UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

USING NOVELTY SEEKING REWARD EVOLUTION STRATEGIES TO TRAIN GENERATIVE ADVERSARIAL NETWORKS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

KHALED JABR
Norman, Oklahoma
2018

USING NOVELTY SEEKING REWARD EVOLUTION STRATEGIES TO
TRAIN GENERATIVE ADVERSARIAL NETWORKS

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Acknowledgements

I would like to thank my advisor, Dr. Amy McGovern, for taking me in as a student, giving me all the space I needed to explore the field of neuroevolution, and helping me to shape my research experience and ideas. I would also like to thank people at the OU Supercomputing Center for Education and Research (OSCER) for providing me with the all necessary support I needed to use their system.

Table of Contents

List Of Figures

Abstract

Generative Adversarial Networks (GANs) are a subclass of deep generative models that aim to implicitly learn to model a data distribution. While GANs have gained wide research attention, and achieved much success, when trained with first-order stochastic gradient descent (SGD), they suffer from training instabilities, such as non-convergence and mode collapse, in which they fail to converge to the Nash equilibrium of the minimax game, and fail to learn all the modes of the data distribution, where the samples of the generator lack diversity. To this end, this thesis investigates the use of evolution strategies (ES) to train GANs, and address the mode collapse issue. The evolution strategies (ES) algorithm used in this work is simplified version of natural evolution strategies (NES). ES achieved very impressive and competitive results against state of the art SGD-based deep reinforcement learning (RL) algorithms. A quality diversity hybrid of ES, known as Novelty Seeking Reward Evolution Strategies (NSR-ES), that aims to encourage exploration and diversity is particularly interesting in relation to the mode collapse problem is also used. In this work we propose two algorithms to train GANs, ES-GAN and NSR-ES-GAN, and we carryout experimentation on a constrained GAN setup where mode collapse exits to study how our algorithms can help overcome the issue. Our results show that using ES and NSR-ES to train GANs fails to overcome the mode collapse issue, and suggests that more robust and domain specific techniques are needed to overcome the problem.

Chapter 1

Introduction

Generative Adversarial Networks (GANs) were first introduced by Goodfellow et al. (2014). GANs are a subclass of generative models that implicitly aim to model a real data distribution without directly interacting with it. GANs frame the generative problem as two competing neural networks playing a minimax game, where both are trying to minimize their own loss function and reach the solution to the game. One network, the generator, generates synthetic data that is intended to resemble data from the real data distribution. The other network, the discriminator, is responsible for classifying whether the data is coming from the real distribution or the model distribution. During training, the discriminator learns a good classifier of synthetic vs real data, while the generator uses feedback from the discriminator to improve the quality of the generated samples, effectively learning to fool the discriminator. The GAN converges when the discriminator is no longer able to classify between real and generated data, and outputs a classification probability of 0.5 for all the data it sees whether it came from the real data distribution or the model distribution.

GANs have enjoyed success in since they were first introduced, and have been used in many domains including, but not limited to, image and video generation (Karras et al. 2018; Chen et al. 2016; Vondrick et al. 2016), image-to-image translation (Isola et al. 2017; Zhu et al. 2017), Semi-supervised learning (Salimans et al. 2016; Kumar et al. 2017), and Reinforcement Learning (Li et al.

2017). Though they enjoy a lot of success, GANs have proven notoriously hard to train at times. Because of their unstable and not well understood training dynamics, GANs suffer from non-convergence and mode collapse problems, in which they fail to learn the full real data distribution, and only learn a subset of the its modes (Salimans et al. 2016; Metz et al. 2016; Mescheder et al. 2017; Li et al. 2018).

GANs are one component of study in this thesis; the other component is a highly parallelizable natural evolution strategies (NES) variant introduced by Salimans et al. (2017). In their work, Salimans et al. (2017) introduced a simplified version of NES, and obtained competitive results on complex and high-dimensional RL benchmarks in comparison to gradient based state-of-the-art RL algorithms. For clarity and simplicity reasons, we will refer to this variant as Evolution Strategies (ES). ES drew attention because of its ability to optimize high-dimensional neural networks in challenging deep RL environments, contrary to the previously held belief that evolutionary algorithms were only suitable for low-dimensional problems. ES has been shown to exhibit interesting features such as qualitatively different exploration behavior (Salimans et al. 2017; Conti et al. 2017), faster training wall-clock time than rival first-order stochastic gradient descent algorithms (Salimans et al. 2017), and the ability to optimize neural networks with respect to different types of gradients that could lead the search to a different, yet more robust, areas of the search space (Lehman et al. 2017; Conti et al. 2017).

The main purpose of this thesis is to investigate the use of ES to train GANs, and how it could be used to address the problem of mode collapse. We investigate the use of complementary neuroevolution techniques, namely novelty search (Lehman and Stanley 2008, 2011a) and quality diversity algorithms

(Pugh et al. 2015), in aiding ES in training of GANs. We hypothesize that using a quality diversity variant of ES, dubbed as Novelty Seeking Reward Evolution Strategies (NSR-ES) (Conti et al. 2017), to train the generator of GANs, while using ES to train the discriminator, will help GANs avoid the mode collapse problem by diversifying the output of the generator to cover more modes of the real data distribution, and avoid mode collapse. We performed experiments on a low-dimensional and artificial dataset, using a simple GAN setup where mode collapse is well known and is easy to visualize.

Chapter 2

Background and Related Work

2.1 Generative Adversarial Networks

GANs are a subclass of deep generative models based on game theory. GANs are formulated as a minimax game between two networks: a generator (G) and a discriminator (D). G aims to learn the true data distribution and to generate samples that are intrinsically similar to it, while the D aims to learn discriminate between the real data distribution p_r and the model's distribution p_g. The adversarial term comes from the training process in which G is trying to maximize the probability of the D being mistaken, while the D is trying maximize the probability of it being correct. In GANs, both D and G are assumed to be differential functions, usually represented by neural networks networks.

GANs are formalized as follows : Let p_r be the true data distribution over real data x. We define the latent variable $z \sim p_z$, where p_z is usually chosen to be a uniform distribution. We define the generator, $G(z) : z \rightarrow x$, as a mapping from the latent variable to the model distribution, p_g, over x. We define the discriminator, $D(x) : x \rightarrow [0, 1]$, to output the probability that x came from p_r.

Then D and G can be formalized as two players playing a minimax game with the following value function $V(G, D)$ (Goodfellow et al. 2014):

$$\min_G \max_D V(D, G) = E_{x \sim p_{\mathrm{r}}}[\log D(x)] + E_{z \sim p_z}[\log(1 - D(G(z)))]. \tag{2.1}$$

Using this setup, the GAN is trying to learn an optimal generator , G^* such that :

$$G^* = \arg\min_G V(G, D^*), \tag{2.2}$$

and

$$D^* = \arg\max_D V(G, D) \tag{2.3}$$

where D^* is the optimal discriminator at each training iteration, and E is the expectation symbol.

Solving for D^* at every training iteration is computationally infeasible, especially when we are using neural networks, instead, we train GANs by alternating gradient updates on D and G (Goodfellow et al. 2014). At every iteration, we sample a mini-batch of real data x drawn from p_r and a mini-batch of z drawn from p_z, then we train D to maximize $E_{x \sim p_{\mathrm{r}}}[\log D(x)] + E_{z \sim p_z}[\log(1 - D(G(z)))$ by ascending its gradient, and we train G to minimize $log(1 - D(G(z)))$ by descending its gradient.

By minimizing $log(1 - D(G(z)))$, G learns to produce samples that have a low probability of being synthetic (Goodfellow et al. 2014). This setup is also commonly referred to as minimax GAN (MM-GAN). In the same paper, Goodfellow et al. (2014) highlighted that this formulation is does not perform well in practice at early stages in training because D is able to reject generated samples with high confidence,i.e $D(G(z))$ is close to zero, which does not provide sufficient learning signal to the generator as the term $log(1-D(G(z)))$ saturates, instead, Goodfellow et al. (2014) suggested an alternative loss objective for

G where it learns to generate samples that have a high probability of being classified as real by maximizing the $log(D(G(z)))$. This formulation is known as the non-saturating GAN (NS-GAN). The relationship between the cost of G and $D(G(z))$ is shown in Figure 2.1. All GANs formulations use the same loss function for D, but differ in the loss used for G.

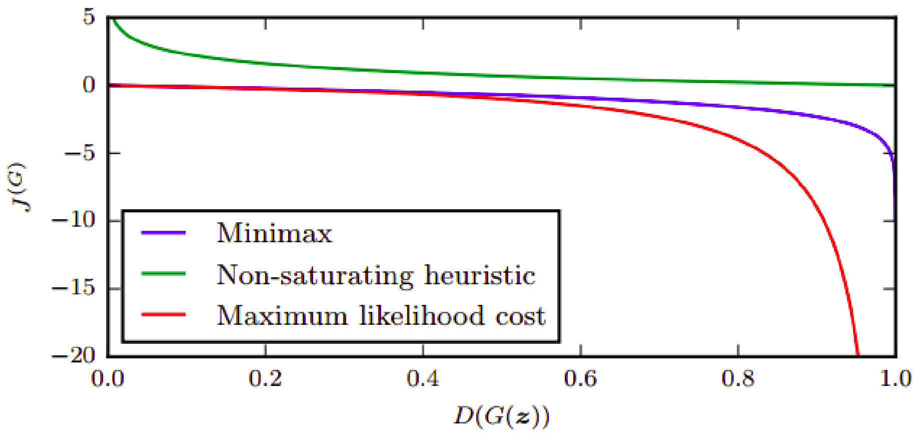

Figure 2.1: Generator cost function as a function of the discriminator response on generated images. Image from Goodfellow (2016)

Minimizing D loss is equivalent to minimizing the Jensen-Shannon Divergence, JSD, between p_r and p_g (Goodfellow et al. 2014). The GAN converges when $JSD(p_r||p_g) = 0$, where G recovers the real p_r and the value of $D(x_{real}) = D(x_{synthetic}) = \frac{1}{2}$ for all $x_{real} \in p_r$ and $x_{synthetic} \in p_g$. This is also known as Nash equilibrium of the miminmax game, when neither player can make a move to unilaterally improve its score.

2.1.1 Mode Collapse

Mode collapse is a common problem when training GANs, in which the generated samples of G lack diversity. While the generator is able to produce data that looks like data from the real distribution, it does not cover all the mode (classes) of the true data distribution, but only a small subset of it. Mode collapse occurs when the generator learns to map different noise values, z, from the latent space to the same output (Goodfellow 2016; Lin et al. 2018). Early on, it was common belief that mode collapse was related to the type of divergence GANs sought to minimize between the p_r and p_g; however, recent research and the current belief is that mode collapse is related to the training dynamics of GANs and its instability, which is still an active area of research, where many aspects are still not well understood(Goodfellow 2016; Metz et al. 2016; Mescheder et al. 2017; Li et al. 2018; Lin et al. 2018).

The main difficulty in understanding and training GANs and its stability lies in the way GANs are formulated as a minimax game in Eq. 2.1. The solution to the minimax game is a saddle point (G^*, D^*), also called Nash equilibrium, where :

$$V(G^*, D) \leq V(G^*, D^*) \leq V(G, D^*) \tag{2.4}$$

In game theory, this saddle point exits if :

$$\min_G \max_D V(G, D) = \max_D \min_G V(G, D) \tag{2.5}$$

Theoretical guarantees for converging to the saddle point, (G^*, D^*), exist using alternating updates on D and G when the minimax problem is of convex outer minimization and concave inner maximization (Goodfellow et al. 2014; Metz et al. 2016). However, since D and G are both represented by neural networks, G is not convex, and D is not concave, thus updates are not constrained in a theoretical way, and there are no guarantees that the GAN will converge to (G^*, D^*), thus Eq. 2.5 does not hold when the GAN converges to a solution, causing mode collapse to happen (Goodfellow et al. 2014; Metz et al. 2016).

This underlines one of the major difficulties of training in practice, which is that it is still a heuristic process.

Another intuitive way to understand mode collapse is through the lens of catastrophic forgetting (Thanh-Tung et al. 2018). Thanh-Tung et al. (2018) frames GANs training as a continual learning problem in which at each iteration t, the discriminator has to learn to discriminate between the real data distribution, p_r and generated data distribution, p_g^t. The problem of mode collapse arises from the fact that at each iteration, D has access to samples from p_r, however, it forgot about previous samples from model distribution, $p_g^0 : p_G^{t-1}$. This results in D being biased towards separating the current synthetic samples from the nearby real ones, ignoring distant real samples, and previously seen synthetic samples. Thanh-Tung et al. (2018) noted that this bias caused D to overemphasize the importance of the current batch of synthetic samples where it assigns higher scores to data samples that are further from current synthetic samples. This makes D unable to guide G correctly to produce more diverse samples, and cover more modes of the real data distribution. This explanation bears a resemblance to predator-prey co-evolution (Nolfi and Floreano 1998), which is left for future work.

Many approaches have been proposed to deal with the issue of mode collapse. Salimans et al. (2016) used an approach to directly encourage diversity through feature matching and mini batch discrimination. Metz et al. (2016) takes an approach where the generator anticipates the counter play of the discriminator by defining its cost function to be the unrolled optimization of the discriminator, which they show that it is a good approximation for the optimal discriminator, D^*. Tolstikhin et al. (2017) approached mode collapse by training multiple GANs at the same time, where each covers a subset of modes. Wang et al.

(2018) used an evolutionary approach in which a population of generators using different loss functions were evolved to adapt to the discriminator.

2.2 Natural Evolution Strategies

Natural Evolution Strategies (NES) (Wierstra et al. 2008, 2014) is a class of black-box optimization algorithms that maintains a search distribution that it iteratively updates using a gradient that follows the direction of higher expected fitness. The main idea of NES is to use search gradients to seek areas in search spaces with higher expected fitness. The basic procedure of NES can be summarized into three operations performed at each generation: Sampling from the search distribution, evaluating sampled solutions fitness, and recombining the results to estimate a gradient and update the search distribution parameters. While this procedure is similar to that of evolution strategies (Rechenberg 1973; Schwefel 1977), it is important to highlight that NES is different in two main ways: the representation of population as a search distribution, and the use of search gradients to update search distribution (Wierstra et al. 2014).

NES characterizes the population it uses to estimate the gradient by a search distribution, and seeks to optimize the expected fitness of population sampled from the search distribution (Wierstra et al. 2014). More formally, let F be a function acting on parameters θ. To optimize F with respect to θ, NES defines a search distribution $p_\psi(\theta)$, where ψ represents the mean and convariance of p_ψ, and finds the gradient of objective function being optimized, $\nabla_\psi E_{\theta\sim p_\psi}\{F(\theta)\}$, with respect to ψ as follows :

$$\nabla_\psi E_{\theta\sim p_\psi} F(\theta) = E_{\theta\sim p_\psi}\{F(\theta)\nabla_\psi \log p_\psi(\theta)\} \tag{2.6}$$

The NES framework (Wierstra et al. 2014) provides nice and clear derivations for search gradients using multi-variate Gaussian distributions as follows: let $\theta = (\mu, \Sigma)$, where $\mu \in R^d$ and $\in R^{d\times d}$ are the mean and covariance of the distribution respectively. To sample from the distribution, we transform a standard normal vector $z \sim \mathcal{N}(0, I)$ into a sample $s \sim \mathcal{N}(\mu, \Sigma)$:

$$z = \mu + A^\intercal S, \tag{2.7}$$

where $A \in R^{d\times d}$ is the square root of covariance matrix that satisfies $A^\intercal A = \Sigma$, and $I \in R^{d\times d}$ is the identity matrix. Next, the density function of the search distribution $\mathcal{N}(\mu, \Sigma)$ is defined as :

$$\pi(z|\theta) = \frac{1}{\sqrt{(2\pi)^d \det(A)}} \cdot \exp\left(-\frac{1}{2}(z-\mu)^\intercal \Sigma^{-1}(z-\mu)\right) \tag{2.8}$$

Using this formulation, we can calculate the gradient by calculating the derivatives $\nabla_\mu \log \pi(z|\theta)$ and $\nabla_\Sigma \log \pi(z|\theta)$ by :

$$\nabla_\mu \log \pi(z|\theta) = \Sigma^{-1}\left(z-\mu\right), \tag{2.9}$$

and

$$\nabla_\Sigma \log \pi(z|\theta) = \frac{1}{2}\Sigma^{-1}\left(z-\mu\right)\left(z-\mu\right)^\intercal \Sigma^{-1} - \frac{1}{2}\Sigma^{-1} \tag{2.10}$$

Using those derivatives, we can calculate the gradient and update the parameters the search distribution, $\theta = (\mu, \Sigma)$ to a new center and covariance matrix. The updates can be applied using $\theta_{t+1} = \theta_t + \eta \nabla_\theta \log \pi(z|\theta)$ where η is the learning rate, or using optimizers such as Adaptive Moment Estimation (Adam) optimizer (Kingma and Ba 2014; Salimans et al. 2017). The Adam optimizer is a first-order gradient optimization algorithm that adapts the functions learning rates using the average of the first and second moments of gradients. Adam is computationally efficient, easy to implement, and is every popular in the deep learning field.

2.2.1 OpenAI ES

Salimans et al. (2017) proposed and used a black-box optimization algorithm that is a variant of NES to solve challenging deep RL problems. The algorithm also takes a similar approach to the REINFORCE algorithm (Williams 1992). Just like NES, the algorithm defines a search distribution, p_ψ, where the parameters of the neural network being optimized, θ, are drawn from. p_ψ is chosen to be an isotropic Gaussian with mean ψ and fixed covariance $\sigma^2 I$. Such setup allows to rewrite our expectation in Eq. 2.6, $E_{\theta \sim p_\psi} F(\theta)$, directly in terms of the neural network parameters, θ, as the following:

$$E_{\theta \sim p_\psi} F(\theta) = E_{\epsilon \sim N(0,I)} \{F(\theta + \sigma\epsilon)\} \tag{2.11}$$

where ϵ is Gaussian noise added to perturb the parameters of the network θ.

Salimans et al. (2017) defines this a Gaussian-blurred version of the original objective function and argue that it helps deal with the discrete and non-smooth nature of the optimized neural network. Using the re-parameterization, we can optimize over the parameters of the network θ directly by means of sampling:

$$\nabla_\theta E_{\epsilon \sim N(0,I)} F(\theta + \sigma\epsilon) = \frac{1}{\sigma} E_{\epsilon \sim N(0,I)} \{F(\theta + \sigma\epsilon)\epsilon\} \approx \frac{1}{n\sigma} \sum_{i=1}^{n} F(\theta + \epsilon_i)\epsilon_i \tag{2.12}$$

This simplified version of NES proposed by Salimans et al. (2017) uses a fixed mutation step size, $\sigma \in R$, and only updates the mean, θ. For the sake of clarity, we will refer to this variant of NES as ES for the remaining of the thesis. Note that ES used in this work is not to be confused by ES introduced by Schwefel (1977). The full ES algorithm is shown in algorithm 1.

Salimans et al. (2017) were able to scale ES to optimize high-dimensional neural network scaling up to millions of parameters by introducing a novel communication strategy between parallel workers in which the workers communicate the final evaluation of the fitness function and the random seed responsible for generating the weight perturbation between each other. This communication strategy made ES highly parallelizable and extremely low bandwidth. The parallelized version of ES is shown in algorithm 2. The implementation of weight perturbations sampling is done by instantiating a large table of Gaussian noise at the beginning of training, and then having all the workers share a copy of this table (Salimans et al. 2017). Thus, random seeds generating perturbations vectors are randomly drawn indices that satisfy $0 \leq i < S(N) + d - 1$, where $S(N)$ and d are the size of the noise table and the dimensionality (number of trainable weights) of the neural network being optimized respectively, and workers share those seeds instead of entire perturbation vectors (Salimans et al. 2017).

2.2.2 Antithetic Sampling and Fitness Shaping

Wierstra et al. (2008) and Salimans et al. (2017) proposed using techniques to improve the performance of ES, most relevant of which are antithetic sampling and fitness shaping. Antithetic sampling (Geweke 1988) is a technique where for each perturbation vector, ϵ, we evaluate the fitness of both $(\epsilon, -\epsilon)$ to reduce the variance of the gradient estimation.

Another technique that was proposed by Wierstra et al. (2014) and used by Salimans et al. (2017) is fitness shaping. Fitness shaping is a technique to help ES avoid outliers by applying a rank-preserving transformation to the returns of the population and using the transformation to calculate the gradient. Rather than using the actual fitness, we rank the individuals of the population by their fitness returns, and then we apply a utility functions to the individual fitness returns to produce augmented fitness values that are proportional to the individual rank in the population.

Algorithm 1 ES

```
1: Input: learning rate $\eta$, noise standard deviation $\sigma$, population size $n$, initial network parameters $\theta_0$,
   optimizer $opt$, fitness function $F$
2: for $t = 0, 1, ...$ do
3:     for $i = 1$ to $n$ do
4:         Sample $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$
5:         Compute $f^+ = F(\theta_t + \sigma \times \epsilon_i)$
6:         Compute $f^- = F(\theta_t - \sigma \times \epsilon_i)$
7:     end for
8:     Compute ranked fitness $r^+ = ranks(f^+)$
9:     Compute ranked fitness $r^- = ranks(f^-)$
10:    Estimate Gradient : $g \approx \frac{1}{n\sigma} \sum_{i=1}^{n} (\epsilon_i (r_i^+ - r_i^-))$
11:    Update Network $\theta_{t+1} = \theta_t + opt(g, \eta)$
12: end for
```

Algorithm 2 Parallelized ES

```
1: Input: learning rate $\eta$, noise standard deviation $\sigma$, population size $n$, initial network parameters $\theta_0$,
   optimizer $opt$, fitness function $F$
2: for $t = 0, 1, ...$ do
3:     for $i = 1$ to $n$ do
4:         Sample $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$
5:         Compute $f^+ = F(\theta_t + \sigma \times \epsilon_i)$
6:         Compute $f^- = F(\theta_t - \sigma \times \epsilon_i)$
7:     end for
8:     for $i = 1$ to $n$ do
9:         reconstruct $\epsilon_i$ for $i = 1, 2...n$ using random seeds from other workers
10:        Compute ranked fitness $r^+ = ranks(f^+)$
11:        Compute ranked fitness $r^- = ranks(f^-)$
12:        Estimate Gradient : $g \approx \frac{1}{n\sigma} \sum_{i=1}^{n} (\epsilon_i (r_i^+ - r_i^-))$
13:        Update Network $\theta_{t+1} = \theta_t + opt(g, \eta)$
14:    end for
15: end for
```

2.2.3 Domains of ES

The ES algorithm introduced by Salimans et al. (2017), and the competitive results it obtained in deep RL challenges, sparked interest in using ES, and other related gradient-fee methods for RL domains (Conti et al. 2017; Such et al. 2017; Mller and Glasmachers 2018; Chrabaszcz et al. 2018). On the other hand, Zhang et al. (2017) took a different direction and studied how ES performed in supervised learning classification problem in comparison to traditional, first-order, stochastic gradient descent (SGD). Zhang et al. (2017) sought to measure the correlation between gradients computed by SGD and ES by optimizing a neural network that learns a classifier over the MNIST dataset. Zhang et al. (2017) also tested and proposed techniques that improved the performance of ES: limited perturbations, and no-mini batch. In limited perturbation ES, each worker in the ES algorithm perturbs a subsets of the weights of the neural network instead of all them, which speeds up the computations related to ES Zhang et al. (2017). These ideas was originally suggested, but not studied, by Salimans et al.. In the no-mini batch ES, workers in each iteration evaluate the perturbed neural network on a unique random subset of data, instead of using the traditional approach of mini-batch where all the workers in one iteration would evaluate the same mini-batch for all the batches in the dataset each iteration. Zhang et al. showed that the no-mini batch approach produced very smooth curves rates and achieved good results. The work by Zhang et al. (2017) inspired us to use the no-mini batch approach in our work.

2.3 Novelty Search and Quality Diversity

2.3.1 Novelty Search

Novelty Search (NS) is a different type of search that questions the effectiveness and the optimization pathology of main-stream objective-based search algorithms. The main idea of novelty search is to get rid of the objective function and instead search for behavioural and functional novelty in the target search space, and to move in the direction of increased behavioral novelty and, subsequently, complexity (Lehman and Stanley 2008, 2011a). Novelty Search was inspired by the inability of objective functions to foresee and avoid deception caused by local optima. Lehman and Stanley (2011a) showed that novelty search alone outperformed fitness-based methods in deceptive maze and biped walking tasks.

The two main components of novelty search are behaviour characterization (bc), and the novelty archive A. Behaviour characterization is a domain dependent metric that characterizes the behaviour of the individual, i.e a neural network, and it is used to compute its novelty with respect to other individuals whose behavioral characterizations are stored in the novelty archive. During training, every time an individual θ is evaluated, its behavioral characterization $bc(\theta)$ is added to the novelty archive A. Novelty archive is an archive that stores that behavior characterizations of the individuals during training. The novelty of a specific individual $N(\theta, A)$ is calculated by selecting the k-th nearest neighbours, where k is predetermined, of $bc(\theta)$ from archive A, then computing the average of the distances between them :

$$N(bc(\theta)) = \frac{1}{|S|} \sum_{i \in S} dist(bc(\theta), bc(\theta_i)) \tag{2.13}$$

where S is the set of k-th nearest neighbours, and *dist* is a domain dependent distance metric.

The more we accumulate similar behavior characterizations in A, the novelty decreases rendering them as less novel, and NS moves away from them. An interesting property of behavior characterization in the context of its relationship with fitness, is alignment. A highly aligned behavior characterization means that more novelty usually leads to higher fitness, while a weakly aligned, or an unaligned one does not impose that relationship (Pugh et al. 2016).

2.3.2 Minimal Criteria Novelty Search

Minimal Criteria Novelty Search (MCNC) is an extension to novelty search that addresses its open-endedness and unconstrained nature (Lehman and Stanley 2010). The behavior space defined by pure novelty search focuses solely on pushing the search to more novel areas in the behavior space, without explicitly taking into account the environment of interest it is deployed to search. This could cause issues in some specific domains when the behavior space is unbounded. Lehman and Stanley (2010, 2011a). To test this phenomenon Lehman and Stanley designed two experiments where a robot was trained to solve a maze. For the two maze experiments, the main difference was that one had closed walls and the other did not. Earlier experiments showed NS being very successful in finding a solution to the closed-walls maze problem. However, when NS was deployed to solve the partially opened maze, its performance degraded greatly. The reason behind that is, while NS was exploring novel behaviours in the behaviour space, it was not doing so efficiently in relation to the maze exploration task, hence the idea of MCNS. MCNS aims to make NS more efficient by connecting it to the domain and pruning the behavioral space. The extension of MCNS is simple: at the time of evaluation, MCNS checks whether the individual meets a certain domain-dependent minimal criteria, if so, the novelty search proceeds as normal; if not, the novelty of the individual is set to zero and special behaviour characterization reflecting this failure is added to the archive or discarded. In the maze example, Lehman and Stanley (2010) set the minimal criterion to be that the robot must end inside the maze at the end of each iteration, and the experiments showed success. One main concern is that at the start of training, MCNS might not have any solution (individual) that meets the minimal criterion, which would effectively render it a random search until the first individual that meets the minimal criteria is found. To approach this, Lehman and Stanley (2010) suggest seeding the initial population with a solution that meets the minimal criterion.

2.3.3 Quality Diversity Algorithms

Quality Diversity (QD) algorithms (Pugh et al. 2015, 2016) are algorithms that are concerned with both quality and diversity of evolved individuals. QD algorithms suggest a new way of looking at search in evolutionary algorithm as a divergent, rather than convergent, search trying to find the best performing individuals while diversifying their behaviors. For a detailed and historical study of quality diversity algorithms, refer to Pugh et al. (2016). In NS, the search optimizes only for behavioral diversity, and ignores the fitness returns completely, although it still offers valuable information about the search space. QD algorithms seek to make use of both quality (fitness) and diversity (novelty). This way, QD searches the problem space for individuals with high fitness and novel behaviors. A QD algorithm can be developed by augmenting NS in a way such that the individual is updated based on the average of their novelty and fitness.

Conti et al. (2017) hybridized NS with ES to promote directed exploration while maintaining the scalability of the ES proposed in algorithm 1, named Novelty Search Reward Evolution Strategies (NSR-ES). In ES, we focus solely on fitness, in NS, we abandon the fitness completely in favor of novelty, but in NSR-ES, we are optimizing for the weighted sum of both fitness and novelty Conti et al. (2017).

Algorithm 3 NSR-ES

```
1: Input: learning rate $\eta$, noise standard deviation $\sigma$, population $n$, initial network parameters $\theta_0$ , optimizer $opt$, fitness function $F$, novelty archive $A$, reward pressure $p$
2: Compute $bc(\theta_0)$
3: Add $bc(\theta_0)$ to $A$
4: for $t = 0, 1, ...$ do
5:     for $i = 1$ to $n$ do
6:         Sample $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$
7:         Compute Novelty $N^+(bc(\theta + \sigma \times \epsilon_i))$
8:         Compute Novelty $N^-(bc(\theta - \sigma \times \epsilon_i))$
9:         Compute $f^+ = F(\theta + \sigma \times \epsilon_i)$
10:        Compute $f^- = F(\theta - \sigma \times \epsilon_i)$
11:    end for
12:    Compute ranked novelty $n^+ = ranks(N^+)$
13:    Compute ranked Novelty $n^- = ranks(N^-)$
14:    Compute ranked fitness $r^+ = ranks(f^+)$
15:    Compute ranked fitness $r^- = ranks(f^-)$
16:    Estimate Gradient : $g \approx \frac{1}{n\sigma} \sum_{i=1}^{n} \epsilon_i \{p \times (r_i^+ - r_i^-) + (1 - p) \times (n_i^+ - n_i^-)\}$
17:    Update Network $\theta_{t+1} = \theta_t + opt(g, \eta)$
18:    Compute $bc(\theta_{t+1})$
19:    Add $bc(\theta_{t+1})$ to $A$
20: end for
```

Chapter 3

Approach

In this section, we discuss our formulation of the problem, and how we plan to use ES and NSR-ES to train GANs.

3.1 Approach

The main research question in this thesis is to study how can we use ES and its variant, NSR-ES, to train GANs and avoid the mode collapse problem. To best understand and analyze this issue, we carry experiments on a small synthetic 2D dataset of a mixture of 8 Gaussians, shown in Figure 3.1, on a fixed neural network setup that exhibits mode collapse. The advantage of using this setup is that we are able to easily visualize mode collapse as well as see how our approach tries to migrate and solve the problem. This small experimentation setup is also a good fit for ES, since ES can be budget-demanding when optimizing large neural networks, let alone two at the same time (G and D). We start by presenting our neural network setup, and then we train it using ES on an even simpler dataset as a proof of concept that our ES code does indeed work. Then we move into experimentation where we train our proposed GANs using ES, then we experiment with NSR-ES

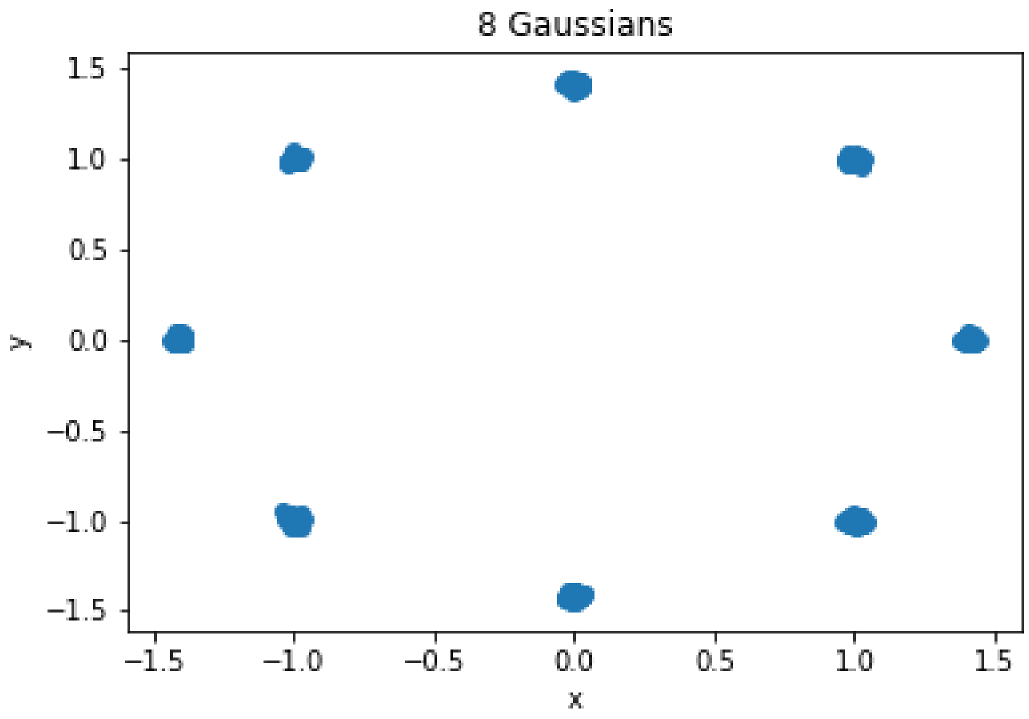

Figure 3.1: A mixture of 8 Gaussians

3.2 GAN Architecture

We use a GAN architecture where both D and G are both multilayer perceptrons (MLPs). Each MLP consists of 2 fully connected layers with 128 hidden units followed by 1 linear output layer with one neuron for D, and two neurons for G. Weights are initialized by sampling from a normal distribution with 0 mean and a standard deviation of 0.05. Rectified Linear Units (Relu) activation functions are used as the non-linearity between fully connected layers. Batch size is set to 256 and the dimension of the latent code, z, is 256. We use the non-saturating GAN formulation (NS-GAN). We call it simple GAN, and its architecture is depicted in Figure 3.2

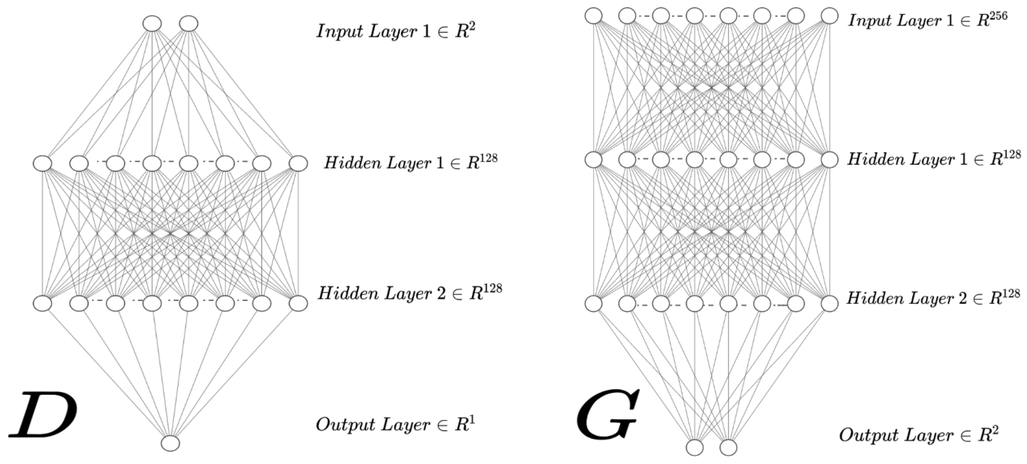
Input Layer $1 \in R^2$
Hidden Layer $1 \in R^{128}$
Hidden Layer $2 \in R^{128}$
D
Output Layer $\in R^1$
Input Layer $1 \in R^{256}$
Hidden Layer $1 \in R^{128}$
Hidden Layer $2 \in R^{128}$
G
Output Layer $\in R^2$

Figure 3.2: Simple GAN architecture.

Under the theoretical framework of GANs, this GAN architecture should be able to learn to model the 8 Gaussian datasets presented earlier. However, it fails to do so, and instead learns to produce one mode only. This is why this setup is considered a good test bed for ideas and research tackling the mode collapse issues where it acts as a proof of concept for the proposed ideas, which is a lot harder to do with larger detests and is used widely in GANs research community (Metz et al. 2016; Nguyen et al. 2017; Akash et al. 2017; Mao et al. 2017; Wang et al. 2018).

3.3 ES-GAN: Evolution Strategies GAN

ES is a blackbox optimization for estimating gradients. ES can be used for training GANs as a drop-in replacement for stochastic gradient descent(SGD). The full algorithm of ES-GAN is shown in Algorithm 4. Lines 3-13 are equivalent to updating the discriminator D, and lines 14-23 are equivalent to updating the generator G.

For fitness shaping, the returns (Lines 7-8 and 17-18) are ranked by ascending order, then the following utility function is used to normalize the returns based on their rank:

$$r_i = \frac{z_i}{n-1} - 0.5 \tag{3.1}$$

where z_i is the rank of the $i-th$ best individual in ascending order, and n is the population size. Since ES is estimating a gradient, we can use optimizers to update the the parameters of D and G. In our work, we use ADAM optimizer (Kingma and Ba 2014). The fitness shaping and optimizer choices in this work are the same as the ones in Salimans et al. (2017) and Conti et al. (2017).

Algorithm 4 ES-GAN

```
 1: Input: learning rates ($\eta_d$, $\eta_g$), noise standard deviations ($\sigma_d$,$\sigma_g$), population $n$, optimizers ($opt_d$,$opt_g$),
    loss functions ($F_d$,$F_g$)
 2: for $t = 0, 1, ...$ do
 3:     for $i = 1$ to $n$ do
 4:         Sample minibatch of m examples from real data $p_r$
 5:         Sample minibatch of m examples from $p_z$
 6:         Sample $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$
 7:         Compute $f_d^+ = F_d(\theta_{dt} + \sigma_d \times \epsilon_i)$
 8:         Compute $f_d^- = F_d(\theta_{dt} - \sigma_d \times \epsilon_i)$
 9:     end for
10:     Compute ranked fitness $r_d^+ = ranks(f_d^+)$
11:     Compute ranked fitness $r_d^- = ranks(f_d^-)$
12:     Estimate D Gradient : $g_d \approx \frac{1}{n\sigma} \sum_{i=1}^{n}(\epsilon_i(r_d^+{}_i - r_d^-{}_i))$
13:     Update D $\theta_{dt+1} = \theta_{dt} + opt_d(g_d, \eta_d)$
14:     for $i = 1$ to $n$ do
15:         Sample minibatch of m examples from noise distribution $p_z$
16:         Sample $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$
17:         Compute $f_g^+ = F_g(\theta_{gt} + \sigma_g \times \epsilon_i)$
18:         Compute $f_g^- = F_g(\theta_{gt} - \sigma_g \times \epsilon_i)$
19:     end for
20:     Compute ranked fitness $r_g^+ = ranks(f_g^+)$
21:     Compute ranked fitness $r_g^- = ranks(f_g^-)$
22:     Estimate D Gradient : $g_g \approx \frac{1}{n\sigma} \sum_{i=1}^{n}(\epsilon_i(r_g^+{}_i - r_g^-{}_i))$
23:     Update G $\theta_{gt+1} = \theta_{gt} + opt_g(g_g, \eta_g)$
24: end for
```

3.4 NSR-ES-GAN: Novelty Seeking Reward Evolution Strategies GAN

The main idea behind using NSR-ES is to diversify the output of the generator, to incentivize diversifying the mappings from the noise vector z to the output of the generator. We hypothesize that diversifying the output of the generator will help both the generator and discriminator to make smoother updates in which the GAN is less likely to suffer from mode collapse. In this work, NSR-ES is applied only to the generator, as we are seeking to diversify its output, while the discriminator is trained using ES.

In NSR-ES, we estimate the gradient of G as the following:

$$g \approx \frac{1}{n\sigma} \sum_{i=1}^{n} p f_g(\theta_t^i)\epsilon_i + (1-p)N(bc(\theta_t^i)|A)\epsilon_i,$$

where n is the population size, σ is the standard deviation of the noise added to perturb the weights, ϵ is the perturbation vector,$p \in [0, 1]$ is the reward pressure, $f_g(\theta)$ is the loss function of the G, and $N(bc(\theta)|A)$ is the novelty of the generator given its behavior characterization, bc, with respect to the novelty archive A.

We update G based on novelty and loss (fitness), and both measures are ranked using fitness shaping to ensure that they are on the same scale. With $p > 0.5$, we push the search towards areas with more emphasis on higher fitness. With $p < 0.5$, we push the search towards areas with more emphasis on novelty based on our behavior characterization. The default value for p is 0.5. In our experiments, we test for $p \in \{0.3, 0.5, 0.7\}$

When using novelty search, we have to specify four main components of interest to our domain: Behavioral Characteristic, Minimal Criterion, Novelty Archive, and Distance Function.

We define our behavior characterization on G, and donate it $bc_g(g)$. $bc_g(g)$ is defined as the average of n of generator samples. That is, at the end of each GAN iteration, we pass n randomly sampled noise vectors, $z \sim p_z$, through G and average the outputs to a single output sample. Our intuition behind this behavioural characteristic is to encourage the diversity of the generator's output which will diversify the mappings from z to x and help alleviate mode collapse. We set $n = 1$ for all our experiments. We avoided using $n > 1$ because, when averaging the samples, it hurt the representation of the behavior characterization.

Running initial experiments using NSR-ES to train the GAN using $bc_g(g)$, without constraining the behavioural space of novelty search led the generator output to spreading continually outwards as shown in Figure 3.3. This behavior is expected as the generated data is still considered novel under our bc definition; however, they are not related or constrained to the environment of interest, which is the dataset to be learned. This is a classic example of why we need to constrain the behavior space of novelty search using minimal criteria. Our minimal criterion is domain-dependant, and defined as the following: Two polygons are defined as follows: first is a square that bounds the real data distribution, $poly_{(mc)}$. The second is a square that bounds $bc(g)$,$poly_{(g)}$. The minimal criterion is defined by requiring $poly_{(g)}$ to be inside $poly_{(mc)}$. $poly_{(g)}$ is computed by computing the bounding box of $bc_g(g)$. Our minimal criterion is shown in Figure 3.4. In our implementation, at the end of each iteration, if the

$bc_g(g)$ fails to meet minimal criterion, the novelty score is assigned zero, and $bc_g(g)$ is discarded.

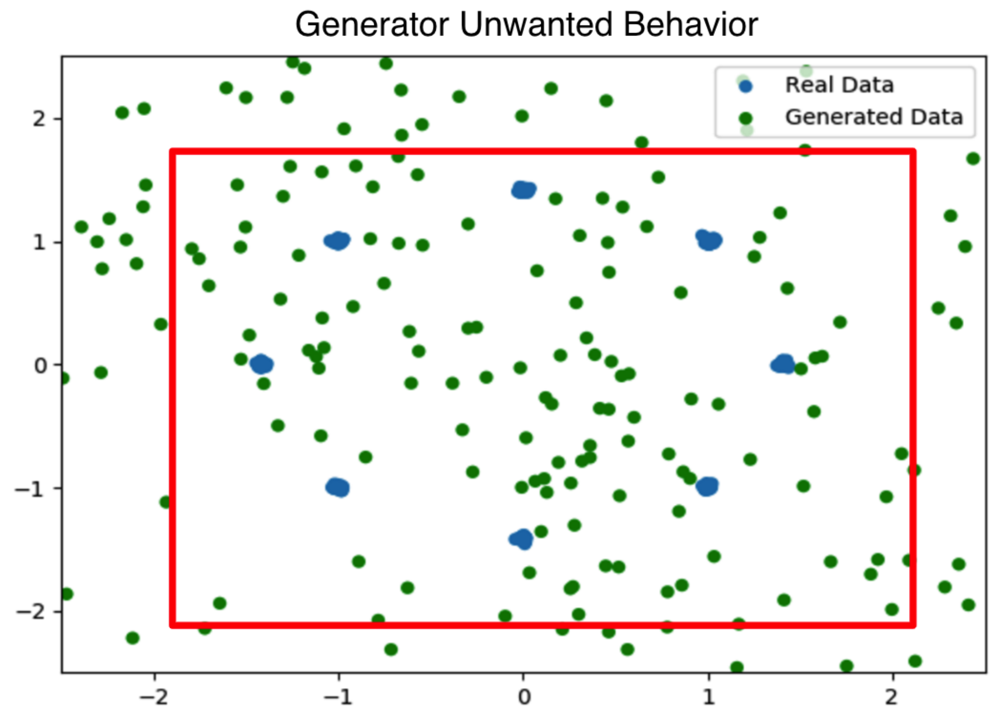

Figure 3.3: Without minimal criteria, novelty search on G tends to spread outwards and away from our target data

For the novelty archive, we do not set a specific upper bound for the size of the archive. For calculating the novelty with respect to the behaviors stored in the novelty archive, A, we have two main approaches: Average (N_{AV}), and k-Nearest Neighbour(V_{kNN}). For the average approach, N_{AV}, the novelty of a proposed $bc_g(g)$ is computed by finding its distance with the average of all the behaviours stored in the novelty archive. In this approach we are effectively searching for behaviours that are novel in average with respect to all the behaviours encountered since the start of the training. N_{AV} is calculated as follows:

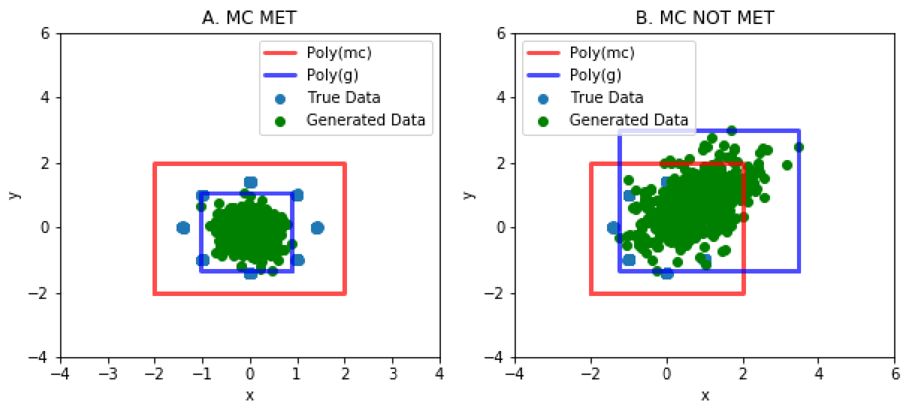

Figure 3.4: To the left, $bc(g)$ met minimal criterion, and will be given a novelty score. To the right $bc(g)$ failed to meet minimal criterion, and will be given a novelty of zero

$$N_{AV}(bc(\theta)) = dist(bc(\theta), \frac{1}{|z|}\sum_{i \in z} bc(\theta_i)), \tag{3.2}$$

where z all the behavior characterizations in A, and $dist$ is a domain dependent distance metric.

The k-Nearest Neighbour approach, N_{kNN}, is the same as originally proposed by Lehman and Stanley (2011a) in equation 2.13, and k is set to 10. For the distance function, we have several options, such as Euclidean, Manhattan, or Hausdorff distance. We choose to follow Lehman and Stanley (2011a) and Conti et al. (2017). Examples of the use of the Euclidean distance function to calculate the difference between two different generator behavior characterizations is show in Figure 3.5.

At the end of each training iteration, our archive is only updated when the minimal criterion is met, and because of this, the generator behavior characterization, $bc_g(g)$, at the start of training needs to be guaranteed to meet this criterion, otherwise, NSR-ES will act as a random search until we find the first

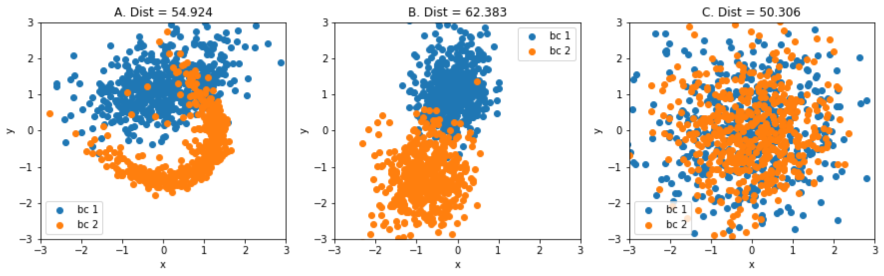

Figure 3.5: Euclidean distance between different generator behavior characterizations.

$bc_g(g)$ that meets the minimal criterion (Lehman and Stanley 2010). To handle this, at the start of training, if $bc_g(g)$ does not meet the minimal criterion, we seed the novelty archive with a dummy $bc_g(g)$ that meets the minimal criterion. The dummy $bc_g(g)$ is a matrix of the same dimension as $bc_g(g)$ and sampled from $\mathcal{N}(-1, 1)$ and is shown in Figure 3.6

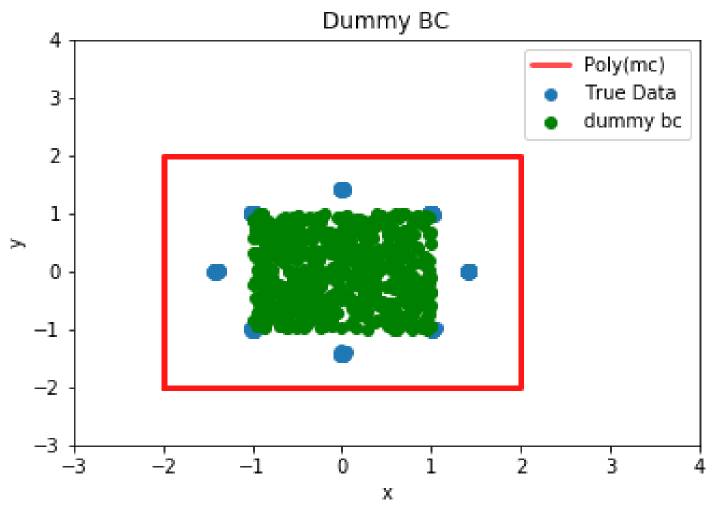

Figure 3.6: Dummy *bc* is seeded to the novelty archive at the start of training to ensure that the novelty search has a starting point and a reference behavioral novelty

The full NSR-ES-GAN algorithm is show in Algorithm 5.

Algorithm 5 NSR-ES-GAN

```
1: Input: learning rates ($\eta_d$, $\eta_g$), noise standard deviations ($\sigma_d$,$\sigma_g$), population $n$, optimizers ($opt_d$,$opt_g$),
   loss functions ($F_d$,$F_g$), reward pressure $p$, minimal criteria $MC$, dummy $bc_g(g)$
2: Compute and Add $bc_g(\theta_{g0})$ to novelty archive $A$ if meets $MC$, else add dummy $bc_g(g)$ to $A$
3: for $t = 0, 1, ...$ do
4:     for $i = 1$ to $n$ do
5:         Sample minibatch of m examples from real data $p_r$
6:         Sample minibatch of m examples from $p_z$
7:         Sample $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$
8:         Compute $f_d^+ = F_d(\theta_{dt} + \sigma_d \times \epsilon_i)$
9:         Compute $f_d^- = F_d(\theta_{dt} - \sigma_d \times \epsilon_i)$
10:    end for
11:    Compute ranked fitness $r_d^+ = ranks(f_d^+)$
12:    Compute ranked fitness $r_d^- = ranks(f_d^-)$
13:    Estimate D Gradient : $g_d \approx \frac{1}{n\sigma} \sum_{i=1}^{n} (\epsilon_i (r_d^+{}_i - r_d^-{}_i))$
14:    Update D $\theta_{dt+1} = \theta_{dt} + opt_d(g_d, \eta_d)$
15:    for $i = 1$ to $n$ do
16:        Sample minibatch of m examples from noise distribution $p_z$
17:        Sample $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$
18:        Compute Novelty $N^+(bc(\theta_{gt} + \sigma_g \times \epsilon_i))$
19:        Compute Novelty $N^-(bc(\theta_{gt} - \sigma_g \times \epsilon_i))$
20:        Compute $f_g^+ = F_g(\theta_{gt} + \sigma_g \times \epsilon_i)$
21:        Compute $f_g^- = F_g(\theta_{gt} - \sigma_g \times \epsilon_i)$
22:    end for
23:    Compute ranked fitness $r_g^+ = ranks(f_d^+)$
24:    Compute ranked fitness $r_g^- = ranks(f_d^-)$
25:    Compute ranked novelty $n_g^+ = normalized\_ranks(N_g^+)$
26:    Compute ranked novelty $n_g^- = normalized\_ranks(N_g^-)$
27:    Estimate G Gradient : $g_g \approx \frac{1}{n\sigma} \sum_{i=1}^{n} (\{(p \times (r_g^+{}_i - r_g^-{}_i)) + (1-p) \times (n_g^+{}_i - n_g^-{}_i)\} \times \epsilon_i)$
28:    Update G $\theta_{gt+1} = \theta_{gt} + opt_g(g_g, \eta_g)$
29:    Compute and Add $bc(\theta_{gt+1})$ to novelty archive $A$ if meets $MC$, else discard
30: end for
```

Chapter 4

Experiments and Results

4.1 Experimental Setup

Our hypothesis is that using NSR-ES to train the generator of the GAN, while using ES to train the discriminator, will help avoid the mode collapse problem by diversifying the output of the generator to cover more modes of the real data distribution. In this chapter we describe our experiments to train our GAN, as proposed in chapter 3, using ES-GAN and NSR-ES-GAN algorithms. For ES-GAN experiments, we report the generator's kernel density estimation (KDE) plot and the training loss curves for D and G. For NSR-ES-GAN, we report the generator's KDE plot, the training loss curves for D and G, and best fit plot for the generator's novelty during training. For all experiments, we set the number of iterations to be 25000, and number of workers (perturbations per generation) to be 39. For the ADAM optimizer (Kingma and Ba 2014), hyper-parameters, we set $\beta_1 = 0.5$ and $\beta_2 = 0.999$ for both D and G. β_1 and β_2 are the exponential decay rates for the first and second moment estimates respectively.

To decided on what to use for the learning rates and the standard deviations for Gaussian noise, we carried out experiments varying both values for $\sigma \in \{0.0002, 0.002, 0.02\}$ and for $\eta \in \{0.0001, 0.001, 0.01\}$, and we found the pair of $\{\sigma = 0.0002, \eta = 0.001\}$ to be the most suitable for both D and G empirically, because they resulted in the expected behavior of mode collapse in our initial experiments.

4.1.1 Experiments with ES-GAN

In ES-GAN, we train the GAN as outlined in algorithm 4, where both D and G are trained using ES.

4.1.1.1 Proof-of-Concept Experiment

In this section, we test our ES-GAN on a dataset of three-Gaussians mixture model, shown in Figure 4.1, as a proof of concept that the underlying algorithm code works. We use the same architecture proposed in chapter 3, but we restrict the dimension of z to be 2, which is the minimum possible dimension of z in relationship to the dimension of the real data set (Goodfellow 2016). The kernel density estimate (KDE) plot is shown in Figure 4.1, and the training loss curves plot is shown in Figure 4.3.

Looking at the results, we can see that the GAN learned all three modes or our dataset as expected. By the end of training, the G learned to produce all the mode of the dataset.

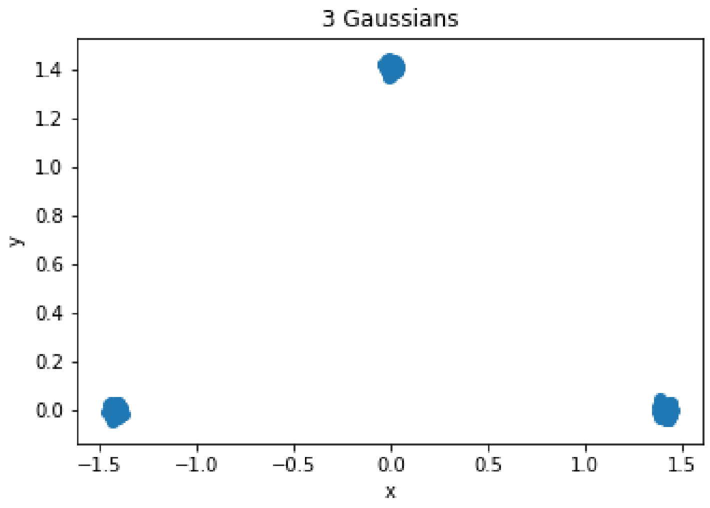

Figure 4.1: A mixture of three Gaussians

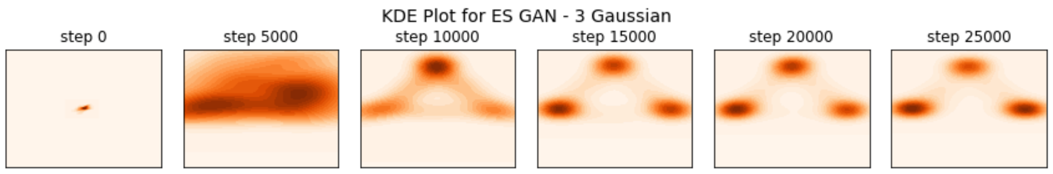

Figure 4.2: KDE plot for 3 Gaussian ES-GAN

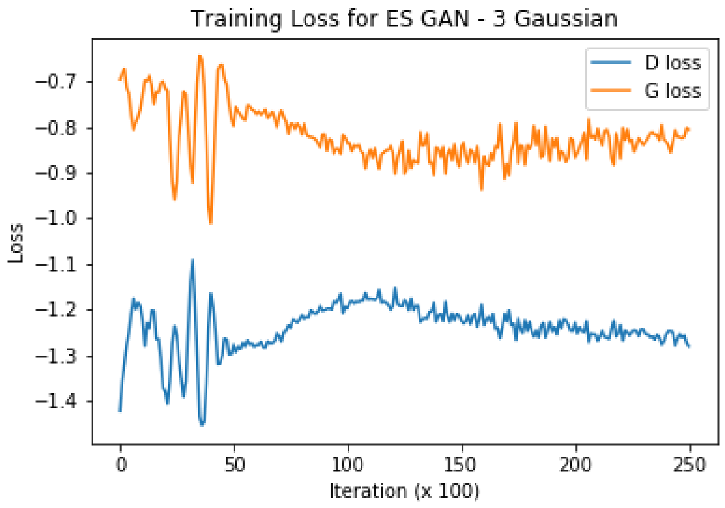

Figure 4.3: ES-GAN training loss curves for 3 Gaussians

4.1.1.2 Simple GAN Experiments

In this section, we experiment with our proposed ES-GAN algorithm to train our simple GAN architecture for the 8 Gaussian mixture experiment. The KDE plot is shown in Figure 4.4, and the training loss plot are shown in Figure 4.5.

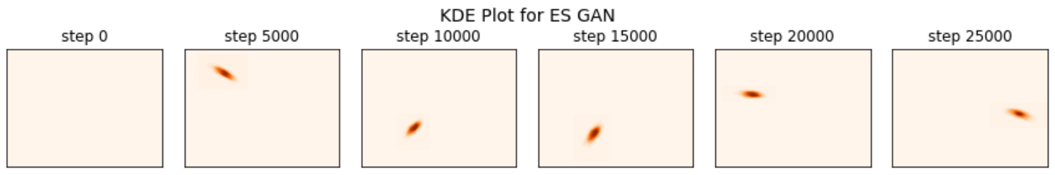

Figure 4.4: KDE plot for ES-GAN

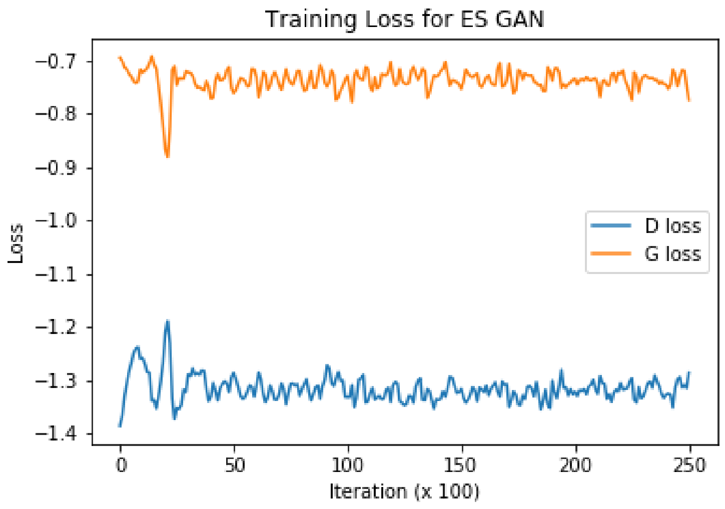

Figure 4.5: Training loss ES-GAN

In these experiments, we used ES to train D and G, and we can see that the GAN collapsed into producing just one mode. This is the classic mode collapse case, in which the generator exhibits a cyclic behavior of learning one mode only, forgetting about the previous modes is learned.

4.1.2 Experiments with NSR-ES-GAN

In this subsection, we train the simple GAN using our proposed NSR-ES-GAN algorithm. We train D using ES, while training G using NSR-ES, as outlined in algorithm 5. We organize our experiments as follows: for each way proposed to calculate novelty with respect to the novelty archive, (N_{AV},N_{kNN}), we carry out three different types of experiments for $p \in \{0.3, 0.5, 0.7\}$. For results, we provide the generator's KDE plot, training loss curves plot, and the generator's novelty plot showing the generator's novelty over training, and the best fit line for the results.

4.1.2.1 Using Archive Average Novelty

In these experiments, the novelty of the current generator is calculated by finding the distance of its behavior characterization, $bc_g(g)$, with respect to the average of all the behavior characterizations stored in the novelty archive.

Experiments with reward pressure $p = 0.3$. For $p = 0.3$, NSR-ES-GAN is emphasizing for increasing the generator's novelty more than minimizing its loss. The KDE plot is shown in Figure 4.6. The training loss is shown in Figure 4.7. The generator's novelty over training is shown in Figure 4.8.

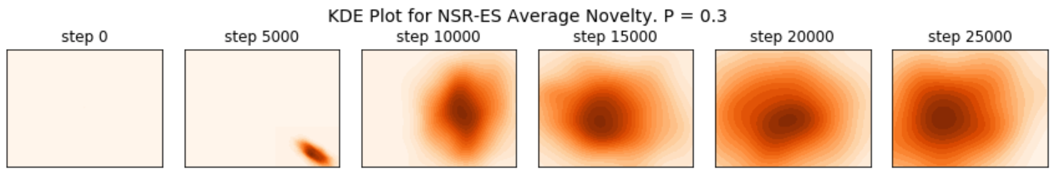

Figure 4.6: KDE plot for NSR-ES-GAN with $p = 0.3$

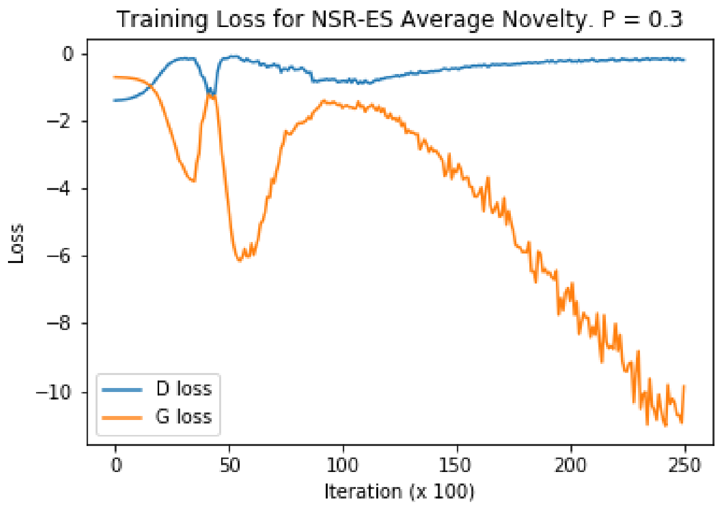

Figure 4.7: Training loss NSR-ES-GAN with $p = 0.3$

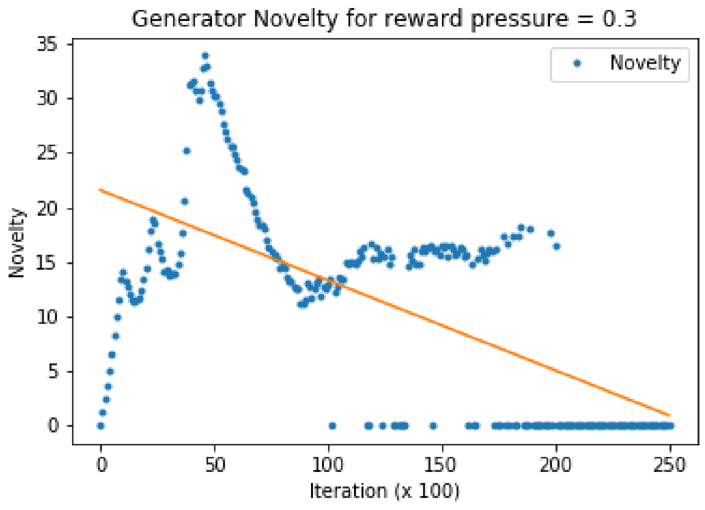

Figure 4.8: Generator average novelty plot with $p = 0.3$

From the KDE plot, we can see that the GAN failed to converge. Looking at the training loss curves, we notice that, as the discriminator gets better during training, the generator gets considerably worse, and its loss decreases dramatically. This suggests that, generator is failing to learn from the signal that it is getting from the discriminator. Looking at the generator's novelty plot, we can see that the novelty fluctuates over the course of training, and it has an overall decreasing tendency.

Experiments with reward pressure $p = 0.5$. For $p = 0.5$, NSR-ES-GAN is placing the same pressure on increasing the generator's novelty as well as minimizing the its loss. The KDE plot is shown in Figure 4.9. The training loss plot is shown in Figure 4.10. The generator's novelty over training is shown in Figure 4.11.

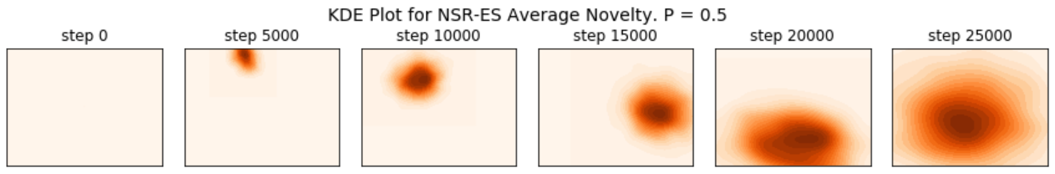

Figure 4.9: KDE plot for NSR-ES-GAN with $p = 0.5$

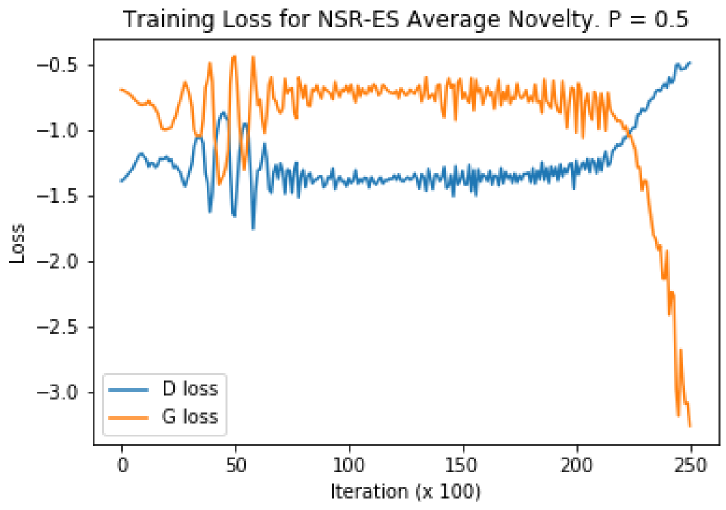

Figure 4.10: Training loss NSR-ES-GAN with $p = 0.5$

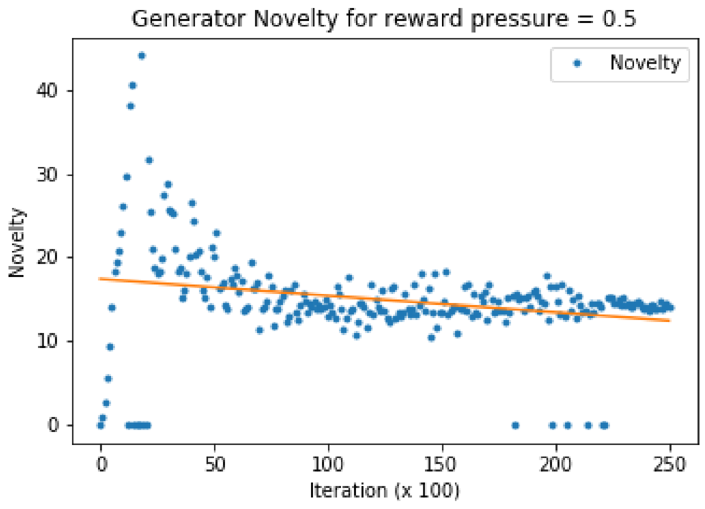

Figure 4.11: Generator average novelty plot with $p = 0.5$

We can see that the results for $p = 0.5$ are very similar to the ones for $p = 0.3$: the GAN fails to converge, the generator loss decreases dramatically in response to the discriminator loss increasing, and the novelty shows a downwards trend.

Experiments with reward pressure $p = 0.7$. For $p = 0.7$, NSR-ES-GAN is emphasizing minimizing the generator loss more than increasing its novelty. The KDE plot is shown in Figure 4.12. The training loss curves plot is shown in Figure 4.13. The generator's novelty over training is shown in Figure 4.14.

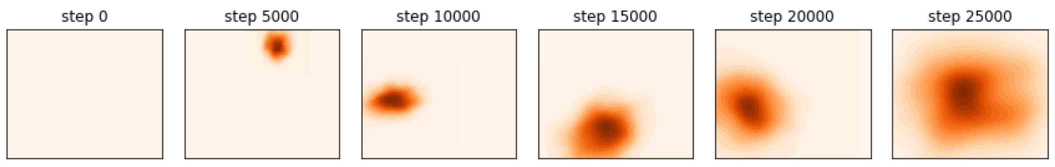

Figure 4.12: KDE plot for NSR-ES-GAN with $p = 0.7$

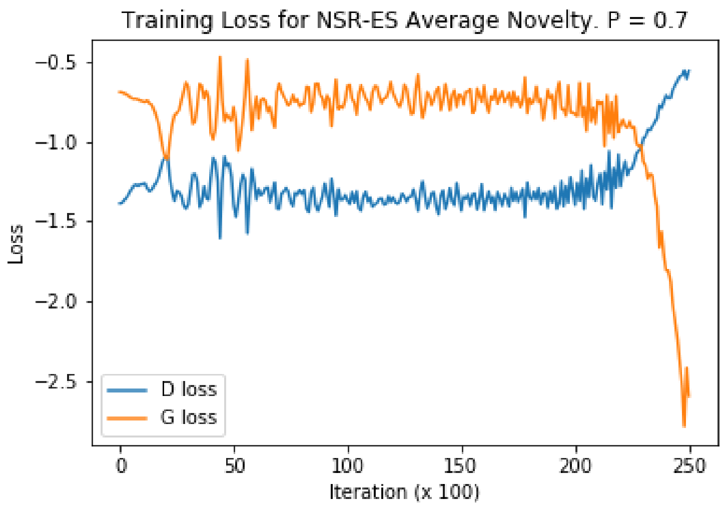

Figure 4.13: Training loss NSR-ES-GAN with $p = 0.7$

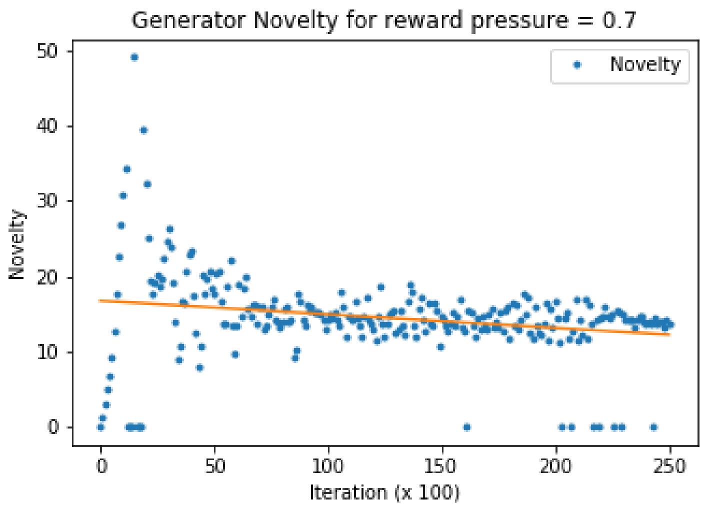

Figure 4.14: Generator average novelty plot with $p = 0.7$

The experiments for $p = 0.7$ show the same trend for both $p = 0.3$ and 0.5. ES-NSR-GAN failed to increase novelty based on our behavior characterization, and the generator failed to learn from the signal of novelty and the discriminator. The GAN failed to converge to a solution that learned our target dataset.

4.1.2.2 Using the k-Nearest Neighbour Novelty Archive

In this section, we carry out experiments in which we calculate the novelty of the generator by calculating the average distances between its behavior characterization and the k-nearest neighbours in the novelty archive, as originally suggested by Lehman and Stanley (2011a). For all our experiments, we set $k = 10$.

Experiments with reward pressure $p = 0.3$. For p= 0.3, NSR-ES-GAN is emphasizing for increasing the generator's novelty more than minimizing its loss. The KDE plot is shown in Figure 4.15. The training loss curves plot is show in Figure 4.16. The generator's novelty over training is shown in Figure 4.17

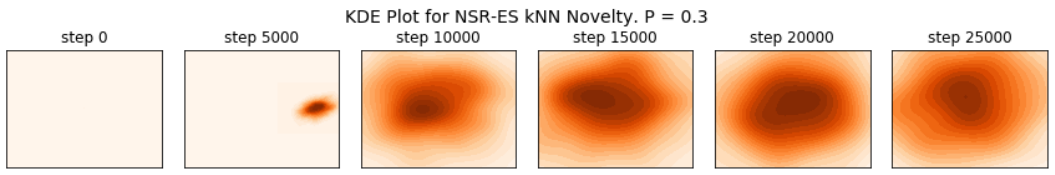

Figure 4.15: KDE plot for NSR-ES-GAN kNN novelty with $p = 0.3$

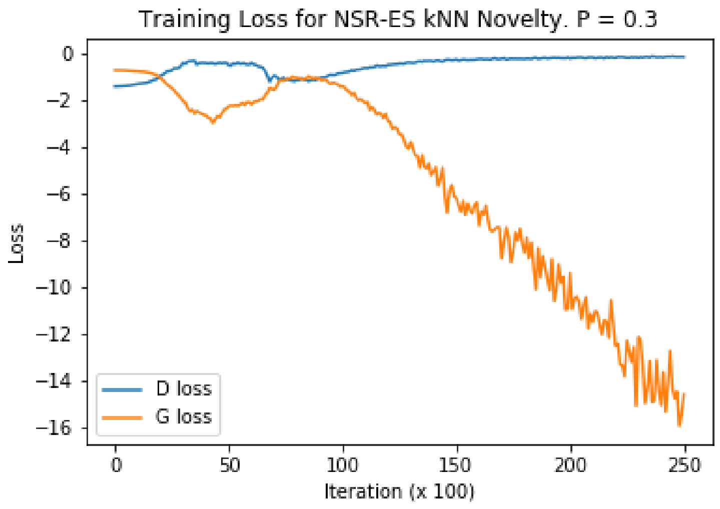

Figure 4.16: Training loss NSR-ES-GAN with $p = 0.3$

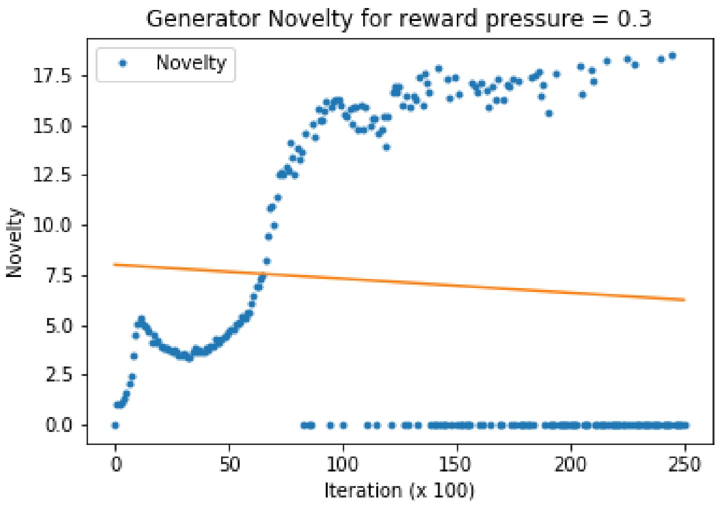

Figure 4.17: Generator kNN novelty plot with $p = 0.3$, and k = 10

We can see from the results that the GAN fails to converge, and that the generator reacts badly to improvements in the discriminator. The novelty of the generator increases over training, however, the overall trend is still downwards.

Experiments with reward pressure $p = 0.5$. For p= 0.5, NSR-ES-GAN is placing the same pressure on increasing the generator's novelty, as well as minimizing the its loss. The KDE plot is shown in Figure 4.18. The training loss curves plot is shown in Figure 4.19. The generator's novelty over training is shown in Figure 4.20

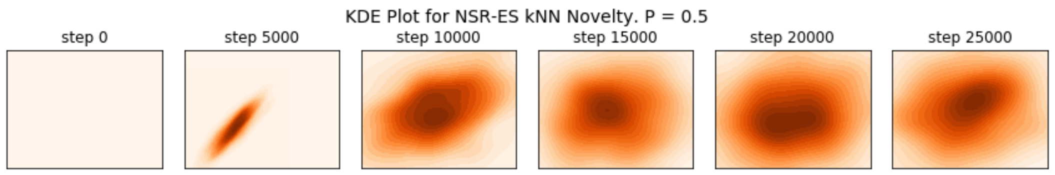

Figure 4.18: KDE plot for NSR-ES-GAN kNN novelty with $p = 0.5$

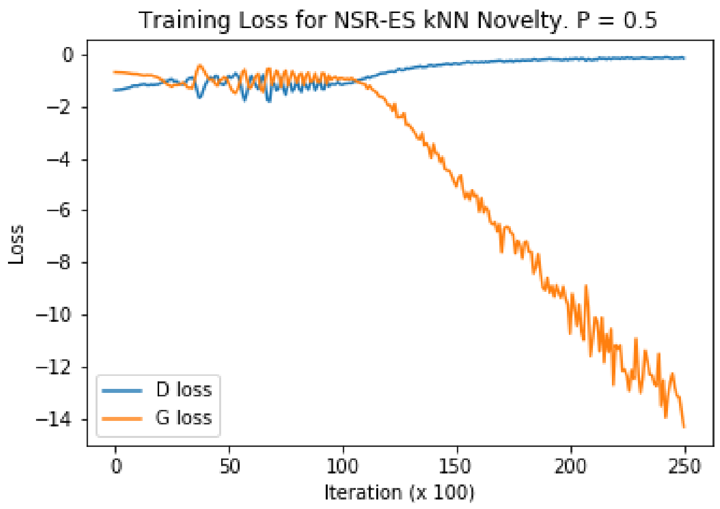

Figure 4.19: Training loss NSR-ES-GAN with $p = 0.3$

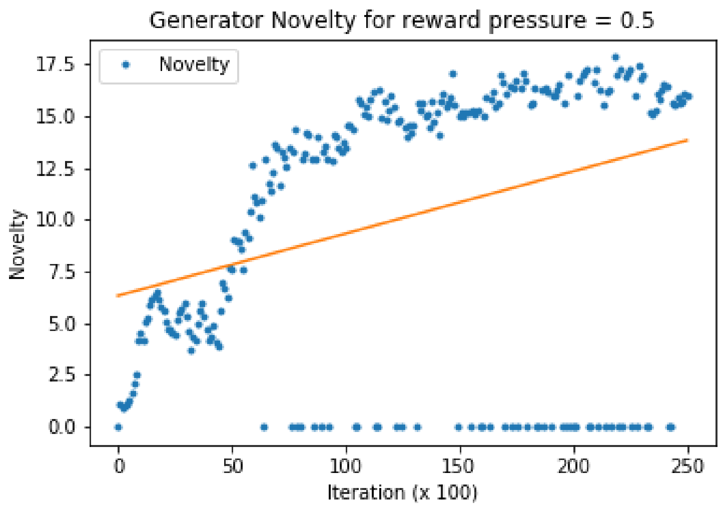

Figure 4.20: Generator kNN novelty plot with $p = 0.5$, and k = 10

The results here are similar to our experiments before. The GANs fails to converge, and the generator's loss deceases dramatically in response to an increase of the discriminator's loss. However, we notice an upward increase in the generators novelty.

Experiments with reward pressure $p = 0.7$.For p= 0.7, NSR-ES-GAN is emphasizing minimizing the generator loss more than increasing its novelty. The KDE plot is shown in Figure 4.21. The training Loss is show in Figure 4.22. The generator's novelty over training is shown in Figure 4.23.

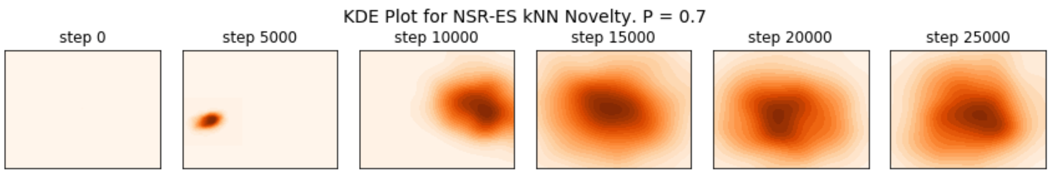

Figure 4.21: KDE Plot for NSR-ES-GAN kNN novelty with $p = 0.7$

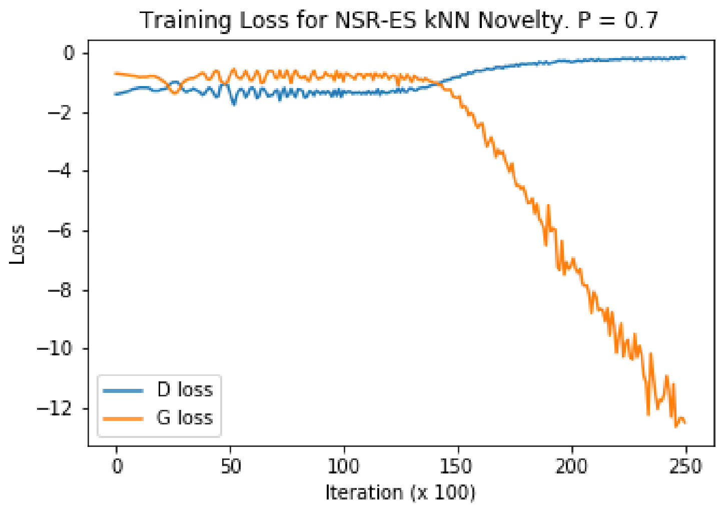

Figure 4.22: Training loss NSR-ES-GAN with $p = 0.7$

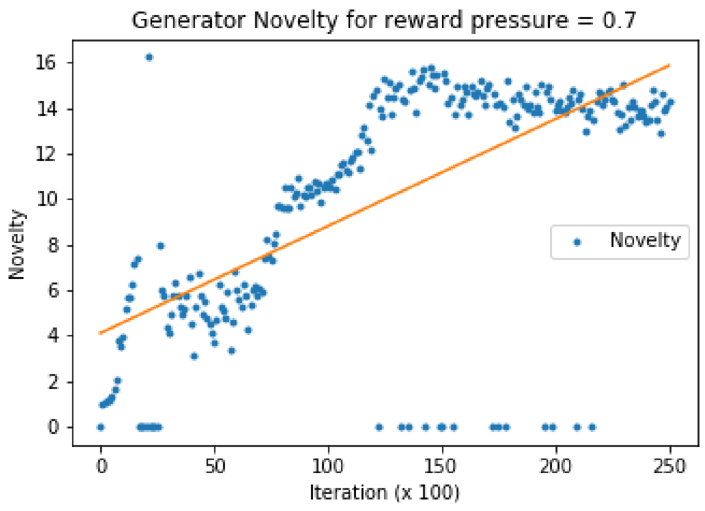

Figure 4.23: Generator kNN novelty plot with $p = 0.7$, and k = 10

Result from this experiment are very similar to the experiment with $p = 5$.

4.1.3 Discussion

All the experiments that we carried under our setup failed to converge to a solution that learned our target dataset. The generator seems to push for novelty, based on our generator's behavior characterization, however, this behavior seems to worsen its learning of the true data distribution, and subsequently causes the GAN to fail to converge, and instead diverging with a dramatic decrease in the generator's loss. Our experiments suggest that, for our proposed behavior characterization, calculating an individual's novelty using the kNN approach, as suggested by Lehman and Stanley (2011a), is the more suitable approach as the novelty of the generator increased during training. However, it is important to note that all these results are based on our proposed behavior characterization, which failed to help the GAN to converge to a solution.

Chapter 5

Conclusion and Future Work

In this section we present conclusions that can be drawn for our experiments and suggest directions for future work.

5.1 Conclusion

In this thesis, we studied the use of Evolution Strategies (ES), and Novelty Seeking Reward Evolution Strategies (NSR-ES) in training Generative Adversarial Networks (GANs), targeting the issue of mode collapse. We used an architecture setup of GANs where mode collapse exits and is easy to visualize. We presented algorithms to use ES and NSR-ES to train GANs. We designed NSR-ES to tackle the issue of mode collapse by incentivizing the generator output to be more diverse to cover more modes of the real data distribution. However, our experimentation using this setup and definition of NSR-ES has failed to help the GAN to converge into a solution that overcame the mode collapse issue, and we conclude that, under this setup, Novelty Seeking Reward Evolution Strategies has failed towards helping the GANs overcome the mode collapse issue.

As highlighted earlier, the dynamics of training GANs are still largely not well understood, thus providing a theoretical analysis of our work work is not an option at the mean time given the current state of research in GANs. However, we would like to point out the reasons we believe that using ES and NSR-ES failed to help the GAN converge in our experimentation. Starting with ES-GAN, we used ES to train both D and G instead of first order stochastic gradient descent (SGD). ES, just like SGD, estimates the gradient of a loss function. In our ES-GAN experiments, the GAN collapsed to producing only one mode, which was identical the the behavior of SGD. This suggests that using ES only to train GANs, the issue of mode collapse is related to the training dynamics of GANs, and the interactions between D and G, rather than the optimization method used, which was the main reason that inspired us to use NSR-ES to augment the behavior of G.

Moving into our experiments with NSR-ES. We proposed NSR-ES-GAN, in which D was trained using ES, while G was trained using NSR-ES. We proposed a behaviour characterization based on the output of G, with the main goal being to incentive it to be different, as in mode collapse the output of the generator tends to converge to a single, or few modes of the datasets. We also proposed two ways to calculate novelty with respect to novelty archive: One that used 10-th nearest neighbours to calculate novelty, and one where we averaged all the behaviours in the novelty archive, then measured the distance with the current generator. We also proposed different reward pressures where we sought to exploit NSR-ES on G to put more pressure on increasing the novelty of the generator, or minimizing its loss. Throughout our experimentation, our GAN failed to coverage, not producing any modes of the real dataset. To better understand what happened, we look at the key difference between ES-GAN, which suffered from mode collapse, and NSR-ES-GAN, which failed to converge to a solution. The key difference is the use of novelty search, defined by our behavior characterization. We believe that the novelty search component of NSR-ES-GAN failed to provide additional and useful learning signal to the generator, and instead worsened its learning process. This can be seen in our training loss curves where the loss of the generator decreased dramatically over training, and failed to converge. There are two main reasons that we believe caused our novelty signal to be unhelpful to G: the alignment of behaviour characterization of the generator, and the novelty landscape of GANs.

The alignment of the behavior characterization is defined as the degree to which finding novel behaviours based on the behavior characterization leads to higher fitness or, in our case, better G (Pugh et al. 2016). However, our proposed behavior characterization for the generator was to search for novelty in terms of diversity, which is, as pointed by Pugh et al. (2016), intrinsically not aligned with fitness. The main idea of using NSR-ES-GAN is push the generator to explore new generating diverse output, rather than collapsing to a single solution, in one hand, and in the other hand using the signal from the discriminator to drive this diversity towards matching the real data distribution. This is the main idea of quality diversity (QD) algorithms, and interestingly enough, much of the published research using QD algorithms to search for high quality, yet, diverse solutions, employed unaligned behavior characterizations (Lehman and Stanley 2011b; Cully and Mouret 2013; Mouret and Clune 2015; Pugh et al. 2016), which was the main motivation of the choice of our behavior characterization.

Another challenging aspect about using novelty search on the GAN framework is the changing novelty landscape of GANs. The use of novelty search has been mainly conducted on reinforcement learning problems, in which the environment stayed constant over the course of training. Unfortunately this is the not the case in GANs. One can look at GANs as a reinforcement problem, in which the generator (the agent) is trying to reach certain level or objective by interacting with the discriminator (the environment). In NSR-ES-GAN, the generator is learning from the discriminator, and is using what it learns to produce more novel behaviors by using novelty search. The challenge is that the discriminator (the environment) itself is changing as the discriminator is learning to be a better classifier, thus the different behaviours produced by the generator, which are indirectly affected by the signal it learned from the discriminator, may not be as relevant throughout training. Thus, the changing novelty landscape of the GANs might have severely harmed the usefulness of the novelty returns, and rendered it useless.

To the best of the author's knowledge, it is the first time that an ES algorithm has been used in the domain of GANs. The ideas and experimentation provided here are to be seen as an approach to take into consideration for any future work or research interest in applying more evolutionary computation algorithms to the same domain. While we had hoped to tackle the issue of mode collapse, our experiments failed, and instead, we are presented with yet another layer of complex dynamics in relation to training GANs while leveraging novelty search. We conclude that this constitutes a good stopping point for the direction of this particular set of ideas, but we hope more researchers investigate this line of research.

5.2 Future Work

For future work, there are very interesting ideas that are worth exploring and taking into consideration, the most important being the behaviour characterization used in NSR-ES. We believe that future work should focus on more robust and inclusive behavior characterization of GANs, that would characterize both the generator and the discriminator together. Researching how to develop and use a more robust and behavior characterization of GANs will help understand the need for its alignment with respect to the optimization landscape, which is yet another interesting question. This could also help in understanding the necessity of minimal criteria, and further understand whether it is needed, or could be removed.

On the use of evolutionary algorithms on the domain of GANs, future work could investigate the use of algorithms other than evolution strategies and novelty search. Two particular approaches that are related to GANs are Predator-Prey Co-evolution and Speciation. In Predator-Prey Co-evolution, two competing sets of co-evolving populations are trying to learn behaviours against each other so one population can beat the other. Nolfi and Floreano (1998) researched the this idea by co-evolving two robots, a predator and a prey, in a co-evolutionary setup, where the predator was rewarded for touching the prey, and the prey was rewarded for escaping from the predator. One can draw the similarities between the co-evolutionary setup and the GAN setup, where G and D can be view as the predator, prey respectively. In their research, Nolfi and Floreano (1998) noticed that the basic predator-prey co-evolutionary setup results in a cycling behavior in which the same behaviours are exhibited by the the predator and prey over and over. The cycling behaviour of the predator-prey is very similar to that of of the issue of mode collapse and non-convergence of GANs. To overcome this behavior, Nolfi and Floreano (1998) proposed the Master Tournament technique, which leveraged the knowledge accumulated from past generations by testing the best individual of each each generation of one population with the best individual of all generations of the competing population. Future research could tackle such problems by drawing from the co-evolutionary research.

Another interesting evolutionary approach is Speciation. Speciation aims to overcome the issue of genetic drift in evolutionary algorithms (EA), which is defined as the EA tendency to converge quickly to a single fit solutions Della Cioppa et al. (2011). In many search spaces, it is of interest to find and maintain multiple diverse solutions. Niching techniques are the most common approach to speciation in evolutionary algorithms, where it is used to maintain and promote diverse sub-population in the search space. Niching has been used in multi objective optimization (Ursem 2002), genetic algorithms (Goldberg and Richardson 1987), and in evolution strategies (Shir and Back 2005). In relation to GANs, future work can frame the mode collapse issue of as lack of diversity of the generator output, as in this work, and research using niching techniques to overcome this problem.

Last, the GANs frame has proven to be a very hard test-bed for our ideas because of the challenges with GANs, and future work should also investigate the use of novelty seeking evolution strategies with other types of generative models than GANs such as Variational Auto Encoders (Kingma and Welling 2013)

Reference List

Akash, S., V. Lazar, R. Chris, G. M. U., and S. Charles, 2017: Veegan: Reducing mode collapse in gans using implicit variational learning. *Neural Information Processing Systems*.

Chen, X., Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, 2016: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds., Curran Associates, Inc., 2172–2180.

Chrabaszcz, P., I. Loshchilov, and F. Hutter, 2018: Back to basics: Benchmarking canonical evolution strategies for playing atari.

Conti, E., V. Madhavan, F. P. Such, J. Lehman, K. O. Stanley, and J. Clune, 2017: Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents.

Cully, A. and J.-B. Mouret, 2013: Behavioral repertoire learning in robotics. *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, 175–182.

Della Cioppa, A., A. Marcelli, and P. Napoli, 2011: Speciation in evolutionary algorithms: Adaptive species discovery. *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, 1053–1060.

Geweke, J., 1988: Antithetic acceleration of monte carlo integration in bayesian inference. *Journal of Econometrics*, **38**, 73–89.

Goldberg, D. E. and J. Richardson, 1987: Genetic algorithms with sharing for multimodal function optimization. *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, 41–49.

Goodfellow, I., 2016: Nips 2016 tutorial: Generative adversarial networks.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, 2014: Generative adversarial nets. *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds., Curran Associates, Inc., 2672–2680.

Isola, P., J.-Y. Zhu, T. Zhou, and A. A. Efros, 2017: Image-to-image translation with conditional adversarial networks. *CVPR*.

Karras, T., T. Aila, S. Laine, and J. Lehtinen, 2018: Progressive growing of GANs for improved quality, stability, and variation. *International Conference on Learning Representations*.

Kingma, D. P. and J. Ba, 2014: Adam: A method for stochastic optimization.

Kingma, D. P. and M. Welling, 2013: Auto-encoding variational bayes.

Kumar, A., P. Sattigeri, and T. Fletcher, 2017: Semi-supervised learning with gans: Manifold invariance with improved inference. *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., Curran Associates, Inc., 5534–5544.

Lehman, J., J. Chen, J. Clune, and K. O. Stanley, 2017: Es is more than just a traditional finite-difference approximator.

Lehman, J. and K. O. Stanley, 2008: Exploiting open-endedness to solve problems through the search for novelty. *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI*, MIT Press.

—, 2010: Revising the evolutionary computation abstraction: Minimal criteria novelty search. *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, 103–110.

—, 2011a: Abandoning objectives: Evolution through the search for novelty alone. *Evol. Comput.*, **19**, 189–223.

—, 2011b: Evolving a diversity of virtual creatures through novelty search and local competition. *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, 211–218.

Li, J., A. Madry, J. Peebles, and L. Schmidt, 2018: On the limitations of first-order approximation in GAN dynamics. *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, eds., PMLR, Stockholmsmssan, Stockholm Sweden, volume 80 of *Proceedings of Machine Learning Research*, 3005–3013.

Li, Y., J. Song, and S. Ermon, 2017: Infogail: Interpretable imitation learning from visual demonstrations. *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., Curran Associates, Inc., 3812–3822.

Lin, Z., A. Khetan, G. Fanti, and S. Oh, 2018: Pacgan: The power of two samples in generative adversarial networks. *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., Curran Associates, Inc., 1504–1513.

Mao, X., Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, 2017: Least squares generative adversarial networks. *The IEEE International Conference on Computer Vision (ICCV)*.

Mescheder, L., S. Nowozin, and A. Geiger, 2017: The numerics of gans. *Advances in neural information processing systems*.

Metz, L., B. Poole, D. Pfau, and J. Sohl-Dickstein, 2016: Unrolled generative adversarial networks.

Mouret, J.-B. and J. Clune, 2015: Illuminating search spaces by mapping elites.

Mller, N. and T. Glasmachers, 2018: Challenges in high-dimensional reinforcement learning with evolution strategies.

Nguyen, T., T. Le, H. Vu, and D. Phung, 2017: Dual discriminator generative adversarial nets. *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., Curran Associates, Inc., 2667–2677.

Nolfi, S. and D. Floreano, 1998: Coevolving predator and prey robots: Do "arms races" arise in artificial evolution? *Artif. Life*, **4**, 311–335.

Pugh, J. K., L. B. Soros, and K. O. Stanley, 2016: Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, **3**, 40.

Pugh, J. K., L. B. Soros, P. A. Szerlip, and K. O. Stanley, 2015: Confronting the challenge of quality diversity. *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, 967–974.

Rechenberg, I., 1973: *Evolutionsstrategie; Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Mit einem Nachwort von Manfred Eigen*. Frommann-Holzboog [Stuttgart-Bad Cannstatt], 170 p. pp.

Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, 2016: Improved techniques for training gans. *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds., Curran Associates, Inc., 2234–2242.

Salimans, T., J. Ho, X. Chen, S. Sidor, and I. Sutskever, 2017: Evolution strategies as a scalable alternative to reinforcement learning.

Schwefel, H.-P., 1977: *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, volume 26 of *ISR*. Birkhaeuser, Basel/Stuttgart, 390 pp.

Shir, O. M. and T. Back, 2005: Dynamic niching in evolution strategies with covariance matrix adaptation. *2005 IEEE Congress on Evolutionary Computation*, volume 3, 2584–2591 Vol. 3.

Such, F. P., V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, 2017: Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning.

Thanh-Tung, H., T. Tran, and S. Venkatesh, 2018: On catastrophic forgetting and mode collapse in generative adversarial networks.

Tolstikhin, I., S. Gelly, O. Bousquet, C. J. Simon-Gabriel, and B. Schölkopf, 2017: Adagan: Boosting generative models. *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., 5430–5439.

Ursem, R. K., 2002: Diversity-guided evolutionary algorithms. *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, PPSN VII, 462–474.

Vondrick, C., H. Pirsiavash, and A. Torralba, 2016: Generating videos with scene dynamics. *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds., Curran Associates, Inc., 613–621.

Wang, C., C. Xu, X. Yao, and D. Tao, 2018: Evolutionary generative adversarial networks.

Wierstra, D., T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, 2014: Natural evolution strategies. *J. Mach. Learn. Res.*, **15**, 949–980.

Wierstra, D., T. Schaul, J. Peters, and J. Schmidhuber, 2008: Natural evolution strategies. *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 3381–3387.

Williams, R. J., 1992: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, **8**, 229–256.

Zhang, X., J. Clune, and K. O. Stanley, 2017: On the relationship between the openai evolution strategy and stochastic gradient descent.

Zhu, J.-Y., T. Park, P. Isola, and A. A. Efros, 2017: Unpaired image-to-image translation using cycle-consistent adversarial networkss. *Computer Vision (ICCV), 2017 IEEE International Conference on*.