Generate more than one child in your co-evolutionary semi-supervised learning GAN

Francisco Sedeño[0009−0000−7797−3562], Jamal Toutouh[0000−0003−1152−0346], and Francisco Chicano[0000−0003−1259−2990]

ITIS Software, University of Malaga, Spain
{fsedenoguerrero,jamal,chicano}@uma.es

Abstract. Generative Adversarial Networks (GANs) are very useful methods to address semi-supervised learning (SSL) datasets, thanks to their ability to generate samples similar to real data. This approach, called SSL-GAN has attracted many researchers in the last decade. Evolutionary algorithms have been used to guide the evolution and training of SSL-GANs with great success. In particular, several co-evolutionary approaches have been applied where the two networks of a GAN (the generator and the discriminator) are evolved in separate populations. The co-evolutionary approaches published to date assume some spatial structure of the populations, based on the ideas of cellular evolutionary algorithms. They also create one single individual per generation and follow a generational replacement strategy in the evolution. In this paper, we re-consider those algorithmic design decisions and propose a new co-evolutionary approach, called *Co-evolutionary Elitist SSL-GAN* (CE-SSLGAN), with panmictic population, elitist replacement, and more than one individual in the offspring. We evaluate the performance of our proposed method using three standard benchmark datasets. The results show that creating more than one offspring per population and using elitism improves the results in comparison with a classical SSL-GAN.

Keywords: Generative adversarial network · Semi-supervised learning · SSL-GAN · Evolutionary machine learning · Co-evolution.

1 Introduction

A *generative adversarial network* (GAN) [4] is composed of two artificial neural networks: a *generator network* and a *discriminator network*. The generator creates data samples trying to follow the probability distribution of the real data samples for a particular application, while the discriminator tries to distinguish between real or generated (fake) data samples. The loss functions used to train the networks are defined in such a way that: 1) the generator is rewarded when it generates data samples the discriminator classifies as real and 2) the discriminator is rewarded when it correctly distinguishes between real and fake data samples. GANs are able to produce a discriminator for a particular application using less samples than other neural networks due to the generative nature of

the network. The generator is able to produce realistic data samples and, for this reason, GANs have been used in medical image generation [14], 3D object generation [1], and image-to-image translation [7] among others.

GAN training is not an easy task. Vanishing gradient, mode collapse or discriminator collapse are some of the problems that can be found during the training. In order to mitigate these problems, researchers have successfully proposed the use of evolutionary algorithms (EA) with a co-evolution approach [11]. One prominent example is Lipizzaner [9], which combines co-evolution with a spatial distribution of the individuals in the population borrowed from cellular EAs [3].

The ability of GANs to learn from less labeled data makes them very appropriate for semi-supervised learning (SSL), where the datasets contain a mix of labeled and unlabeled data samples. The use of adversarial networks in SSL, sometimes denoted with the acronym SSL-GAN, is very old [10] and there are many different proposals in the scientific literature. The interested reader is referred to the recent surveys by Sajun and Zualkernan [8] and Ma et al. [6]. The application of co-evolutionary approaches to SSL-GANs is recent, with only two works to date [11, 12], up to authors knowledge. The previous work on co-evolutionary SSL-GANs propose the use of a particular spatial distribution of the individuals, one offspring per generation, and a generational replacement strategy for the population.

In this work, we present a novel co-evolutionary approach for SSL-GANs and evaluate different parameters of the approach. The novel features of the proposal are the use of panmictic population, elitist replacement, and more than one individual in the offspring. The final goal of this study is to create a guide that could be useful in future studies to set the parameters of a co-evolutionary SSL-GAN.

The remainder of the paper is organized as follows. Section 2 presents the definitions and background information required to understand the rest of the paper. Our novel SSL-GAN approach is presented in Section 3. We propose our research questions and describe the experimental methodology in Section 4. Finally, Section 5 presents the results obtained in the experiments and the paper concludes in Section 6.

2 Background

Let us assume that we have a partially labeled real samples dataset containing samples classified in K classes. We assume that the samples are d-dimensional real vectors $x \in \mathbb{R}^d$. For each labeled real sample we have its class represented using one-hot encoding $y \in \{0,1\}^K$. We will denote the labeled real samples with $p_{real,la} \subseteq \mathbb{R}^d \times \{0,1\}^K$, and the unlabeled real samples with $p_{real,un} \subseteq \mathbb{R}^d$.

An SSL-GAN (Fig. 1) is composed of two artificial neural networks (ANN): a generator, G_u, which generates a data sample from a random input vector; and a discriminator, D_v, which classifies each sample as real or fake and provides one of the K classes for real data. The u in G_u and the v in D_v denote the set of parameters of the generator and discriminator, respectively. Formally, G_u is

a function $G_u : \mathbb{R}^\ell \to \mathbb{R}^d$ that maps a vector in $\mathbb{R}^\ell$, called the latent space, to a generated sample in $\mathbb{R}^d$. The discriminator D_v is a function $D_v : \mathbb{R}^d \to [0,1]^{K+1}$ that maps a sample to a $K+1$-dimensional vector of real values in $[0,1]$. The first K values of that vector are probabilities for the K classes in the real data and we will denote them as $D_v^{class}(x) \in [0,1]^K$. The $(K+1)$-th value in the vector is the probability for the sample to be real and we will denote it as $D_v^{real}(x) \in [0,1]$, where $D_v^{real}(x) = 1$ means that sample x is classified as real with certainty and $D_v^{real}(x) = 0$ means it is classified as fake with certainty.

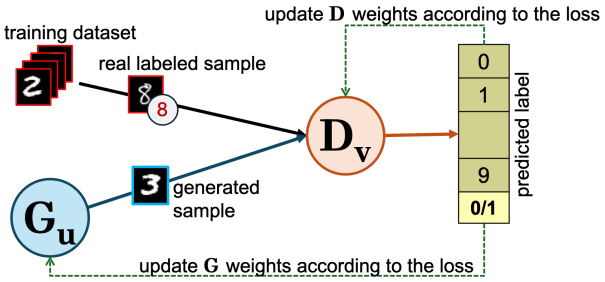


Fig. 1: Example of SSL-GAN for MNIST digits generation

The goal of the generator is to generate samples similar to the real samples. Thus, its loss function should take lower values when $D_v^{real}(G_u(z))$ is higher. The loss function for the generator can be written as:

$$\mathcal{L}_G = \mathbb{E}_{z\sim\mathcal{N}^\ell(\mathbf{0},\mathbf{I})}[\phi(D_v^{real}(G_u(z)))], \tag{1}$$

where ϕ is a monotone decreasing function and $\mathcal{N}^\ell(\mathbf{0},\mathbf{I})$ is a ℓ-dimensional multivariate normal distribution with zero mean and identity covariant matrix (all components of the vector are independent and follow a standard normal distribution). In our case, we use cross-entropy for the loss function, and $\phi(y) = -\log y$.

The discriminator has two goals: 1) classifying each sample as real or fake and 2) for real data, classifying the sample in one of the K classes. The loss function of the discriminator is

$$\mathcal{L}_D = \mathcal{L}_{D,u} + \mathcal{L}_{D,s}, \tag{2}$$

where $\mathcal{L}_{D,u}$ is the *unsupervised loss*, based on the fake samples and unlabeled real data, and $\mathcal{L}_{D,s}$ is the *supervised loss*, based on the real labeled data. These losses are

$$\mathcal{L}_{D,u} = \mathbb{E}_{z\sim\mathcal{N}^\ell(\mathbf{0},\mathbf{I})}\left[\phi(1 - D_v^{real}(G_u(z)))\right] + \mathbb{E}_{x\sim p_{real,un}}\left[\phi(D_v^{real}(x))\right], \tag{3}$$

$$\mathcal{L}_{D,s} = \mathbb{E}_{(x,y)\sim p_{real,la}}\left[\sum_{i=1}^{K} y_i \phi(D_{v,i}^{class}(x))\right], \tag{4}$$

where we used $D_{v,i}^{class}(x)$ to denote the i-th component of the output vector $D_v^{class}(x)$ in the discriminator.

2.1 SSL-GAN training

SSL-GAN extends the traditional unsupervised GAN by introducing an additional objective for the discriminator: classifying the real labeled images. Algorithm 1 summarizes the SSL-GAN training process. The training proceeds over T epochs, iterating through batches of labeled, B_L, and unlabeled, B_U, real samples to compute $\mathcal{L}_{D,s}$, $\mathcal{L}_{D,u}$, $\mathcal{L}_D$, and $\mathcal{L}_G$. The two loss functions, $\mathcal{L}_D$ and $\mathcal{L}_G$, are used to update the network weights of the discriminator (Lines 5 to 10) and the generator (Lines 11 to 14), respectively, by stochastic gradient descent.

Algorithm 1 SSL-GAN training
Input: T: Epochs, $p_{real,la}$: Dataset with labeled data, $p_{real,un}$: Dataset with unlabeled data, B_s: Batch size, θ_g, θ_d: Initial generator and discriminator parameters
Return: g, d: Trained generator and discriminator

1: $g \leftarrow$ initializeNetworks(θ_g) ▷ Initialize the generator
2: $d \leftarrow$ initializeNetworks(θ_d) ▷ Initialize the discriminator
3: **for** $t = 1$ **to** T **do** ▷ Loop over the T training epochs
4: **for** $B_L \subseteq p_{real,la}, B_U \subseteq p_{real,un}$ **do** ▷ Loop over the batches in the dataset
5: $\mathbf{z} \leftarrow$ getRandomNoise(B_s) ▷ Get samples from latent space, $\mathbf{z} \sim \mathcal{N}^{\ell}(\mathbf{0}, \mathbf{I})$
6: $X_f \leftarrow g(\mathbf{z})$ ▷ Generate fake samples
7: $\mathcal{L}_{D,s} \leftarrow$ getSupervisedLoss(d, B_L) ▷ Compute supervised loss
8: $\mathcal{L}_{D,u} \leftarrow$ getUnsupervisedLoss(d, g, B_U, X_f) ▷ Compute unsupervised loss
9: $\mathcal{L}_D \leftarrow \mathcal{L}_{D,s} + \mathcal{L}_{D,u}$ ▷ Compute total discriminator loss
10: $d \leftarrow$ gradientDescent($d, \mathcal{L}_D, \theta_d$) ▷ Update discriminator parameters
11: $\mathbf{z} \leftarrow$ getRandomNoise(B_s) ▷ Get samples from latent space, $\mathbf{z} \sim \mathcal{N}^{\ell}(\mathbf{0}, \mathbf{I})$
12: $X_f \leftarrow g(\mathbf{z})$ ▷ Generate fake samples
13: $\mathcal{L}_G \leftarrow$ generatorLoss(d, g, X_f) ▷ Compute generator loss
14: $g \leftarrow$ gradientDescent($g, \mathcal{L}_G, \theta_g$) ▷ Update generator parameters
15: **end for**
16: **end for**
17: **return** g, d ▷ Return trained g and d

3 Co-evolutionary Elitist SSL-GAN

This section presents CE-SSLGAN, a competitive co-evolutionary method to train SSL-GANs with limited resources, using small populations. CE-SSLGAN combines a $(\mu + \lambda)$ competitive co-evolutionary algorithm with the SSL-GAN training approach (see Section 2). CE-SSLGAN operates over two populations of ANNs of the same size μ: a population of generators $\mathbf{g}$, $\{g_1, \ldots, g_\mu\}$, and a population of discriminators $\mathbf{d}$, $\{d_1, \ldots, d_\mu\}$. In each generation, a subset of λ

individuals (ANNs) of **g** and **d** are selected to evolve. The variation operator of this co-evolutionary algorithm is the mutation of the parameters (weights) of the ANNs, which is performed by applying stochastic gradient descent during a training process. This training is applied to pairs of individuals g, d of both populations by applying a number n_t of training epochs of Algorithm 1. Thus, given a computational budget T_B of maximum training epochs, the number of the generations, ι, of the main co-evolutionary loop is given by the expression

$$\iota = \left\lfloor \frac{T_B}{n_t \lambda} \right\rfloor . \tag{5}$$

Algorithm 2 describes the main steps of CE-SSLGAN. The algorithm begins by initializing each individual, i.e., ANN, of the generator and discriminator populations using parameters θ_g and θ_d, respectively (Lines 1 and 2). The number of generations ι is computed. Then, all the individuals of both populations are evaluated. The fitness values $\mathcal{L}_{g,d}$ are computed by aggregating the loss of each individual against all the individuals of the adversary population (Line 4). A number n_e of batches is used to calculate the loss. The selected value for n_e can affect the individual evaluation, because a small n_e can produce imprecise loss estimations, whereas a large n_e could incur significant computational costs.

The main CE-SSLGAN loop runs over ι generations. In each generation, tournament selection with size τ is applied to create an offspring of generators ($\mathbf{g}^*$) and discriminators ($\mathbf{d}^*$) with size λ (Lines 6 and 7). The individuals of the offspring are coupled in λ SSL-GAN to perform the training for n_t epochs (Line 12). After training, $\mathbf{g}^*$ and $\mathbf{d}^*$ are added to their populations (Lines 14 and 15, respectively). The updated populations are then evaluated to update the fitness values, $\mathcal{L}_{g,d}$ (Line 16). The best μ individuals in each population is selected to be the population for the next generation (Lines 17 and 18). Once all generations are completed, CE-SSLGAN returns the best generator and discriminator according to $\mathcal{L}_{g,d}$ (Lines 20 to 22).

4 Experimental setup

We perform an empirical analysis of CE-SSLGAN by examining both the discriminator classification accuracy and the quality of the samples produced by the generator[1]. Our goal is to answer the following research questions:

- **RQ1**: How do the number of training epochs during mutation, n_t, population size, μ, and offspring size, λ, influence the performance of CE-SSLGAN?
- **RQ2**: How does CE-SSLGAN perform compared to the basic SSL-GAN?
- **RQ3**: How sensitive is the performance of CE-SSLGAN and the basic SSL-GAN to the proportion of labeled data?

[1] The replication package of the paper is at `https://doi.org/10.5281/zenodo.14729793`

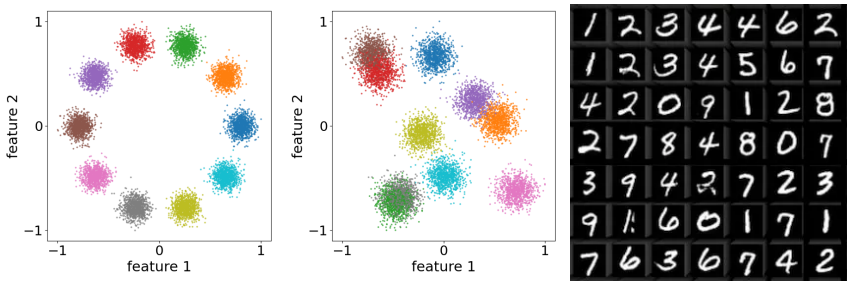
Algorithm 2 CE-SSLGAN training
Input: T_B: Computational budget (training epochs), $p_{real,la}$: Dataset with labeled data, $p_{real,un}$: Dataset with unlabeled data, B_s: Batch size, θ_g, θ_d: Initial generator and discriminator parameters, μ: Population size, λ: Offspring size, τ: Tournament size, n_e: Number of evaluation batches, n_t: Number of training epochs per generator-discriminator couple
Return: g, d: Trained generator and discriminator

$\mathbf{g} \leftarrow$ initializePopulation(θ_g) ▷ Initialize population $\mathbf{g}$
$\mathbf{d} \leftarrow$ initializePopulation(θ_d) ▷ Initialize population $\mathbf{d}$
$\iota \leftarrow \left\lfloor \frac{T_B}{n_t \lambda} \right\rfloor$ ▷ Compute the number of generations to perform
$\mathcal{L}_{g,d} \leftarrow$ evaluate($\mathbf{g}, \mathbf{d}, n_e$) ▷ Evaluate populations
for $i = 1$ **to** ι **do** ▷ Loop over generations
  $\mathbf{g}^* \leftarrow$ select($\lambda, \tau, \mathcal{L}_{g,d}$) ▷ Tournament selection to get offspring population $\mathbf{g}^*$
  $\mathbf{d}^* \leftarrow$ select($\lambda, \tau, \mathcal{L}_{g,d}$) ▷ Tournament selection to get offspring population $\mathbf{d}^*$
  $\mathbf{d}' \leftarrow \mathbf{d}^*$ ▷ Initialize the set of discriminators to couple with generators
  for $g \in \mathbf{g}^*$ **do** ▷ Loop over g in the offspring $\mathbf{g}^*$
    $d \leftarrow$ pickOneRandomly($\mathbf{d}'$) ▷ Select a couple discriminator for g
    $\mathbf{d}' \leftarrow \mathbf{d}' - \{d\}$ ▷ Remove the selected discriminator from the set
    $g, d \leftarrow$ train($g, d, n_t, p_{real,la}, p_{real,un}, B_s, \theta_g, \theta_d$) ▷ Update g, d parameters
  end for
  $\mathbf{g} \leftarrow \mathbf{g} \cup \mathbf{g}^*$ ▷ Add offspring to population
  $\mathbf{d} \leftarrow \mathbf{d} \cup \mathbf{d}^*$ ▷ Add offspring to population
  $\mathcal{L}_{g,d} \leftarrow$ evaluate($\mathbf{g}, \mathbf{d}, n_e$) ▷ Evaluate populations
  $\mathbf{g} \leftarrow$ updatePopulation($\mathbf{g}$) ▷ Apply elitism (best μ individuals survive)
  $\mathbf{d} \leftarrow$ updatePopulation($\mathbf{d}$) ▷ Apply elitism (best μ individuals survive)
end for
$g \leftarrow$ selectBestGenerator($\mathbf{g}$)
$d \leftarrow$ selectBestDiscriminator($\mathbf{d}$)
return g, d ▷ Return g, d trained SSL-GAN

The experiments are performed on three datasets: mixture of ten 2D Gaussian distributions arranged in a ring (referred to as `RING`), a mixture of eight 2D Gaussian distributions randomly arranged (referred to as `BLOB`), and the Modified National Institute of Standards and Technology (`MNIST`) dataset [2]. Fig. 2 illustrates the distributions of `RING` and `BLOB` and some samples of `MNIST`.

`RING` and `BLOB` datasets consist of 2D points in the range [-1, 1] grouped in ten classes (Gaussian distributions), with 10,000 samples in the training set and 1,000 in the test set. These types of datasets are widely used in GANs research because, even if they may seem simple, they present a challenge in the generative component and can lead to mode collapse. The `RING` dataset mixtures are well separated in the feature space, which makes it easier to classify the samples (see Fig. 2a). In contrast, the mixtures in the `BLOB` dataset are randomly distributed, with several mixtures sharing feature space, which means that a classifier cannot learn to classify all the points in the test set correctly. In our analysis, the BLOB dataset used in the whole experimentation is the same (see Fig. 2b). A classifier was trained using all available labels in the training dataset, achieving a maximum classification accuracy of 0.889.

The MNIST dataset contains gray-scale images of handwritten digits (0 to 9) with a resolution of 28×28 pixels. The training set includes 60,000 images, while the test set contains 10,000 images.



(a) RING dataset (b) BLOB dataset (c) MNIST dataset

Fig. 2: Representation of the datasets applied in our empirical analysis

The classification accuracy is calculated by applying the classifier defined by the discriminator on the test dataset. The generative models in RING and BLOB experiments are evaluated according to the Wasserstein distance (W_D) [13], while in MNIST are assessed regarding Fréchet Inception Distance (FID) [5].

W_D measures the minimal cost required to transform one probability distribution into another. Given two probability distributions P and Q on a metric space $\mathcal{X}$, $W_D(P,Q)$ is defined in Eq. (6), where $\Pi(P,Q)$ denotes the set of all joint distributions $\gamma(x,y)$ with marginals P and Q, and $d(x,y)$ is the distance between points x and y in $\mathcal{X}$.

$$W_D(P,Q) = \inf_{\gamma \in \Pi(P,Q)} \int_{\mathcal{X}\times\mathcal{X}} d(x,y)\, d\gamma(x,y) \tag{6}$$

FID computes the similarity between two sets of images. Specifically, given two distributions of features extracted from ANNs, P (e.g., real images) and Q (e.g., generated images), FID is computed using Eq. (7), where μ_P and Σ_P denote the mean and covariance of the features from P, and μ_Q and Σ_Q denote those of Q.

$$\text{FID}(P,Q) = ||\mu_P - \mu_Q||^2 + \text{Tr}(\Sigma_P + \Sigma_Q - 2(\Sigma_P\Sigma_Q)^{1/2}) \tag{7}$$

The experiments are performed on two different architectures for the SSL-GANs: one based on multilayer perceptrons (MLP) to learn RING and BLOB datasets, and one based on convolutional neural networks (CNN) to address MNIST dataset.

For RING and BLOB datasets, the generator consists of a three-layer MLP. It maps a latent vector to a 2D point (data sample) through a hidden layer with 64 units using ReLU activations. The final layer applies tanh activation

to constrain the output range to $[-1, 1]$, fitting the RING and BLOB distribution. The discriminator is a two-layer MLP that maps a 2D input to a 64-dimensional feature space using LeakyReLU activations. It has two outputs: a binary classifier with a sigmoid activation to determine real or fake samples and a multiclass classification layer that outputs class predictions.

For MNIST, the generator's input layer is fully connected and converts a latent vector into an initial 7×7 feature map. This feature map is then upsampled through two layers of transposed convolutions, using ReLU activations and batch normalization, finally producing a 28×28 output with a final activation function of tanh. The discriminator architecture includes four convolutional blocks with LeakyReLU activations, dropout, and batch normalization, reducing the input image dimensions from 28×28 (input image) to 4×4 (features map). Then, it has two output layers: a binary classifier (real/fake), which applies a sigmoid activation function, and a layer for multiclass classification of the MNIST digits.

The experimental design focuses on assessing the impact of three key factors in a co-evolutionary algorithm (**RQ1**): population size, offspring size, and exploitation of individuals. We explore population sizes $\mu \in \{3, 5, 7, 9\}$, and adjust the offspring size λ from 1 up to the ceiling of half the population size, e.g., $\lambda \in \{1, 2, 3\}$ for μ=5. We examine the effect of the different levels of exploitation of solutions by varying the number of training epochs applied to update the network weights each generation (n_t). We consider $n_t \in \{1, 5, 10\}$. This parameter is essential because the number of generations (ι) that CE-SSLGAN performs depends on the total training epochs to be conducted (T_B), λ, and n_t (see Eq. 5). Thus, when n_t=1 CE-SSLGAN completes 10 times more generations than when it is set n_t=10. To evaluate the effectiveness of the proposed method, we compare it to a standard SSL-GAN training approach (**RQ2**), offering insights into the performance of the co-evolutionary algorithm across different parameter settings.

The main hyper-parameters used are set according to preliminary experiments. Generators and discriminators apply the Adam optimizer with the same learning rate 0.0003. The batch size is 100 samples for RING and BLOB and 600 for MNIST. The total training epochs T_B is set to 300λ. We decided to set the budget proportional to the number of offspring to simulate the parallel training of the offspring. Thus, the standard SSL-GAN runs for 300 training epochs (i.e., T=T_B because it is considered as λ=1).

We denote with n_s the number of labeled samples per class. For the RING and BLOB datasets, the experiments are performed with one labeled sample per class, $n_s = 1$ (i.e., 10 labeled samples which is a portion of 0.001 labeled data in the training dataset). For the experiments on the MNIST dataset, the primary analysis is performed using $n_s = 10$ labeled samples per class (i.e., 100 labeled samples which is 0.00167 labeled data in the training dataset). A sensitivity analysis is performed on the number of labeled data for MNIST (**RQ3**), the most challenging dataset, to see the impact on the results. Thus, additional experiments are performed with $n_s = 60$ and $n_s = 100$ labeled samples per class.

5 Experimental results

In this section, we present the results of the experiments, split by dataset. In each dataset, we first analyze the influence of the number of training epochs during offspring generation, n_t, and then the population and the size of the offspring, μ and λ, respectively (**RQ1**). We also show the results obtained by a standard SSL-GAN (**RQ2**). Thirty independent runs were performed in all the cases, except in the analysis of the sensitivity of the methods to the number of labels (**RQ3**) in `MNIST`, where 15 independent runs were performed.

5.1 `RING` experimental results

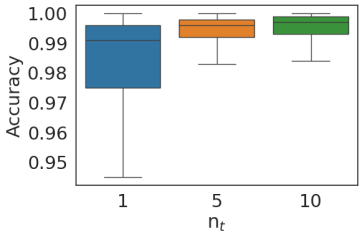
Table 1 shows the accuracy for the discriminator and the Wasserstein distance for the generator in the `RING` dataset. We aggregate the results by value of n_t to analyze the influence of that parameter. Figs. 3a and 3b show the same results using boxplots. We can observe that the accuracy of the discriminator is higher when n_t is 5 or 10, this means, more training epochs during the offspring generation. There is no statistically significant difference (at $\alpha = 0.05$ level) between 5 and 10, but differences are statistically significant between these two values and $n_t = 1$. The same observations can be made for the generator (W_D), where, again, $n_t = 5$ and $n_t = 10$ are statistically significantly better (lower value for W_D) than $n_t = 1$.

Table 1: Classification accuracy of discriminator and Wasserstein distance of generator grouped by n_t for `RING` dataset.

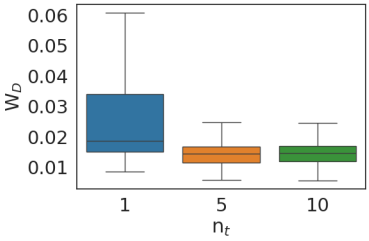
	Accuracy				W_D			
n_t	Min	Median	IQR	Max	Min	Median	IQR	Max
1	0.550	0.991	0.021	1.000	0.009	0.019	0.019	0.668
5	0.787	0.996	0.006	1.000	0.006	0.014	0.005	0.110
10	0.709	0.997	0.006	1.000	0.006	0.015	0.005	0.111

Figs. 3c and 3d show the evolution of the median accuracy and the Wasserstein distance during the search in the experiments for different values of n_t. We can observe a faster convergence and a more stable evolution when n_t is 5 and 10. The evolution for $n_t = 1$ is noisy and slower.

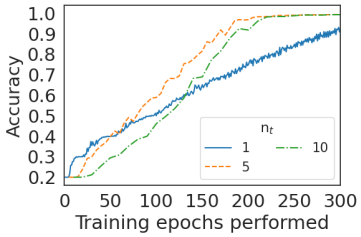
Now, we move to the influence of the population size, μ, and offspring size, λ, on the performance of the networks. Figs. 4a and 4b show the accuracy of the discriminator and the Wasserstein distance of the generator for the `RING` dataset for different combinations of μ and λ. Regarding the accuracy of the discriminators, we observe that the basic SSL-GAN has lower accuracy compared to CE-SSLGAN. The differences are statistically significant except in the comparison with the CE-SSLGANvariations with $\mu = 7$, $\lambda = 1$ and $\mu = 9$, $\lambda = 1$. There is no clear influence of the population size, μ, on the accuracy. We can



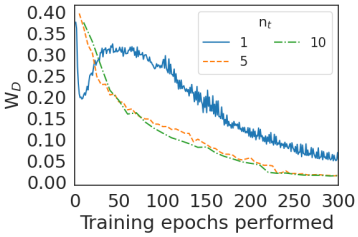
(a) Classification accuracy



(b) W_D



(c) Median classification acc. evolution (first 300 training epochs)

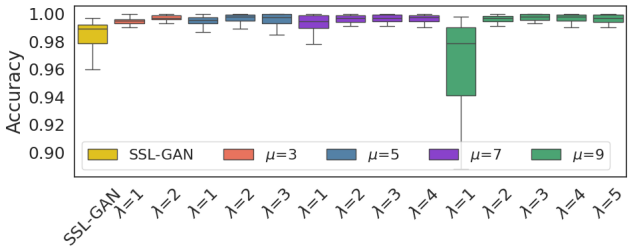


(d) Median W_D evolution (first 300 training epochs)

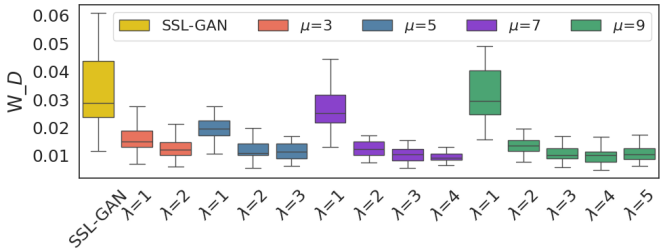
Fig. 3: Influence of n_t on RING dataset

observe that the accuracy is higher when the number of offspring, λ, is greater than 1. The conclusion here is that we should set $\lambda > 1$. Similar results, but with a more clear trend, can be observed in the case of the generator performance, using W_D. The generators of SSL-GAN have a higher value for W_D, compared to those of CE-SSLGAN, with statistically significant differences. Here again, a value of $\lambda > 1$ decreases the Wasserstein distance. We can also appreciate a reduction in W_D when μ is increased, but the statistical tests show no significant differences. We conclude that CE-SSLGAN outperforms the results of the basic SSL-GAN (**RQ2**), the population size, μ, has no significant difference on the results, and the number of offsprings should be more than one (**RQ1**).

In Figs. 5a and 5b we show the evolution of the discriminator accuracy and the Wasserstein distance (for generators) in the RING dataset for some selected configurations of CE-SSLGAN and SSL-GAN. We can observe in both figures that the evolution of SSL-GAN is noisy and slow for W_D. CE-SSLGAN, on the other hand, shows a soft evolution with a convergence time that depends on the population size, μ, in the case of the accuracy. The fastest configuration of CE-SSLGAN is the one with $\mu = 3$. Increasing the population size also increases the time to reach the maximum accuracy. Offspring's generation can be done in parallel. Parallelization would compress the curves to the left and they could overlap if we consider the evolution with respect to the wall clock time.



(a) Classification results for RING dataset when using n_t=10



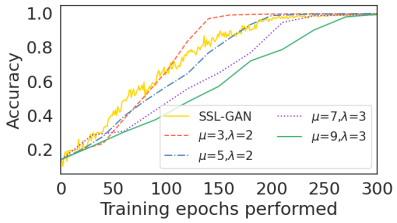
(b) W_D results for RING experiments when using n_t=10

Fig. 4: Influence of population and offspring size in the performance for RING

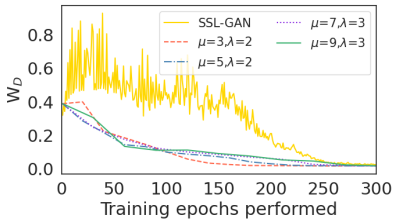
5.2 BLOB experimental results

The influence of n_t in the discriminator accuracy and the Wasserstein distance obtained by the generators in the BLOB dataset are presented in Table 2. Figs. 6a and 6b shows the evolution of the median accuracy and median W_D for the discriminator and generator, respectively, during the search. We observe the same behavior we saw in the RING dataset: n_t should be larger than 1, but there are no differences between 5 and 10 (**RQ1**).

Regarding the influence of the population size and the number of offspring per generation, we also observe the same trends as in the RING dataset in Figs. 7a and 7b: CE-SSLGAN outperform the results of SSL-GAN (**RQ2**), and the best



(a) Median classification acc. evolution

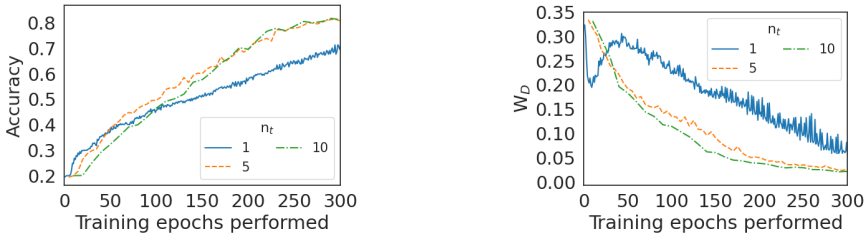


(b) Median W_D evolution

Fig. 5: Evolution of the classification accuracy and W_D through each training epoch performed for the selected methods on RING (first 300 training epochs)

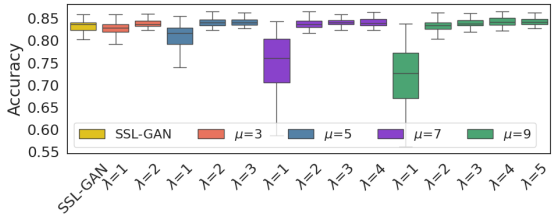
Table 2: Classification accuracy of discriminator and Wasserstein distance of generator grouped by n_t for BLOB dataset.

	Accuracy				W_D			
n_t	Min	Median	IQR	Max	Min	Median	IQR	Max
1	0.473	0.826	0.040	0.861	0.006	0.018	0.021	0.449
5	0.516	0.829	0.012	0.861	0.006	0.013	0.009	0.291
10	0.568	0.830	0.011	0.857	0.005	0.013	0.008	0.075

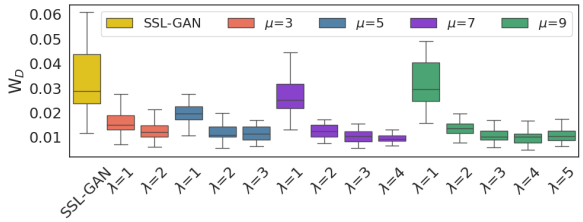


(a) Median classification acc. evolution (b) Median W_D evolution

Fig. 6: Influence of n_t in the BLOB dataset (first 300 training epochs)



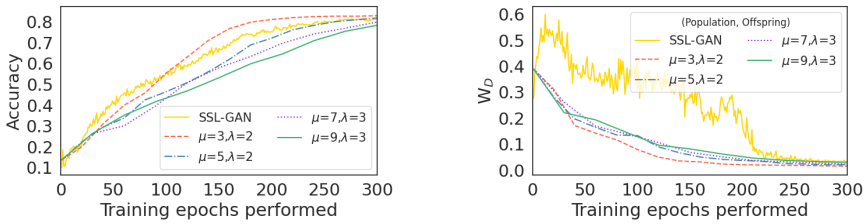
(a) Classification results for BLOB experiments when using n_t=10



(b) W_D results for BLOB experiments when using n_t=10

Fig. 7: Influence of population and offspring size in the performance for BLOB

performance is obtained when $\lambda > 1$, while μ does not seem to have much influence on the results (**RQ1**).



(a) Median classification acc. evolution (b) Median W_D evolution

Fig. 8: Evolution of the classification accuracy and W_D through each training epoch performed for the selected methods (first 300 training epochs) for BLOB

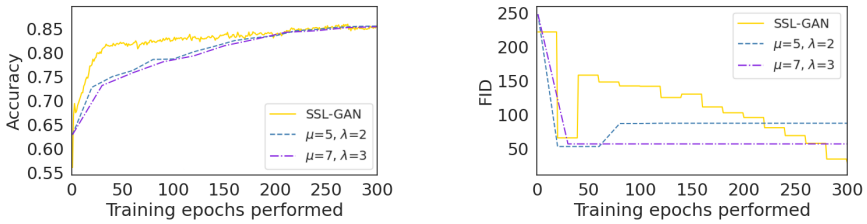
5.3 MNIST experimental results

Taking into account the trends observed in the results for RING and BLOB, we only show results of CE-SSLGAN for $n_t = 10$ and two combinations of population and offspring size: μ=5, λ=2; and μ=7, λ=3. In Table 3 we show the statistics of the classification accuracy of the discriminators and the FID obtained by the generators. The main conclusion from these results is that CE-SSLGAN outperforms the results of SSL-GAN (**RQ2**). The differences observed in the results between SSL-GAN and CE-SSLGAN are statistically significant. The differences between the two population configurations ($\mu = 5, \lambda = 2$ and $\mu = 7, \lambda = 3$) are not statistically significant. In Figs. 9a and 9b we present the evolution of the accuracy and FID during the search. We can observe that the SSL-GAN is very fast at the beginning, it reaches better values for the accuracy and FID than CE-SSLGAN in the first 300 training epochs. However, in the last 300 training epochs (not shown) CE-SSLGAN outperforms SSL-GAN.

Table 3: Classification accuracy of discriminator and FID of generator grouped for CE-SSLGAN variations and SSL-GAN addressing MNIST dataset.

	Accuracy				FID			
	Min	Median	IQR	Max	Min	Median	IQR	Max
CE-SSLGAN								
$\mu = 5, \lambda = 2$	0.759	0.883	0.012	0.898	22.590	27.134	14.744	44.734
$\mu = 7, \lambda = 3$	0.759	0.880	0.013	0.897	22.570	24.952	16.663	60.253
SSL-GAN	0.751	0.857	0.020	0.879	24.769	29.335	34.140	96.689

We analyze in the MNIST dataset what is the sensitivity of SSL-GAN and CE-SSLGAN to the number of labels in the dataset (**RQ3**). We evaluate the accuracy of the discriminator when the number of samples per class n_s in the labeled data is 10 (the base case), 60, and 100. Table 4 shows the median accuracy and interquartile range (IQR). We observe that as n_s increases the accuracy also increases for all the methods. CE-SSLGAN is still better than SSL-GAN in all



(a) Median classification acc. evolution (b) Median FID evolution

Fig. 9: Evolution of the classification accuracy and FID through each training epoch performed for the selected methods (first 300 training epochs) for `MNIST`

the cases. However, the difference in the accuracy is decreasing between the two methods with a larger number of samples per class.

6 Conclusions and future work

In this paper, we have proposed an elitist co-evolutionary algorithm for SSL-GAN training with a panmictic population. Our goal was to analyze the influence of different hyper-parameters of the method in the performance of the generators and discriminators. We can answer **RQ1** by concluding that having more than one individual in the population ($\mu > 1$) and generating more than one individual in each generation ($\lambda > 1$) is beneficial for the search. We did not observe a clear influence of the population size μ on the results when $\mu > 1$. We also observe that our CE-SSLGAN method outperforms a standard SSL-GAN, answering **RQ2**. Finally, we answer **RQ3** concluding that increasing the number of labels per class, n_s, improves the performance of both CE-SSLGAN and SSL-GAN, and the differences between them are reduced when n_s increases. Although our conclusions are only for the three datasets used and more experiments should be done to confirm these findings, we think the results in this paper could be used as a guide for parameter tuning of SSL-GAN training using EAs.

Table 4: Classification accuracy of discriminator for CE-SSLGAN and SSL-GAN in the `MNIST` dataset when the number of samples per class n_s changes.

	$n_s = 10$		$n_s = 60$		$n_s = 100$	
	Median	IQR	Median	IQR	Median	IQR
CE-SSLGAN						
$\mu = 5, \lambda = 2$	0.883	0.012	0.953	0.002	0.956	0.011
$\mu = 7, \lambda = 3$	0.880	0.013	0.952	0.003	0.956	0.010
SSL-GAN	0.857	0.020	0.939	0.008	0.950	0.009

Although we treat the training problem as a single-objective problem, discriminator training could be formulated using a multi-objective approach, where supervised and unsupervised losses, are considered as two different objectives of

the training process. Besides, CE-SSLGAN could be implemented using parallelism. In addition, it would be interesting to analyze the influence of the spatial distribution of the population in the performance of the GAN. This could be done comparing the results of CE-SSLGAN with that of Lipizzaner [11].

Acknowledgements

This work is partially funded by the Junta de Andalucia, Spain, under contract QUAL21 010UMA; and Universidad de Málaga under the grant B1-2022_18. We thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI center of the University of Malaga.

References

1. Byrd, R., Damani, K., Che, H., Calandra, A., Kim, D.: A systematic literature review of volumetric 3d model reconstruction methodologies using generative adversarial networks. J. Inf. Sci. Eng. **38**(6), 1243–1263 (2022), http://jise.iis.sinica.edu.tw/JISESearch/pages/View/PaperView.jsf?keyId=189_2545
2. Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**(6), 141 – 142 (2012). https://doi.org/10.1109/MSP.2012.2211477, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85032752689&doi=10.1109%2fMSP.2012.2211477&partnerID=40&md5=efb588a7093fa4ddb9e65d13b64dcc89, cited by: 3047
3. Dorronsoro, B., Alba, E.: Cellular Genetic Algorithms. Springer, New York, USA (2008). https://doi.org/10.1007/978-0-387-77610-1
4. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada. pp. 2672–2680 (2014), https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html
5. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017)
6. Ma, Y., Zheng, Y., Zhang, W., Wei, B., Lin, Z., Liu, W., Li, Z.: Optimization of semi-supervised generative adversarial network models: a survey. International Journal of Intelligent Computing and Cybernetics **17**(4) (2024). https://doi.org/10.1108/IJICC-05-2024-0202
7. Qin, Y., Wang, Z., Xi, D.: Tree cyclegan with maximum diversity loss for image augmentation and its application into gear pitting detection. Appl. Soft Comput. **114**, 108130 (2022). https://doi.org/10.1016/J.ASOC.2021.108130, https://doi.org/10.1016/j.asoc.2021.108130
8. Sajun, A.R., Zualkernan, I.: Survey on implementations of generative adversarial networks for semi-supervised learning. Applied Sciences **12**(3) (2022). https://doi.org/10.3390/app12031718, https://www.mdpi.com/2076-3417/12/3/1718
9. Schmiedlechner, T., Yong, I.N.Z., Al-Dujaili, A., Hemberg, E., O'Reilly, U.: Lipizzaner: A system that scales robust generative adversarial network training. CoRR **abs/1811.12843** (2018), http://arxiv.org/abs/1811.12843

10. Tachibana, R., Matsubara, T., Uehara, K.: Semi-supervised learning using adversarial networks. In: 15th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2016, Okayama, Japan, June 26-29, 2016. pp. 1–6. IEEE Computer Society (2016). https://doi.org/10.1109/ICIS.2016.7550881, https://doi.org/10.1109/ICIS.2016.7550881
11. Toutouh, J., Nalluru, S., Hemberg, E., O'Reilly, U.: Semi-supervised generative adversarial networks with spatial coevolution for enhanced image generation and classification. Appl. Soft Comput. **148**, 110890 (2023). https://doi.org/10.1016/J.ASOC.2023.110890, https://doi.org/10.1016/j.asoc.2023.110890
12. Toutouh, J., Nalluru, S., Hemberg, E., O'Reilly, U.: Semi-supervised learning with coevolutionary generative adversarial networks. In: Silva, S., Paquete, L. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2023, Lisbon, Portugal, July 15-19, 2023. pp. 568–576. ACM (2023). https://doi.org/10.1145/3583131.3590426, https://doi.org/10.1145/3583131.3590426
13. Vallender, S.: Calculation of the wasserstein distance between probability distributions on the line. Theory of Probability & Its Applications **18**(4), 784–786 (1974)
14. Wang, H., Won, D., Yoon, S.W.: An adaptive neural architecture optimization model for retinal disorder diagnosis on 3d medical images. Appl. Soft Comput. **111**, 107686 (2021). https://doi.org/10.1016/J.ASOC.2021.107686, https://doi.org/10.1016/j.asoc.2021.107686