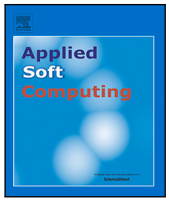
Applied Soft Computing Journal 148 (2023) 110890

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc



Semi-supervised generative adversarial networks with spatial coevolution for enhanced image generation and classification

Jamal Toutouh [a,b,*], Subhash Nalluru [b], Erik Hemberg [b], Una-May O'Reilly [b]

[a] ITIS Software, University of Malaga, Málaga, 29071, Málaga, Spain
[b] MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, 02139, MA, USA

ARTICLE INFO

Keywords:
Generative adversarial network
Semi supervised learning
Coevolution
Spatial distribution

ABSTRACT

Labeling images for classification can be expensive. Semi-Supervised Learning (SSL) Generative Adversarial Network (GAN) methods train good classifiers with a few labeled images. However, authors generally do not train SSL-GAN generators to produce new high-quality images, but as a component to train the classifier. In this article, we use a coevolutionary algorithm (CoEA) with SSL-GANs to train both the classifier and the image generative model using a few labeled images. A CoEA introduces diversity into the GAN training and mitigates training pathologies. We use a two-dimensional grid of GANs to inject diversity via distributed training that exchanges GAN components between neighboring cells based on performance and population-based hyperparameter tuning. In addition, we identify simple and efficient SSL-GAN architectures. We demonstrate the utility on three separate benchmark datasets, achieving good classification accuracy and high image quality generation while using fewer labeled data exemplars. The image quality and classification accuracy are also competitive with State-of-the-Art methods.

Code metadata

Permanent link to reproducible Capsule: https://doi.org/10.24433/CO.4207221.v1.

1. Introduction

Labeling image data is often both expensive and time-consuming. In Machine Learning (ML) data is categorized as labeled (supervised) or unlabeled (unsupervised). Semi-Supervised Learning (SSL) ML models are trained with datasets that contain both some labeled data and unlabeled data [1], thus reducing the need for labeled image data. Furthermore, Generative Adversarial Networks (GANs) can be used for SSL [2,3]. A GAN estimates the distribution from which real data samples are obtained [4]. The GAN consists of two artificial neural networks (ANNs), a generator that generates data samples from a latent space input and a discriminator which discriminates between generated samples ("fake") and samples from a dataset ("real"). The objective and loss functions are adversarially defined to train the generator to generate a sample that cannot be discriminated from a real sample by the discriminator. GANs produce realistic data samples due to the adversarial interactions [4] and can use few samples due to their generative nature. Numerous applications use GANs to produce synthesized images, e.g. 3D object generation [5], image-to-image translation [6], multispectral and panchromatic images fusion [7], and medical image generation [8], and other types of data samples [9–11].

GANs can be quite difficult to train with gradient based methods, the adversarial loss term creates training pathologies, e.g. mode collapse [12], discriminator collapse [13], and vanishing gradients [14]. There are some black-box ML algorithms that use evolutionary computation [15] to overcome this when training GANs [16,17]. For GANs and SSL others have applied different loss functions and complex GAN architectures [2,18–20], see Section 3. In general, SSL-GAN methods produce good classifiers and unsupervised GAN methods train high-quality image generators. We are interested in investigating if it is possible to, at the same time, train both a good generator and classifier. This contrasts with related works that either train a good classifier with a given method [18,20,21] or train a good generator with another method [16,17].

The research questions this paper addresses are:

• **RQ-1**: Can a simple GAN composition and architecture be used for SSL?

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author at: MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, 02139, MA, USA.
E-mail addresses: jamal@uma.es (J. Toutouh), hembergerik@csail.mit.edu (E. Hemberg), unamay@csail.mit.edu (U.-M. O'Reilly).
URL: http://www.jamal.es (J. Toutouh).

https://doi.org/10.1016/j.asoc.2023.110890
Received 8 February 2023; Received in revised form 20 September 2023; Accepted 26 September 2023
Available online 30 September 2023

• **RQ-2**: Can an evolutionary GAN training method find the parameters of both a high-accuracy classifier and a high-quality generative model?

• **RQ-3**: Can higher quality samples be generated from GANs with some labels (SSL) compared to no labels (unsupervised)?

First, we investigate variants of simple SSL-GAN architectures for both accuracy and image generation quality in Section 4. Then we combine SSL-GANs with spatial coevolution and gradient-based learning to improve the GAN training. We extend Lipizzaner [22], which combines spatial coevolution with gradient-based learning to improve the robustness of GAN training, see Section 5. Lipizzaner has many features of evolutionary computation, such as training parameter diversity, spatial communication, local (sub-population) selection, plus the evolution of generator ensemble weights and the learning rate under stochastic gradient descent. Lipizzaner results are diverse and can recover from mode collapses [23–25]. We add SSL with majority voting classification, spatial diversity loss and balanced label batch sampling to create Lipi-SSL.

We describe the setup of empirical experiments in for our method called Lipi-SSL Section 6. The experiments for Lipi-SSL are on the MNIST [26] dataset and the more complex CIFAR-10 [27] and SVHN [28] datasets. The experiments show that Lipi-SSL can achieve both good accuracy and generated image quality, also in comparison to State-of-the-Art (SoA) methods, see Section 7. We also provide a discussion regarding the results, the configurations for Lipi-SSL training and how it relates to previous work, as well as limitations of our study in Section 8. Our contributions are:

• **Simple SSL GAN architectures and training for both classification an generation performance**: We identify simple GAN architectures for semi-supervised learning (SSL-GANs). In addition, we provide a method, Lipi-SSL, to train parameters for both classification accuracy and image classification with majority voting classification, spatial diversity loss and balanced label batch sampling. We provide ablation studies to empirically show the importance of the new components in Lipi-SSL.

• **Improved classification accuracies with few labels**: We achieve accuracies around 96% using only 100 (0.167%) labeled data points for training out of the 60 000 data points in the MNIST training data. We observe higher accuracies with an increase in labels. Similar trend is shown for CIFAR-10 and SVHN datasets.

• **Improved generative models**: The quality of the generated images is quantified by either the Fréchet Inception Distance (FID) [29] or an Inception Score (IS) [21]. The reported scores for images generated by the Lipi-SSL are competitive. The comparison shows that Lipi-SSL generated images are visually comparable (or better) to the images produced by SoA (SoA methods do not provide quantitative metrics).

The article structure is the following. Sections 2 and 3 present main background and related work. Section 4 introduces and evaluates SSL-GAN architectures. Section 5 describes distributed spatial adversarial evolutionary SSL-GAN training with Lipi-SSL. Section 6 presents the experimental setup for training both generator and classifier with Lipi-SSL. Section 7 discusses the results and analysis of the experiments with Lipi-SSL. Section 8 comments the overall results and limitations of Lipi-SSL. Finally, Section 9 draws the main conclusions and future work.

2. Background

This section introduces the concepts of unsupervised and semi-supervised GANs, SSL-GAN training, and details for the dataset and image quality measures used in the experiments.

2.1. Semi-supervised Generative Adversarial Networks

A GAN consists of two neural networks, the generator and the discriminator [30], see Fig. 1(a). The generator learns the features of samples obtained from a "real" data distribution and mimics them to generate "fake" data. The discriminator learns the features that differentiate a real image from a fake image.

The generator produces images from a random vector z drawn from a *latent space*, usually defined as a normal distribution, $\mathcal{N}(0,1)$. The discriminator observes real images from the training dataset and fake images from the generator, and it tries to discriminate real from fake images. The generator uses the feedback from the discriminator to update its parameters and generates images from latent space vectors. Fig. 1(a) summarizes GAN training. Often the latent space is smaller than the space of the real images dataset. The dimensionality reduction tries to group related data and maintains a minimal amount of information to reconstruct an image and reduce noise.

Formally, the goal of GAN training is to learn two sets of parameters, u and v, that approximate and discriminate the real data distribution, G_* (or p_{data}). The generator is obtained from a class of generators $\mathcal{G}$, each characterized by the variables $u \in \mathcal{U}$, the discriminator is obtained from a class of discriminators D, each characterized by a variable $v \in \mathcal{V}$; where $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^p$ are the parameters space of the generators and discriminators, respectively.

GANs apply adversarial learning by optimizing their parameters according to a minmax optimization problem defined in Eq. (1), which considers the loss function presented in Eq. (2).

$$\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} \mathcal{L}(u, v) \tag{1}$$

$$\mathcal{L}(u, v) = \mathbb{E}_{x \sim G_*}[\phi(D_v(x))] + \mathbb{E}_{x \sim G_u}[\phi(1 - D_v(x))] \tag{2}$$

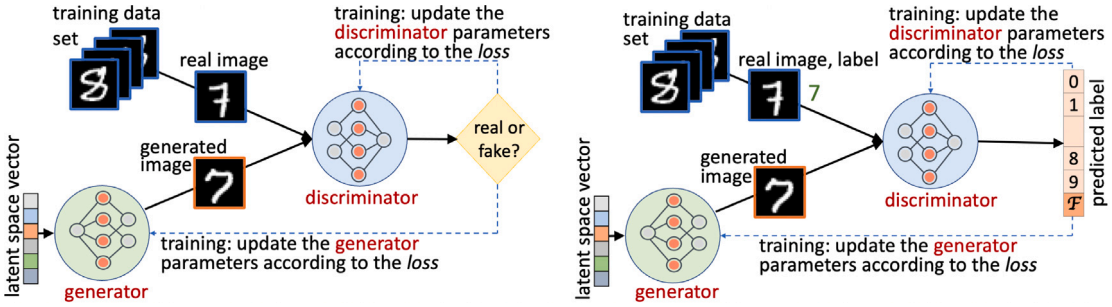
Function $\phi : [0,1] \to \mathbb{R}$ is a concave function, commonly referred to as the *measuring function*. The left operand is the probability that when x is taken from the real data distribution (G_*), it is classified as real. The right operand identifies the data point taken from the generator's fake data distribution (G_u) as fake. $D_v(\dots)$ corresponds to the probability that the data point is real, which is why the $1 - D_v(x)$ is used in the generator's loss component. Thus, GAN training is applied to find the parameters that make the generator maximize the probability that the discriminator classifies fake images as real ones. This learning approach is known as (fully) unsupervised GAN because the training dataset does not consider labeled classes.

This work focuses on semi-supervised GANs (SSL-GANs), which are applied to partially labeled datasets. The idea is to simultaneously train the generator to produce images (as in the traditional unsupervised GANs) and also the discriminator as a classifier (to label images from the dataset), see Fig. 1(b). In unsupervised GANs, the discriminator outputs either a real or fake label. The discriminator of an SSL-GAN has $K+1$ possible labels when it is trained on a dataset with K classes. The first K labels are associated with the real image classes, and the $(K+1)$th one is the fake label. Thus, the discriminator is penalized for incorrectly predicting a real or fake image with the wrong label.

2.2. SSL-GAN training enhancements

GANs are challenging to train [31–33], so enhancements have been used to improve the SSL-GAN convergence and training stability.

Instance Noise (IN). An issue that arises when training GANs is that the real and fake data distributions are concentrated with non-overlapping support. Here, support refers to the closure of the set of values for a random variable corresponding to a distribution. When the supports of each distribution do not overlap the discriminators always determine whether or not the data is from a fake distribution. This makes it difficult for the generator to improve. Snderby et al. proposed the Instance Noise [34], adding noise to data samples obtained from each distribution. This expands their support and increases the likelihood of overlap between the distributions. This overlap can weaken discriminators and thus improve generator performance.



(a) GAN (b) SSL-GAN

Fig. 1. Example of GAN (1(a)) and SSL-GAN (1(b)) for MNIST image generation. $\mathcal{F}$ is the fake image class.

Feature Matching (FM). In GANs, Feature Matching is a regularizing objective for the generator to prevent overtraining on the current discriminator [21]. Instead of directly maximizing the output of the discriminator, the objective of the generator is to generate data that matches the statistics of the real data distribution. Thus, the discriminator specifies the statistics to match. Specifically, the generator is trained to match the expected value of the features on an intermediate layer of the discriminator. Simultaneously, the discriminator learns the most distinctive features of real data versus data produced by the current generative model.

Dropout and Batch Normalization. Dropout between layers of the discriminator architecture is used to weaken the discriminator [35]. The dropout forces the discriminator to rely more on the features that distinguishes between each class, which also improves generalization. Furthermore, batch normalization standardizes the inputs of each layer to improve the stability of the training process [36].

One-Sided Label Smoothing (OS-LS). Salimans et al. proposed the idea of One-Sided Label Smoothing, or Positive Label Smoothing [21]. For example, if the discriminator generally considers two labels: 0 to classify fake data and 1 to identify real data samples, the idea behind OS-LS is to smooth real data labels from 1 to 0.9. Thus, the generalization of the discriminator can improve. Nevertheless, label smoothing is not applied to generated data since it prevents the computation of relevant gradients from which the generator learns how to make its samples more realistic.

2.3. Image datasets and quality metrics

The empirical experiments used to evaluate the Lipi-SSL considers three standard benchmark image datasets:

MNIST (Modified *National Institute of Standards and Technology*) [26] is an image dataset of handwritten digits (0 to 9) with a training set of 60 000 images and a test set of 10 000 images. Each image is 28 × 28 gray-scale pixels.

CIFAR-10 (Canadian Institute For Advanced Research 10 classes) [27] is a dataset of 50 000 training images and 10 000 test images of 32 × 32 color images of 10 object classes. The classes are airplanes, birds, cars, cats, deer, dogs, frogs, horses, ships, and trucks. CIFAR-10 is uniformly balanced.

SVHN (Street View House Numbers) [28] dataset comprises color images that are similar to MNIST, featuring small cropped digits. It contains 73 257 images for training, 26 032 images for testing, and an additional 531 131 images.

The generative models are evaluated according to the quality and diversity of the images produced. We use Inception Score (IS), Fréchet Inception Distance (FID) and Kernel Inception Distance (KID) as image quality metrics. The diversity of the generated images is explicitly evaluated in terms of the Total Variation Distance (TVD).

IS was proposed by Saliman et al. [21] to objectively evaluate the quality of the generated images. IS is a summary of the predictions obtained from a pretrained classifier, most use the Inception V3 model [37]. The predictions of the inception model are used to obtain the conditional label distribution denoted as $p(y|x)$. IS is calculated as the divergence between the conditional and marginal probabilities distributions (see Eq. (3)). Here, $KL(p \parallel q)$ refers to the Kullback–Leibler divergence.

$$IS = e^{\mathbb{E}_x KL(p(y|x)\|p(y))} \tag{3}$$

FID calculates the distance between feature vectors calculated for real (x) and generated images (g) [29]. It uses an intermediate layer of an inception neural network to extract defining features of an image, and it models these features as a Gaussian distribution with mean μ and covariance Σ (see Eq. (4)). The $Diag$ function sums up the elements along the diagonal of a matrix. Note that the lower the FID score value, the better the image quality.

$$FID(x, g) = \|\mu_x - \mu_g\|_2^2 + Diag(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{1/2}) \tag{4}$$

KID is a metric proposed to overcome size dependency and biases in FID and IS limitations. KID is defined as the squared maximum discrepancy (MMD) between Inception representations [38]. Given samples $X = \{x_i\}_{i=1}^m$ drawn i.i.d. from distribution $\mathbb{P}$, and $Y = \{y_j\}_{j=1}^n$ drawn i.i.d. from distribution $\mathbb{Q}$.

$$\begin{aligned} \mathrm{KID}(X, Y) = &\frac{1}{m(m-1)} \sum_{i \neq j}^{m} k(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i \neq j}^{n} k(x_i, x_j) \\ &- \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} k(x_i, x_j) \end{aligned} \tag{5}$$

A polynomial kernel, $k(x, y) = (\frac{1}{d} x^T y + 1)^3$, is used, where d is the representation dimension. KID does not assume a parametric form for the distribution of activations. The cubic kernel of the KID compares skewness, mean and variance. Finally, the KID has a simple unbiased estimator. Note that the lower the KID score value, the better the image quality.

Finally, TVD captures the diversity of the generated images. An inception network is used to obtain the frequency of appearance of each class from a batch of real images (x). This frequency is compared against the frequency of each class from a batch of generated images (g). The computed TVD score is proportional to the sum of the absolute difference of frequencies for each class (see Eq. (6)). The f_c term denotes the frequency of labels predicted by the inception network in a given class.

$$TVD(x, g) = k \sum_{c \in classes} |f_{c_x} - f_{c_g}| \tag{6}$$

3. Related work

This section presents related work on Semi-supervised GANs, Evolutionary GANs, and Distributed Evolutionary GAN training. In turn, it identifies the benefits and limitations of some techniques implemented by state-of-the-art (SoA) approaches to train GANs for semi-supervised learning.

Semi-supervised GANs. Semi-Supervised learning (SSL) aims to reduce the amount of labeled data needed for accurate inference. In SSL, models are trained with datasets containing some labeled data, and the rest is unlabeled. SSL-GANs simultaneously train the generator to produce images (unsupervised), and both discriminate and classify the generated images (supervised).

Numerous methods have been employed in SSL to improve performance. For example, Salimans et al. recommend using the Feature Matching loss (FM) for the generator rather than maximizing the modified generator loss [21]. Feature matching performs well theoretically and yields good classification results in practice. The quality of images produced is quite low and can easily be distinguished as fake by humans. In practice, the FM loss is calculated using the output of the penultimate layer of the discriminator and with the mean of the L_1 norm of the real and fake data moments. Another example of SSL-GAN specific loss is triplet loss (TL), in which the input is structured as triplets of a query, a positive example, and a negative example. The triplet loss aims to increase the distance between the query and a negative example while decreasing the distance between the query and a positive example. Applying TL, the model performs much better in classification tasks. Zieba et al. [18] used the triplet loss as one of the discriminator's loss components, resulting in improved classification performance. Finally, Denton et al. introduce Context-Conditional GAN (CC-GAN) [2], which takes a different approach with semi-supervised learning using in-painting. They use weights for the supervised and unsupervised loss components.

GAN training has pathologies, e.g., mode collapse [12], and these also occur for SSL GANs. For example, Salimans et al. propose the idea of Minibatch Discrimination (MB D.) [21] to work around this issue where the discriminator looks at multiple points simultaneously, and the correlation of input samples is provided as input in the intermediate layers.

Another common issue faced by SSL-GANs is that they are limited by the amount and nature of labeled data used. Li et al. proposed the Unified GAN (UGAN) [19] that uses "good" and "bad" data samples (adversarial examples) to train the classifier of a GAN, making it less susceptible to such variations in the labeled data. A UGAN uses four networks — two generators, one classifier, and one discriminator. One generator generates unlabeled data, and the other generates labeled data. Another example of an SSL-GAN method that employs more components is MarginGAN [20], which was primarily designed to achieve very high classification performance on a dataset while using very minimal amounts of labeled data. The idea for MarginGANs was motivated by the success of large margin classifiers such as Support Vector Machines (SVMs) [39]. In addition to a generator and discriminator, a MarginGAN contains a third component, a classifier.

Evolutionary GANs. Evolutionary computation (EC) is a broad class of population-based optimization algorithms modeled on genetic evolution. EC defines an input (genome) space for individual solutions in a population that are subject: in each iteration (generation), the top individuals are selected from the population and varied, then the new individuals are evaluated and assigned a score (fitness) by an objective function (fitness function). EC has been used to address deep learning problems, e.g., optimizing the deep neural network (DNN) parameters through neuroevolution and evolving DNN architectures [40–43].

Wang et al. proposed Evolutionary GAN (EGAN) [16]. In each epoch of the training, an EGAN uses multiple loss functions, each associated with a generator, and it evolves a population of generators against a single discriminator by selecting the best-performing generator for the next epoch. EGAN introduces diversity through a mutation operator, randomly applying a loss function picked from a set of different loss functions.

An evolutionary method to train GANs is the COEGAN approach, which evolves the neural network architectures for the generator and the discriminator [17,44]. Then coevolution is used to train the adversarial components of GANs. Each neural network architecture is evaluated against a sample of k. The best GAN pair performs a gradient update step.

Evolved GAN applies a genetic algorithm (GA) to evolve the architectures of the generators and discriminators [45]. A cooperative coevolutionary algorithm has been proposed to conduct adversarial multi-objective optimization [46]. The minmax optimization problem is decomposed into two sub-problems (generation and discrimination). Each problem is solved by separated populations of generators and discriminators that evolve by their own evolutionary algorithm (EA).

Distributed Evolutionary GAN Training. Coevolutionary algorithms (CoEA) generally refer to environments where agents interact and are trained with different objective functions. Competitive CoEAs have adversarial populations that simultaneously evolve solutions against each other with individuals engaging in two-player games [47]. These algorithms use fitness functions that rate a solution relative to its adversaries and can sometimes be described as a minimax optimization.

A spatial topology (i.e., two-dimensional toroidal grid) can efficiently control the mixing of adversarial populations in CoEAs [25]. Theoretical studies and empirical results showed that the spatially distributed competitive coevolutionary GAN training mitigates non-convergence pathologies [23,25,48]. The research presented in this article focuses on Lipizzaner, which locates the individuals of the generator and discriminator populations on each cell of a toroidal grid (i.e., each cell contains a generator–discriminator pair) [22,23].

Several studies have been published on Lipizzaner. For example, Mustangs, a variation of Lipizzaner based on EGANs, randomly selects a loss function to train each cell for each generation to increase diversity in the populations [49]. The spatial distribution of the cells allows data decomposition to train GANs in each cell with different subsets of data, which fosters diversity across the grid [50]. Lipizzaner returns a generative model that consists of an ensemble of generators defined by the best sub-population of generators. EAs are used to learn this ensemble [51]. Lipizzaner and its variations have shown competitive results on standard benchmarks.

Distributed GAN training represents a high-dimensional optimization problem with high computational requirements. Parallel/distributed implementation of algorithms improves their scalability. Lipizzaner maximizes the scalability by running the sub-population training on independent computational resources, CPUs or GPUs, applying asynchronous parallelism [23,52].

Related work summary. In this paper, we take advantage of distributed evolutionary GAN training for SSL to use a simple GAN architecture with diverse loss weights to improve classification performance with limited impact on data generation quality. Table 1 shows an overview of related work based on GAN architecture, such as NN loss, complexity, population (pop.), and training objectives of discrimination and generation. The comparison of loss is only applied when it is declared as a main contribution of the method, otherwise, we treat it as not applicable. Our approach Lipi-SSL is the only evolutionary GAN approach that focuses on both image classification and data generation with simple GAN architecture regarding loss functions and DNNs. Previous research studies focus on only one training objective and/or have complex architectures.

Table 1
Overview of related work discussed. It presents the main GAN training features: loss applied (Loss), models trained (DNNs), and populations defined (Pop.); as well as, the training objectives of classification and generation. Notation is Generator(G), Discriminator(D), Classifier(C), and not applicable (NA).

Name	GAN training			Training objectives	
	Loss	DNNs	Pop.	Classification	Generation
FM-GAN [21]	FM	G, D	No	Yes	No
MB D [21]	NA	G, D	No	No	Yes
MarginGAN [20]	NA	G, D, C	No	Yes	No
UGAN [19]	NA	2 G, D, C	No	Yes	Yes
SSL-GAN-TL [18]	TL	G, D	No	Yes	No
EGAN [16]	Multiple	G, D	G	No	Yes
COEGAN [17]	NA	G, D	G, D	No	Yes
Lipizzaner [22,23]	NA	G, D	G, D	No	Yes
Lipi-SSL	Softmax	G, D	G, D	Yes	Yes

4. SSL GAN architecture and training

This section introduces and evaluates different architectures and training for stand-alone SSL-GAN (from now on SSL-GAN) aimed at training both a high-accuracy classifier (discriminator) and a high-quality generative model. The proposed spatially distributed SSL GAN method uses an SSL-GAN variant. First, we study three approaches with different loss functions and architectures. Second, we propose several enhancements to the GAN training to improve convergence and stability. Finally, experimental evaluation is conducted to determine a deep neural network architecture for SSL.

4.1. SSL-GAN architectures and loss functions

SSL-GAN is built on a standard unsupervised GAN by providing another task to the discriminator, classifying the real labeled images it receives as input. As a result, the loss of a SSL-GAN discriminator requires a new supervised loss term.

The generator's loss is unchanged from a standard GAN (see Eq. (7)). While the discriminator's loss in Eq. (8) is defined according to an unsupervised component for fake and unlabeled real images ($\mathcal{L}_{Du}$), and a supervised component for labeled real images ($\mathcal{L}_{Ds}$). The pseudocode in Algorithm 1 summarizes the SSL-GAN training in the SSL-GAN variants evaluated. Lines from 4 to 7 illustrate the steps used to train the discriminator d. The computation of $\mathcal{L}_{Ds}$ using real and fake images depends on the SSL-GAN variant used and is described in the following sub-sections. Lines from 8 to 11 present the steps to train the generator g, which are the same ones performed in unsupervised learning.

$$\mathcal{L}_G = \mathbb{E}_{z\sim p_u(u)}[\phi(1 - D_v(G(z)))] \tag{7}$$

$$\mathcal{L}_D = \mathcal{L}_{Du} + \mathcal{L}_{Ds} \tag{8}$$

Algorithm 1 SSL-GAN training
Input: T: Epochs, θ_D : Dataset, B_s : Batch size, λ : Learning rate
Return: g, d : Trained generator and discriminator

1: $g, d \leftarrow$ initializeNetworks() ▷ Initialize generator and discriminator
2: **for** t **do** $\in [0, \dots, T]$ ▷ Loop over the T training epochs
3: **for** $B \in \theta_D$ **do** ▷ Loop over the batches in θ_D
4: $\mathbf{z} \leftarrow$ getRandomNoise(B_s) ▷ Get random generator input
5: $X_f \leftarrow$ generateImages($g, \mathbf{z}$) ▷ Compute $G(z)$ to generate images
6: $\mathcal{L}_D \leftarrow$ getDiscriminatorLoss(d, B, X_f) ▷ Compute $\mathcal{L}_D$ Eq. (8)
7: $d \leftarrow$ gradientDescent($d, \mathcal{L}_D, \lambda$) ▷ Update parameters of d
8: $\mathbf{z} \leftarrow$ getRandomNoise(B_s) ▷ Get random generator input
9: $X_f \leftarrow$ generateImages($g, \mathbf{z}$) ▷ Compute $G(z)$ to generate images
10: $\mathcal{L}_G \leftarrow$ generatorLoss(g, X_f) ▷ Compute $\mathcal{L}_G$ Eq. (7)
11: $g \leftarrow$ gradientDescent($g, \mathcal{L}_G, \lambda$) ▷ Update parameters of g
12: **end for**
13: **end for**
14: **return** g, d ▷ Return trained g and d

The investigated SSL-GAN variants are:

- Softmax SSL-GAN (SX SSL-GAN), in which the discriminator computes the loss from the output layer with a softmax activation function;
- Sigmoid-softmax SSL-GAN (SS SSL-GAN), in which the discriminator computes the two components of the loss from the output layer with a softmax activation and a sigmoid activation functions; and
- Hybrid SSL-GAN (H SSL-GAN), in which the discriminator combines the two previous approaches.

Softmax SSL-GAN (SX SSL-GAN). In SSL, images from the real distribution are classified into K classes. Softmax SSL-GAN aims to train a discriminator to learn to classify real and fake samples and simultaneously distinguish the real samples among the K classes. The classification problem is extended by adding a $(K+1)^{th}$ class to label fake samples. Therefore, the discriminator is trained to label real data from 1 to K (according to the sample class) and fake data as K+1.

When the discriminator is trained with real labeled data, i.e., $(x, y) \sim G_*$, it computes the supervised loss aimed at identifying the correct label for the data as a general supervised classifier does. The $\mathcal{L}_{Ds}$ loss function is defined in Eq. (9).

$$\mathcal{L}_{Ds} = \mathbb{E}_{(x,y)\sim G_*(x)}[\phi(p(y|x; y < K+1))] \tag{9}$$

The $\mathcal{L}_{Du}$ loss in Eq. (10) is applied to train a discriminator with fake or unlabeled data. The first component focuses on real data. The term $\mathbb{E}_{x\sim G_*(x)}[\phi(D_v(x))]$ is computed as the sum of the first K outputs of the softmax layer corresponding to the real classes and returns the probability of the sample being real. The second component enforces the discriminator to predict fake images as $K+1$.

$$\mathcal{L}_{Du} = \mathbb{E}_{x\sim G_*(x)}[\phi(D_v(x))] + \mathbb{E}_{z\sim p_u(u)}[\phi(y = K+1|G(z))] \tag{10}$$

Sigmoid-softmax SSL-GAN (SS SSL-GAN). Sigmoid-softmax SSL-GAN (SS SSL-GAN) incorporates an additional sigmoid output for the discriminator, i.e., the discriminator uses two different outputs (see Fig. A.14). Thus, the $\mathcal{L}_{Du}$ loss incorporates a new term to Eq. (10), $\mathbb{E}_{z\sim p_u(u)}[\phi(1 - D_{sig}(G(z)))]$, which allows the discriminator to distinguish between real and fake images. The $\mathcal{L}_{Ds}$ computation is the same.

$$\begin{aligned}\mathcal{L}_{Du} =& \mathbb{E}_{x\sim G_*(x)}[\phi(D_v(x))] + \mathbb{E}_{z\sim p_u(u)}[\phi(y = K+1|G(z))] + \\ & \mathbb{E}_{z\sim p_u(u)}[\phi(1 - D_{sig}(G(z)))]\end{aligned} \tag{11}$$

Hybrid SSL-GAN (H SSL-GAN). One of the primary challenges with semi-supervised learning is to minimize the number of labeled samples used during training. However, when using fewer labeled samples, the difficulty of both image generation and classification increases. To address this challenge, we evaluated a two-phase SSL-GAN training approach named Hybrid SSL-GAN (H SSL-GAN).

In the first phase, the GAN is trained in an unsupervised setting and takes only the output of the sigmoid layer into account when it computes the loss (i.e., discarding the output of the softmax activation). In the second phase, the GAN is trained by taking only the softmax

layer into account for the semi-supervised task (i.e., Softmax SSL-GAN is applied, and therefore, the sigmoid activation is discarded). Thus, the discriminator's architecture is the same as for Sigmoid-softmax SSL-GAN because it requires both output layers, sigmoid and softmax (see Fig. A.14).

4.2. Determining SSL-GAN variant

Our goal is to find a simple SSL GANs training setup to see if it benefits from spatially distributed coevolutionary training. Besides, the coevolutionary approach must simultaneously train good classifiers and generative models. Thus, we study if Lipi-SSL (Section 5) addressing SSL yields competitive results, as demonstrated by Lipizzaner for unsupervised learning [23,24]. According to the preliminary analysis, see Appendix A, a simple SSL-GAN setup that provided a trade off between image classification and generation is SX SSL-GAN without FM. Therefore, Lipi-SSL will use SX SSL-GAN.

5. Method

This section describes Lipi-SSL method by introducing the main properties and presenting the Lipi-SSL distributed GAN training algorithm.[1] Finally, we provide a theoretical analysis in Section 5.3.

5.1. Lipi-SSL properties

The Lipi-SSL method extends Lipizzaner's spatially distributed training for fully unsupervised GANs [23]. Lipi-SSL trains a generative model by co-evolving a population of generators $\mathbf{G} = \{g_1, \dots, g_N\}$ against a population of discriminators $\mathbf{D} = \{d_1, \dots, d_N\}$. The g_i and d_i ANNs (the individuals in each population) are based on the Softmax SSL-GAN ones described in Section 4.1.

The individuals in each population are distributed in each cell of a spatial toroidal grid. Thus, each cell of the grid contains a generator–discriminator pair named *center*. According to overlapping Von Neumann neighborhoods, two sub-populations, one of generators ($\mathbf{G}$) and one of discriminators ($\mathbf{D}$), are defined in each cell. Fig. 2 illustrates a 3×3 toroidal grid and how the overlapped sub-populations are defined in cell 4 and 6 (in this example, the cells 3 and 7 are overlapped). These sub-populations participate in the coEA training phase `Algorithm 4` to update the *center* of the cell. In Lipi-SSL, the generators and discriminators optimize their parameters by applying a semi-supervised learning approach based on the `Algorithm 1`. In turn, the overlapping neighborhoods define the communication among the cells, in which the best individual of each cell sub-population (the ANN parameters) is propagated to the four neighboring cells.

During the training process of Lipi-SSL, each cell learns a generative model defined as an ensemble of its sub-population of generators and mixture weights (computed using evolutionary strategies to optimize the quality of the samples produced). Thus, the generative model returned by Lipi-SSL is the best ensemble of generators according to a given quality metric. The most distinctive new properties of Lipi-SSL over Lipizzaner are: the **majority voting classification**, the **spatial diversity loss** used to train the discriminator, and the **balanced labels batch sampler** used to generate the training batches.

[1] Lipi-SSL source code is publicly available at https://github.com/jamaltoutouh/lipi-ssl/.

Majority voting classification. By taking advantage of sub-populations, Lipi-SSL returns a generative model based on an ensemble of generators and a classifier defined by a sub-population of discriminators that apply a **majority voting classification scheme**. In this scheme, on each batch of images, the classifications of all the discriminators in a sub-population are combined (frequency of labels), and the label with the highest frequency is used as the final classification label. This type of image classification ensemble can improve classification accuracy over a single model [53]. Formally, given a set of m classifiers $C = \{C_1, \dots, C_m\}$ and an input image $\mathbf{X}$, the class label $\hat{y}$ is predicted via majority voting of each classifier C_j as follows

$$C(\mathbf{X}) = \text{mode}\{C_1(\mathbf{X}), \dots, C_m(\mathbf{X})\}$$

where $C_j(\mathbf{x})$ returns the class label predicted by C_j.

Spatial diversity loss. In SSL-GAN, the loss function of the discriminator is defined as an aggregation of the unsupervised loss, $\mathcal{L}_{Du}$, and the supervised loss, $\mathcal{L}_{Ds}$ (see Eq. (8)). Including two weight parameters (i.e., w_u and w_s), the loss function of the discriminator is defined as a weighted sum in which w_u and w_s weigh the importance of $\mathcal{L}_{Du}$ and $\mathcal{L}_{Ds}$, respectively (see Eq. (12)).

$$\mathcal{L}_D = w_u\mathcal{L}_{Du} + w_s\mathcal{L}_{Ds} \quad where \quad w_u + w_s = 1 \tag{12}$$

An SSL-GAN with the weights w_u=1 and w_s=0 is equivalent to an unsupervised GAN (standard GAN). Conversely, an SSL-GAN with the weight w_u=0 and w_s=1 specializes in classifying data. Finally, when $w_u = 0.5$ and $w_s = 0.5$, it is equivalent to the SSL-GAN shown in Eq. (8).

Training the cells of the grid with different loss functions fosters diversity in the populations (i.e., the ANNs parameters), which results in improving the GAN training [23,49]. Based on this evidence, Lipi-SSL applies **spatial diversity loss**, i.e., it uses different w_u and w_s values to train the discriminators in each cell. The different grid cells train the networks with different weights on the generative and classification tasks, which can maintain diversity in the populations. In addition, communication between cells by overlapping neighborhoods enables the best individuals (ANNs), the *center* of the cell, to propagate through the grid and to be trained with different supervised and unsupervised loss weights. The selection and replacement operators in the sub-populations propagate the best individuals during the training process.

To distribute different convex combinations of w_u and w_s through the grid, the id i of the cell is used. Eq. (13) defines the values of the weights w_u and w_s in the cell i, where populations size as N. Fig. 3 illustrates the different values of w_u and w_s applied to two grid layouts 2×2 and 3×3.

$$w_u = \frac{i}{N-1}, \quad w_s = 1 - w_u, \quad i \in [0, N) \tag{13}$$

Balanced labels batch sampler. Another new property of Lipi-SSL is the **balanced labels batch sampler**, which is used when sampling batches of data for SSL. This data loader was motivated by literature on deep learning in data-constrained classification problems, such as label imbalance [54]. The main idea behind this data loader is to produce image data batches for the training considering two properties of the labeled samples used for the supervised loss: 1. the same labeled data points in each batch are used throughout training, and 2. the classes of labeled data points are balanced equally. Preliminary experiments showed that applying the balanced labels batch sampler improved classification accuracy. The labeled data sampler, $(\mathbf{x}, y), \mathbf{x} \in \mathbb{X}, 0 < y \leq K \in \mathbb{N}_{\geq 1}$, is described `Algorithm 2`.

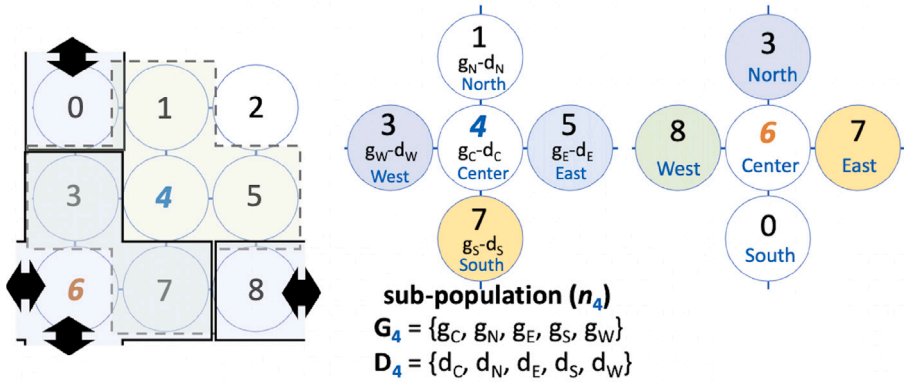


Fig. 2. Example of Lipi-SSL grid and overlapping neighbors. It highlights the von Neumann neighborhoods of cells 4 and 6. The black arrows indicate where the cell's neighbor is, i.e., the neighbors to the West and South of cell 6 are cells 8 and 0, respectively. Cells 4 and 6 have two overlapping cells (cells 3 and 7).

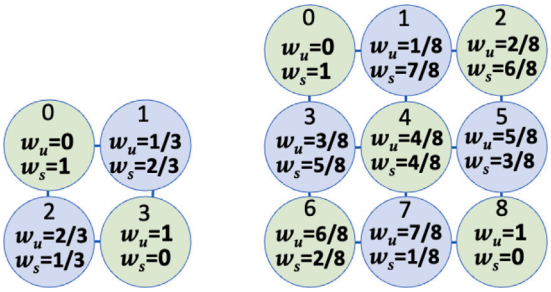


Fig. 3. Layout of a 2×2 (left) and 3×3 (right) grid with different loss weights w_u and w_s in the cells to promote diversity.

Algorithm 2 Balanced labels batch sampler (BLBS)
Input: s: bath size n: number of batches D: labeled data, $(\mathbf{x}, y)$, $|\mathbf{x}| = k$, $D^{m\times k+1}$,
Return: D_B: batches of balanced data, $D_B^{n\times s\times m\times k+1}$

1: $D_B \leftarrow \emptyset$ ▷ Balanced data batches
2: **for** i **do** $\in [0, \dots, n]$ ▷ Iterate over batches
3: $B \leftarrow \emptyset$ ▷ Batch
4: **for** k **do** $\in [0, \dots, K]$ ▷ Iterate over labels
5: $B \cup \{\mathbf{x}, y | (\mathbf{x}, y) \in D, y = k\}$ ▷ Sample s/K with replacement
6: **end for**
7: $D_B \cup B$ ▷ Add batch to batches
8: **end for**
9: **return** D_B ▷ Balanced batches

Algorithm 3 Lipi-SSL main steps
Input: T: Training epochs, E: Cells, s: Neighborhood size, χ_B: Training dataset, χ_R: Ratio of labeled data, θ_{COEV}: CoevSSLGAN parameters `Algorithm 4` θ_{ES}: ES parameters for ensemble creation, λ: Initial learning rate, θ_{GM}: GM parameters
Return: n_g: neighborhood of generators, ω: mixture weights, n_d: neighborhood of discriminators,

1: $\chi_B \leftarrow$ createBatches(χ_D, χ_R) ▷ Create training batches, see `Algorithm 2`
2: **parfor** $k \in E$ ▷ Asynchronous parallel execution of all cells
3: $n, \omega \leftarrow$ initializeCell(k, χ_B) ▷ Initialization of cell
4: **for** t **do** $\in [0, \dots, T]$ ▷ Iterate over epochs
5: $n \leftarrow$ copyNeighbours(k) ▷ Collect neighbor cells
6: $n \leftarrow$ CoevSSLGAN $(n, \chi_B, \theta_{COEV}, \lambda)$ ▷ Coevolve SSL-GANs `Algorithm 4`
7: $\lambda \leftarrow$ updateLearningRate(λ, θ_{GM}) ▷ Update learning rate
8: **end for**
9: $\omega \leftarrow$ generatorEnsembleES(ω, n, θ_{ES}) ▷ Optimize generators ensemble
10: $a_k \leftarrow$ classificationScore(n_d) ▷ Compute the classification accuracy
11: $q_k \leftarrow$ qualityScore(n_g, ω) ▷ Compute the generative quality score
12: **end parfor**
13: **return** $n_d^*, (n_g, \omega)^*$ ▷ Best classifier and generator sub-populations

5.2. Lipi-SSL algorithm details

Based on the main properties of the Lipi-SSL, algorithm we now delve into more detail, see `Algorithm 3`. The Lipi-SSL algorithm starts by creating the set of batches χ_B to train the models by using the balanced labels batch sampler (Line 1) and `Algorithm 2`. Then, it begins the distributed execution of the training on each cell asynchronously in parallel by setting their learning hyper-parameters (Line 3). Next, a loop of T training epochs starts. In each epoch and each cell, three main phases of the coevolutionary training are performed [23]: *sub-population update*, gathering a copy of the ANNs in the neighborhood to build/update the sub-population; *SSL-GAN training*, in which the cell trains a pair generator–discriminator of the sub-population by applying the coEA SSL-GAN training described in `Algorithm 4`; and *learning rate (λ) optimization* by using a Gaussian mutation (GM) operator.

Once the distributed training loop is completed, each cell constructs a generative model by learning an ensemble of generators, computing the mixture weights ω using evolutionary strategies (ES), Line 9. Then, for each sub-population of discriminators, the algorithm computes the classification score that consists in computing the classification accuracy on the test training dataset by using the majority voting prediction. For each ensemble of generators, it gets the quality score by computing a metric for the quality of the generated samples, such as the inception score (IS). Finally, the best sub-population of discriminators n_d^* to create the majority voting classification ensemble, and the best ensemble of generators $(n_g, \omega)^*$ are returned.

For each training epoch, Lipi-SSL applies the competitive coevolution SSL GAN (CoevSSLGAN) (see Line 6 in `Algorithm 3`) training step shown in `Algorithm 4` in each cell in parallel. This training approach extends the unsupervised training defined in Lipizzaner by including the SSL learning steps in `Algorithm 1`. The alternating coevolutionary algorithm method starts with evaluating both sub-populations using an *all-vs-all* scheme. The fitness $\mathcal{L}_{g,d}$ of a given individual is evaluated according to the loss functions, $\mathcal{L}_G$ in the case of the generators and $\mathcal{L}_D$ given w_u and w_s values for the discriminators.

Then, the *best* generator g^* and discriminator d^* are selected (Lines 1 to 4). The adversarial learning process for g^* and d^* applies stochastic gradient descent against randomly selected adversaries from the sub-populations (Lines 5 to 9). Note, the discriminator applies the diversity loss defined in Eq. (12) considering w_u and w_s. Then, the sub-populations are updated by including the trained g^* and d^*. The last step is the replacement, in which the population updates the *center* of the cell by selecting the best generator and discriminator in terms of fitness (loss) values (Lines 12 to 14).

Algorithm 4 CoevSSLGAN cell sub-population training in Lipi-SSL
Input: n: Cell neighborhood, χ_B : Data batches, τ : Tournament size, λ : Learning rate
Return: n : Cell neighborhood of trained sub-population

1: $B_r \leftarrow$ getRandomBatch(χ_B) ▷ Random batch to evaluate GAN pairs
2: $w_u, w_s \leftarrow$ getLossWeights(n) ▷ Loss weights w_u, w_s to compute $\mathcal{L}_D$
3: $\mathcal{L}_{G,D} \leftarrow$ evaluate($\mathbf{G}, \mathbf{D}, w_u, w_s, B_r$) ▷ Evaluate sub-population, i.e., GAN pairs
4: $g^*, d^* \leftarrow$ select(n, τ) ▷ Tournament selection
5: **for** $B \in \chi_B$ **do** ▷ Loop over the batches in χ_B
6: $d \leftarrow$ getRandomOpponent($\mathbf{D}$) ▷ Get random discriminator
7: $g^* \leftarrow$ updateNN(g^*, d, B, λ) ▷ Update g^* with gradient
8: $g \leftarrow$ getRandomOpponent($\mathbf{G}$) ▷ Get uniform random generator
9: $d^* \leftarrow$ updateNN($d^*, g, w_u, w_s, B, \lambda$) ▷ Update d^* with gradient and loss weights
10: **end for**
11: $\mathbf{G}, \mathbf{D} \leftarrow$ updatePopulations($\mathbf{G}, \mathbf{D}, g^*, d^*$) ▷ Add g^* and d^*
12: $\mathcal{L}_{G,D} \leftarrow$ evaluate($\mathbf{G}, \mathbf{D}, w_u, w_s, B_r$) ▷ Evaluate the GAN pairs in the sub-population
13: $n \leftarrow$ replace($n, \mathbf{G}, \mathbf{D}$) ▷ Replace the networks with worst loss
14: $n \leftarrow$ setCenter(n) ▷ Set best generator and discriminator in center
15: **return** n ▷ Return cell neighborhood with trained sub-population

Finally, each run of Lipi-SSL returns two models (Line 13 of `Algorithm 3`): *(A)* a classification model consisting of the ensemble (sub-population) of discriminators that have the best classification score n_d^* (based on majority voting); *(B)* a generative model consisting of the ensemble (sub-population) of generators $(n_g, \omega)^*$ that produces images with the best quality according to a given quality metric, such as FID or IS.

5.3. Theoretical analysis

In this section, we provide a more formal grounding for the optimality of solutions for GAN training with Lipizzaner (see Section 5.3.1), and then, for SSL-GAN training with Lipi-SSL (see Section 5.3.2).

We show that Lipizzaner and Lipi-SSL adhere to the existing GAN theory regarding optimal solutions [4,21]. Here is a sketch for optimal solutions:

• Lipizzaner loss is like a GAN, so existing GAN theory holds. Lipizzaner uses an ensemble, this does not impact the GAN optimality.

• Lipi-SSL loss is like the SSL-GAN loss, so existing SSL-GAN theory still holds. Lipi-SSL uses majority voting, this does not impact GAN optimality.

We expand on these and the algorithmic implementation of Lipi-SSL. Note, we only provide a theoretical analysis based on existing work. This is what most related work does as well [16–18,20,23]. The theoretical analysis is more in depth in only a few [4,19,21].

5.3.1. GAN optimality

From Proposition 1 in [4] we get that $p_g = p_{data}$. Briefly, this is based on the optimal discriminator D^* for any given generator G. Then for a fixed G the optimal discriminator D is

$$D_G^*(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{X}) + p_g(\mathbf{x})} \tag{14}$$

The loss in Lipizzaner is described in Eq. (2). From [4], we see that Lipizzaner has the same optimality as in Eq. (14).

Note, the training performed by Lipizzaner consists in training individually the individuals of the population (i.e., the ANNs update their weights according to the GANs loss function), see Lines 5 to 9 of `Algorithm 4`. When the generative capabilities of Lipizzaner are evaluated, an ensemble of the trained generators (n_g, ω) is used, see Line 13 of `Algorithm 3`. Future work will investigate the theory of the generator ensemble, as in e.g. boosting [55].

5.3.2. SSL-GAN optimality

The optimality for SSL-GAN comes from [21]. Given a standard classifier $C : \mathbb{X} \rightarrow (0,1)^K, C(\mathbf{x}) = [l_1, \dots, l_K], l_i \in (0,1)$. The l term refers to logit $p = \ln \frac{p}{1-p}, p \in (0,1)$. The class probabilities come from applying softmax, $p_C(y = j|\mathbf{x}) = \frac{e^{l_j}}{\sum_{k=1}^{K} e^{l_k}}$.

For SSL-GAN, $p_C(y = K+1|\mathbf{x})$ is added as the probability that $\mathbf{x}$ is fake (as for the discriminator in unsupervised GAN, which is $1 - D(\mathbf{x})$). In SSL-GAN, the discriminator rears from the unlabeled data because it is known that it is real and these data should have a label between 1 and K. Thus, the following expression is used $\max \ln p_C(y \in \{1, \dots, K\}|\mathbf{x})$.

With, arbitrarily, half the data real and half fake the loss function is:

$$\begin{aligned} L &= -\mathbb{E}_{\mathbf{x},y\sim p_{data}(x,y)} \log[p_C(y|\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim G} \log[p_C(y = K+1|\mathbf{x})] \\ &= L_{D_s} + L_{D_u} \end{aligned}$$

where

$$\begin{aligned} L_{D_s} &= -\mathbb{E}_{\mathbf{x},y\sim p_{data}(\mathbf{x},y)} \log p_C(y|\mathbf{x}, y < K+1) \\ L_{D_u} &= -\{\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})} \log 1 - p_C(y = K+1|\mathbf{x}) + \mathbb{E}_{\mathbf{x}\sim G} \log[p_C(y = K+1|\mathbf{x})]\} \\ &= -\{\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{\mathbf{z}\sim noise} \log(1 - DG(\mathbf{z}))\} \end{aligned}$$

The optimal solution for minimizing L_{D_s}, L_{D_u} is

$$e^{l_j(\mathbf{x})} = c(\mathbf{x})p(y = j, \mathbf{x}) \forall j \leq K+1, e^{l_{K+1}(\mathbf{x})} = c(\mathbf{x})p_g(\mathbf{X}) \tag{15}$$

where $c(\mathbf{x})$ is some scaling function

$$L = L_{D_s} + L_{D_u} \tag{16}$$

From Eq. (16), we see that Lipi-SSL has the same optimality as Eqs. (10) and (9). We also observe that Lipi-SSL uses Eq. (12). We see that Lipi-SSL still has the same optimality as Eq. (15).

Note, Lipi-SSL (like Lipizzaner) trains (update the weights of the ANNs) the GANs individually, see Lines 5 to 9 of `Algorithm 4`. When the classification capabilities of Lipi-SSL are evaluated an ensemble of classifiers n_d^* is used (see Line 11 of `Algorithm 3`). Furthermore, as in [21], Lipi-SSL uses the softmax loss (see Section 4.1). Future work will investigate the theory of the classification ensemble, as in e.g. boosting [56].

6. Experimental setup

The experimental analysis empirically assesses the capability of Lipi-SSL training on MNIST, CIFAR-10, and SVHN datasets (see Section 2.3). Our study considers both the classifier model accuracy and the quality of the images produced by the generative model in an SSL GAN. The classification accuracy is calculated by applying the classifier model (sub-population of classifiers) obtained by running Lipi-SSL to the test dataset. The generative model is evaluated according to FID, IS, KID, and TVD (see Section 2.3).

We investigate how to train a GAN for both classification accuracy and image generation quality with low-complexity ANNs architecture. The two first row in Table 2 summarizes the experimental study for how much labeled data to use, the number of datasets investigated, the grid size, m (which determines the population size as m^2), the complexity of GAN, and measures of generated data quality (Ex. means that there is only a qualitative example of a generated image). The other rows in the table show the experimental analysis of the SoA SSL GAN methods used for comparison purposes.

This section describes the main aspects of the experimental setup carried out in this study: the ANNs architectures, the configuration parameter settings, the experimental run time environment.

Table 2
Overview of experimental setup based on the evaluated datasets, grid size (Grid), measures of image quality from the generator (Quality), and complexity (Models trained). Note, Ex. estates for qualitative evaluation by showing generated images, M. means image quality metrics are used, e.g. IS, FID, or TVD. G is Generator, D is Discriminator, and C is classifier.

Name	Evaluated datasets	Grid	Quality	Models trained
Lipi-SSL	MNIST, CIFAR, SVHN	1, 2, 3	M., Ex	G, D
SSL-GAN [3]	MNIST, CIFAR	No	M.	G, D
FM-GAN [21]	MNIST, CIFAR, SVHN	No	Ex	G, D
MarginGAN [20]	MNIST, CIFAR, SVHN	No	Ex	G, D, C
UGAN [19]	MNIST, CIFAR, SVHN	No	Ex	2 G, D
TL SSL-GAN [18]	MNIST, CIFAR, SVHN	No	Ex	G, D

Table 3
Neural networks configuration applied by Lipi-SSL in the experiments.

Parameter	MNIST	CIFAR-10 and SVHN
Input neurons to generator	100	100
Discriminator hidden layers	4	7
Generator hidden layers	4	5
Neurons per hidden layer	16 384 - 131 072	16 384 - 262 144
Network complexity	128	64
Generator output neurons	128 × 28 × 28	64 × 64 × 64
Activation function	*tanh*	*tanh*

Table 4
Configurations for the Lipi-SSL method used in the experiments.

Parameter	MNIST	CIFAR-10	SVHN
Training settings			
Batch size	300, 400, 600, 600, 600	250, 200	366, 366
Label rates	0.01, 0.005, 0.00167, 0.0167, 0.05	0.08, 0.1	0.025, 0.05
Coevolutionary settings			
Stop condition	100 epochs	90 epochs	90 epochs
Tournament size	2	2	
Grid sizes	1×1, 2×2 and 3×3	1×1 and 2×2	2×2
Generator quality metric	FID	IS	IS
Hyperparameter mutation			
Optimizer	Adam	Adam	Adam
Initial discriminator LR	0.001	0.0007	0.0007
Initial generator LR	0.0002	0.0003	0.0003

6.1. Generator and discriminator deep neural network architectures

The preliminary experimental analysis presented in Section 4.2 showed competitive results when addressing SSL with GANs based on MLP. However, the classification accuracy when using 100 labels was lower than 50%, highlighting the need for a slightly more complex and convoluted architecture. Thus, in the experimental analysis of Lipi-SSL, two variations of DCGAN architecture [57], that applies SX SSL-GAN, were used.

We proposed a smaller variant of DCGAN tailored to MNIST images to reduce the model complexity and computational resources. Originally, DCGAN is applied on 64×64 images and the MNIST dataset consists of 28×28 images. Fig. B.15 in Appendix B shows the major details the SSL-DCGAN architecture used in the experimental analysis. The DCGAN architecture is designed to address RGB color images. Therefore, the discriminator and generator have three input and output channels, respectively. As MNIST images are gray-scale, the SSL-DCGAN proposed has one input and output channel in the discriminator and generator.

Another DCGAN architecture variant was devised for the CIFAR-10 and SVHN datasets (see Fig. B.16 in Appendix B). The generator is unchanged from DCGAN and the discriminator has two more layers and a smaller kernel size (three instead of four). The size of the CIFAR-10 and SVHN images is 32×32, so we reshape the input images to 64×64. Note that upsampling the input image can also make it easier for the discriminator to identify which features differentiate classes.

6.2. Evaluation of the trained models

The evaluation of the trained models is performed following standard practice from the GAN literature. The classification accuracy of the obtained classifier models is calculated on the test set of each dataset. Regarding the IS, FID, and KID scores, these metrics are derived by analyzing real images alongside synthetic images produced by the generative models using a pre-trained inception model. The scores are computed by generating 50 000 synthetic images by the evaluated generative model. We use the recommended number of samples reported from standard literature on GANs [58,59]. The computation of IS and KID uses the pre-trained Inception V3 network [60]. Therefore, the images are scaled from 32 × 32 to 299 × 299 (as proposed by Salimans et al. [21]). FID applied on MNIST is computed using a pre-trained CNN inception model specifically designed and trained to learn MNIST dataset features to calculate FID on such dataset.[2]

6.3. Lipi-SSL configuration parameters

The hyperparameter configuration of Lipi-SSL is outlined in Table 3. Due to computational resource limitations, Lipi-SSL was tuned for 3×3. The table outlines the coevolutionary settings and the details of the network architectures. Important training settings include the batch size, the label rate, and the initial learning rates (LR) of the generator

[2] The pre-trained MNIST inception model is included in the repository of the project.

Table 5
SSL-GAN parameter settings applied in Lipi-SSL (IN is instance noise).

Parameter	MNIST	CIFAR-10	SVHN
IN mean	0.0	0.0	0.0
IN initial stdev	0.1	0.7	0.1
IN stdev decay rate	0.0005/iteration	0.0005/iteration	0.0005/iteration
IN stdev decay limit	0.065	None	0.05
Dropout	No	Yes	Yes
Batch normalization	Yes	Yes	Yes
Positive label smoothing value	0.9	Yes	Yes

and discriminator. Note, the label rate was set in order to ensure that there was at least one labeled image per batch.

After setting the parameters, see Tables 3 and 4, a series of empirical parameter tuning experiments were performed to enhance the SSL-GAN training, i.e., IN, FM, dropout, batch normalization, and OS-LS, see Section 2.2. Table 5 presents parameters used for Lipi-SSL on MNIST, CIFAR-10, and SVHN.

6.4. Runtime environment

All the methods used to conduct experiments were written using `Python3` and `PyTorch`.[3] We conducted our MNIST experiments on an HPC platform which uses an IBM Power System AC922 (8335-GTG) with a Power9 architecture with 1TB RAM and 3.6 GHz cores and NVIDIA Tesla V100 GPUs with 32 GB RAM. For the CIFAR-10 and SVHN dataset, we used a Google Cloud instance with 16 virtual CPUs each of which had 64 GB memory and 4 T T4 GPUs with 16 GB memory each.

7. Experimental analysis

This section presents the results and the analyses of the proposed distributed Lipi-SSL training method on the MNIST, CIFAR-10, and SVHN datasets. The classification accuracy and image generation of the trained models, the evolutionary learning process, and the comparison of Lipi-SSL against State-of-the-Art (SoA) methods are reported. We focus our analysis on challenging scenarios, i.e., datasets with few labels. Finally, we perform an ablation study to establish the relevance of the new components added to Lipizzaner to perform SSL. Note, due to computational constraints different number of independent runs were performed for each data set.

7.1. MNIST experimental results

Here we evaluate Lipi-SSL on the MNIST dataset according to the classification accuracy (which is maximized) and generative model quality regarding FID (which is minimized) when training with a few (100) labels for different grid sizes. We analyze the sensitivity of the classification accuracy and FID to the number of labeled samples in the training dataset. It overviews the evolution of the learning process. Finally, we compare the best Lipi-SSL configuration against the published results of SoA methods to address SSL, i.e., UGAN, FM-GAN, TL SSL-GAN, and Margin-GAN. For MNIST, 15 independent runs were performed for each configuration.

7.1.1. MNIST classification accuracy

SSL accuracy with few labels. The key SSL challenge is to train the classifier with few labeled samples, 100 (0.17%) labels. Table 6 summarizes the classification and image generation quality (FID) results by reporting the minimum (Min), median (Med), inter-quartile range (IQR), and maximum (Max) of the distribution of the results for each grid size over 15 runs. The best values for each metric are in bold.

The results in Table 6 show that Lipi-SSL 3×3 provided the classifier with the highest classification accuracy (96.99%). This grid size had the best minimum and median results. In addition, it had the smallest IQR value indicating it had the lowest variation in results. The results were not normally distributed, so we performed a pairwise comparison between Lipi-SSL 3×3 and the other grid sizes by using Wilcoxon statistical tests. This test confirmed that Lipi-SSL 3×3 had the best accuracy results with a statistical significance higher than 99% (i.e., p-value <0.01).

SSL accuracy sensitivity to the number of labels. Table 7 reports the average classification accuracy over 15 runs for the classifiers for SSL-GAN and Lipi-SSL with varying the number of labeled samples used in training. Note accuracy is not reported when using only unsupervised learning (0 labels).

The models that train a single pair generator–discriminator, i.e., SSL-GAN and Lipi-SSL 1×1, i.e. no populations-based GAN training. Table 7 shows that the classification accuracy of the Lipi-SSL 1×1 was higher than the accuracy of the SSL-GAN, which implies that the Lipi-SSL method (see Section 5) can improve the semi-supervised learning. Furthermore, in Table 7 we see that the accuracy is improving with the number of labeled data samples in the training data set, as expected.

For populations of generators and discriminators, Lipi-SSL with grids larger than 1×1, the classification accuracies are higher than 90% when training the model with 100 samples. The highest average classification accuracy is observed for Lipi-SSL 3×3, with 95.96% accuracy when trained with 100 samples. Moreover, results in Table 7 confirm the trend in Table 6, i.e., Lipi-SSL 3×3 trained the best classifiers.

7.1.2. MNIST image generation

FID and TVD scores are used to assess the image quality of the trained generative models. Table 6 summarizes the FID results by reporting the minimum, median, IQR, and maximum of the results distribution for each grid size over 15 runs. Table 7 includes the averaged FID scores when we use 100 labeled images and the whole dataset, respectively. In addition, this table includes the unsupervised GAN training method, Lipizzaner. Further results on FID and TVD scores are shown in Table C.17 in Appendix C.

From Table 6, we see that Lipi-SSL 3×3 models generated the highest quality images based on FID. The results are not normally distributed, so we performed a pairwise comparison between Lipi-SSL 3×3 and the other grid sizes by using Wilcoxon statistical tests. This statistical analysis confirmed that Lipi-SSL 3×3 offered the best FID results with a statistical significance higher than 99% (i.e., p-value <0.01). Fig. 5(f) illustrates samples produced by the generative model trained by Lipi-SSL 3×3 with 100 labels.

Table 7 shows that the generative models improved with more labeled images, i.e., the quality and diversity of the samples improved when using 60 000 labels to train the models. For the single GAN training methods (i.e., SSL-GAN vs. Lipi-SSL 1×1), the generative model trained using Lipi-SSL 1×1 generated samples with better quality (lower FID) and higher diversity (lower TVD) than the generative model trained with SSL-GAN. This confirms that the enhancements applied in simple Lipi-SSL and ANN architecture improve the learning process and provides an answer to **RQ-1**.

[3] Lipi-SSL python source code is publicly available at https://github.com/jamaltoutouh/lipi-ssl/.

Table 6
Descriptive statistics for classification accuracy and FID for experiments with 100 (0.0017) labeled images with Lipi-SSL for 15 independent runs. The best values are in **bold**, accuracy is maximized and FID is minimized.

Model & grid size	Accuracy (%)				FID			
	Min	Med	IQR	Max	Min	Med	IQR	Max
Lipi-SSL 1×1	55.42	85.45	16.87	93.17	5.92	11.89	10.34	29.17
Lipi-SSL 2×2	80.36	91.45	3.25	95.06	4.49	6.68	2.49	7.36
Lipi-SSL 3×3	**90.36**	**96.27**	**0.72**	**96.99**	**3.36**	**4.34**	**2.15**	**7.51**

Table 7
Average (15 runs) accuracy and FID scores from different methods with 100 and 60 000 labeled images when applying semi-supervised learning, and 0 labels when using unsupervised learning. Best values are in **bold**.

Model & grid size	# labels: 0		# labels: 100		# labels: 60 000	
	Acc. (%).	FID	Acc. (%)	FID	Acc. (%)	FID
Lipizzaner 1×1	–	17.96	–	–	–	–
Lipizzaner 3×3	–	3.38	–	–	–	–
SSL-GAN	–	–	49.28	403.39	84.10	73.93
Lipi-SSL 1×1	–	–	86.03	13.77	94.98	8.94
Lipi-SSL 2×2	–	–	91.16	7.02	99.14	5.62
Lipi-SSL 3×3	–	–	**95.96**	**5.22**	**99.38**	**2.72**

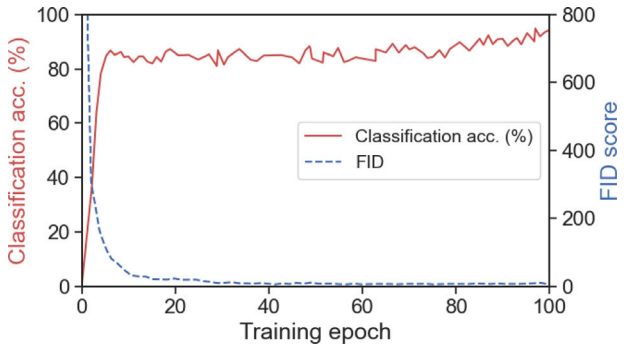


Fig. 4. Evolution of values for classification accuracy and FID for MNIST dataset Lipi-SSL 3×3 training when using 100 labels in the median accuracy run.

When comparing the different Lipi-SSL grid sizes, results improve with grid size (lower FID), as well as observed from previous work [23]. As for classification accuracy, Lipi-SSL 3×3 had the best results. Moreover, for semi-supervised (Lipi-SSL) versus unsupervised (Lipizzaner) learning, lower FID scores were obtained with Lipi-SSL using the whole dataset labeled (i.e., 60 000 labels) for the two grid sizes studied. This provides and answer to **RQ-3**, that higher quality samples be generated when training GANs with SSL (with some labels) compared to unsupervised learning (without labels).

7.1.3. MNIST Lipi-SSL learning evolution

We investigate the evolution of the model training in terms of classification accuracy and image quality generation for Lipi-SSL 3×3 with 100 labeled samples from the run with the median classification accuracy. Fig. 4 shows the evolution of the average classification accuracy and FID at different training epochs. In addition, Fig. 5 shows the evolution of the per-class accuracy for epochs 5, 50, and 100 and an example of generated image for the same training epochs.

From classification accuracy evolution shown in Fig. 4 for MNIST, we observe a steep slope indicating that the classifier quickly learned to correctly classify many images. Specifically, the model had an average classification accuracy of 85.04% after the first five iterations, with an accuracy of 97.76% and 94.09% for classes “0” and “6”, respectively (see Fig. 5(a)). As the training epochs progressed, the slope flattened. Fig. 5(c) shows that a similar classification accuracy of 94.29% has been achieved for all classes in epoch 100. Note that the trend in the last 30 epochs indicates that the model accuracy might not have converged because the accuracy trend is increasing.

The evolution of FID, see Fig. 4, illustrates that FID looks like a monotonically decreasing function with small oscillations. FID exhibited a sharp drop in the first epochs, which explains the considerable difference between the quality of images produced in epochs 5 and 50 (see Figs. 5(d) and 5(e)). It can be observed in Fig. 4 that the training process stabilized to FID lower than 10 after the first 40 epochs, visualized by the quality of images between epochs 50 and 100 (see Figs. 5(e) and 5(f)).

7.1.4. MNIST comparison with state-of-the-art

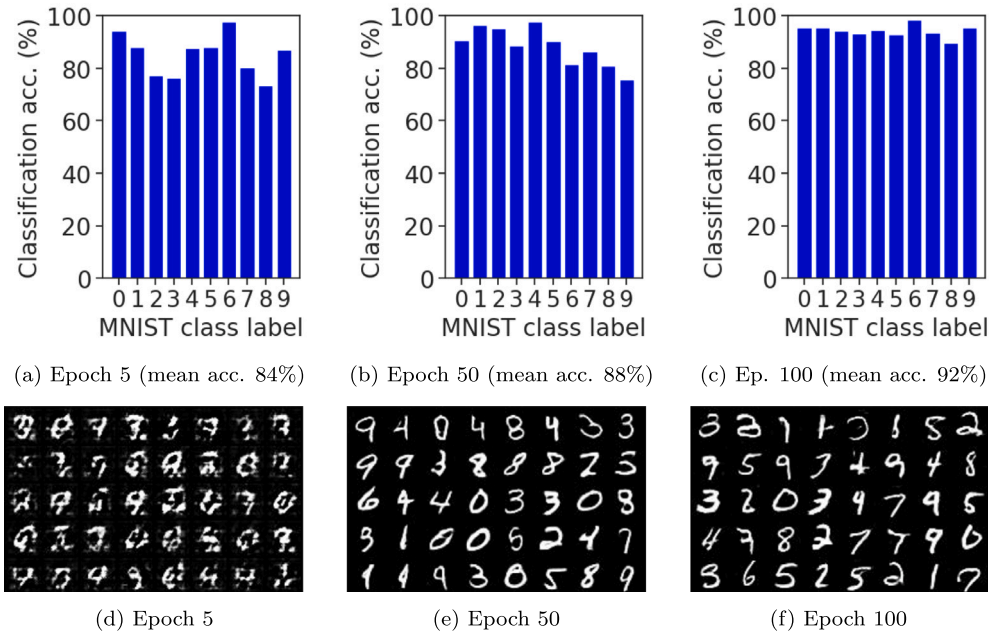
We compare Lipi-SSL against related SoA approaches, see Table 1 for an overview of these methods. Table 8 summarizes the classification accuracies results that were published and Lipi-SSL 3×3 for 50, 100, 200, 600, 1000, and 3000 labeled images. UGAN and FM-GAN achieved accuracy above 99%, TL SSL scored lower, around 97%–98%. Lipi-SSL and MarginGAN have lower comparable accuracies around 96%–97%. Moreover, Lipi-SSL provides more competitive results than MarginGAN when increasing the number of labeled images (i.e., 600, 1000, and 3000 labels).

One reason for the differences in the results is the difference in the complexity of the neural network models used, see Table 2. The SoA methods have complex convoluted models with numerous layers and train for more epochs (about 400 epochs or longer), Lipi-SSL achieved very close results with only 100 epochs and a 4-layered discriminator. Thus, the spatially distributed GAN training improves training efficacy, overcoming a sub-optimal use of a less complex network architecture. These results are in line with the results shown by similar approaches applied to unsupervised learning [24].

The SoA GAN training approaches prioritize the classification quality of the GAN to optimize the classification accuracy. Instead, Lipi-SSL focuses simultaneously on both the discriminator and generator quality. The authors of the SoA methods do not usually measure the quality and diversity of the samples generated. Thus, only a qualitative comparison can be made between SoA and Lipi-SSL. Fig. 6 shows generated images by the compared methods. The samples from FM-GAN and MarginGAN have incomplete digits, and many do not even appear visually as valid digits. UGAN produced higher quality images with a complex GAN architecture. Lipi-SSL generates images of similar quality to the UGAN with a simpler GAN architecture. This answers **RQ-2**, that we can devise a GAN method to train both a high-accuracy classifier and a high-quality generative model.

7.2. CIFAR-10 experimental results

This section evaluates Lipi-SSL on the CIFAR-10 dataset considering the classification accuracy (which is maximized) and generative model



(a) Epoch 5 (mean acc. 84%) (b) Epoch 50 (mean acc. 88%) (c) Ep. 100 (mean acc. 92%)

(d) Epoch 5 (e) Epoch 50 (f) Epoch 100

Fig. 5. Evolution of per-class accuracy and generated images for MNIST at different epochs for the median accuracy run with 100 labels for Lipi-SSL 3×3.

Table 8
Lipi-SSL vs. SoA mean±standard deviation classification accuracy (%) on MNIST for varying number of labeled images.

Model	Labeled images					
	50	100	200	600	1 000	3 000
UGAN	**98.92 ± 0.13**	**99.91 ± 0.08**	**99.35 ± 0.05**	–	–	–
FM-GAN	97.79 ± 1.36	99.07 ± 0.07	99.10 ± 0.05	–	–	–
TL SSL-GAN	–	97.71 ± 0.89	98.50 ± 0.92	**98.64 ± 0.96**	**98.86 ± 0.97**	–
MarginGAN	–	96.47 ± 0.57	–	96.97 ± 0.69	97.13 ± 0.71	97.94 ± 0.20
Lipi-SSL 3×3	94.87 ± 1.55	95.96 ± 1.42	96.07 ± 0.77	97.07 ± 0.68	98.09 ± 0.31	**98.95 ± 0.19**

(a) UGAN (Accuracy 99.2%) (b) FM-GAN (Accuracy 99.1%) (c) MarginGAN (Accuracy 96.5%) (d) Lipi-SSL (Accuracy 96.0%)

Fig. 6. Comparison of generated MNIST images by SoA methods and Lipi-SSL.

quality regarding IS (which is maximized), when training with 5000 labels (10% of training set). We investigate the sensitivity of the classification accuracy to the number of labeled samples in the training dataset. We discuss the evolution of the learning process. Finally, we compare the best Lipi-SSL configuration against the results published by the SoA algorithms, i.e., UGAN, FM-GAN, TL SSL-GAN, and MarginGAN. For CIFAR-10, seven independent runs were performed for each Lipi-SSL configuration.

7.2.1. CIFAR-10 classification accuracy

SSL accuracy with few labels. Here, Lipi-SSL was evaluated on 5000 labeled samples, i.e., only 10% of the training dataset. Table 9 summarizes the classification and image generation quality (IS) results by reporting the minimum (Min), median (Med), inter-quartile range (IQR), and maximum (Max) of the distribution of the results for each grid size over seven runs. The best values for each metric are in bold. Table 9 includes SSL-GAN as a baseline for comparison.

The results in Table 9 indicate that Lipi-SSL 2×2 provided the best classification results for all the descriptive statistics evaluated. The statistical analysis confirmed that Lipi-SSL 2×2 had the best accuracy results with a statistical significance higher than 99% (i.e., p-value <0.01). There is a significant improvement in Lipi-SSL 1×1 over a standalone implementation, highlighting the benefits of enhancements introduced by Lipi-SSL during training (regardless of distributed training). The accuracy classification improved when applying spatially distributed training.

Table 9
Descriptive statistics for classification accuracy and IS for experiments with 5000 (0.1) labeled images with Lipi-SSL for 7 independent runs. The best values are in **bold**, accuracy and IS are maximized.

Model & grid size	Accuracy (%)				IS			
	Min	Med	IQR	Max	Min	Med	IQR	Max
SSL-GAN	55.54	59.79	2.44	65.7	2.41	2.71	0.19	2.96
Lipi-SSL 1×1	65.43	70.13	3.29	75.79	3.43	3.77	0.34	3.96
Lipi-SSL 2×2	**73.19**	**76.29**	**2.70**	**79.84**	**4.78**	**4.92**	**0.14**	**5.23**

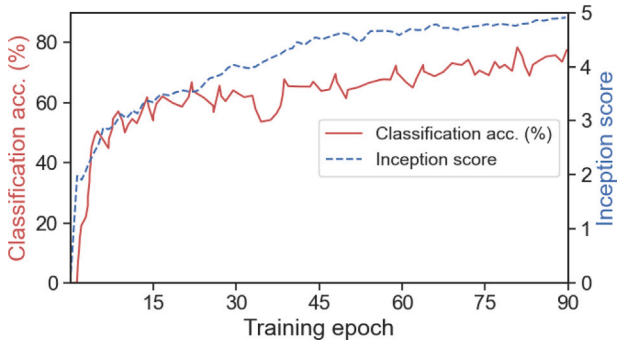


Fig. 7. Evolution of values for classification accuracy and IS for CIFAR-10 dataset Lipi-SSL 2×2 training when using 5000 labels in the median accuracy run.

Table 10
Lipi-SSL vs. SoA mean±standard deviation classification accuracy (%) on CIFAR-10 for varying number of labeled images (# of labels). The number between parenthesis indicates the fraction of labeled samples used.

Model	# labels: 5000 (0.1)	# labels: 50 000 (1.0)
UGAN	86.12 ± 0.06	–
FM-GAN	82.09 ± 2.32	–
TL SSL-GAN	80.97±NA	88.04±NA
MarginGAN	**93.56 ± 0.10**	–
Lipi-SSL 2×2	74.79 ± 2.76	**97.41 ± 2.40**

These results confirmed that increasing the grid size improves the classification accuracy of the trained classifiers, as seen in the MNIST experimental results (see Section 7.1.1).

SSL accuracy sensitivity to the number of labels. Further experiments by increasing the labeled samples from 5000 to 50 000 led to significantly better accuracy results. The average accuracy classification over seven runs improved from 68.44% to 95.47% for Lipi-SSL 1×1 and from 74.79% to 97.41% for Lipi-SSL 2×2.

7.2.2. CIFAR-10 image generation

One of the objectives of Lipi-SSL is to generate images of high quality and achieve high classification accuracy. Table 9 quantitatively assess the image quality generation by reporting the minimum, median, IQR, and maximum of the IS results distribution on the CIFAR-10 dataset when using 5000 labeled samples over the seven independent runs. Further results on IS and TVD (diversity) scores are shown in Table C.18 in Appendix C.

The SSL-GAN generative model produced the worst quality and the least diverse images (i.e., the lowest IS and the highest TVD), which validates the enhancements applied on Lipi-SSL 1×1. The populations of generators and discriminators provide an improvement in the quality and diversity of generated images is experienced (see Lipi-SSL 2×2 in Table 9). This improvement is attributed to the diversity introduced in the genome space by Lipi-SSL (i.e., Lipizzaner [23]).

7.2.3. CIFAR-10 Lipi-SSL learning evolution

We now investigate the evolution of the models during the training in terms of classification accuracy and IS for Lipi-SSL 2×2 with 5000 labeled samples. Fig. 7 illustrates the evolution of the average classification accuracy and IS score over different training epochs for the run that provided the median classification accuracy. The general trend of the plots in Fig. 7 shows that the training had not converged in iteration 90. The plot has an upward slope even at the last epochs. The classification accuracy for Lipi-SSL 2×2 is 74.8% at the end of the run (see Fig. 7).

Fig. 8 shows the evolution of the per-class accuracy at 5, 45, and 90 epochs. The per-label classification accuracy for classes 2, 3, 4, and 5 increased slower than for the other classes. This leads to differences in the classification accuracy results, especially in the early epochs. For example, in epoch 5, the accuracies of classes 5 and 9 were 36.28% and 71.73%, respectively. The differences were reduced in the later stages of training but still exist. For example, at epoch 90, the accuracies of the same classes (5 and 9) were 71.04% and 87.26%.

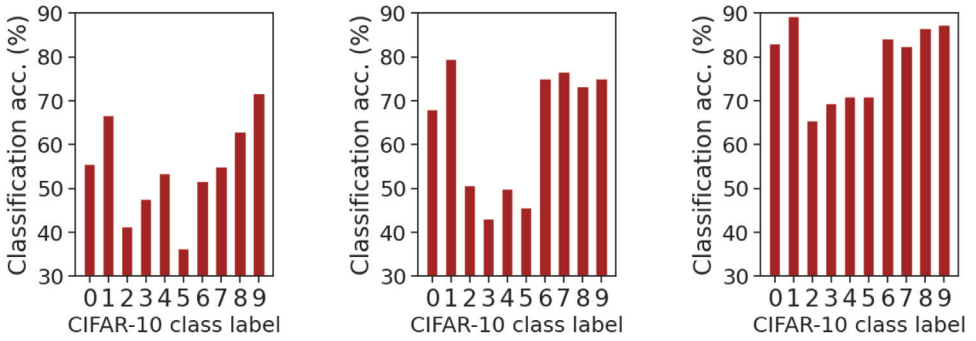
7.2.4. CIFAR-10 comparison with State-of-the-Art

We compare Lipi-SSL against SoA approaches. Table 10 summarizes the classification accuracies published and Lipi-SSL 2×2 for 5000 and 50 000 labeled samples. Note that TL SSL-GAN [18] did not report the standard deviation [18]. In addition, Fig. 9 shows samples generated by the compared methods. Note that there were no publicly available CIFAR-10 samples generated using UGAN and Triplet Loss SSL-GAN.

For 5000 labeled samples, the most competitive accuracy results are shown by MarginGAN, 93.56%. The other SoA methods had accuracies higher than 80%, while Lipi-SSL 2×2 got 74.79%. On the other hand, with the fully labeled dataset, Lipi-SSL 2×2 had the highest accuracy of 97.41%. The main findings from Table 10 and Fig. 9 are similar to the outcomes with MNIST in Section 7.1.4. Note that MarginGAN and UGAN training methods use more models (neural networks) besides one generator and one discriminator, Lipi-SSL uses simpler models (neural networks with fewer neurons), and trains for fewer epochs, see Tables 2 and 3. Moreover, Lipi-SSL trains a generative model that produces higher-quality images according to the qualitative analysis of the images provided by the SoA authors.

7.3. SVHN experimental results

In this section, we evaluate Lipi-SSL on the SVHN dataset considering the classification accuracy and generative model quality regarding IS, when training with 500 (0.68% of training set) and 1000 (1.39% of training set) labels. For the SVHN dataset we use Lipi-SSL 2×2, since in the previous sections we found that a larger grid size produces better results, and this is the largest grid size we used for a color image data set (CIFAR-10). Our analysis focuses on the classification accuracy and IS of Lipi-SSL, while varying the number of labels used. In addition, we discuss the evolution of the learning process. Finally, we compare Lipi-SSL against the results published by the SoA algorithms, i.e., UGAN, FM-GAN, TL SSL-GAN, and Margin-GAN. For SVHN, seven independent runs were performed for each number of labels.



(a) Epoch 5 (mean acc. 53%) (b) Epoch 45 (mean acc. 62%) (c) Epoch 90 (mean acc. 79%)

Fig. 8. Evolution of per-class accuracy for the CIFAR-10 at some epochs for the median accuracy run with 5000 labels for Lipi-SSL 2×2.

(a) FM-GAN (Accuracy 82.1%) (b) MarginGAN (Accuracy 93.6%) (c) Lipi-SSL 2×2 (Accuracy 74.8%)

Fig. 9. CIFAR-10 images produced by various SoA methods and Lipi-SSL.

Table 11
Descriptive statistics for classification accuracy and IS for experiments with 500 (0.68% of training set) and 1000 (1.39% of training set) labeled images with Lipi-SSL 2×2 for 7 independent runs. The best values are in **bold**, accuracy and IS are maximized.

Number of labels	Accuracy (%)				IS			
	Min	Med	IQR	Max	Min	Med	IQR	Max
500	83.05	85,03	2.14	86.84	**2.58**	**2.70**	0.22	**2.91**
1 000	**91.23**	**93.06**	**1.87**	**94.07**	2.32	2.67	**0.06**	2.76

7.3.1. Classification accuracy and image generation label sensitivity

Table 11 summarizes the classification accuracy and IS results by reporting the minimum, median, inter-quartile range, and maximum of the distribution of the results for the experiments with 500 and 1000 labeled images over seven runs. The best values for each metric are in bold.

The outcomes in Table 11 show that Lipi-SSL 2×2 trained with 1000 labels produced the most competitive classification results. The Wilcoxon statistical test confirms that increasing the labeled images from 500 to 1000 led to better accuracy results with a statistical significance greater than 99% (i.e., p-value <0.01).

Regarding image generation, the generative models trained using 500 labels provided the highest values for the descriptive statistics evaluated, except IQR. The IS results when training with more labels, i.e., 1000 labeled samples, are more robust (1000 label's IQR is lower than the same of 500). According to Wilcoxon statistical test, there is no statistically significant difference between the IS for both analyzed quantities of labeled samples.

7.3.2. SVHN Lipi-SSL learning evolution

In this section, we explore the progress of the models during the training process concerning classification accuracy and IS for Lipi-SSL 2×2, using a dataset of 500 labeled images. Fig. 10 illustrates the average classification accuracy and IS score evolution across different

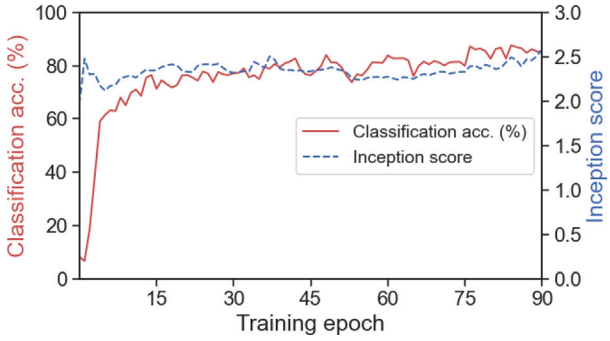
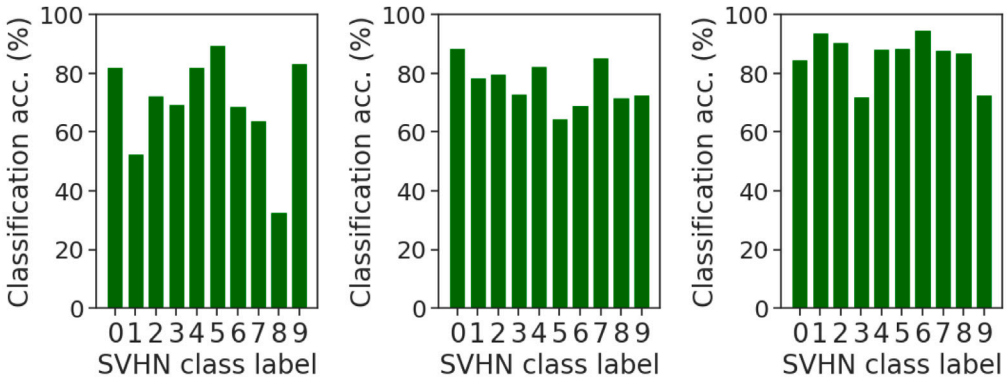


Fig. 10. Evolution of values for classification accuracy and IS for SVHN dataset Lipi-SSL 2×2 training when using 500 labels in the median accuracy run.

training epochs for the run that yielded the median classification accuracy. Similar to the findings in our CIFAR-10 experiments (see Section 7.2.3), the plot in Fig. 10 indicates that the classifier's training had not yet converged even at iteration 90. The plot maintains an upward trend even towards the final epochs. Finally, after the run, the classification accuracy for Lipi-SSL 2×2 stands at 85.03%.

Fig. 11 shows the progression of the per-class accuracy at 5, 45, and 90 epochs. The per-label classification accuracy for classes 8 and 1 increased slower than the others. Consequently, variations in the classification accuracy results were observed, particularly in the early epochs. For example, in epoch 5, the accuracies of classes 8 and 5 were 32.97% and 89.65%, respectively. The differences were reduced in the later stages of training. Towards the final epochs, there is a change in trend and the lowest classification accuracies are shown by classes 3 and 9.



(a) Epoch 5 (mean acc. 65%) (b) Epoch 45 (mean acc. 76%) (c) Epoch 90 (mean acc. 85%)

Fig. 11. Evolution of per-class accuracy for the SVHN at some epochs for the median accuracy run with 500 labels for Lipi-SSL 2×2.

(a) UGAN (Accuracy 96.5%) (b) MarginGAN (Accuracy 94.0%) (c) Lipi-SSL 2×2 (Accuracy 92.6%)

Fig. 12. Comparison of generated SVHN images by various SoA methods and Lipi-SSL.

Table 12
Lipi-SSL vs. SoA mean±standard deviation classification accuracy (%) on SVHN for varying number of labeled images (# of labels).

Model & grid size	# labels: 500	# labels: 1 000
UGAN	–	**96.49 ± 0.09**
FM-GAN	–	91.89 ± 1.30
TL SSL-GAN	–	94.23 ± 0.17
MarginGAN	**93.97 ± 0.43**	–
Lipi-SSL 2×2	84.95 ± 1.51	92.60 ± 1.14

7.3.3. SVHN comparison with State-of-the-Art

We compare Lipi-SSL against SoA approaches on SVHN dataset. Table 12 summarizes the classification accuracies published and Lipi-SSL 2×2 for 500 and 1000 labeled images. In addition, Fig. 12 shows samples generated by the compared methods. Note that there were no publicly available SVHN samples generated using FM-GAN and TL SSL-GAN.

The experiments involving 500 labeled images only have MarginGAN results available for comparison. In this scenario, MarginGAN shows more competitive outcomes than Lipi-SSL. For 1000 labeled samples, the UGAN displayed the most competitive accuracy results at 96.49%. The other SoA methods achieved accuracies above 91%, with Lipi-SSL reaching 92.60%. The images produced by Lipi-SSL in Fig. 12 are sharper and less blurred than those produced by the other methods. Overall, the main findings from Table 12 and Fig. 12 are similar to the outcomes with MNIST and CIFAR-10. Lipi-SSL provided high competitive accuracy results while producing high-quality images using simpler models and shorter training processes.

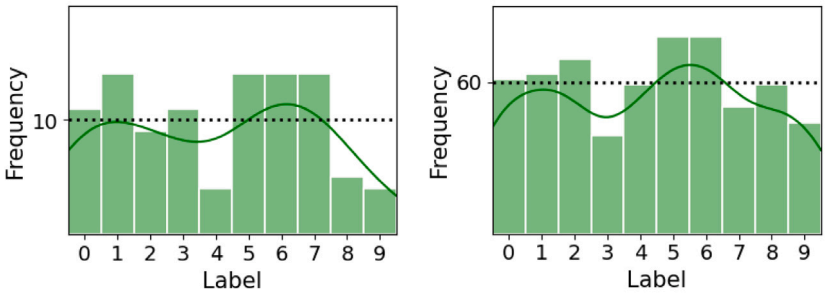
7.4. Ablation experiments

The experiments in this section aim at establishing the relevance of the new features added to Lipizzaner to perform SSL. We proceed through an ablation study on grayscale and color image datasets, i.e., MNIST and SVHN datasets. Thus, this section first outlines the ablation experimental analysis setup and, second, presents and discusses the results. Note that authors of Lipizzaner conducted an ablation study about the relevant features of the spatially distributed co-evolutionary GAN training method [23]. Thus, we focus on the specific SSL features of Lipi-SSL.

7.4.1. Ablation setup

We have performed three ablation experiments. The `Baseline` model for MNIST is Lipi-SSL 3×3, while the `Baseline` for SVHN is Lipi-SSL 2×2, with both models include all proposed method features. The three ablation configurations analyzed are the elimination of spatial diversity loss (`No diversity loss`), balanced labels batch sampler (`No balanced sampler`), and the combination of removing these two features (`No Lipi-SSL features`). The ablations are listed in Table 13. We have conducted 15 and seven independent runs of each ablation experiment for MNIST and SVHN datasets, respectively. Note that ensemble methods are well established [61] so we do not ablate the majority voting of the classifier.

These ablation experiments were performed on the MNIST dataset using Lipi-SSL 3×3, which was the grid size that performed the best for that dataset. We have examined two scenarios, one with 100 labeled samples and the other with 600, to assess the influence of the new Lipi-SSL features in situations with limited labels and as the number of labels increases.



(a) 100 labeled images (b) 600 labeled images

Fig. 13. Examples of the distribution of the labeled images used to train `No balanced sampler` and `None Lipi-SSL features` ablation for 100 and 600 labels. The dotted line represents the frequency expected for uniformly distributed labels, i.e., Lipi-SSL using balanced labels batch sampler.

Table 13
Lipi-SSL features ablation.

Ablation	Spatial diversity loss	Balanced labels batch sampler
`Baseline`	✓	✓
`No diversity loss`		✓
`No balanced sampler`	✓	
`No Lipi-SSL features`		

When Lipi-SSL does not apply spatial diversity loss (i.e., `No diversity loss` experiment) all the cells in the grid apply the same values for the discriminator loss weights, i.e., w_u=0.5 and w_s=0.5.

The labeled images in the full Lipi-SSL, i.e, our `Baseline`, are uniformly distributed throughout the classes. In contrast, the `No balanced sampler` ablation randomly simply samples the labeled images used to build the training dataset without any criteria. Thus, when this ablation is applied, the labeled images are distributed differently for each run. Fig. 13 shows an examples of how the labels of the samples were distributed in one run of this ablation experiment for 100 and 600 labeled samples. The relative difference among the label frequencies in the same run is higher when using fewer labels (i.e., 100 labeled images experiments). This can lead to more unbalanced datasets when the number of labeled samples is small.

Another set of ablation experiments was performed on SVHN. The idea is to investigate whether the ablation experiments report similar outcomes to MNIST when Lipi-SSL is applied to color images and with larger ANNs.

7.4.2. Ablation results

The results obtained from the ablation experiments on Lipi-SSL are summarized in Tables 14 and 15, which shows the minimum (Min), median (Med), interquartile range (IQR), and maximum (Max) values of the distribution of the results for each metric and ablation experiment for MNIST and SVHN, respectively.

Following the previous analysis, we have evaluated FID on MNIST and IS on SVHN to assess the quality of the obtained generative models, i.e., the quality of the images synthesized by the generators. The results from these two metrics do not shown any significant differences between the generative models evaluated when the ablations were applied. Thus, KID is assessed to provide further observation regarding the impact of the ablations on the generative model training.

Classification accuracy results in Tables 14 and 15 show that each ablation notably leads to a deterioration in the performance (Accuracy) of the obtained classifier. According to pairwise Wilcoxon statistical tests comparisons between all the evaluated ablations, `Baseline` significantly outperforms the three ablations for MNIST (both 100 and 600 labeled samples) and SVHN.

Focusing on MNIST when using 100 and 600 labeled images, the difference between Lipi-SSL and the other evaluated approaches are higher when using fewer labeled images, i.e., 100 labeled samples. Thus, the two new features to Lipizzaner are relevant for classification problems with limited labels.

When Lipi-SSL does not use the spatial diversity loss feature (`No diversity loss`), the classification accuracy results are less competitive than the results of `Baseline`. Previous research has shown that applying diversity in the loss functions when training GANs keeps higher diversity in the populations and allows training better unsupervised GANs [16,49]. This ablation study shows that the same occurs when training SSL-GANs with a coevolutionary training method, i.e., using diversity in loss functions trains more competitive SSL-GANs.

The `No balanced sampler` ablation results are negatively affected by biases when generating the training dataset. Nevertheless, it can be observed that the spatial diversity loss feature kept the results in competitive margins. This result is in line with the idea that keeping diversity in the populations of coevolutionary GAN training improves the learning process [23]. This can explain how the results of the `No balanced sampler` ablation are significantly better than those of the `None Lipi-SSL features` ablation.

Thus, combining the spatial diversity loss and balanced labels batch sampler is essential to keep the classification accuracy results competitive. Note that the difference between the baseline Lipi-SSL and the other three Lipi-SSL ablations is larger for experiments with 100 labels than with 600 labels because the biases are stronger when producing datasets with fewer labels (see Fig. 13). Thus, applying the whole Lipi-SSL in classification problems with very few labeled images seems to be the proper strategy for competitive classification results. Finally, there is no statistical difference between the accuracy results of `No balanced sampler` and `No diversity loss`.

Regarding the quality of the trained generators, when comparing the four Lipi-SSL variants analyzed, there is no statistical difference between the FID results on MINST, see Table 14 (i.e., p-value $\gg$0.01). The same occurs with SVHN experiments, i.e., there is no statistical significance among the IS computed for `Baseline` and the ablations, see Table 15. Likewise, no statistically differences were observed when evaluating the KID results for both datasets. Therefore, the two SSL-GAN features studied in this ablation study do not have an impact on the trained generative models.

The findings of this ablation study underscore the significance of the spatial diversity loss and balanced labels batch sampler in Lipi-SSL for achieving its competitive performance in terms of classification accuracy results and quality of the generative models.

Table 14
Descriptive statistics for classification accuracy and FID for MNIST ablation experiments with 100 (0.0017) and 600 (0.01) labeled images for 15 independent runs. The best values are in **bold**, accuracy is maximized and FID and KID are minimized.

Model	Accuracy (%)				FID				KID ($\times 10^{-2}$)			
	Min	Med	IQR	Max	Min	Med	IQR	Max	Min	Med	IQR	Max
100 labeled samples												
Baseline	**90.36**	**96.34**	**0.71**	**96.99**	3.36	4.35	0.93	7.51	3.21	3.68	0.17	3.73
No diversity loss	75.30	86.98	5.25	**96.99**	3.01	4.29	**0.32**	5.55	3.35	3.64	**0.08**	3.72
No balanced sampler	80.30	87.48	4.36	93.84	**2.33**	**4.16**	1.57	**5.22**	3.28	3.64	0.21	3.70
None Lipi-SSL features	64.01	78.43	12.82	92.24	3.08	4.22	1.77	7.22	**3.15**	**3.63**	0.11	**3.70**
600 labeled samples												
Baseline	**95.77**	**97.25**	0.92	**98.52**	3.10	4.18	1.37	5.50	3.19	3.68	0.21	**3.72**
No diversity loss	91.96	95.88	**0.44**	98.22	**2.99**	3.64	**0.58**	5.11	3.27	3.64	0.21	3.73
No balanced sampler	95.38	96.29	0.90	98.11	**2.99**	**3.57**	1.39	**5.01**	**3.15**	**3.63**	**0.09**	3.73
None Lipi-SSL features	73.87	90.68	7.84	98.40	3.01	4.34	1.41	5.80	3.20	3.65	0.12	3.73

Table 15
Descriptive statistics for classification accuracy, IS, and KID for SVHN ablation experiments with 1000 labeled images for 7 independent runs. The best values are in **bold**, accuracy and IS are maximized, and KID is minimized.

Model	Accuracy (%)				IS				KID ($\times 10^{-2}$)			
	Min	Med	IQR	Max	Min	Med	IQR	Max	Min	Med	IQR	Max
Baseline	**91.23**	**93.06**	1.87	**94.07**	2.32	2.67	**0.06**	2.76	2.42	2.47	0.17	2.67
No diversity loss	85.18	87.08	**1.85**	87.98	2.33	2.53	0.15	2.66	2.43	2.63	0.14	2.96
No balanced sampler	86.28	87.70	1.58	88.91	2.37	2.54	0.14	2.74	**2.39**	2.50	0.13	**2.55**
None Lipi-SSL features	81.59	83.23	2.05	85.47	**2.56**	**2.68**	0.13	**2.84**	**2.39**	**2.46**	0.09	2.56

8. Discussion

We investigated the quality of both the generator and the discriminator when trained at the same time. This is in contrast to related works that either train a good generator with a given method or a good classifier with another method. More specifically, the empirical experiments provided the answer to **RQ-2** by demonstrating that with Lipi-SSL it is possible to devise a GAN method that can train both a high-accuracy classifier and a high-quality generative model (see Section 7). In addition, the answer to **RQ-1** was shown by the utility of SX SSL-GAN (see Section 4) in Lipi-SSL. Finally, the Lipi-SSL experiments also answered **RQ-3** by generating high quality samples when training with some labels (SSL) compared to training without labels(unsupervised) (see Section 7).

State-of-the-art comparisons. The state-of-the-art methods UGAN and MarginGAN have in addition to the generator and discriminator, more neural networks in their GAN architectures. MarginGAN includes an additional classifier, and UGAN applies a classifier and a second generator. Instead, Lipi-SSL applied the basic DCGAN architecture.

Lipi-SSL behaves as expected from previous work, i.e., increasing grid size can increase image generation performance [23,24]. We also expect that Lipi-SSL can improve the SoA methods by creating a population of those GAN architectures. In [24], a distributed evolutionary GAN has a limit to how much it can improve an ANN architecture. In addition, one focus of this work was to use simple ANN architectures

Lipi-SSL loss diversity training. The diversity of the generators and discriminators has been demonstrated to have an impact on the performance of Lipizzaner [23,24]. A diversity of loss functions, which led to a diversity of loss also improved performance [23]. This leads us to conjecture that the diversity of the supervised and unsupervised loss weights in the grid contributes to performance improvement. Further ablation studies are required to provide more evidence for this conjecture.

Lipi-SSL training configuration. We see that Lipi-SSL achieves good performance for 2×2 and 3×3. This is an indication of the effect of the diverse SSL loss weights can have on both classification accuracy and generated sample quality. Furthermore, we see that a fully labeled data set has the highest sample generation quality with semi-supervised learning compared to unsupervised learning. In addition, for MNIST, the image generation converged in the first half of the run. The classification accuracy can still be improved by further training.

When training with all CIFAR-10 labels, Lipi-SSL got training accuracies larger than 99.0% in only 40 epochs when the fully labeled dataset was used (result not shown), indicating how the GAN training stabilized with a simple neural network architecture. These observations and the trend in the plots in Figs. 4, 7, and 10 indicate that it would take longer for the training process to stabilize when a small fraction of labeled samples were used, enabling further improvement in classification performance (and image generation quality).

Limitations. For SoA comparisons, we cannot test the differences for statistical significance since only aggregate values were reported. The CIFAR-10 and SVHN results were obtained even when training was stopped (due to computational constraints) before training converged and stabilized.

9. Conclusions and future work

Our main objectives were to achieve good performances for both classification and image generation, using a simple architecture, low training duration, and a few labeled samples. We demonstrated that spatially distributed evolutionary GAN for semi-supervised learning method, Lipi-SSL, can provide this. With the use of SX SSL-GAN, we introduced a simple architecture. In addition, Lipi-SSL generated high-quality samples when training with some labels (semi-supervised) compared to training without labels (unsupervised).

The coevolutionary approach of Lipi-SSL injects diversity into the GAN training, which can improve performance. In addition, Lipi-SSL uses optimizations such as Feature Matching and Instance Noise to stabilize the training. Moreover, Lipi-SSL has a simple architecture both in terms of the number of layers and number of convolutional channels. Additionally, Lipi-SSL ensures a balanced subset of data points associated with labels is used during training. We evaluated our approach on the MNIST, CIFAR-10, and SVHN datasets. Lipi-SSL achieved good

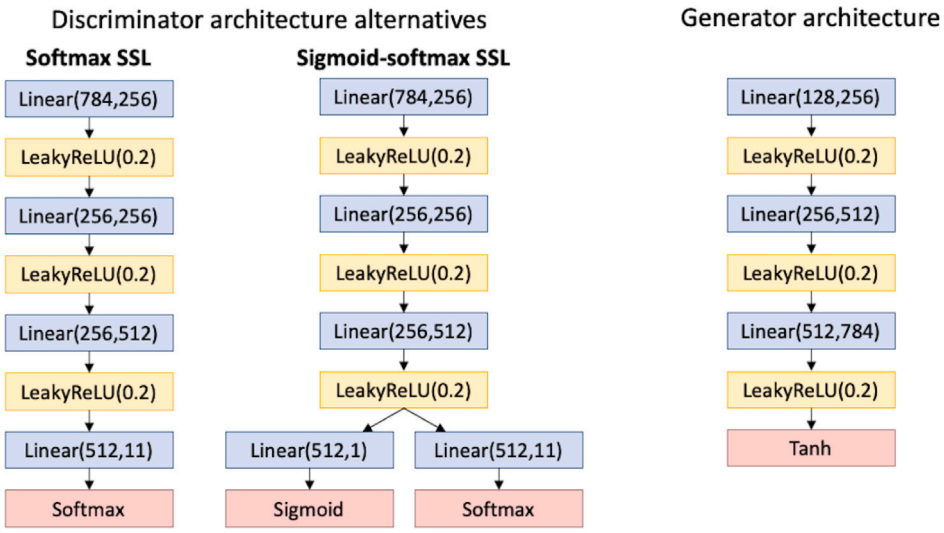


Fig. A.14. Base SSL-GAN architectures.

classification accuracy, and the images generated by our models were of a quality that is visually comparable to SoA approaches. Note that the compared SoA approaches did not provide quantitative measures of the image quality, so the potentially more biased visual inspection must be used. Finally, our ablation study verified how the discriminators' loss function diversity and balancing the labels in the training dataset impact the performance of Lipi-SSL.

The main lines for future work are related to extending the empirical analysis with different RGB and hyperspectral image datasets, e.g., TinyImageNet, and Indian Pines; neural network architectures; and Lipi-SSL configurations. Finally, SoA and Lipi-SSL will be run with the same computational budget for the comparisons.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research was partially funded by the Universidad de Málaga, Spain; the project PID 2020-116727RB-I00 (HUmove) funded by MCIN/AEI/10.13039/501100011033, Spain; TAILOR ICT-48 Network (No. 952215) funded by EU Horizon 2020 research and innovation programme; and by Universidad de Málaga, Spain (grant B1-2022_18). The Systems that learn initiative at MIT CSAIL.

Appendix A. SSL GAN architecture and training

This appendix extends the analysis performed in Section 4. It presents a preliminary set of experiments on the MNIST dataset to evaluate the three SSL-GAN variants (architectures and loss functions) presented in Section 4.1 and FM in Section 2.2. The experiments here are used to select a simple and high-performing SSL-GAN variant to investigate in Section 7.

Fig. A.14 shows the main components of the SSL-GAN architectures evaluated here.

This section presents a preliminary set of experiments on the MNIST dataset to evaluate the three SSL-GAN variants (architectures and loss functions) presented in Section 4.1 and FM in Section 2.2. The experiments here are used to select a simple and high-performing SSL-GAN variant to investigate in Section 7.

Setup. The main parameter settings are the following: the generator input was defined by a normal distribution with a mean of 0 and a standard deviation of 0.1; a dropout of 0.2 was included between each layer of the discriminator architecture; the batch normalization layers were added in the generator and the discriminator as indicated; and positive label smoothing with a smoothed label of 0.9 was used. Finally, these preliminary experiments considered ten runs of each alternative over 100 epochs, using 100 (0.00167), 300 (0.005), 600 (0.01), and 60 000 (1.0) labeled samples from the MNIST dataset.

The experiments were run on a HPC computing platform that uses an IBM Power System AC922 (8335-GTG) with a Power9 architecture, 1 TB RAM, 3.6 GHz CPU cores, and NVIDIA Tesla V100 GPUs with 32 GB RAM.

Results. Table A.16 presents the classification accuracies and the image quality generation (FID score) obtained by each SA-SSL variant with a different number of labeled data points over the ten independent runs. For readability Table A.16 shows integer values. Note that the differences among the evaluated methods can be observed without decimals.

The results in Table A.16 confirm that increasing the number of labeled images improved the results, for both accuracy and image generation. There is a markable difference between using the entire labeled dataset as opposed to a fraction of it. On the one hand, SX SSL-GAN provided the most competitive classification accuracy results (see Table A.16). On the other hand, SS SSL-GAN trained the best generative models in terms of FID (see Table A.16). The use of FM improved the classification accuracy while reducing the quality of the generated images.

SS SSL-GAN forces the discriminator to learn the correlation between images produced by the generator, to be labeled as fake by the sigmoid layer while simultaneously also being labeled with the $(K+1)^{th}$ class by the softmax layer. In contrast, SX SSL-GAN only does the latter, focusing more on classification. As a result, the quality of images produced by SX SSL-GAN was lower than that of SS SSL-GAN. However, its classification performance was much better than that of the SS

Table A.16
Classification accuracy (Acc.) and FID on MNIST for the SSL-GAN methods with (FM) and without feature matching (No FM) for different amounts of labels (# labels (fraction)). **Bold** indicates the best result, accuracy is maximized and FID is minimized.

# labels	SX SSL-GAN				SS SSL-GAN				H SSL-GAN			
	No FM		FM		No FM		FM		No FM		FM	
	Acc.	FID	Acc.	FID	Acc.	FID	Acc.	FID	Acc.	FID	Acc.	FID
100 (0.0017)	42	362	**49**	403	25	**274**	28	328	38	303	45	356
300 (0.005)	46	315	**62**	374	28	**243**	39	282	41	280	57	320
600 (0.01)	51	288	**72**	331	38	**201**	48	256	48	262	65	274
60 000 (1.0)	73	64	**84**	73	58	**48**	64	56	69	55	82	65

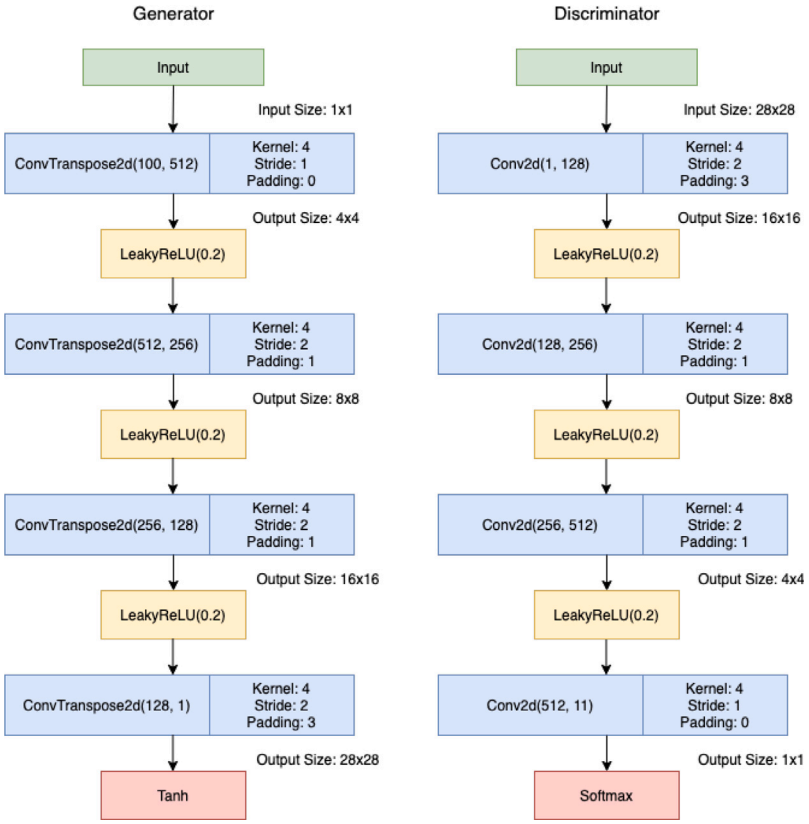


Fig. B.15. Proposed SSL-DCGAN architecture to deal with MNIST (with PyTorch layers).

SSL-GAN. H SSL-GAN is worse than SX SSL-GAN in its classification performance and worse than SS SSL-GAN in its generation capability.

The H SSL-GAN implements a training process that, after a specific duration, switches training from unsupervised learning to semi-supervised learning. The selection of the optimal time to switch adds another parameter to the training process. This training method did not provide the best results for both metrics.

Using FM improved the classification accuracy while worsening the FID scores. Thus, we selected SX SSL-GAN, the method that got the best classification results without using FM (because FM limits the capability of the trained generative model to produce quality images).

Appendix B. Neural network architectures for Lipi-SSL experiments

Figs. B.15 and B.16 illustrate the architectures defined to perform the experimental analysis of Lipi-SSL on MNIST and CIFAR-10, respectively. In these two figures, `Conv2d` [62] and `ConvTranspose2d` [63] refer to the PyTorch implementations of layers for a convolutional operator and a two-dimensional transposed convolutional operator, respectively.

Appendix C. Further empirical analysis results

Tables C.17 and C.18 summarize the main experimental results. These two tables present the classification accuracy, image generation quality (in terms of FID for MNIST and in terms of IS for CIFAR-10) and image generation diversity (TVD) results by reporting the minimum, median, inter-quartile range (IQR), and maximum of the distribution of the results for each grid size. The MNIST results were computed over 15 independent runs, while the CIFAR-10 results were obtained over seven independent runs.

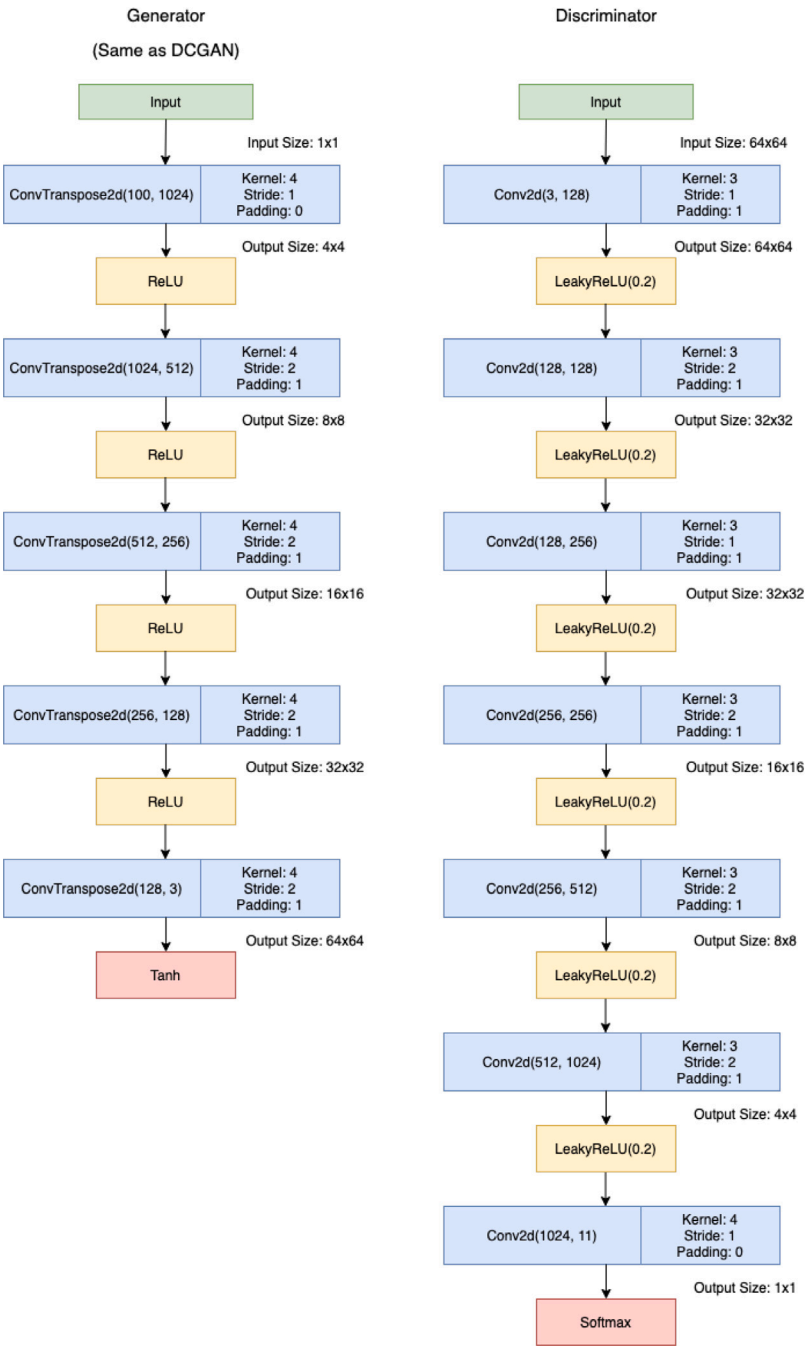


Fig. B.16. Proposed SSL-DCGAN architecture to deal with CIFAR (with PyTorch layers).

Table C.17
Accuracy, FID, and TVD scores using different models with 100 and 50 000 labeled data points when applying semi-supervised learning and 0 labels when using unsupervised learning (averaged across 15 runs). Best values are in **bold**, accuracy is maximized, and FID and TVD is minimized.

Model & grid size	# labels: 0 (0)			# labels: 100 (0.0017)			# labels: 50 000 (0.83)		
	Acc.	FID	TVD	Acc.	FID	TVD	Acc.	FID	TVD
Lipizzaner 1×1	–	17.96	0.10	–	–	–	–	–	–
Lipizzaner 3×3	–	3.38	0.04	–	–	–	–	–	–
SSL-GAN	–	–	–	49.28	403.39	0.16	84.10	73.93	0.14
Lipi-SSL 1×1	–	–	–	86.03	13.77	0.08	94.98	8.94	0.07
Lipi-SSL 2×2	–	–	–	91.16	7.02	**0.07**	99.14	5.62	0.06
Lipi-SSL 3×3	–	–	–	**95.96**	**5.22**	**0.07**	**99.38**	**2.72**	**0.04**

Table C.18
Classification accuracy using varying number of labeled data points considering Lipi-SSL and SSL-GAN, averaged across 7 runs. The best values for each number of labeled samples used are in **bold**, accuracy and IS are maximized, and TVD is minimized.

Model & grid size	# labels: 5 000 (0.1)			# labels: 50 000 (1.0)		
	Acc.	IS	TVD	Acc.	IS	TVD
SSL-GAN	59.26%	2.74	0.081	88.27%	–	–
Lipi-SSL 1×1	68.44%	3.64	0.043	95.47%	–	–
Lipi-SSL 2×2	**74.79%**	**4.92**	**0.031**	**97.41%**	–	–

References

[1] J.E. Van Engelen, H.H. Hoos, A survey on semi-supervised learning, Mach. Learn. 109 (2) (2020) 373–440.

[2] E. Denton, S. Gross, R. Fergus, Semi-supervised learning with context-conditional generative adversarial networks, 2016, http://arxiv.org/abs/1611.06430.

[3] A. Odena, Semi-supervised learning with generative adversarial networks, 2016, arXiv preprint arXiv:1606.01583.

[4] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014.

[5] R. Byrd, K. Damani, H. Che, A. Calandra, D.-K. Kim, A systematic literature review of volumetric 3D model reconstruction methodologies using generative adversarial networks., J. Inf. Sci. Eng. 38 (6) (2022) 1243–1263.

[6] Y. Qin, Z. Wang, D. Xi, Tree CycleGAN with maximum diversity loss for image augmentation and its application into gear pitting detection, Appl. Soft Comput. 114 (2022) 1–11.

[7] W. Diao, F. Zhang, J. Sun, Y. Xing, K. Zhang, L. Bruzzone, ZeRGAN: Zero-reference GAN for fusion of multispectral and panchromatic images, IEEE Trans. Neural Netw. Learn. Syst. (2022) 1–15, http://dx.doi.org/10.1109/TNNLS.2021.3137373.

[8] H. Wang, D. Won, S.W. Yoon, An adaptive neural architecture optimization model for retinal disorder diagnosis on 3d medical images, Appl. Soft Comput. 111 (2021) 1–13.

[9] Y. Li, M. Sun, X. Zhang, Perception-guided generative adversarial network for end-to-end speech enhancement, Appl. Soft Comput. 128 (2022) 1–9.

[10] H. Ohno, Training data augmentation: An empirical study using generative adversarial net-based approach with normalizing flow models for materials informatics, Appl. Soft Comput. 86 (2020) 1–10.

[11] F. Zhao, Y. Lu, X. Li, L. Wang, Y. Song, D. Fan, C. Zhang, X. Chen, Multiple imputation method of missing credit risk assessment data based on generative adversarial networks, Appl. Soft Comput. 126 (2022) 1–11.

[12] S. Arora, A. Risteski, Y. Zhang, Do GANs learn the distribution? Some theory and empirics, in: International Conference on Learning Representations, 2018, pp. 1–16.

[13] J. Li, A. Madry, J. Peebles, L. Schmidt, On the limitations of first-order approximation in GAN dynamics, in: 35th International Conference on Machine Learning, Vol. 7, ICML 2018, 2017, pp. 4672–4689.

[14] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International Conference on Machine Learning, 2017, pp. 214–223.

[15] X. Yu, M. Gen, Introduction to evolutionary algorithms, 2010.

[16] C. Wang, C. Xu, X. Yao, D. Tao, Evolutionary generative adversarial networks, IEEE Trans. Evol. Comput. 23 (6) (2019) 921–934.

[17] V. Costa, N. Lourenço, P. Machado, Coevolution of Generative Adversarial Networks, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11454 LNCS, 2019, pp. 473–487.

[18] M. Zieba, L. Wang, Training triplet networks with GAN, in: 5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings, 2017, pp. 1–6.

[19] W. Li, Z. Wang, Y. Yue, J. Li, W. Speier, M. Zhou, C. Arnold, Semi-supervised learning using adversarial training with good and bad samples, Mach. Vis. Appl. 31 (2020) 1–11.

[20] J. Dong, T. Lin, Margingan: adversarial training in semi-supervised learning, in: NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019, pp. 10440–10449.

[21] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training GANs, in: NIPS'2016: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 2234–2242.

[22] T. Schmiedlechner, I. Ng Zhi Yong, A. Al-Dujaili, E. Hemberg, U.-M. O'Reilly, Lipizzaner: A system that scales robust generative adversarial network training, in: NIPS'18: Workshop on Systems for Machine Learning, 2018, pp. 1–7.

[23] E. Hemberg, J. Toutouh, A. Al-Dujaili, T. Schmiedlechner, U.-M. O'Reilly, Spatial coevolution for generative adversarial network training, ACM Trans. Evol. Learn. Optimtim. 1 (2) (2021) 1–28.

[24] D. Flores, E. Hemberg, J. Toutouh, U.-M. O'Reily, Coevolutionary generative adversarial networks for medical image augumentation at scale, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2022, pp. 367–376.

[25] J. Toutouh, U.-M. O'Reilly, Signal propagation in a gradient-based and evolutionary learning system, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2021, pp. 377–385.

[26] L. Deng, The MNIST database of handwritten digit images for machine learning research [best of the web], IEEE Signal Process. Mag. 29 (6) (2012) 141–142.

[27] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, Tech. rep, University of Toronto, 2009, URL https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf.

[28] N. Yuval, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in: Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.

[29] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, Gans trained by a two time-scale update rule converge to a local nash equilibrium, in: NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, 2017, pp. 6629–6640.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[31] C. Donahue, J. McAuley, M. Puckette, Synthesizing audio with generative adversarial networks, 2018, arXiv preprint arXiv:1802.04208.

[32] W.J. Baddar, G. Gu, S. Lee, Y.M. Ro, Dynamics transfer GAN: Generating video by transferring arbitrary temporal dynamics from a source video to a single target image, 2017, arXiv preprint arXiv:1712.03534.

[33] S. Tulyakov, M.-Y. Liu, X. Yang, J. Kautz, Mocogan: Decomposing motion and content for video generation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 1526–1535.

[34] C.K. Sønderby, J. Caballero, L. Theis, W. Shi, F. Huszár, Amortised MAP inference for image super-resolution, in: ICLR 2017: International Conference on Learning Representations, 2017, pp. 1–17.

[35] S. Wieluch, F. Schwenker, Dropout induced noise for co-creative gan systems, in: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019, pp. 3137–3140.

[36] S. Xiang, H. Li, On the effects of batch and weight normalization in generative adversarial networks, 2017, arXiv preprint arXiv:1704.03971.

[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.

[38] M. Bińkowski, D.J. Sutherland, M. Arbel, A. Gretton, Demystifying mmd gans, in: International Conference on Learning Representations, 2018.

[39] A. Mammone, M. Turchi, N. Cristianini, Support vector machines, Wiley Interdiscip. Rev. Comput. Stat. 1 (3) (2009) 283–289.

[40] K.O. Stanley, J. Clune, J. Lehman, R. Miikkulainen, Designing neural networks through neuroevolution, Nat. Mach. Intell. 1 (1) (2019) 24–35.

[41] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, B. Hodjat, Chapter 15 - evolving deep neural networks, in: Artificial Intelligence in the Age of Neural Networks and Brain Computing, 2019, pp. 293–312.

[42] A. Camero, J. Toutouh, E. Alba, Random error sampling-based recurrent neural network architecture optimization, Eng. Appl. Artif. Intell. 96 (2020) 1–8.

[43] P.-K. Wong, M.-L. Wong, K.-S. Leung, Probabilistic grammar-based deep neuroevolution, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2019, pp. 87–88.

[44] V. Costa, N. Lourenço, J. Correia, P. Machado, Neuroevolution of generative adversarial networks, in: Deep Neural Evolution, Springer, 2020, pp. 293–322.

[45] U. Garciarena, R. Santana, A. Mendiburu, Evolved gans for generating pareto set approximations, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2018, pp. 434–441.

[46] S. Chen, W. Wang, B. Xia, X. You, Q. Peng, Z. Cao, W. Ding, CDE-GAN: Cooperative dual evolution-based generative adversarial network, IEEE Trans. Evol. Comput. 25 (5) (2021) 986–1000.

[47] E. Popovici, A. Bucci, R.P. Wiegand, E.D. De Jong, Coevolutionary principles, in: Handbook of Natural Computing, Springer, 2012, pp. 987–1033.

[48] A. Al-Dujaili, T. Schmiedlechner, E. Hemberg, U.-M. O'Reilly, Towards distributed coevolutionary GANs, in: Association for the Advancement of Artificial Intelligence (AAAI) Fall Symposium, 2018, pp. 1–6.
[49] J. Toutouh, E. Hemberg, U.-M. O'Reilly, Spatial evolutionary generative adversarial networks, in: GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference, 2019, pp. 472–480.
[50] J. Toutouh, E. Hemberg, U.-M. O'Reilly, Data dieting in gan training, in: H. Iba, N. Noman (Eds.), Deep Neural Evolution: Deep Learning with Evolutionary Computation, Springer Singapore, Singapore, 2020, pp. 379–400.
[51] J. Toutouh, E. Hemberg, U.-M. O'Reily, Re-purposing heterogeneous generative ensembles with evolutionary computation, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, 2020, pp. 425–434.
[52] E. Pérez, S. Nesmachnow, J. Toutouh, E. Hemberg, U.-M. O'Reily, Parallel/distributed implementation of cellular training for generative adversarial neural networks, in: 2020 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPSW, IEEE, 2020, pp. 512–518.
[53] C. Ju, A. Bibaut, M. van der Laan, The relative performance of ensemble methods with deep convolutional neural networks for image classification, J. Appl. Stat. 45 (15) (2018) 2800–2818.
[54] J.M. Johnson, T.M. Khoshgoftaar, Survey on deep learning with class imbalance, J. Big Data 6 (1) (2019) 1–54.
[55] A. Grover, S. Ermon, Boosted generative models, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.
[56] Y. Freund, R.E. Schapire, et al., Experiments with a new boosting algorithm, in: Icml, 96, Citeseer, 1996, pp. 148–156.
[57] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, in: ICLR 2016: 4th International Conference on Learning Representations, 2016, pp. 1–16.
[58] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of gans for improved quality, stability, and variation, in: International Conference on Learning Representations, 2018.
[59] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4401–4410.
[60] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, 2016, pp. 2818–2826.
[61] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, CRC Press, p. 2012.
[62] Conv2D - PyTorch master documentation, 2022, URL https://pytorch.org/docs/master/generated/torch.nn.Conv2d.html. (Accessed 22 December 2022).
[63] ConvTranspose2D - PyTorch master documentation, 2022, URL https://pytorch.org/docs/master/generated/torch.nn.ConvTranspose2d.html. (Accessed 22 December 2022).