Signal Propagation in a Gradient-Based and Evolutionary Learning System

Jamal Toutouh
Massachusetts Institute of Technology, CSAIL, USA
University of Malaga, LCC, Spain
toutouh@mit.edu,jamal@lcc.uma.es

Una-May O’Reilly
Massachusetts Institute of Technology, CSAIL
unamay@csail.mit.edu

1 INTRODUCTION

Generative adversarial networks (GANs) exhibit training pathologies that can lead to convergence-related degenerative behaviors, whereas spatially-distributed, coevolutionary algorithms (CEAs) for GAN training, e.g., Lipizzaner, are empirically robust to them. The robustness arises from diversity that occurs by training populations of generators and discriminators in each cell of a toroidal grid. Communication, where signals in the form of parameters of the best GAN in a cell propagate in four directions: North, South, West and East, also plays a role, by communicating adaptations that are both new and fit. We propose Lipi-Ring, a distributed CEA like Lipizzaner, except that it uses a different spatial topology, i.e. a ring. Our central question is whether the different directionality of signal propagation (effectively migration to one or more neighbors on each side of a cell) meets or exceeds the performance quality and training efficiency of Lipizzaner Experimental analysis on different datasets (i.e., MNIST, CelebA, and COVID-19 chest X-ray images) shows that there are no significant differences between the performances of the trained generative models by both methods. However, Lipi-Ring significantly reduces the computational time (14.2%…41.2%). Thus, Lipi-Ring offers an alternative to Lipizzaner when the computational cost of training matters.

CCS CONCEPTS
• Computing methodologies → Bio-inspired approaches; Neural networks; Search methodologies; Ensemble methods;

KEYWORDS
Generative adversarial networks, ensembles, genetic algorithms, diversity

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

GECCO ‘21, July 10–14, 2021, Lille, France
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/Y/MM… $15.00
https://doi.org/10.1145/nnnnnn.nnnnnn
This section introduces the main concepts in GANs training and variable from a latent space $z$. The goal of GAN training is to find $G$ to approximate the latent data distribution in general, both networks are trained by applying back-propagation.

## 2.2 Related work

Mode collapse and vanishing gradients are the most frequent GAN training pathologies [2, 5], leading to inconsistent results. Prior studies tried to mitigate degenerate GAN dynamics with new generator or discriminator objectives (loss functions) [3, 18, 19, 31] and applying heuristics [14, 22].

Others have integrated EC into GAN training. Evolutionary GAN (E-GAN) evolves a population of generators [29]. The mutation selection among three optimization objectives (loss functions) updates the weights of the generators, which are adversarially trained against a single discriminator. Multi-objective E-GAN (MO-EGAN) has been defined by reformulating E-GAN training as a multi-objective optimization problem by using Pareto dominance to select the best solutions in terms of diversity and quality [6].

Two genetic algorithms (GAs) have been applied to learn mixtures of heterogeneous pre-trained generators to specifically deal with mode collapse [28]. Finally, in [10], a GA evolves a population of GANs (represented by the architectures of the generator and the discriminator and the training hyperparameters). The variation operators exchange the networks between the individuals and evolves the architectures and the hyperparameters. The fitness is computed after training the GAN encoded by the genotype.

Another line of research uses CEAs to train a population of generators against a population of discriminators. Coevolutionary GAN (COEGAN) combines neuroevolution with CEAs [8]. Neuroevolution is used to evolve the main networks’ parameters. COEGAN applies an all-vs-best Comp-CEA (with $k$-best individuals) for the competitions to mitigate the computational cost of all-vs-all.

CEAs showed similar pathologies as the ones reported in GAN training, such as focusing, and lost of gradient, which have been attributed to a lack of diversity [21]. Thus, spatially distributed populations have been demonstrated to be particularly effective at maintaining diversity, while reducing the computational cost from quadratic to linear form [30]. Lipizzaner locates the individuals of a population of GANs (pairs of generators and discriminators) in a 2D toroidal grid. A neighborhood is defined by the cell itself and its adjacent cells according to Von Neumann neighborhood. Coevolution proceeds at each cell with sub-populations drawn from the neighborhood. Gradient-based learning is used to update the weights of the networks while evolutionary selection and variation are used for hyperparameter learning [1, 24]. After each training iteration, the (weights of the) best generator and discriminator are kept while the other sub-population members are refreshed by new copies from the neighborhood. A cell’s update of its GAN is effectively propagated to the adjacent cells in four directions (i.e., North, South, East, and West) once the neighbors of the cell refresh their sub-populations from neighborhood copies. Thus, each cell’s sub-populations are updated with new fit individuals, moving them closer towards convergence, while fostering diversity. Another approach, Mustangs, combines Lipizzaner and E-GAN [25].
Thus, the mutation operator randomly selects among a set of loss functions instead of applying always the same one in order to increase variability. Finally, taking advantage of the spatial grid of Lipizzaner, a data dieting approach has been proposed [27]. The main idea is to train each cell with different subsets of data to foster diversity among the cells and to reduce the training resource requirements.

In this study, we propose Lipi-Ring, spatially distributed GAN training that uses a ring topology instead of a 2D grid. We contrast Lipi-Ring to Lipizzaner in the next section.

3 RING-DISTRIBUTED GAN TRAINING

This section describes the Lipi-Ring CEA GAN training, which applies the same principles (definitions and methods) as Lipizzaner [24]. We introduce both 2D grid and ring topologies applied by Lipizzaner and Lipi-Ring, respectively. We summarize the spatially distributed GAN training method. We present the main distinctive features between Lipizzaner and Lipi-Ring.

3.1 Spatially Structured Populations

Lipi-Ring and Lipizzaner use a population of generators $g = \{g_1, \ldots, g_Z\}$ and a population of discriminators $d = \{d_1, \ldots, d_Z\}$, which are trained against each other (where $Z$ is the size of the population). A generator-discriminator pair named center is placed in each cell, which belongs to a ring in the case of Lipi-Ring or to a 2D toroidal grid in the case of Lipizzaner. According to the topology’s neighborhood, sub-populations of networks, generators and discriminators, of size $s$ are formed.

For the $k$-th neighborhood, we refer the center generator by $g^k_{center} \subseteq g$, the set of generators in the neighborhood by $g^k = \{g^{k,1}, \ldots, g^{k,s}\}$, and the generators in this $k$-th sub-population by $g^k_s = \cup_{i=1}^s g^{k,i} \subseteq g$. The same is stated for discriminators $d$.

Lipizzaner uses Von Neumann neighborhoods with radius 1, which includes the cell itself and the ones in the adjacent cells to the North, South, East, and West [24], i.e. $s=5$ (see Figure 1.a). This defines the migration policy (i.e., the directionality of signal propagation) through the cells in four directions.

Figure 1.a shows an example of a 2D toroidal grid with $Z=16$ (4x4 grid). The shaded areas illustrate the overlapping neighborhoods of the (1,1) and (0,2) cells, with dotted and solid outlines, respectively. The updates in the center of (1,1) will be propagated to the (0,1), (2,1), (1,0), and (1,2) cells.

In Lipi-Ring, the cells are distributed in a one-dimensional grid of size $1 \times Z$ and neighbors are sideways, e.g. left/or right, i.e. an index position or more away. The best GAN (center) after an evolutionary epoch at a cell is updated. Neighborhood cells retrieve this update when they refresh their sub-population membership at the end of their training epochs, effectively forming two non-ending pathways, around the ring, carrying signals in two directions. Figures 1.b and 1.c show populations of six individuals ($Z=6$) organized in a ring topology with neighborhood radius one ($r=1$) and two ($r=2$), respectively. The shaded areas illustrate the overlapping neighborhoods of the cells (0) and (4), with dotted and solid outlines, respectively. The updates in the center of (0) will be propagated to the (5) and (1) for $r=1$ (Figure 1.b) and to the (4), (5), (1), and (2) for $r=2$ (Figure 1.c).

Figure 1: Overlapping neighborhoods and sub-population definitions according topologies and neighborhood radius.

Thus the topologies of Lipizzaner and Lipi-Ring influence communication (of the parameters of the best cell each epoch) and this, in turn, affects how the populations converge.

3.2 Spatially Distributed GAN Training

Algorithm 1 illustrates the main steps of the applied training algorithm. First, it starts the parallel execution of the training on each cell by initializing their own learning hyperparameters (Line 2). Then, the training process consists of a loop with two main phases: first, migration, in which the cells gather the GANs (neighbors) to build the sub-population ($n$), and second, train and evolve when each cell updates the center by applying the coevolutionary GANs training (see Algorithm 2). These steps are repeated $T$ (generations or training epochs) times. After that, each cell learns an ensemble of generators by using an Evolutionary Strategy, ES-(1+1) [17, Algorithm 2.1], to compute the mixture weights $\omega$ to optimize the accuracy of the generative model returned ($n, \omega$)* [24].

For each training generation, the cells apply the CEA in Algorithm 2 in parallel. The CEA starts by selecting best pair generator and discriminator (called $g_b$ and $d_b$) according to a tournament selection of size $\tau$. It applies an all-vs-all strategy to evaluate all the GAN pairs in the sub-population according to a randomly chosen batch of data $Br$ (Lines 1 to 4). Then, for each batch of data in the training dataset (Lines 5 to 10), the learning rate $n_b$ is updated applying Gaussian Mutation [24] and the offspring is created by training $g_b$ and $d_b$ against a randomly chosen discriminator and
generator from the sub-population (i.e., applying gradient-based mutations). Thus, the sub-populations are updated with the new individuals. Finally, a replacement procedure is applied to remove from the sub-populations the weakest individuals and the center is updated with the individuals with the best fitness (Lines 11 to 15). The fitness $L_{g,d}$ of a given generator (discriminator) is evaluated according to the binary-cross-entropy loss, where the model’s objective is to minimize the Jensen-Shannon divergence between the real and fake data [11].

Algorithm 1 Lipizzaner main steps.  
Input: $E$: Total generations, $E$: Grid cells, $s$: Neighborhood size, $\theta_{\text{TD}}$: Training dataset, $\theta_{\text{COEV}}$: Parameters for CoevGANsTraining, $\theta_{\text{EA}}$: Parameters for MixtureEA  
Return: $n$: neighborhood, $\omega$: mixture weights

1. parfor $c \in E$ do \texttt{Asynchronous parallel execution of all cells in grid}
2. $n, \omega \leftarrow \text{initializeCells}(c, k, \theta_{\text{TD}})$ \texttt{Initialization of cells}
3. for generation $d \in [0, \ldots, T]$ do \texttt{Iterate over generations}
    4. $n \leftarrow \text{copyNeighbours}(c, k)$ \texttt{Collect neighbor cells}
    5. $n \leftarrow \text{CoevGANsTraining}(n, \theta_{\text{TD}}, \theta_{\text{COEV}})$ \texttt{Coevolve GANs}
    6. $\omega \leftarrow \text{MixtureEA}(n, \omega, \theta_{\text{EA}})$ \texttt{Build optimal ensemble}
4. end parfor
5. return $(n, \omega)^*$ \texttt{Cell with best generator mixture}

Here, we have summarized the spatially distributed method applied. More detailed definitions of this spatially distributed GAN training method can be found in [1, 24].

3.3 Lipi-Ring vs. Lipizzaner

The population structured in a ring with $r=2$ provides a signal propagation similar to the Lipizzaner toroidal grid because both topologies and migration models allow the center individual to reach four cells (two located in the West and two in the East in the case of this ring), see figures 1.a and 1.b. Thus, they provide the same propagation speed and, as their sub-populations are of size $s=5$, the same selection pressure. The signal propagation of the ring with $r=1$ is slower because, after a given training step, the center only reaches two cells (the one in the West and the one in the East), see Figure 1.b. In this case, the sub-populations have three individuals, which reduces the diversity in the sub-population and accelerates convergence (it has 40% fewer individuals than Lipizzaner sub-populations). Thus, Lipi-Ring with $r=1$ reduces the propagation speed, while accelerating the population’s convergence.

Lipi-Ring with $r=1$ has two main advantages over Lipizzaner: a) it mitigates the overhead due to the communication is carried out only with two cells (instead of four) and the sub-populations are smaller, which reduces the number of operations for fitness evaluation and selection/replacement; and b) like all Lipi-Ring with any radius, it does not require to have a rectangular grid of cells, but Z may be any natural number.

Given the infeasibility of analyzing the change in selection and other algorithm elements, we proceed empirically.

4 EXPERIMENTAL SETUP

We evaluate different distributed CEA GAN training on image data sets: the well known, MNIST [9] and CelebA [16]; and a dataset of chest X-ray images of patients with COVID-19 [7].

In our experiments, we evaluate the following algorithms: Lipizzaner; two variations of Lipi-Ring both with $r=1$, Ring(1) that performs the same training epochs than Lipizzaner and Ring'(1) that runs for the same computational cost (wall clock time) than Lipizzaner; and Ring(2) which is the Lipi-Ring with $r=2$ performing the same number of iterations than Lipizzaner. These represent a variety of topologies and migration models. Ring'(1) is analyzed to make a proper comparison between Ring(1) ($r=1$) and Lipizzaner taking into account the computational cost (time). Table 1 summarizes the main characteristics of the Lipi-Ring variations studied.

The parameters are set according to the authors of Lipizzaner [24, 25]. Thus, all these CEsAs apply a tournament selection of size two. The main settings used for the experiments are summarized in Table 2.

Table 1: Lipi-Ring variations analyzed.

| Parameter | Value |
|-----------|-------|
| $r$ | 1 |
| $r'$ | 1 |
| $r''$ | 2 |
| $s$ | 3 |
| $s'$ | 3 |
| $s''$ | 5 |

Table 2: Main GAN training parameters.

| Parameter | Value |
|-----------|-------|
| Network topology | MNIST: MLP, CelebA: DCGAN, COVID-19: DCGAN |
| Input neurons | 64, 100, 100 |
| Number of hidden layers | 2, 4, 4 |
| Neurons per hidden layer | 256, 16,384 - 131,072, 32,768 - 524,288 |
| Output neurons | 784, 64x64 x 64, 128x128 |
| Activation function | tanh, tanh, tanh |
| Batch size | 100, 128, 69 |
| Skip N disc. steps | 1, - |
| Learning rate | 0.0002, 0.00005, 0.0002 |
| Initial learning rate | 0.0001, 0.0001, 0.0001 |
| Mutation rate | 0.01, 0.5, 0.5 |
For MNIST experiments, the generators and the discriminators are multilayer perceptrons (MLP). The stop condition of each method is defined as follows: a) \( \text{Ring}(1) \), \( \text{Ring}(2) \), and Lipizzaner perform 200 training epochs to evaluate the impact of the topology on the performance and computational time required; and b) \( \text{Ring}(1) \) stops after running for the same time than Lipizzaner to compare the methods taking into account the same computational cost (time). The population sizes are 9, 16, and 25, which means Lipizzaner uses grid of sizes 3×3, 4×4, and 5×5. Besides, we study the impact of the size of the populations on \( \text{Ring}(1) \) by training rings of size between 2 and 9.

For CelebA and COVID-19 experiments, deep convolutional GANs (DCGAN) are trained. DCGANs have much more parameters than MLP (see Table 2). Here, we compare \( \text{Ring}(1) \) and Lipizzaner. They stop after performing 20 training epochs for CelebA and 1,000 for COVID-19 (because COVID-19 training dataset has much fewer samples). This will discern the differences in terms of performance and computational cost with more complex networks. In these cases, the population size is 9.

The experimental analysis is performed on a cloud computing platform that provides 16 Intel Xeon cores 2.8GHz with 64 GB RAM and an NVIDIA Tesla P100 GPU with 16 GB RAM. We run multiple independent runs for each method.

We have implemented all variations of the Lipi-Ring by extending Lipizzaner framework [24] using Python and Pytorch [20].

5 RESULTS AND DISCUSSION

This section presents the results and the analyses of the presented GAN training methods. The first subsections evaluate them with the MNIST dataset. They are measured in terms of: the FID score, the diversity of the generated samples by evaluating the total variation distance (TVD) [15], and the diversity in the genome space (network parameters). Then, we analyze the CelebA and COVID-19 results with measurements of Inception Score (IS) and computational time. The next subsection presents the results incrementally increasing the ring size of \( \text{Ring}(1) \). Finally, we compare the computational times needed by \( \text{Ring}(1) \) and Lipizzaner.

5.1 Quality of the MNIST Generated Data

Table 3 shows the best FID value from each of the 30 independent runs performed for each method. All the evaluated methods improve their performance when increasing the population size, while maintaining the same budget. This could be explained by diversity increasing during the training, as the populations get bigger.

\( \text{Ring}(1) \) has the lowest (best) median and mean FID results. In turn, it returned the generative model that provided the best quality samples (minimum FID score) for all the evaluated population sizes. The second best results are provided by \( \text{Ring}(1) \). This indicates that the methods with smaller sub-populations converge faster, even though the ring migration model \( (r=1) \) slows down the propagation of the best individuals.

Comparing \( \text{Ring}(2) \) and Lipizzaner, they provide close results. Though they use different topologies, their signal propagation and selection operate equally.

As the results do not follow a normal distribution, we rank the studied methods using the Friedman rank statistical test and we apply Holm correction post-hoc analysis to assess the statistical significance. For all the population sizes, the Friedman ranks \( \text{Ring}(1) \), Lipi-Ring, Lipizzaner, and \( \text{Ring}(2) \) as the firsts, second, third, and fourth, respectively. However, the significance (p-value) varies from \( p\text{-value}=5.67\times10^{-6} \) for population size 9 to \( p\text{-value}=1.13\times10^{-2} \) (i.e., \( p\text{-value} \geq 0.01 \)) for population sizes 16 and 25. According to the Holm post-hoc analysis, \( \text{Ring}(1) \) and \( \text{Ring}(1) \) are statistically better than \( \text{Ring}(2) \) and Lipizzaner (which have no statistical differences between each other) for population size 9. For the other population sizes, \( \text{Ring}(1) \) provides statistically better results than \( \text{Ring}(2) \) and Lipizzaner and there is no significant difference among \( \text{Ring}(1) \), \( \text{Ring}(2) \), and Lipizzaner.

Table 3 shows \( \text{Ring}(1) \) is better than \( \text{Ring}(2) \) and Lipizzaner (lower FID is better). However, the difference between their FIDs decreases when population size increases. This indicates that migration modes that allow faster propagation take better advantage of bigger populations.

Next, we evaluate the FID score throughout the GAN training process. Figure 2 illustrates the changes of the median FID during the training process. \( \text{Ring}(1) \) is not included because it operates the same as \( \text{Ring}(1) \). Although the FID score almost behaves like a monotonically decreasing function with oscillations. The methods with larger sub-populations, i.e., \( \text{Ring}(2) \) and Lipizzaner, show smaller oscillations which implies more robustness (less variance).

Focusing on the methods with a ring topology, we clearly see the faster convergence when \( r=1 \). \( \text{Ring}(1) \) most of the time provides smaller FID values than \( \text{Ring}(2) \). The reduced sub-population \( (s=3) \) favors the best individual from the sub-population to be selected during the tournament (it increases the selection pressure).

Figure 2 shows \( \text{Ring}(1) \), \( \text{Ring}(2) \), and Lipizzaner converge to similar values. This is in accordance with the results in Table 3 that indicate these three methods provide comparable FID scores.

Finally, Figure 3 illustrates some samples synthesized by generators trained using populations of size 16. As it can be seen, the four sets of samples show comparable quality.
5.2 Diversity of the Generated MNIST Outputs

This section evaluates the diversity of the samples generated by the best generative models obtained each run. Table 4 reports the TVD for each method and population size. Recall we prefer low TVDs (high diversity) to show the quality of the generative model.

Table 4: TVD MNIST results in terms of best mean, standard deviation, Iqr, minimum, and maximum. Best values in bold. (Low TVD indicates more diverse samples).

| Method   | Means ± Std | Median | Iqr | Min | Max |
|----------|-------------|--------|-----|-----|-----|
| Ring(1)  | 0.11 ± 0.02 | 0.11   | 0.03| 0.07| 0.16|
| Ring(1)  | 0.11 ± 0.02 | 0.11   | 0.03| 0.07| 0.14|
| Ring(2)  | 0.12 ± 0.02 | 0.12   | 0.02| 0.09| 0.19|
| Lipizzaner | 0.12 ± 0.02 | 0.12   | 0.03| 0.08| 0.18|
| Population size: 9 |
| Ring(1)  | 0.09 ± 0.02 | 0.09   | 0.02| 0.07| 0.13|
| Ring(1)  | 0.10 ± 0.01 | 0.09   | 0.03| 0.07| 0.12|
| Ring(2)  | 0.10 ± 0.01 | 0.10   | 0.02| 0.06| 0.13|
| Lipizzaner | 0.10 ± 0.02 | 0.10   | 0.02| 0.05| 0.16|
| Population size: 16 |
| Ring(1)  | 0.09 ± 0.02 | 0.09   | 0.02| 0.05| 0.12|
| Ring(1)  | 0.09 ± 0.02 | 0.09   | 0.02| 0.05| 0.11|
| Ring(2)  | 0.09 ± 0.02 | 0.09   | 0.01| 0.06| 0.12|
| Lipizzaner | 0.09 ± 0.02 | 0.10   | 0.02| 0.06| 0.16|
| Population size: 25 |

The results in Table 4 demonstrate that, as the population size increases, the resulting trained generative models are able to provide more diverse samples, that is, they have better coverage of the latent distribution. Thus, again, all the methods take advantage of bigger populations.

For population size 9, the mean and median TVD of Ring(1) and Ring’(1) are the lowest. According to Friedman ranking Ring’(1), Ring(1), Lipizzaner, and Ring(2) are first, second, third, and fourth, respectively (p-value=0.0004). The Holm post-hoc correction confirms that there are no significant differences between Ring(1) and Ring’(1) and they are statistically more competitive than Ring(2) and Lipizzaner (p-values<0.01).

For bigger populations, the statistical analyses report that there are no significant differences between the Ring(1), Ring(2), and Lipizzaner, and Ring’(1) is statistically better than Ring(2) and Lipizzaner (p-values<0.01).

With the support of the FID and TVD results, we can answer RQ1: What is the effect on the generative model trained when changing the directionality of the signal propagation from four directions to two? Answer: The impact on the results of performing migration to one or more neighbors is higher than the directionality itself. If we isolate the directionality, i.e., Lipi-Ring vs. Lipizzaner, the main differences are revealed when r=1. Thus, Ring’(1) generators create statistically better samples (FID and TVD results) than Lipizzaner. However, when r=2, there is no significant difference between Ring(2) and Lipizzaner for all the evaluated population sizes. So, we observe that directionality is irrelevant.

We can also answer RQ2: When the signal is propagated to only two directions, what is the impact of performing migration in one or more neighbors? Answer: When comparing between Lipi-Ring with r=1 and with r=2 using the same number of training epochs (i.e., Ring(1) and Ring(2)), they perform the same although Ring(1) converges faster. The smallest sub-population of Ring(1) likely increases the selection pressure and the convergence speed of the sub-populations, despite slower signal propagation.

5.3 Genome Diversity of MNIST

This section analyzes the diversity in the genome space (i.e., the distance between the weights of the evolved networks). We evaluate the populations of size 9 and 16 to see how the migration model and sub-population size affect their diversity. L2 distance between neural network parameters in a given population is used to measure...
diversity and Table 5 summarizes the results. Figure 4 presents the $L_2$ distances for the populations that represent the median $L_2$.

Table 5: $L_2$ distance values between the weights of the networks in different cells on MNIST dataset. High values indicate more networks diversity. Highest value is in bold.

| Method   | Mean±Std | Median | Iqr  | Min  | Max  |
|----------|----------|--------|------|------|------|
| Ring(1)  | 16.92±7.14 | 15.53  | 9.60 | 6.79 | 40.19 |
| Ring'(1) | 13.38±4.97  | 12.07  | 6.49 | 6.67 | 22.56 |
| Ring(2)  | 12.69±5.89  | 13.98  | 10.85| 4.10 | 25.33 |
| Lipizzaner| 13.85±8.29  | 18.06  | 15.32| 4.09 | 23.46 |

Focusing on the Lipi–Ring methods with $r=1$, the population diversity diminishes with more training, i.e., $L_2$ distances between the networks in Ring(1) are higher than in Ring'(1) for both population sizes, see Table 5 and Figure 4. This shows that as the ring runs longer the genotypes are starting to converge.

Taking into account the Lipi–Ring methods with $r=1$ and $r=2$ that performed the same number of training epochs, i.e., Ring(1) and Ring(2), the first one shows higher $L_2$ distances (darker colors in Figure 4). This confirms that the populations are more diverse when signal propagation is slower.

Comparing Ring(2) and Lipizzaner, that have the same sub-population size and migration to four neighbors, there is not a clear trend because Lipizzaner generated more diverse populations for population size 9 and Ring(2) for population size 16. Therefore, we have not found a clear effect of changing the migration from four to two directions.

According to these results, we can answer the questions formulated in RQ3: How does diversity change over time in a ring topology? When Ring(1) performs more training the diversity decreases. How does diversity compare with a ring topology with different neighborhood radius? As the radius increases, the diversity is lower because the propagation of the best individuals is faster.

5.4 CelebA and COVID-19

According to the empirical results, in general, Ring(1) and Lipizzaner compute generative models with comparable quality for MNIST (training MLP networks). Here, we study both methods training generative models for generating synthesized samples of CelebA and COVID-19 datasets using convolutional DNN, which are bigger networks (having more parameters). Table 6 shows the best IS value from each of the 10 independent runs for each method with population size 9.

In this case, it is more clear that both methods demonstrated the same performance. Focusing on CelebA dataset, Ring(1) computes higher (better) mean, minimum, and maximum IS, but Lipizzaner shows higher median. For the COVID-19 dataset, Lipizzaner shows better mean, median, and maximum IS values. The statistical analysis (ANOVA test) corroborates that there is not significant differences between both methods for both datasets (i.e., CelebA p-value=3.049 and COVID-19 p-value=0.731).

Table 6: IS results in terms of best mean, standard deviation, median, Iqr, minimum, and maximum for CelebA and COVID-19 experiments. Best values in bold. (High IS indicates better samples)

| Method   | Mean±Std | Median | Iqr  | Min  | Max  |
|----------|----------|--------|------|------|------|
| Dataset: CelebA | | | | | |
| Ring(1)  | 36.61±1.95 | 35.94  | 2.31 | 34.40| 40.40 |
| Lipizzaner| 36.30±1.77  | 36.34  | 2.86 | 32.95| 39.02 |
| Dataset: COVID-19 | | | | | |
| Ring(1)  | 1.79±0.12  | 1.79   | 0.20 | 1.63 | 2.00 |
| Lipizzaner| 1.86±0.19  | 1.84   | 0.16 | 1.61 | 2.24 |

Finally, Figure 5 illustrates some samples of CelebA and COVID-19 synthesized by the generators trained using Ring(1) and Lipizzaner. As it can be seen, the samples show similar quality.
We know that training with smaller sub-populations sizes takes running times as the population size increases. bigger sub-populations (5 instead of 3 individuals) affects less the spectively. This indicates that the computation effort of using 40% longer times than Ring(1) could generate some discrepancies. For population sizes 9, 16, and 25, respectively. This indicates that the computation effort of using 40% longer times than Ring(1) (3 individuals per cell) to analyze the effect of reducing the high segregation ratio of training epochs (see Section 4) and provide comparable quality results. In terms of diversity, Ring(1) shows the most diverse populations which diminishes with more training. Focusing on ring topology, when the migration radius increases, i.e., Ring(1) (r=1) vs. Ring(2) (r=2), the diversity decreases. Finally, we have not found a marked difference on the diversity of the populations when changing the migration directionality, i.e., comparing between Ring(2) and Lipizzaner.

Table 7: Computational time cost in terms of mean and standard deviation (minutes).

| Dataset   | Population size | Ring(1) | Lipizzaner |
|-----------|-----------------|---------|------------|
| MNIST     | 9               | 65.83±1.02 | 72.95±0.22 |
|           | 16              | 72.95±0.22 | 91.19±1.80 |
|           | 25              | 91.19±1.80 | 105.54±1.98 |
| CelebA    | 9               | 224.37±2.36 | 118.87±0.95 |
| COVID-19  | 9               | 276.47±2.36 | 168.57±0.32 |

For CelebA and COVID-19 experiments, Ring(1) reduces the mean computational time by 23.22% in CelebA experiments and by 41.18% in COVID-19. The time saving is higher for COVID-19 mainly because the training methods performed more epochs (1,000) for this dataset than for CelebA (20 epochs). The sub-population size principally affects the required effort to perform the evaluation of the whole sub-population and the selection/replacement operation which are carried out for each training iteration. This explains why the time saving are higher for COVID-19 experiments.

6 CONCLUSIONS AND FUTURE WORK

The empirical analysis of different spatially distributed CEA GAN training methods shows that the use of a ring topology instead of a 2D grid does not lead to a loss of quality in the computed generative models, but it may improve them (it depends on the setup).

Ring(r), which uses a ring topology with neighborhood radius \( r=1 \) and run for the same time than Lipizzaner, produced the best generative models. Ring(1), Ring(2), and Lipizzaner, which were trained for the same training epochs, trained comparable generative models on the MNIST, CelebA, and COVID-19 datasets (similar FID and TVD for MNIST and IS for CelebA and COVID-19).

In terms of diversity, Ring(1) shows the most diverse populations which diminishes with more training. Focusing on ring topology, when the migration radius increases, i.e., Ring(1) (r=1) vs. Ring(2) (r=2), the diversity decreases. Finally, we have not found a marked difference on the diversity of the populations when changing the migration directionality, i.e., comparing between Ring(2) and Lipizzaner.

Ring(1), changing the signal propagation from four to two directions and using a migration of radius one, reduced the computational time cost of Lipizzaner by between 14.2% and 41.2%, while keeping comparable quality results.

Future work will include the evaluation of Ring(1) on more datasets, bigger populations, and for longer training epochs. We will apply specific strategies to deal with small sub-populations (3 individuals per cell) to analyze the effect of reducing the high selection pressure. We will perform a convergence analysis to provide appropriate Lipizzaner and Ring(1) setups to address MNIST. Finally, we are exploring new techniques to evolve the network architectures during the CEA training.

REFERENCES

[1] Abdullah Al-Dujaili, Tom Schmiedeblecher, Erik Henberg, and Una-May O’Reilly. 2018. Towards distributed coevolutionary GANs. In AAAI 2018 Fall Symposium.
[2] Martin Arjovsky and Léon Bottou. 2017. Towards Principled Methods for Training Generative Adversarial Networks. arXiv e-prints, art. arXiv preprint arXiv:1701.04862 (2017).
