Multi-Objective Evolutionary GAN

Marco Baioletti
University of Perugia
marco.baioletti@unipg.it

Carlos Artemio Coello Coello
CINVESTAV-IPN
ccoello@cs.cinvestav.mx

Gabriele Di Bari
University of Perugia
gabriele.dibari@unifi.it

Valentina Poggioni
University of Perugia
valentina.poggioni@unipg.it

ABSTRACT

Generative Adversarial Network (GAN) is a generative model proposed to imitate real data distributions. The original GAN algorithm has been found to be able to achieve excellent results for the image generation task, but it suffers from problems such as instability and mode collapse. To tackle these problems, many variants of the original model have been proposed; one of them is the Evolutionary GAN (EGAN), where a population of generators is evolved.

Inspired by EGAN, we propose here a new algorithm, called Multi-Objective Evolutionary Generative Adversarial Network (MO-EGAN), which reformulates the problem of training GANs as a multi-objective optimization problem. Thus, Pareto dominance is used to select the best solutions, evaluated using diversity and quality fitness functions.

Preliminary experimental results on synthetic datasets show how the proposed approach can achieve better results than EGAN.

KEYWORDS

General Adversarial Network, Multi Objective, Evolutionary Algorithms, Deep Generative Models

ACM Reference Format:
Marco Baioletti, Carlos Artemio Coello Coello, Gabriele Di Bari, and Valentina Poggioni. 2020. Multi-Objective Evolutionary GAN. In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3377929.3398138

1 INTRODUCTION

Generative models have a long history in the machine learning field; these tools are a powerful way to learn a distribution from a set of real data.

The introduction of deep learning gives a new life to this research field, thus, new models have been proposed such as PixelRNN [13], PixelCNN[14], Variational Autoencoders [10], and many others, which can tackle this task in a surprising way.

Using these new models, we can generate realistic data distribution such as images, videos, audios and so on. A family of these models is based on *Generative Adversarial Networks* (GANs). GANs train two neural networks, called *discriminator* and *generator*, that compete with each other in a two-player *minimax* game. This model was designed to learn from large scale real-data distributions like for example CalebA [7], LSUN [19] and ImageNet [2].

Nevertheless, this model can learn a limited distribution path since it suffers from the vanishing gradient issue and mode collapse, which leads to training a generator able to generate a limited diversity of samples, diverging from the real-data distribution to be imitated.

Thus, a huge number of recent works have focused on overcoming these issues by proposing several generator's objective functions. In [15] the authors proposed the *Evolutionary GAN* (*EGAN*) model. In that work, a population of generators is evolved using different training objectives to produce new candidates for the next generation. In *EGAN*'s selection step, the new population of generators is reduced using a fitness function which takes into account quality and diversity of the solutions.

In this work, we propose the Multi-Objective Evolutionary Generative Adversarial Network (*MO-EGAN*) model, where the evaluation of generators is re-defined as a multi-objective problem, using quality and diversity as two distinct conflicting objectives. Thus, in contrast to *EGAN*, *MO-EGAN* performs the selection between the parents and offspring using a technique taken from the evolutionary multi-objective optimization field. Our preliminary results using synthetic data have shown how the two objectives selected are indeed conflicting. Additionally, we show how this new approach can actually provide better results than *EGAN*.

This paper is organized as follows. In Section 2 we provide a brief summary of the *GAN* algorithm and the formulation of the multi-objective problem of our interest. Section 3 describes in detail the proposed algorithm: *MO-EGAN*. Section 4 provides details of our preliminary experiments aimed to validate our proposed approach. In Section 5 the analysis of the results obtained in terms of qualitative and quantitative measures is given. We also provide a short discussion on the Pareto fronts obtained and the complexity of the algorithm proposed. Finally, in Section 6, we present our conclusions and some possible paths for future research.

2 BACKGROUND

In this section, we provide a short description of the Generative adversarial networks, and some basic definitions used in multi-objective optimization.

GECCO '20 Companion, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7127-8/20/07.
https://doi.org/10.1145/3377929.3398138

2.1 Generative Adversarial Networks

The Generative Adversarial Network was originally proposed by Goodfellow et al. [3].

This framework re-defines the generative model as a two-player game, where a generator network competes with a discriminator one. The generator has the task to generate real-looking distributions (e.g. images) while the discriminator has to distinguish fake distributions from original ones. The two adversaries play a continuous match as the generator tries to fool the discriminator, while the other tries not to be fooled. To generate the best distribution we need a very good generator and a very good discriminator: this is because if the generator is not good enough, it will never be able to fool the discriminator and the model will never converge. On the other hand, if the discriminator is not good enough, then distributions without any sense are classified as real and the model will never learn.

Formally, given a noise sample Z generated from a uniform distribution p_{noise} (i.e. fake data), and a data sample X generated from the data distribution p_{data} (i.e. real data), the generator G produces an output according to a distribution $G(Z)$, which is as close as possible to p_{data}, while a discriminator D tries to detect which are the true samples (X) and which one came from the $G(Z)$ fake distribution.

The original formulation of the two-players minimax game described in [3] is:

$$\min_{G} \max_{D} \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim p_{noise}} [1 - \log D(G(x))]$$

In the specialized literature, many different reformulations of the original one are available [9, 16, 17].

2.2 Multi-Objective Problem

In multiobjective optimization, the aim is to solve problems of the type[1]:

$$\text{minimize } \vec{f}(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})] \tag{1}$$

subject to:

$$g_i(\vec{x}) \leq 0 \quad i = 1, 2, \ldots, m \tag{2}$$

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{3}$$

where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables, $f_i : \mathbf{R}^n \to \mathbf{R}$, $i = 1, \ldots, k$ are the objective functions and $g_i, h_j : \mathbf{R}^n \to \mathbf{R}$, $i = 1, \ldots, m$, $j = 1, \ldots, p$ are the constraint functions of the problem.

A few additional definitions are required to introduce the notion of optimality used in multiobjective optimization:

Definition 1. Given two vectors $\vec{x}, \vec{y} \in \mathbf{R}^k$, we say that $\vec{x} \leq \vec{y}$ if $x_i \leq y_i$ for $i = 1, ..., k$, and that $\vec{x}$ **dominates** $\vec{y}$ (denoted by $\vec{x} \prec \vec{y}$) if $\vec{x} \leq \vec{y}$ and $\vec{x} \neq \vec{y}$.

Definition 2. We say that a vector of decision variables $\vec{x} \in \mathcal{X} \subset \mathbf{R}^n$ is **nondominated** with respect to $\mathcal{X}$, if there does not exist another $\vec{x}' \in \mathcal{X}$ such that $\vec{f}(\vec{x}') \prec \vec{f}(\vec{x})$.

Definition 3. We say that a vector of decision variables $\vec{x}^* \in \mathcal{F} \subset \mathbf{R}^n$ ($\mathcal{F}$ is the feasible region) is **Pareto-optimal** if it is nondominated with respect to $\mathcal{F}$.

Definition 4. The **Pareto Optimal Set** $\mathcal{P}^*$ is defined by:

$$\mathcal{P}^* = \{\vec{x} \in \mathcal{F} | \vec{x} \text{ is Pareto-optimal}\}$$

Definition 5. The **Pareto Front** $\mathcal{PF}^*$ is defined by:

$$\mathcal{PF}^* = \{\vec{f}(\vec{x}) \in \mathbf{R}^k | \vec{x} \in \mathcal{P}^*\}$$

Therefore, our aim is to obtain the Pareto optimal set from the set $\mathcal{F}$ of all the decision variable vectors that satisfy (2) and (3). Thus, given a Multi-Objective Problem (MOP), the goal of a Multi-Objective Evolutionary Algorithm (MOEA) is to produce a good approximation of the Pareto front.

One of the most widely used MOEAs for problems having only two or three objectives is the elitist Nondominated Sorting Genetic Algorithm (*NSGA-II*) [1]. This algorithm solves a MOP using nondominated sorting and a crowding-comparison operator that acts as its density estimator.

3 MO-EGAN

MO-EGAN algorithm resembles the classical *GAN* algorithm [3]: it evolves a population of μ generators $G_1, \ldots, G_\mu$, represented by the corresponding parameters [2] $\Theta^1, \ldots, \Theta^\mu$, respectively, and a single discriminator D, represented by the parameter vector w.

Its main loop is composed of two phases: in the first phase D is trained and in the second phase all the generators are trained. These two phase are alternately executed until a satisfactory solution is found. In contrast with *GAN* and analogously with the Evolutionary *GAN* [15], *MO-EGAN*, during the generator training phase, evolves the population of generators using three steps: Variation, Evaluation, and Selection.

The description of *MO-GAN* is provided in Algorithm 1. The discriminator training phase is composed of N_d steps described in lines 3-7. At each step a sample $\vec{x}$ of m elements is generated from p_{data} and a sample $\vec{z}$ of the same size is generated from a noise distribution and divided in μ batches of size m/μ. Each generator G_j uses its own batch to generate fake data. One step of the Adam neural network optimizer algorithm is then used to update the weights w of D.

The generator training phase is described between lines 9 and 13. During this phase, the Variation procedure is applied to all the parents, producing a subpopulation of $N_m \cdot \mu$ children.

In the evaluation step, two fitness functions are computed for each child and also for each parent. Finally, multi-objective selection is applied.

---

[1] Without loss of generality, we will assume only minimization problems.

[2] the network connection weights

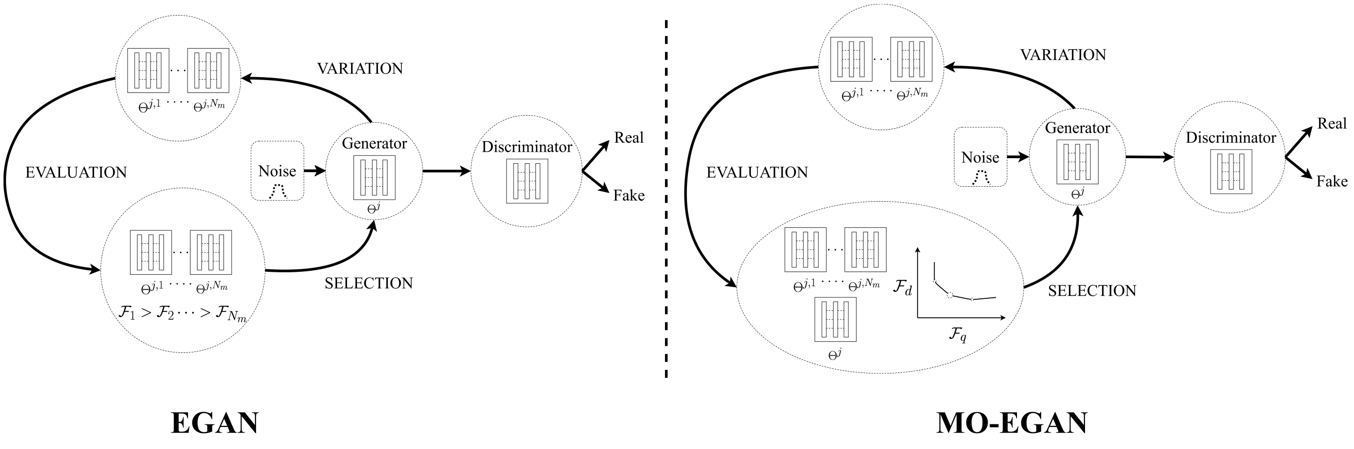

Figure 1: The *EGAN* algorithm diagram compared to the *MO-EGAN* one. The *EGAN* evolve a population of generator $\{\Theta^j\}$ and select the best on the new population by a fitness function $\mathcal{F}$. Instead the *MO-EGAN* algorithm uses the Pareto dominance on a population $\{\Theta^j, \Theta^{j,1}, \ldots \Theta^{j,N_m}\}$ to select the bests ones for the next generation.

Algorithm 1 MO-EGAN

```
Require: batch size $m$, number of parents $\mu$, discriminator's updating steps per iteration $N_d$, number of objective functions $N_m$, Adam hyperparameters $\alpha$, $\beta_1$, $\beta_2$
Require: Initial discriminator's parameters $w$
Require: Initial generators' parameters $\Theta^1, \ldots, \Theta^\mu$
for number of training iterations do
    for $k \leftarrow 1 \ldots N_d$ do
        Sample a batch $\vec{x}$ from dataset $P_{data}$
        Generate noise batch $\vec{z}$ from a uniform distribution
        $g_w \leftarrow \nabla_w \Big[ \frac{1}{m} \sum_{i=1}^{m} \log D_w(x_i)$
                $+ \frac{1}{m} \sum_{j=1}^{\mu} \sum_{i=1}^{m/\mu} \log(1 - D_w(G_j(z_i))) \Big]$
        $w \leftarrow Adam(g_w, w, \alpha, \beta_1, \beta_2)$
    end for
    for $j \leftarrow 1 \ldots \mu$ do
        Generate noise batch $\vec{z}$ from a uniform distribution
        $\Theta^{j,1 \ldots N_m}_{child} \leftarrow$ Variation $(D, \Theta^j, \vec{z}, \alpha, \beta_1, \beta_2)$
        Evaluate all the children
    end for
    Use the last discriminator to re-evaluate the parents
    $PQ \leftarrow \Theta^1, \ldots, \Theta^\mu, \Theta^{1,1}_{child}, \ldots, \Theta^{\mu,N_m}_{child}$
    $\mathcal{PF} \leftarrow$ non_dominated_sorting$(PQ)$
    crowding_alignment_assignment$(\mathcal{PF})$
    $\Theta^1, \ldots, \Theta^\mu \leftarrow$ sorting$(\mathcal{PF})$
end for
```

Algorithm 2 Variation

```
Require: Generator's parameters $\Theta^j$
Require: The number of objective functions $N_m$
Require: Noise Batch $\vec{z}$
Require: Adam hyperparameters $\alpha$, $\beta_1$, $\beta_2$
for $h \leftarrow 1 \ldots N_m$ do
    $g^{j,h}_{\Theta} \leftarrow \nabla_\Theta \mathcal{M}^h(G_{\theta^j}(\vec{z}))$
    $\Theta^{j,h}_{child} \leftarrow Adam(g^{j,h}_{\Theta}, \Theta^j, \alpha, \beta_1, \beta_2)$
end for
```

3.1 Variation

The Variation step is an *asexual reproduction*, where a single parent generates N_m children obtained by applying one step of the Adam algorithm with N_m different objective functions.

A pseudo-code of this step is provided in Algorithm 2, where given a parent Θ^j a child $\Theta^{j,h}_{child}$ is generated for each objective function $\mathcal{M}^h$.

In particular, in this work, we used the same objective functions proposed in [15].

The first objective function is called *minimax* and is derived from the original *GAN* framework

$$\mathcal{M}^{minimax} = \frac{1}{2}\mathbb{E}_{z \sim q(z)}[\log(1 - D(G(Z)))]$$

According to [3], $\mathcal{M}^{minimax}$ minimizes the Jensen-Shannon divergence (JSD) between the data distribution and the generated distribution. The primary issue of this objective function is the generator's vanishing gradient.

The second objective function is called *heuristic*, which unlike *minimax*, maximizes the log probability of the discriminator being mistaken

$$\mathcal{M}^{heuristic} = -\frac{1}{2}\mathbb{E}_{z \sim q(z)}[\log(D(G(Z)))]$$

Compared with *minimax*, *heuristic* does not saturate when the discriminator rejects the generated samples. Thus, this objective function does not suffer from the problem of vanishing gradient for the generators. On the other hand, *heuristic* tends to push the two distributions away from each other. Thus, this may lead to training instability and generative quality fluctuations [5].

The last objective function is *least-square*, used in *LSGAN* algorithm [8]

$$\mathcal{M}^{least-square} = \mathbb{E}_{z \sim q(z)}[\log(D(G(Z)) - 1)^2]$$

This function does not saturate when the discriminator has classified with absolute certainty the generated image as fake (i.e.,

$D(G(Z)) = 0$). This objective function, like *heuristic*, does not suffer from the vanishing gradient problem.

Moreover, unlike *heuristic*, *least-square* tries to avoid that the model collapses by assigning an extremely low cost to generating fake samples.

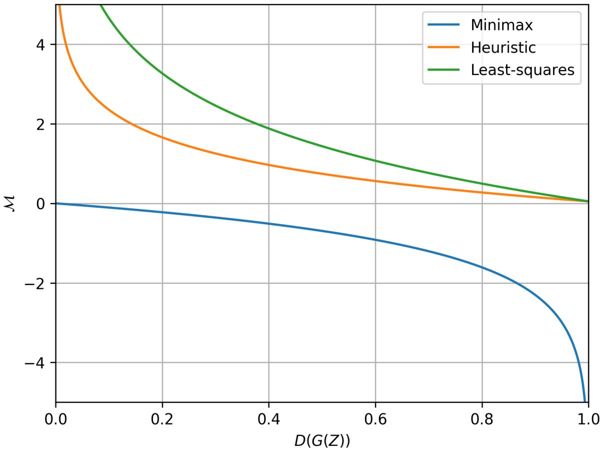

Figure 2: Plot of the variations of the objective functions with respect to the discriminator output.

In order to get a better understanding of the way in which these objective functions work, we illustrate their behaviour in Figure 2 with respect to the discriminator D given a generated distribution $G(Z)$.

3.2 Evaluation

The Evaluation step introduced in [15] uses two fitness functions: the first one computes the *quality* of a generator, and the second one is used to *measure* the diversity. The quality function is defined as :

$$\mathcal{F}_q = \mathbb{E}_z[D(G(Z))]$$

The diversity fitness is defined as :

$$\mathcal{F}_d = -\log ||\nabla_w - \mathbb{E}_x[\log D(X)] - \mathbb{E}_z[\log(1 - D(G(Z)))]||$$

They are described and commented in [15].

Since the discriminator is not the same at each iteration, the diversity fitness is computed as the logarithm of the diversity score weighted by the discriminator's gradient. This can be seen as a weight of the diversity. If the discriminator network applied a countermeasure (hence it is strongly changed), the diversity score of a generator is increased; otherwise, if the discriminator is not changed, the diversity score is reduced. Thus, the collapse issue can be mitigated and the discriminator will smoothly change, which helps to improve the training stability.

Finally, we defined the multi-objective problem of our interest as

$$minimize\ \mathcal{F}_{qd}(\Theta) = \{\mathcal{F}_q(\Theta), \mathcal{F}_d(\Theta)\}$$

where Θ is the generator to be evaluated.

In Algorithm 1, the evaluation step is applied to the children at line 12. Moreover, since the discriminator is changing all the time, at line 14, the parents are re-evaluated. We do that in order to allow a fair comparison between the child and the parent in the selection step.

Algorithm 3 Non Dominated Sorting

```
Require: set of points $P$
for $p \in P$ do
    $S_p \leftarrow \emptyset, n_p \leftarrow 0$
    for $q \in P$ do
        if $p \prec q$ then $S_p \leftarrow S_p \cup \{p\}$
        else if $q \prec p$ then $n_p \leftarrow n_p + 1$
    end for
    if $n_p = 0$ then
        $p_{rank} \leftarrow 1$
        $\mathcal{F}_1 \leftarrow \mathcal{F}_1 \cup \{p\}$
    end if
end for
$i \leftarrow 1$
while $\mathcal{F}_i \neq \emptyset$ do
    for $p \in \mathcal{F}_i$ do
        for $q \in S_p$ do
            $n_p \leftarrow n_p - 1$
            if $n_p = 0$ then
                $q_{rank} \leftarrow q_{rank} + 1$
                $Q \leftarrow Q \cup \{q\}$
            end if
        end for
    end for
    $i \leftarrow i + 1$
    $\mathcal{F}_i \leftarrow Q$
end while
return $(\mathcal{F}_1, \mathcal{F}_2, \ldots)$
```

3.3 Selection

In this section, we describe how *MO-EGAN* selects the generator for each generation. In fact, unlike *EGAN* [15], *MO-EGAN* is based on the *Pareto dominance*, as *NSGA-II* [1].

Therefore, the set of parents and children generators is $PQ = \{\Theta^1, \ldots, \Theta^\mu, \Theta^{1,1}_{child}, \ldots, \Theta^{\mu,N_m}_{child}\}$, whose size is $\mu \cdot (1 + N_m)$.

The *nondominated sorting*, described in Algorithm 3, is applied to PQ. This algorithm divides the set of the points $P = \{\mathcal{F}_{qd}(\Theta) \,|\, \forall \Theta \in PQ\ \}$ into *nondominated fronts* $\mathcal{PF}$ according to *non dominance* relation.

The next step computes the crowding distance for each individual in the same *nondominated* front (see lines 6-12 of the Algorithm 4). It is worth noting that the borders of the front have an infinite crowding distance which means they are always preferred in the selection step.

The last step is the *sorting* procedure, as described in Algorithm 5. In this case, all individuals of all fronts are combined on single set $\bar{P}$, and sorted by the following partial order:

$$i \prec_n j \equiv i_{rank} < j_{rank} \ \vee \ (i_{rank} = j_{rank} \ \wedge \ i_{distance} < j_{distance})$$

where i_{rank} is the value computed in the *nondominated sorting step*, and $i_{distance}$ is the crowding distance.

Given the set $\bar{P}$, the best μ generators are selected for the the next iteration of the algorithm.

Algorithm 4 Crowding Alignment Assignment

```
Require: set of nondominated fronts $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \dots)$
for $\mathcal{I} \in \mathcal{F}$ do
    $l \leftarrow |\mathcal{I}|$
    for $i \leftarrow 1 \dots l$ do
        $\mathcal{I}[i]_{distance} \leftarrow 0$
    end for
    for $m \leftarrow 1 \dots N_m$ do
        sort $\mathcal{I}$ by $m$-th objective function
        $\mathcal{I}[1]_{distance} \leftarrow \infty$
        $\mathcal{I}[l]_{distance} \leftarrow \infty$
        for $i \leftarrow 2 \dots (l-1)$ do
            $\mathcal{I}[i]_{distance} \leftarrow \mathcal{I}[i]_{distance} + \frac{\mathcal{I}[i+1].m - \mathcal{I}[i-1].m}{f_m^{max} - f_m^{min}}$
        end for
    end for
end for
```

Algorithm 5 Sorting

```
Require: set of nondominated fronts $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \dots)$
$\bar{P} \leftarrow \emptyset$
for $i = 1 \dots l$ do $\bar{P} \leftarrow \bar{P} \cup \mathcal{F}_i$
sort $\bar{P}$ by $\prec_n$
return $\bar{P}$
```

4 EXPERIMENTS

In this section, a comparison of MO-EGAN's algorithm performance with respect to GAN and EGAN is presented.

The preliminary experimental analysis was performed on the same synthetics datasets used in [15]. These datasets will help us to get a proof-of-principle about the behavior of MO-EGAN, but we clearly need to assess its performance with more complex datasets in the future.

4.1 Datasets

The synthetic data are two datasets generated from 2D Gaussian mixture distributions.

The first one is a mixture of 8 Gaussians arranged in a circle, and the second one is a mixture of 25 Gaussians arranged in a grid of size 5 × 5.

Those Synthetic datasets are designed to verify the ability of the algorithm to train a generator to reproduce a 2D Gaussian mixture distributions in those cases where the original GAN algorithm does not achieve satisfying results.

4.2 Metric

Because of the nature of the data, the metric measure adopted for our tests is the *Maximum Mean Discrepancy* (MMD) [11] [4]. The MMD metric represents distances between distributions as distances between mean embedding of features.

Formally, let x and y be random variables on a topological space $\mathcal{X}$, and given observations $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$, independently and identically distributed from distribution generators p and q ($p \neq q$), we define the empirical MMD as

$$MMD(\mathcal{F}, p, q) = \sup_{f \in \mathcal{F}} (\mathbb{E}_{x \sim p}[f(X)] - \mathbb{E}_{y \sim q}[f(Y)])$$

where the $\mathcal{F}$ be a class of functions $f : X \to \mathbb{R}$.

Replacing the expectations with the averages computed on the X and Y samples, we obtain the biased empirical estimation of MMD

$$MMD(\mathcal{F}, X, Y) = \sup_{f \in \mathcal{F}} (\frac{1}{n} \sum_{i=1}^{n} f(x_i) - \frac{1}{m} \sum_{i=1}^{m} f(y_i))$$

This metric is not commonly used on datasets of real images since identifying the function f is a hard task. However, according to [4], MMD is easy to estimate if a Gaussian kernel is used and the σ and bias parameters are known.

Thus, since the datasets have been generated with mixtures of Gaussian distributions, this metric can be easily computed giving us an accurate estimation of the difference between a target distribution and the generated one.

4.3 Implementation details

Layer	Size	Activation function
Generator		
Input	2	-
Fully connected	512	ReLU
Fully connected	512	ReLU
Fully connected	512	ReLU
Output	2	-
Discriminator		
Input	2	-
Fully connected	512	ReLU
Fully connected	512	ReLU
Fully connected	512	ReLU
Output	1	Sigmoid

Table 1: Networks layouts

In order to have a fair comparison, we used the same network used in [15]. Since the original document does not describe the network structure used for Synthetic Data, we decided to use the same networks implemented in *EGAN*'s source code available at GitHub [3]. For clarity, the layouts are reported in Table 1, i.e., both networks have more than half a million parameters to optimize. It is also worth noting that we have released *MO-EGAN*'s source code on GitHub [4].

Following this line, we used the same parameters of the *EGAN* implementation: the batch size is $m = 64$, the number of discriminator's updating steps is $N_d = 1$. Moreover the Adam parameters are $\alpha = 0.0001, \beta_1 = 0.0, \beta_2 = 0.999$.

All experiments of this work were performed using a personal computer with a GPU Nvidia Tesla V40d, CPU Intel i7-4770 and 8 GB of RAM.

---

[3] https://github.com/WANG-Chaoyue/EvolutionaryGAN

[4] https://github.com/Gabriele91/MO-EGAN

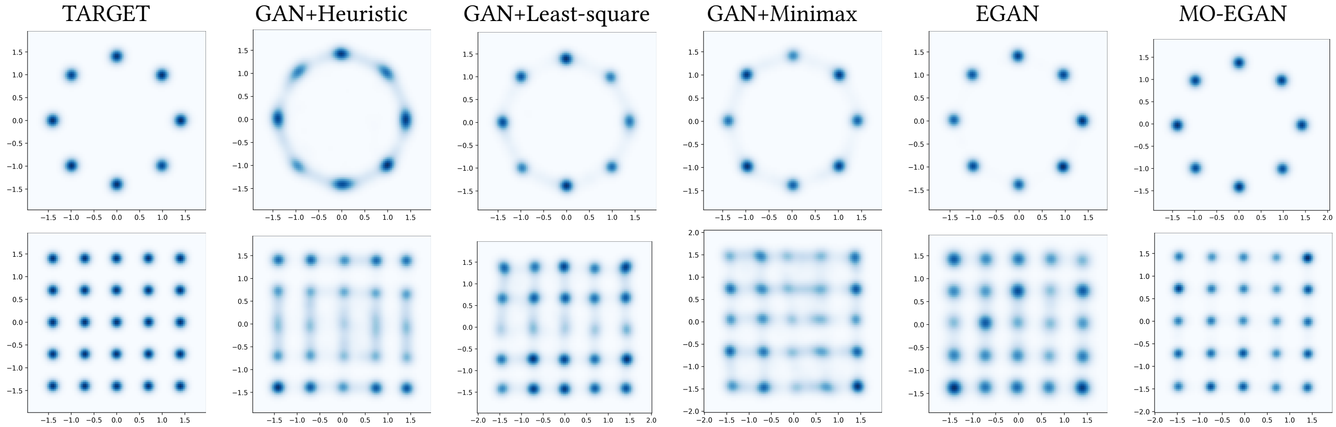

Figure 3: Kernel density estimation plots of generated data compared to target data. The first row reported the mixture of 8 Gaussians, The second one are the plots of 25 Gaussians.

Algorithm	8 Gaussians		25 Gaussians	
	Average	Best	Average	Best
GAN-Heuristic	45.27	33.2	2.80	2.19
GAN-Least-square	3.99	3.16	1.83	1.72
GAN-Minimax	2.94	1.89	1.65	1.55
EGAN (without diversity function)	11.54	7.31	1.69	1.60
EGAN (with diversity function)	2.36	1.17	1.20	1.04
MO-EGAN	**0.912**	**0.677**	**1.08**	**0.948**

Table 2: MMD ($\times 10^2$) on the Synthetic datasets. Lower values are better because they indicate that the generated distribution is closer to the target one. MO-EGAN finds the best solutions in both the problems.

5 ANALYSIS

Our analysis was conducted on the synthetic data. This allows us to understand the collapse issue since we can observe and measure the distribution of the generated data quite easily.

We performed three types of analysis. The first one is qualitative: a comparison between the distributions generated by the generators trained with different algorithms with respect to the target distributions through a Kernel density estimation plot. The second one is a quantitive analysis that comes out of the computation of the *MMD* metric on all the generators trained. The last one is an analysis of the Pareto fronts produced during the training phase with *MO-EGAN*.

Moreover, we reported a brief study on the computational complexity of the new selection step proposed in this work.

5.1 Qualitative analysis

Regarding the qualitative analysis, Figure 3 shows the Kernel Density Estimations (KDE) of the target data with respect to data generated by the best generator trained by *GAN* (with *Heuristic*, *Least-Square* and *Minimax*) , *EGAN*, and *MO-EGAN*.

The *GAN+Heuristic* algorithm does not achieve good results on the tested data: the plots of the 8 Gaussians results are blurry and unclear; this is also true for the middle lines on the 25 Gaussians grid. *GAN+Least-Square* achieves better results on the first dataset while the middle lines on the 25 Gaussians grid still remains blurry. Also *GAN+Minimax* achieves a good result on the first problem, while the second one remains a hard task; in this case, the most blurred part is in the middle columns instead of the middle rows.

Regarding *EGAN*, we can note that the generators produce Gaussian mixture distributions extremely close to the target distributions. This is also true for the generator trained by *MO-EGAN*. Nevertheless, *MO-EGAN*'s generators generate the cleanest distributions, in that a few samples are wrong, in both mixtures. Thus, *MO-EGAN*'s generators produce the closest distributions to the target ones.

5.2 Quantitative analysis

For the quantitative analysis, we report in Table 2 the MMD values computed on each distribution generated in both problems. More precisely, in the right column is written the algorithm used to train the generators, and in the second and third one, we report the MMD's average across 10 generators and the MMD of the best one for the mixture with 8 Gaussians, respectively. In the same way, the fourth and fifth columns are respectively the MMD's average between 10 generators and the best MMD for the mixture with 25 Gaussians.

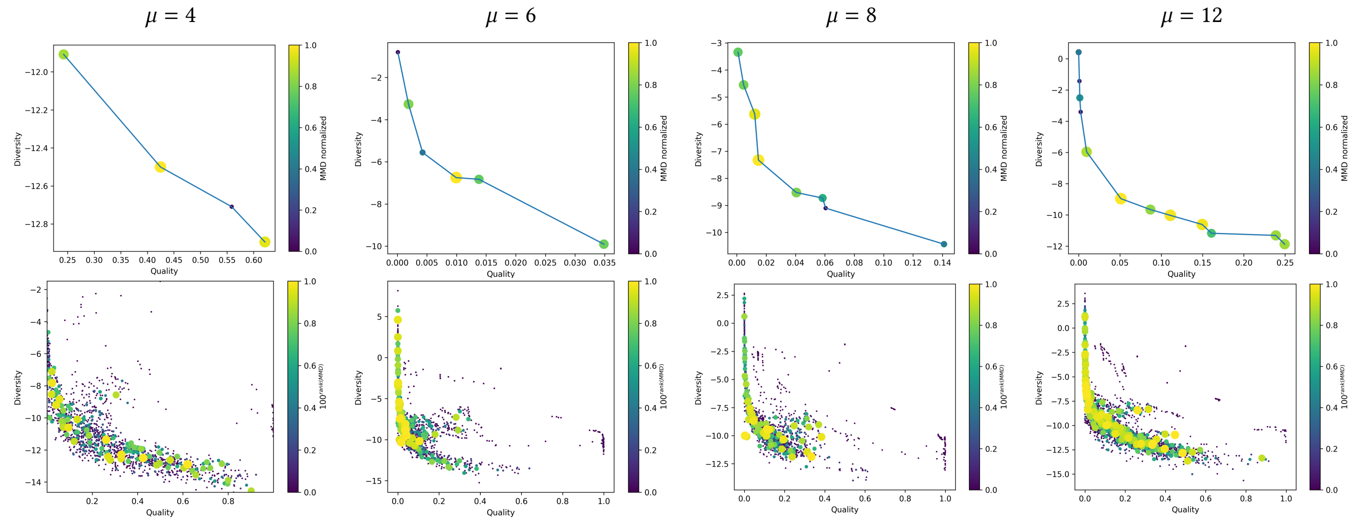

Figure 4: Plots of the Pareto fronts across different population sizes. The first row, we reported the Pareto fronts where we were found the best solution, The second one, we show all points computed during the evolution.

Regarding the first problem, our algorithm performs significantly better, in fact in terms of average *MO-EGAN* has halved the MMD value, and in terms of the best generator computed it improves performance by +75%. Also, the performance on the second dataset was improved by +11% for the average and +9% for the best solution.

5.3 Pareto front analysis

Finally, we analyze the Pareto fronts. The objective of this experimental analysis is to figure out if the selection step based on *Pareto dominance* is a viable option for this problem.

In fact, if the best solutions are in the middle of the front, this motivates the application of this kind of selection. While, if the best solutions are located on one of the two extremes, one of the two functions is privileged over the other. In other words, in the latter case, we do not need to optimize two objective functions and then we could re-define this problem just as a single-objective optimization problem, considering only the privileged objective.

To this aim, our analysis has shown the Pareto fronts and all the points found during the evolutionary process in Figure 4, for the following population sizes: $\mu \in \{4, 6, 8, 12\}$. In the first row, we show the Pareto fronts for the iterations where the best solutions ware found. We normalized the MMD between the worst and best solution. Thus, the big yellow circle is the solution with smallest MMD value, and vice-versa, the small purple circle corresponds to the solution with the highest MMD value. In the second row, with the aims of providing a better visualization, we drew the normalized rank of the MMD values raised to 100. Thus, like the first plots, the big yellow circles are the solutions with smaller MMD values and the small purple circles correspond to the solutions with the highest MMD values.

From these plots, we can see how the best solutions are found in the middle of the Pareto front independently of the population size. This is also visible in the second line, where all the better solutions were found in the curve's inflection point.

5.4 Computational complexity

In this section a computational complexity comparison between *MO-EGAN* and *EGAN* is presented.

EGAN and *MO-EGAN* have the same computational complexity except for the selection step. *EGAN* sorts the generators with respect to the fitness values, hence the cost of this step is $O(N \log N)$, where N is the number of the offsprings produced during the variation step. Instead, *MO-EGAN* selection step uses *Non Dominated Sorting*, *Crowding Alignment Assignment* and *Sorting* procedures.

The complexities of these procedures are, respectively, $O(N^2)$, $O(N \log N)$ and $O(N \log N)$, since *MO-EGAN* has only two objective functions [5].

We also found the empirical contribution to the computational time of the diversity function and the MO-EGAN selection step. The result of this analysis is shown in Table 3. This comparison shows the computation times of *EGAN with diversity function* and *MO-EGAN* for each population size μ with respect to the faster executions on 8 Gaussians and 25 Gaussians, indicated with t_1 and t_2, respectively.

For both datasets, we recorded that the computational cost grows around 70% when the diversity function is used, while the contribution of using of the multi-objective selection is smaller, since the cost is increased by another 15.4%.

6 CONCLUSION AND FUTURE WORK

In this work we redefined the single-objective generator optimization problem of *GAN* as a multi-objective one. We did this by redesigning *EGAN* as a multi-objective algorithm. More precisely, we replaced the selection step with the *non_dominated_sorting* and

---

[5] The cost of the *Non Dominated Sorting* and *Crowding Alignment Assignment* in the general case are $O(MN^2)$ and $O(MN \log N)$, respectively, where M is the number of objective functions

Algorithm	8 Gaussians				25 Gaussians			
	$\mu = 4$	$\mu = 6$	$\mu = 8$	$\mu = 12$	$\mu = 4$	$\mu = 6$	$\mu = 8$	$\mu = 12$
EGAN (with diversity function)	$1.00 \times t_1$	$1.42 \times t_1$	$1.84 \times t_1$	$2.76 \times t_1$	$1.00 \times t_2$	$1.51 \times t_2$	$1.94 \times t_2$	$3.00 \times t_2$
MO-EGAN	$1.10 \times t_1$	$1.61 \times t_1$	$2.16 \times t_1$	$3.32 \times t_1$	$1.17 \times t_2$	$1.73 \times t_2$	$2.60 \times t_2$	$3.42 \times t_2$

Table 3: Times comparison between *EGAN* and *MO-EGAN*, where t_1 and t_2 are the faster computational times for the 8 Gaussians and 25 Gaussians, respectively.

crowding_alignment_assignment procedures, that manage the quality and diversity fitness functions in the same way as the *NSGA-II* does for problems with two objective functions.

The preliminary results of *MO-EGAN* show that the application of the Pareto dominance for the selection phase gets better results than the original *EGAN* algorithm on the synthetic datasets adopted.

As part of our future work, we will focus on three points: experiments on real-world distributions, improving the stability of *MO-EGAN* and reducing the time complexity of our approach.

For the first point, we are going to apply this method on databases such as FashionMNIST [18], CIFAR10 [6], and CelebA [7] in order to see if the same behavior observed on the synthetic datasets persists on these other types of distributions.

About the second point, because of the ability of *MO-EGAN* to manage more than one objective function, the objective functions $\mathcal{F}_{qd}$ could be re-defined using more than one discriminator for measuring the quality and also for the diversity evaluation of the generators.

Regarding the way of reducing the time complexity, we are going to re-define the variation step. A possible idea is to use a strategy such as the roulette wheel proposed in [12], or the upper confidence bound algorithm, version1 (UCB1) for a smart selection of the objective function to be applied.

ACKNOWLEDGMENTS

This work is partially supported by the DMI-Unipg Ricerca di Base 2017 project "Evolutionary Neural Network Optimization".

Carlos A. Coello Coello gratefully acknowledges support from CONACyt grant no. 2016-01-1920 (Investigación en Fronteras de la Ciencia 2016) and from a project from the 2018 SEP-Cinvestav Fund (application no. 4).

REFERENCES

[1] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation 6, 2 (2002), 182–197.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Advances in neural information processing systems. 2672–2680.

[4] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. Journal of Machine Learning Research 13, Mar (2012), 723–773.

[5] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved Training of Wasserstein GANs. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 5769–5779.

[6] Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Technical Report.

[7] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In Proceedings of International Conference on Computer Vision (ICCV).

[8] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least squares generative adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision. 2794–2802.

[9] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng. 2019. Recent Progress on Generative Adversarial Networks (GANs): A Survey. IEEE Access 7 (2019), 36322–36333.

[10] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14). JMLR.org, II–1278–II–1286.

[11] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. 2007. A Hilbert space embedding for distributions. In International Conference on Algorithmic Learning Theory. Springer, 13–31.

[12] Jamal Toutouh, Erik Hemberg, and Una-May O'Reilly. 2019. Spatial evolutionary generative adversarial networks. In Proceedings of the Genetic and Evolutionary Computation Conference. 472–480.

[13] Aäron van den Oord and Nal Kalchbrenner. 2016. Pixel RNN. In ICML.

[14] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alexander Graves. 2016. Conditional Image Generation with PixelCNN Decoders. In Advances in Neural Information Processing Systems 29. 4790–4798.

[15] Chaoyue Wang, Chang Xu, Xin Yao, and Dacheng Tao. 2019. Evolutionary generative adversarial networks. IEEE Transactions on Evolutionary Computation 23, 6 (2019), 921–934.

[16] Zhengwei Wang, Qi She, and Tomas E Ward. 2019. Generative adversarial networks: A survey and taxonomy. arXiv preprint arXiv:1906.01529 (2019).

[17] Xian Wu, Kun Xu, and Peter Hall. 2017. A survey of image synthesis and editing with generative adversarial networks. Tsinghua Science and Technology 22, 6 (2017), 660–674.

[18] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:cs.LG/cs.LG/1708.07747

[19] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. 2015. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. arXiv preprint arXiv:1506.03365 (2015).