN+ EWC	-	7.58±0.07	25.51	-
DO-GAN/P	1.39 ± 0.09 (+0.35)	7.20 ± 0.16 (+3.36)	31.44 (−40.00)	-
DO-DCGAN/P	1.42 ± 0.11 (+0.16)	7.86 ± 0.14 (+1.54)	22.25 (−15.41)	7.11 (−3.81)
DO-SNGAN/P	**1.42±0.07** (+0.07)	8.55 ± 0.08 (+0.97)	18.20 (−7.30)	6.92 (−0.70)
DO-SGAN/P	1.42 ± 0.09 (+0.03)	**8.69±0.10** (+0.07)	**16.56** (−8.27)	**6.32** (−0.66)
DO-GAN/C	1.42 ± 0.10 (+0.38)	7.32 ± 0.30 (+3.48)	26.93 (−44.51)	-
DO-DCGAN/C	1.43 ± 0.13 (+0.17)	8.04 ± 0.22 (+1.72)	21.50 (−16.16)	7.16 (−3.76)
DO-SNGAN/C	1.43 ± 0.08 (+0.08)	8.54 ± 0.16 (+0.96)	19.57 (−5.93)	6.74 (−0.88)
DO-SGAN/C	**1.45±0.15** (+0.06)	**9.78±0.11** (+1.16)	**16.07** (−8.76)	**6.30** (−0.68)

Note: MIX+DCGAN and MGAN results are directly copied from [2, 10]. The magenta values are the improvements from non-DO counterparts.

dataset. We also compared with DCGAN+EWC which uses continual learning without the double oracle framework for CIFAR-10 dataset and obtained better results where DCGAN+EWC obtained inception score of 7.58 ± 0.07 and FID score of 25.51.

Table 2. Wall-clock time comparison of SGAN variants

	SGAN	DO-SGAN/P	DO-SGAN/C
Time (GPU-Hrs)	143.28	119.04	**97.54**

Note: SGAN is trained for 500 epochs on CIFAR-10. DO-SGAN/P converged at 288 epochs and DO-SGAN/C at 236 epochs.

From the results, we can see that DO framework performs better than each of their original counterpart architectures with both methods: pruning and continual learning. More details can be found in Appendix G. We can also see that continual learning method that uses least storage space can obtain competitive results with DO-GAN/P without memory storage limitations or pruning the players' strategies.

We also compare the wall-clock running times of DO-SGAN/P and DO-SGAN/C with SGAN as shown in Table 2. We let SGAN train for 500 epochs on CIFAR-10 dataset. Meanwhile, DO-SGAN/P converged at 288 epochs and DO-SGAN/C converged at 236 epochs. The recorded GPU hours of 143.28 for SGAN, 119.04 for DO-SGAN/P and 97.54 for DO-SGAN/C show that DO variants are more efficient.

7. Conclusion

We propose a novel double oracle framework to GANs, which starts with a restricted game and incrementally adds the best responses of the generator and the discriminator oracles as the players' strategies. We then compute the mixed NE to get the players' meta-strategies by using a linear program. We also propose two approaches to make the solution scalable including pruning the support strategy set and continual learning with an adaptive architecture to store the multiple networks of generators and discriminators. We apply DO-GAN approach to established GAN architectures such as vanilla GAN, DCGAN, SNGAN and SGAN. Extensive experiments with the synthetic 2D Gaussian mixture dataset as well as real-world datasets such as MNIST, CIFAR-10 and CelebA show that DO-GAN variants have significant improvements in comparison to their respective GAN architectures in terms of both subjective image quality and quantitative metrics.

Acknowledgement

This research was supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2019-0013), National Satellite of Excellence in Trustworthy Software Systems (Award No: NSOE-TSS2019-01), and NTU.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, pages 214–223, 2017. 1

[2] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (GANs). In *ICML*, pages 224–232, 2017. 1, 2, 8

[3] Ulrich Berger. Brown's original fictitious play. *Journal of Economic Theory*, 135(1):572–578, 2007. 3

[4] Branislav Bošanský, Christopher Kiekintveld, Viliam Lisy, Jiri Cermak, and Michal Pechoucek. Double-oracle algorithm for computing an exact Nash equilibrium in zero-sum extensive-form games. In *AAMAS*, pages 335–342, 2013. 2

[5] B Bosansky, Christopher Kiekintveld, Viliam Lisy, and Michal Pechoucek. Iterative algorithm for solving two-player zero-sum extensive-form games with imperfect information. In *ECAI*, pages 193–198, 2012. 2

[6] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. In *ICLR*, 2017. 3

[7] Farzan Farnia and Asuman Ozdaglar. GANs may have no Nash equilibria. *arXiv preprint arXiv:2002.09124*, 2020. 1

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014. 1, 3, 6

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, pages 6626–6637, 2017. 7, vi

[10] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. MGAN: Training generative adversarial nets with multiple generators. In *ICLR*, 2018. 1, 2, 3, 8

[11] Ya-Ping Hsieh, Chen Liu, and Volkan Cevher. Finding mixed nash equilibria of generative adversarial networks. In *ICML*, pages 2810–2819, 2019. 1, 2, 3, 4

[12] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. In *CVPR*, pages 5077–5086, 2017. 2, 6

[13] 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 *AAMAS*, pages 327–334, 2011. 2, 4

[14] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research). https://www.cs.toronto.edu/ kriz/cifar.html, 2009. 7

[15] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *NeurIPS*, pages 4190–4203, 2017. 2, 3, 4, 5

[16] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. 6

[17] Yijun Li, Richard Zhang, Jingwan Lu, and Eli Shechtman. Few-shot image generation with elastic weight consolidation. In *NeurIPS*, 2020. 6, iii

[18] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NeurIPS*, pages 700–708, 2017. 1

[19] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738, 2015. 7

[20] H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *ICML*, pages 536–543, 2003. 2, 3

[21] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of GANs. In *NeurIPS*, pages 1825–1835, 2017. 1, 2

[22] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017. 6

[23] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1

[24] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *ICLR*, 2018. 2, 6

[25] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2018. 1

[26] Paul Muller, Shayegan Omidshafiei, Mark Rowland, Karl Tuyls, Julien Perolat, Siqi Liu, Daniel Hennes, Luke Marris, Marc Lanctot, Edward Hughes, et al. A generalized training approach for multiagent learning. In *ICLR*, 2020. 2, 4

[27] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. In *NeurIPS*, pages 2352–2360, 2016. 1

[28] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 1, 2, 6, vi

[29] Mohamed Ragab, Zhenghua Chen, Min Wu, Chuan Sheng Foo, Chee Keong Kwoh, Ruqiang Yan, and Xiaoli Li. Contrastive adversarial domain adaptation for machine remaining useful life prediction. *IEEE Transactions on Industrial Informatics*, 17(8):5239–5249, 2020. 1

[30] Mohamed Ragab, Zhenghua Chen, Min Wu, Haoliang Li, Chee-Keong Kwoh, Ruqiang Yan, and Xiaoli Li. Adversarial multiple-target domain adaptation for fault classification. *IEEE Transactions on Instrumentation and Measurement*, 70:1–11, 2020. 1

[31] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016. 1

[32] Tim Roughgarden. Algorithmic game theory. *Communications of the ACM*, 53(7):78–86, 2010. 3

[33] Yunus Saatchi and Andrew Gordon Wilson. Bayesian gan. *arXiv preprint arXiv:1705.09558*, 2017. 3

[34] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *NeurIPS*, pages 2234–2242, 2016. 7

[35] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998. 2, 4

[36] Ari Seff, Alex Beatson, Daniel Suo, and Han Liu. Continual learning in generative adversarial nets. *arXiv preprint arXiv:1705.08395*, 2017. 3, 6, iii

[37] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. 7

[38] Jason Tsai, Thanh H Nguyen, and Milind Tambe. Security games for controlling contagion. In *AAAI*, pages 1464–1470, 2012. 2

[39] Zhengwei Wang, Qi She, and Tomas E Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *arXiv preprint arXiv:1906.01529*, 2019. vii, xi

[40] Kevin Waugh, Nolan Bard, and Michael Bowling. Strategy grafting in extensive games. In *NeurIPS*, pages 2026–2034, 2009. 2