Towards Distributed Coevolutionary GANs

Abdullah Al-Dujaili, Tom Schmiedlechner, Erik Hemberg and Una-May O’Reilly
aldujail@mit.edu, tschmied@mit.edu, hembergerik@csail.mit.edu, unamay@csail.mit.edu
CSAIL, MIT, USA

Abstract
Generative Adversarial Networks (GANs) have become one of the dominant methods for deep generative modeling. Despite their demonstrated success on multiple vision tasks, GANs are difficult to train and much research has been dedicated towards understanding and improving their gradient-based learning dynamics. Here, we investigate the use of coevolution, a class of black-box (gradient-free) co-optimization techniques and a powerful tool in evolutionary computing, as a supplement to gradient-based GAN training techniques. Experiments on a simple model that exhibits several of the GAN gradient-based dynamics (e.g., mode collapse, oscillatory behavior, and vanishing gradients) show that coevolution is a promising framework for escaping degenerate GAN training behaviors.

Introduction
Generative modeling aims to learn functions that express distributional outputs. In a standard setup, generative models take a training set drawn from a specific distribution and learn to represent an estimate of that distribution. By estimate, we mean either an explicit density estimation, the ability to generate samples, or the ability to do both [14]. GANs [15] are a framework for training generative deep models via an adversarial process. They have been applied with celebrated success to a growing body of applications. Typically, a GAN pairs two networks, viz. a generator and a discriminator. The goal of the generator is to produce a sample (e.g., an image) from a latent code such that the distribution of the produced samples is indistinguishable from the true data (training set) distribution. In tandem, the discriminator plays the role of a critic to do the assessment and tell whether the samples are true data or generated by the generator. Concurrently, the discriminator is trained to discriminate optimally (maximize its accuracy), while the generator is trained to fool the discriminator (minimize its accuracy). Despite their witnessed success, it is well known that GANs are difficult to optimize. From a game theory perspective, GAN training can be seen as a two-player minimax game. Since the two networks are differentiable, optimizing the minimax GAN objective is typically carried out by (variants of) simultaneous gradient-based updates to their parameters. While it has been shown that simultaneous gradient updates converge if they are made in function space, the same proof does not apply to these updates in parameter space [15]. On the one hand, a zero gradient is a necessary condition for standard optimization to converge. On the other hand, equilibrium is the corresponding necessary condition in a two-player game [4]. In practice, gradient-based GAN training often oscillates without ultimately reaching an equilibrium. Moreover, a variety of degenerate behaviors have been observed—e.g., mode collapse [3], discriminator collapse [27], and vanishing gradients [2]. These unstable learning dynamics have been the focus of several investigations by the deep learning community, seeking a principled theoretical understanding as well as practical algorithmic improvements and heuristics [2, 8, 16]. Two-player minimax black-box optimization and games have been a topic of recurrent interest in the evolutionary computing community [24, 38]. In seminal work, Hillis [19] showed that more efficient sorting programs can be produced by competitively co-evolving them versus their testing programs. Likewise, Herrmann [18] proposed a two-space genetic algorithm as a general technique to solve minimax optimization problems and used it to solve a parallel machine scheduling problem with uncertain processing times. In competitive co-evolution, two different populations, namely solutions and tests, coevolve against each other [13]. The quality of a solution is determined by its performance when interacting with the tests. Reciprocally, a test’s quality is determined by its performance when interacting with the solutions, leading to what is commonly referred to as evolutionary arms race [10].

In this paper, we propose to pair a coevolutionary algorithm with conventional GAN training, asking whether the combination is powerful enough to more frequently avoid degenerate training behaviors. The motivation behind our proposition is of two-fold. First, most of the pathological behaviors encountered with gradient-based GAN training have been identified and studied by the evolutionary computing community decades ago—e.g., focusing, relativism, and loss of gradients [35, 47]. Second, there has been a growing body of work, which shows that the performance of gradient-based methods can be rivaled by evolutionary-based counterparts when combined with sufficient computing resources and data [42, 43]. The aim of this paper is to bridge the gap between works of the deep learning and evolutionary computing communities towards a better understanding of gradient-based and gradient-free GAN dynamics. Indeed,
one can see that the Nash Equilibrium solution concept in coevolutionary literature \cite{37} is not that different from the notion of GAN mixtures in GAN literature \cite{4}.

We report the following contributions: i) For a simple parametric generative modeling problem \cite{27} that exhibits several degenerate behaviors with gradient-based training, we validate the effectiveness of combining coevolution with gradient-based updates (mutations). ii) We present Lipizzaner, a coevolutionary framework to train GANs with gradient-based mutations (for neural net parameters) and gradient-free mutations (for hyperparameters) and learn a mixture of GANs.

**Related Work**

**Training GANs** Several gradient-based GAN training variants have been proposed to improve and stabilize its dynamics. One variant category is focused on improving training techniques for single-generator single-discriminator networks. Examples include modifying the generator’s objective \cite{46}, the discriminator’s objective \cite{30}, or both \cite{3, 41}. Some of these propositions are theoretically well-founded, but convergence still remains elusive in practice. The second category employs a framework of multiple generators and/or multiple discriminators. Examples include training multiple discriminators \cite{11}; training an array of specialized discriminators, each of which looks at a different random low-dimensional projection of the data \cite{33}; sequentially training and adding new generators with boosting techniques \cite{44}; training a cascade of GANs \cite{45}; training multiple generators and discriminators in parallel (GAP) \cite{22}; training a classifier, a discriminator, and a set of generators \cite{20}; and optimizing a weighted average reward over pairs of generators and discriminators (MIX+GAN) \cite{4}. For a theoretical view on GAN training, the reader may refer to \cite{22, 27, 4}.

**Coevolutionary Algorithms for Minimax Problems.** Variants of competitive coevolutionary algorithms have been used to solve minimax formulations in the domains of constrained optimization \cite{17}, mechanical structure optimization \cite{6}, and machine scheduling \cite{13}. These early coevolutionary propositions were tailored to symmetric minimax problems. In practice, the symmetry property may not always hold. In fact, mode collapse in GANs may arise from asymmetry \cite{14}. To address this issue, asymmetric fitness evaluation was presented in \cite{21} and analyzed in \cite{8}. Further, Qiu et al. \cite{38} attempt to overcome the limitations of existing coevolutionary approaches in solving minimax optimization problems using differential evolution.

**Methods**

**Notation** We adopt a mix of notation used in \cite{4, 27}. Let $\mathcal{G} = \{G_u, u \in \mathcal{U}\}$ denote the class of generators, where $G_u$ is a function indexed by $u$ that denotes the parameters of the generators. Likewise, let $\mathcal{D} = \{D_v, v \in \mathcal{V}\}$ denote the class of discriminators, where $D_v$ is a function parameterized by $v$. Here $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^p$ represent the parameters space of the generators and discriminators, respectively. Further, let $G_s$ be the target unknown distribution that we would like to fit our generative model to. Formally, the goal of GAN training is to find parameters $u$ and $v$ so as to optimize the objective function

$$\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \mathcal{L}(u, v),$$

where $\mathcal{L}(u, v) = \mathbb{E}_{x \sim G_s}[\phi(D_v(x))] + \mathbb{E}_{x \sim G_u}[\phi(1 - D_v(x))]$, \cite{1}

and $\phi : [0, 1] \rightarrow \mathbb{R}$, is a concave function, commonly referred to as the measuring function. In the recently proposed Wasserstein GAN \cite{3}, $\phi(x) = x$, and we use the same for the rest of the paper. In practice, we have access to a finite number of training samples $x_1, \ldots, x_T \sim G_s$. Therefore, one can use the empirical version $\frac{1}{T} \sum_{i=1}^{T} \phi(D_v(x_i))$ to estimate $\mathbb{E}_{x \sim G_s}[\phi(D_v(x))]$. The same holds for $G_u$. Further, let $S_u$ be a distribution supported on $\mathcal{U}$ and $S_v$ be a distribution supported on $\mathcal{V}$.

**Basic Coevolutionary Dynamics.** With coevolutionary algorithms, the two search spaces $\mathcal{U}$ and $\mathcal{V}$ can be searched with two different populations: the generator population $P_u = \{u_1, \ldots, u_T\}$ and the discriminator population $P_v = \{v_1, \ldots, v_T\}$, where $T$ is the population size. In a predator-prey interaction, the two populations coevolve: the generator population $P_u$ aims to find generators which evaluate to low $\mathcal{L}$ values with the discriminator population $P_v$ whose goal is to find discriminators which evaluate to high $\mathcal{L}$ values with the generator population. This is realized by harnessing the neo-Darwinian notions of heredity and survival of the fittest, as outlined in Algorithm\cite{1}. Over multiple generations (iterations), the fitness of each generator $u_i \in P_u$ and discriminator $v_j \in P_v$ are evaluated based on their interactions with one or more discriminators from $P_v$ and generators from $P_u$, respectively (Lines 2 to 7). Based on their fitness rank (Lines 8 to 11), the current population individuals are employed in producing next population of generators and discriminators with the help of mutation: a genetic-like variation operator (Lines 12 to 13), where the mutated individuals replace the current population if they exhibit a better fitness. In gradient-free scenarios, Gaussian mutations are usually applied \cite{15, 41}. With GANs (which are differentiable nets), we propose to use gradient-based mutations for the generators and discriminators net parameters, i.e., $I_u$ and $I_v$ are mutated with a gradient step computed by back-propagating through one (or more) of their fitness updates (right-hand side of Lines 5 and 6). Note that the coevolutionary dynamics are not restricted to tuning net parameters. Non-differentiable (hyper)parameters can also be incorporated. In our framework, we tune the learning rates for the generator and discriminator populations with Gaussian mutations.

**Spatial Coevolution Dynamics.** The basic coevolutionary setup (as adapted for GAN training in Algorithm\cite{1}) has been the subject of several studies (e.g., \cite{47, 31}) analyzing degenerate behaviors such as focusing, relativism, and loss of gradients; which correspond to mode collapse, discriminator collapse, and vanishing gradients in the GAN literature, respectively. Consequently, this has led to the emergence of more stable setups such as spatial coevolution, where individuals from both populations are distributed spatially (e.g.,
Algorithm 1 BasicCoevGANs($P_u$, $P_v$, $L$, $\{\alpha_i\}$, $\{\beta_i\}$, $I$)

Input:
$P_u$: generator population
$P_v$: discriminator population
$\{\alpha_i\}$: selection probability
$\{\beta_i\}$: mutation probability
$L$: number of generations
$I$: GAN objective function

Return:
$P_u$: evolved generator population
$P_v$: evolved discriminator population

1: for $i$ in range($I$) do
2: // Evaluate $P_u$ and $P_v$
3: $f_{u,1} \rightarrow w_i \leftarrow 0$
4: $f_{v,1} \rightarrow v_i \leftarrow 0$
5: for each $u_i$ in $P_u$, each $v_j$ in $P_v$ do
6: $f_{u,i} \leftarrow \{u_i, v_j\}$
7: end for
8: // Sort $P_u$ and $P_v$
9: $u_{1:T} \leftarrow u_{1:T}$ with $s(i) = \text{argsort}(f_{u_1:t}, i)$
10: $v_{1:T} \leftarrow v_{1:T}$ with $s(i) = \text{argsort}(f_{v_1:t}, j)$
11: // Selection
12: $u_{1:T} \leftarrow \text{replace}(u_{1:T}, v_{1:T})$ with $w_i' \leftarrow \text{mutate}(u_{1:T}, \beta_i)$
13: $v_{1:T} \leftarrow \text{replace}(v_{1:T}, v_{1:T})$ with $v_j' \leftarrow \text{mutate}(v_{1:T}, j)$
14: end for
15: return $P_u$, $P_v$

Generator Neighborhood As A Generator Mixture. Towards the end of training, $|P_u|$ generators will be available for use as generative models. And instead of using one, we propose to choose one of the generator neighborhoods $\{P_k\}^{1 \leq k \leq m^2}$ as a mixture of generators according to a given performance metric $g : \mathbb{U}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$ (e.g., inception score $\mathcal{I}$). That is, the best generator mixture $P^*_u \in \mathbb{U}^N$ and the corresponding mixture weights $w^*_i \in [0,1]^N$.—Recall that in a neighborhood, there are $N$ generators (and $N$ discriminators). Hence, the $N$-dimensional mixture weight vector $w_i$ is defined as follows

\[
P^*_u, w^*_i = \arg \max_{P^*_u, w^*_i} g \left( \sum_{k=1}^{m^2} w_{i,k} G_{u,k} \right),
\]

where $w_i$ represents the mixture weight of (or the probability that a data point comes from) the $i$th generator in the neighborhood, with $\sum_{k=1}^{m^2} w_{i,k} = 1$. One may think of $\{w^*_i\}^{1 \leq k \leq m^2}$ as hyperparameters of the proposed framework that can be set a priori (e.g., uniform mixture weights $w_i = \frac{1}{N}$). Nevertheless, the system is flexible enough to incorporate learning these weights in tandem with the coevolutionary dynamics as discussed next.

Evolving Mixture Weights. With an $m \times m$-grid, we have $m^2$ mixture weight vectors $\{w^*_i\}^{1 \leq k \leq m^2}$, which would like to learn and optimize such that our performance metric $g$ is maximized across all the $m^2$ generator neighborhoods. To this end, we view $\{w^*_i\}^{1 \leq k \leq m^2}$ as a population of $m^2$ individuals whose fitness measures are evaluated by $g$ given the corresponding generator neighborhoods. In other words, the fitness of the $k$th individual (weight vector $w^*_k$) is $g(\sum_{u_i \in P^*_u, w_i \in w^*_k} w_{i,k} G_{u,i})$. After each step of spatial coevolution of the generator and discriminator populations, the mixture weight vectors $\{w^*_k\}^{1 \leq k \leq m^2}$ are updated with an evolution strategy (e.g., (1+1)-ES [29, Algorithm 2.1]), where selection and mutation based on the neighborhoods’ $g$ values (Line 9 of Algorithm 2). This concludes the description of our coevolutionary proposition for training GANs with gradient-based mutations as summarized in Algorithm 2. Fig. 1 provides a pictorial illustration of the grid. We refer to our python implementation of Algorithm 2 by Lipizzaner.
Two different types of experiments were conducted: 1) To elaborate the capability of coevolutionary algorithms to solve typical problems of GANs, we used the theoretical model proposed in [27] that exhibits degenerate training behavior in a typical framework and compare them when trained with a simple coevolutionary counterpart. 2) We then show the ability of Lipizzaner to match state-of-the-art GANs on commonly used image-generation datasets [4,3].

Experiments

Two different types of experiments were conducted: 1) To elaborate the capability of coevolutionary algorithms to solve typical problems of GANs, we used the theoretical model proposed in [27] that exhibits degenerate training behavior in a typical framework and compare them when trained with a simple coevolutionary counterpart. 2) We then show the ability of Lipizzaner to match state-of-the-art GANs on commonly used image-generation datasets [4,3].

Theoretical GAN Model

Setup. To investigate coevolutionary dynamics for GAN training, we make use of the simple problem introduced in [27]. Formally, the generator set is defined as

$$G = \left\{ \frac{1}{2} N(\mu_1, 1) + \frac{1}{2} N(\mu_2, 1) \mid \mu \in \mathbb{R}^2 \right\}. \quad (3)$$

On the other hand, the discriminator set is defined as follows.

$$D = \{ \mathbb{1}_{[\mathbb{R}^2]} \} \quad (4)$$

Given a true distribution $G_\star$ with parameters $\mu^\star$, the GAN objective of this simple problem can be written as

$$\min_{\mu} \max_{\mathbb{E}_{\mathcal{D}, \mathcal{R}}} \mathcal{L}(\mu, \mathbb{E}_{\mathcal{D}, \mathcal{R}}),$$

where

$$\mathcal{L}(\mu, \mathbb{E}_{\mathcal{D}, \mathcal{R}}) = \mathbb{E}_{x \sim G_\star}[D_{\mathbb{E}_{\mathcal{D}, \mathcal{R}}}(x)] + \mathbb{E}_{x \sim G_\mu}[1 - D_{\mathbb{E}_{\mathcal{D}, \mathcal{R}}}(x)]. \quad (5)$$

While being simple to understand and demonstrate, this GAN variant exhibits the relevant dynamics we are elaborating. We conducted several experiments to understand the performance of the coevolutionary framework in its simplest form in comparison to the standard gradient-based dynamic. Unless stated otherwise, we used Algorithm 1 with 120 runs per experiment, each run is set with 100 generations and a population size of 10. We also use Gaussian mutation with a step size of 1 as the only genetic operator.

Results. Fig. 4 shows the convergence of the parameters $\ell_1, \ell_2, \eta_1, \eta_2, \mu_1, \mu_2$ using different variants of gradient-based and coevolutionary dynamics. One can observe that the
Fig. 2: Heatmap of success probability for random generator ($\mu_1$ and $\mu_2$) initializations for (a) a variant of gradient-based dynamics (adapted from [27]) and (b) coevolutionary GAN training dynamics. For each square, the individuals of the generator population are initialized within the corresponding range.

Fig. 3: (a) Heatmap of success probability for random discriminator ($\ell_1, r_1, \ell_2, r_2$) initializations for coevolutionary GAN training dynamics. The axes refer to initial fitness values for both the left ($[\ell_1, r_1]$) and the right ($[\ell_2, r_2]$) bounds, leading to four different quadrants. (b) First and (c) last generation of a coevolutionary trained GAN, initialized in a discriminator collapse setup, which corresponds to the bottom left quadrant of (a). In other words, for each square, the individuals of the discriminator population are initialized randomly such that the signs of their fitness values match those of the corresponding square.

is less than 0. Fig. 3 (b) shows an example of such bounds. In this setup, gradient-based dynamics force the bounds to collapse (i.e., $\ell_1 = r_1, \ell_2 = r_2$, see [27] Fig. 2 (c)). On the other hand, coevolution is able to step out of the local minimum and converges to near-optimality as shown in Fig. 3 (c)—with more generations, the left bound asymptotically moves towards $-\infty$. For this scenario, the parameters of $G$ were fixed to $\mu_1 = -1, \mu_2 = 2.5$ during the whole evolutionary process.

**GAN for Images**

**Setup.** If not stated otherwise, the experiments were conducted with Algorithm [2] on a grid size of 2x2, and a population size of one per cell (i.e., one generator and one discriminator); despite this small size, the shown results are already promising in solving the pathologies described above. We leave experiments with larger grid size for future work and upcoming versions of this paper. At the end of each generation, the current cells individual is replaced with the highest ranked offspring individual created from the neighborhood. For gradient-based mutations of the neural net parameters, we use the Adam optimizer [23] with an initial learning rate of 0.0002, which is altered with a mutation space of $\mathcal{N}(0, 1e^{-7})$ per generation. The mixture weights are updated by an (1 + 1) ES, with the mutation space of $\mathcal{N}(0, 0.01)$. Regarding the neural network topology, we used a four-layer perceptron with 700 neurons for MNIST [25], and the more complex deconvolutional GAN architecture [39] for the CelebA [28] dataset. We use the classic GAN setup [15] instead of recent propositions (e.g., WGAN [3]). This simplifies the observation of interesting pathologies, which can be more complicated to precipitate with stable GAN implementations.

**Results.** As stated, we conducted our experiments on two different datasets, which were selected because of their ability to show the behaviors we are primarily interested in. The MNIST dataset [25] has been widely used in research, and especially appropriate for showing mode collapse due to its limited target space (namely the characters 0-9). Fig. 5 illustrates this behavior, and how Lipizzaner is able to prevent collapsing on few specific modes (numbers). Both results were generated after 400 generations of training on the MNIST dataset with the above-mentioned four-layer perceptron. We furthermore show comparable promising results for the CelebA dataset [28], which contains more than 200,000 images of over 10,000 celebrities’ faces. Fig. 6 shows that a non-coevolutionary DCGAN [39] collapses at a certain point and is unable to recover even after 10 more generations (with
We furthermore note that, while the DCGAN collapse is easy to observe, coevolution is able to create higher diversity among generated samples from a generator trained with Lipizzaner in (c) shows that coevolution is able to create higher diversity among the covered modes.

Fig. 5: Results on the MNIST dataset. (a) contains samples from the original dataset, while (b) shows a typical example for mode collapse; the generator is primarily focused on the characters 1, 9 and 7. The data sampled from a generator trained with Lipizzaner in (c) shows that coevolution is able to create higher diversity among the covered modes.

Fig. 6: Sequence of images generated without Lipizzaner based on the CelebA dataset (b) before, (c) during, and (d) 10 generations after the systems mode or discriminator collapses. The original images in (a) are shown for comparison. This figure illustrates that, without further optimizations, DCGAN is mostly not able to step out of this scenario.

each generation processing the whole dataset). Fig. 7 shows that the same GAN wrapped in the Lipizzaner framework is able to step out of the collapse in only the next generation. We furthermore note that, while the DCGAN collapse is easily repeatable, Lipizzaner was able to completely avoid this scenario in most of our experiments.

Conclusion
In this paper, we have investigated coevolutionary (in particular competitive) algorithms as an option to enhance the performance of gradient-based GAN training methods. We presented Lipizzaner, a framework that combines the advantages of gradient-based optimization for GANs with those of coevolutionary systems, and allows scaling over a distributed spatial grid topology. As demonstrated, our framework shows promising results on the conducted experiments, even without scaling to larger dimensions than other comparable approaches do. Even better results may be achieved by including improved GAN types like the recently introduced WGAN.

References
[1] Al-Dujaili et al. On the application of Danskin’s theorem to derivative-free minimax optimization. Int. Workshop on Global Optimization, 2018.
[2] Arjovsky and Bottou. Towards principled methods for training generative adversarial networks. arXiv:1701.04862, 2017.
[3] Arjovsky et al.. Wasserstein gan. arXiv:1701.07875, 2017.
[4] Arora et al. Generalization and equilibrium in generative adversarial nets (gans). arXiv:1701.05577, 2017.
[5] Arora and Zhang. Do gans actually learn the distribution? an empirical study. arXiv:1706.08224, 2017.
[6] Barbosa. A coevolutionary genetic algorithm for a game approach to structural optimization. In ICGA, 1997.
[7] Barbosa. A coevolutionary genetic algorithm for constrained optimization. In CEC, 1999.
[8] Bäck et al. New approaches to coevolutionary worst-case optimization. In PPSN, 2008.
[9] Cliff and Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In ZEAAL, 1995.
[10] Dosikins and Kirbs. Arms races between and within species. Proc. R. Soc. Lond. B, 1979.
[11] Dururkak et al. Generative multi-adversarial networks. arXiv:1611.02657, 2016.
[12] Ficici and Pollack. A game-theoretic memory mechanism for coevolution. In GECCO, 1999.
[13] Gers and Schraudolph. Novel efficient asynchronous cooperative co-evolutionary multi-objective algorithms. In CEC, 2012.
[14] Goodfellow. Nips tutorial: Generative adversarial networks. arXiv:1701.00010, 2016.
[15] Goodfellow et al. Generative adversarial nets. In NIPS, 2014.
[16] Guclu et al. Improved training of wasserstein gans. In NIPS, 2017.
[17] Harper. Evolving robocode tanks for evo robocode. Genetic Programming and Evolvable Machines, 2014.
[18] Herrmann. A genetic algorithm for minimax optimization problems. In CEC, 1999.
[19] Hilles. Co-evolving parasites improve simulated evolution as an optimization procedure. Physica D: Nonlinear Phenomena, 1990.
[20] Hoang et al. Multi-generator generative adversarial nets. arXiv:1708.02556, 2017.
[21] Jensen. A new look at solving minimax problems with coevolutionary genetic algorithms. In Metaheuristics: computer decision-making, 2003.
[22] Jiao et al. Generative adversarial parallelization. arXiv:1612.00991, 2016.
[23] Kingma and Ba. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014.
[24] Laskari et al. Particle swarm optimization for mimimix problems. In CEC, 2002.
[25] LeCun. The mnist database of handwritten digits, 1998.
[26] Lehman et al. Es is more than just a traditional finite-difference approximator. arXiv:1712.05658, 2017.
[27] Li et al. Towards understanding the dynamics of generative adversarial networks. arXiv:1706.08564, 2017.
[28] Liu et al. Deep learning face attributes in the wild. In CVPR, 2015.
[29] Loshchilov. Surrogate-assisted evolutionary algorithms. PhD thesis, 2013.
[30] Metz et al. Untangling generative adversarial networks. arXiv:1611.02684, 2016.
[31] Mitchell. Coevolutionary learning with spatially distributed populations. Computational intelligence: principles and practice, 2006.
[32] Morse et al. Simple evolutionary optimization can rival stochastic gradient descent in neural networks. In GECCO, pages 477–484. ACM, 2016.
[33] Niyazhuba et al. Stabilizing gan training with multiple random projections. arXiv:1705.07831, 2017.
[34] Nielsen et al. Novel efficient asynchronous cooperative co-evolutionary multi-objective algorithms. In CEC, 2012.
[35] Nolfi and Floreano. Coevolving predator and prey robots: Do “arms races” arise in artificial evolution? Artificial life, 1999.
[36] Oliehoek et al. The parallel rash memory for asymmetric games. In GECCO, 2006.
[37] Popovic et al. Coevolutionary principles. In Handbook of natural computing, 2012.
[38] Qin et al. A new differential evolution algorithm for minimax optimization in robust design. IEEE transactions on cybernetics, 2017.
[39] Raffard and Wofford. Surpassing search: An integer hole search algorithm. In GECCO, pages 1045–1052. ACM, 2016.
[40] Rustin and Belot. New methods for competitive coevolution. Evolutionary computation, 5(1):1–29, 1997.
[41] Salimans et al. Improved techniques for training gans. In NIPS, 2016.
[42] Salimans et al. Evolution strategies as a scalable alternative to reinforcement learning. arXiv:1703.03864, 2017.
[43] Stanley and Clune. Welcoming the era of deep neuroevolution. 2017.
[44] Toloskikhin et al. Adagan: Boosting generative models. In NIPS, 2017.
[45] Wang et al. Ensembles of generative adversarial networks. arXiv:1612.09991, 2016.
[46] Wardle-Farley et al. Improving generative adversarial networks with denoising feature matching. 2016.
[47] Watson and Pollack. Coevolutionary dynamics in a minimal substrate. In GECCO, 2001.
[48] Wiatrak et al. Nature-inspired evolution strategies. In Congress on Computational Intelligence, 2008.
[49] Williams and Mitchell. Investigating the success of spatial coevolution. In GECCO, 2005.