Applied Intelligence (2025) 55:1095
https://doi.org/10.1007/s10489-025-06871-x

Generating survey databases with Wasserstein Generative Adversarial Networks

Hugues Annoye[1,2] · **Cédric Heuchenne**[1,2,3]

Received: 23 May 2025 / Accepted: 23 August 2025 / Published online: 11 November 2025

Abstract
In a world increasingly surrounded by data, data privacy and anonymization are becoming more and more important. Under these circumstances, the need for synthetic databases that replicate the characteristics of the population while preserving privacy is arising. In this article, we investigate how we can use Wasserstein Generative Adversarial Networks (WGANs), developed by Arjovsky et al. [1] in the context of image generation, to create synthetic survey databases. Survey data have both categorical and continuous variables and especially contain sampling weights that will be introduced in the proposed procedures. We evaluate the quality of the (generated) synthetic data through different indicators and especially a new practical and intuitive measure based on Support Vector Data Description (SVDD). All our analyses are achieved with the Labour Force Survey (LFS) data for Belgium.

Keywords Generative adversarial network · Wasserstein generative adversarial network · Synthetic data set · Wasserstein distance · Sampling weights · Support vector data description

1 Introduction

In every domain, there exists a significant demand for data, yet this demand is juxtaposed with the necessity for privacy, thereby complicating access to data. There is an undeniable necessity for methods of anonymization, particularly in light of recent legislative developments such as the General Data Protection Regulation (GDPR) in Europe or the California Consumer Privacy Act (CCPA). The development of synthetic databases presents a viable solution to this dilemma, the demand for such data increasing alongside the growing volume of existing data.

Generative models can be constructed to generate synthetic data sets that share characteristics with the original data. Generative Adversarial Networks (GANs) [2] and Variational Autoencoders (VAEs) [3] are the two primary methods used in the context of synthetic data generation. A third one approach involves the use of denoising diffusion models as used by [4, 5].

Survey data present several unique challenges that distinguish them fundamentally from tasks such as image or signal generation. First, survey data sets are rarely collected through simple random sampling due to cost constraints and nonresponse issues. As a result, each observation must be associated with a sampling weight, typically proportional to the inverse of its probability of selection, in order to obtain unbiased estimators. This aspect is critical and has no direct analogue in image or signal data. Further details and examples about sampling weights can be found in [6]. Second, survey data are typically tabular and include both continuous and categorical variables, which require distinct modeling strategies. Unlike image data, where spatial structure and local correlations justify the use of convolutional layers, tabular data do not benefit from such architectures. Therefore, our generative approach must be specifically adapted to the structure and statistical properties of survey

✉ Hugues Annoye
hugues.annoye@uclouvain.be

Cédric Heuchenne
cedric.heuchenne@uclouvain.be

[1] CAPE, UCLouvain Saint-Louis Bruxelles, 43 Boulevard du Jardin Botanique, Brussels 1000, Belgium

[2] ISBA, Université Catholique de Louvain, 20 Voie du Roman Pays, Louvain-la-Neuve 1348, Belgium

[3] HEC, Université de Liège, 14 Rue Louvrex, Liège 4000, Belgium

data. These differences highlight the need for specialized methodologies when generating synthetic survey data sets.

A comprehensive literature review providing a thorough exploration of the potential applications of GANs is presented by Cheng et al. [7]. However, GANs have already been utilized for privacy purposes. Some efforts have been made to apply GANs, originally developed for generating fake images, to tabular data. For instance, Xu and Veeramachaneni [8]employed GANs to forecast tabular data such as educational or medical records. Additionally, Camino et al. [9] devised techniques to apply GAN and Wasserstein Generative Adversarial Network (WGAN) models to handle multiple categorical variables using softmax. Furthermore, Walia et al. [10] employed the Wasserstein Conditional Generative Adversarial Network with Gradient Penalty (WCGAN-GP), utilizing the WGAN with the Gradient Penalty framework (WGAN-GP) introduced by Gulrajani et al. [11]. WCGAN-GP is specifically tailored for tabular data, where critics (discriminators) incorporate label information.

In this article, we incorporate the methodologies proposed by Gulrajani et al. [11] and Camino et al. [9] for tabular data, specifically addressing scenarios where original data sets are associated with sampling weights. The inclusion of these weights is essential to uphold the representativeness of synthetic databases, particularly given that a significant portion of data sets incorporate sampling weights. This necessity is particularly evident in European surveys like Statistics on Income and Living Conditions (EU-SILC), Household Budget Survey (HBS), and Labor Force Survey (LFS), among others.

Assessing the quality of our synthetic data set is another pivotal aspect of this article. In Borji [12, 13], various methods are compared for assessing the quality of the results generated by GANs, particularly focusing on images. In this article, we have opted to utilize several relevant metrics from these studies, propose a new method for assessing the reliability of generated data sets, utilizing Support Vector Data Description (SVDD), and additionally incorporate the Wasserstein distance algorithm introduced by Schuhmacher et al. [14]. The advantage of our SVDD-based measure is that it provides a percentage indicating the extent to which synthetic data deviate from the original data.

To summarize, the main contributions of this work are as follows:

- We propose algorithms for generating tabular survey data that account for sampling weights and incorporate various methodological advances from the literature. Additionally, sampling weights are integrated into all evaluation metrics used to assess the quality of the generated data sets.
- We compare the performance of our approach with other existing methods in the literature.
- Finally, we introduce a novel method for evaluating the quality of generated data, based on SVVD. This method has the advantage of providing a percentage that quantifies the extent to which the synthetic data deviate from the original data.

The structure of the article is outlined as follows. Firstly, in Sections 2 and 3, we offer background information about GANs and detail the methodologies that will be applied in this study. In Section 4, we introduce the different measurement techniques, including our new approach. Finally, in Section 5, we present an application wherein we generate a synthetic version of the Labour Force Survey (LFS) data set for Belgium and compare the results using the various techniques outlined in Section 4.

2 GAN: Literature review

Generative Adversarial Networks (GANs), initially proposed in the 2014 preprint by Goodfellow et al. [2], were devised to generate synthetic images. In this article, GANs are introduced as a two-player minimax game between two multilayer perceptrons:

$$\min_G \max_D \mathbb{E}_{X\sim p_{\text{data}}(x)} \left[\log\left(D\left(X\right)\right)\right] + \mathbb{E}_{Z\sim p_z(z)} \left[1 - \log\left(D\left(G(Z)\right)\right)\right], \tag{1}$$

where G is the perceptron called generator, which generates synthetic data from noise Z coming from a distribution p_z and D is the perceptron called discriminator, which aims to correctly differentiate true data X, coming from the distribution p_{data}, from synthetic data $G(z)$, as represented in Fig. 1.

Since their inception, GANs have been utilized in a growing number of applications, with image generation being the principal one. Additionally, Lu et al. [15] demonstrated their application in discrete time series analysis.

In the context of anonymization, the study described in Fang et al. [16] focuses on GANs for graphs, aiming to generate an anonymous graph with connection features similar to those of the original graph. For evaluation, they employed a combination of utility metrics and privacy metrics. In Feutry et al. [17], GANs are employed to build a representation$U$ from which a specific type of label Y can be extracted. The construction of U also aims to prevent the extraction of private information contained in other labels Z, leading to a trade-off between these two objectives. This approach is applied to images to preserve crucial information (e.g., sentiment) while anonymizing the image. In the study conducted by Bae et al. [18], GANs were employed

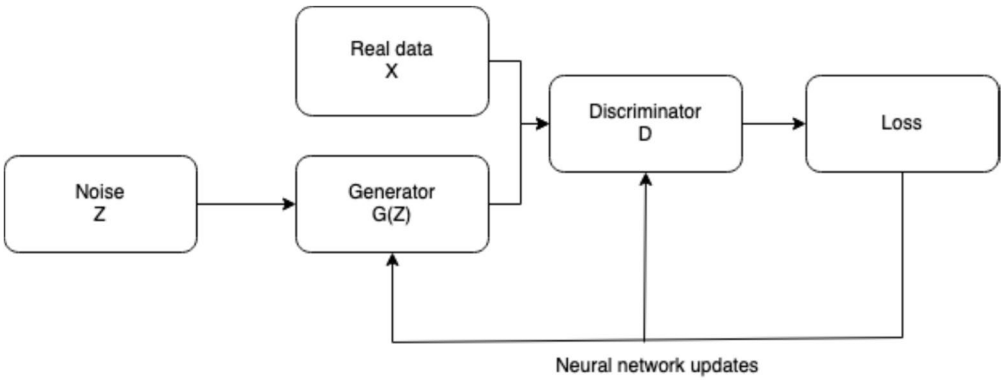

Fig. 1 Schematic representation of a GAN architecture. A random noise vector z is transformed by the generator G into a synthetic sample $G(z)$. The discriminator D receives both real data x and generated data $G(z)$, and outputs the probability that its input is real. During training, the two networks engage in a minimax game

to anonymize medical and DNA data by introducing noise through a generator from which the input is the true data*x* (and not only noise). The article outlines a procedure involving a generator (referred to as an encoder), a discriminator, and a target classifier. From an input x, an $\hat{x}$ is created. The learning objective of the encoder is to minimize the discriminator cross entropy (together with the distance between original and generated data) while maximizing the prediction score of the target classifier. The loss is calculated using the following formula:

$$L_E\left(\theta_E, x, r, \delta\right) = \lambda_e\left(\mathrm{d}\left(x, \hat{x}\right) * \delta\right) + \lambda_d\left(L_{D1} + L_{D2}\right) \tag{2}$$

where d is the Euclidean distance, δ controls the privacy level, L_{D1} is the loss of the discriminator (cross entropy) and L_{D2} is the target classifier score $C\left(f\left(\hat{x}\right)\right)$(with a pretrained perceptron, for instance, one can train it to predict diseases for patients). In the context of tabular data, we can cite Park et al. [19], an article that uses an adaptation of traditional Image-GAN (using 2-d convolutional networks) on table data sets. They transformed categorical data into matrices on which they use 2-d convolutional networks. This approach is criticized by Engelmann and Lessmann [20]. Indeed, convolutions are valuable for image processing but they appear unsuitable for tabular data where columns contain entirely heterogeneous content and their order is arbitrary. Engelmann and Lessmann [20] use GAN and WGAN to create tabular data but in the context of oversampling methods (like SMOTE). They apply a GAN to estimate the distribution of the data and after, the generator provides additional samples of the minority classes using the Conditional Generative Adversarial Network structure (CGAN). CGAN was also examined by Xu et al. [21] for its Conditional Tabular Generative Adversarial Network (CTGAN). In their approach, data are generated given each modality of certain categorical variables, which are randomly chosen. Moreover, the continuous variables are transformed through Gaussian mixture models according to their number of modes. Walia et al. [10] also proposed a technique based on Wasserstein GAN with gradient penalty, similar to the approach outlined in Section3.3, with the distinction that both the discriminator and generator are conditioned on additional information such as class labels (e.g., sick or not, payment default or not, etc.) and they do not use sampling weights. Vega-Márquez et al. [22] have also explored a similar concept in the case of a simple CGAN.

In [9], various methods adapted to multicategorical variables are compared. Two methods employ GANs. First, Gumble-GAN is a traditional GAN utilizing a Gumble softmax layer for each categorical variable:

$$\sigma(\mathrm{x})_i = \frac{\exp\left(\frac{\log(x_i)+g_i}{\tau}\right)}{\sum_{j=1}^{K}\exp\left(\frac{\log(x_j)+g_j}{\tau}\right)}, \quad i = 1, \cdots, K, \tag{3}$$

where, $\mathrm{x} = \left(x_1, \cdots, x_K\right)^T$ represents the input, $g_1, \cdots, g_K$ are *i.i.d.* random variables drawn from a $\mathrm{Gumble}\,(0,1)$ distribution, K is the number of levels of the categorical variable and τ is a positive hyperparameter. The second method involves a Wasserstein GAN utilizing the softmax layer:

$$\sigma(\mathrm{x})_j = \frac{\exp\left(x_j\right)}{\sum_{k=1}^{K}\exp\left(x_k\right)}, j = 1, \ldots, K. \tag{4}$$

It is somewhat similar to what we present in Sections 3.3 and 3.4, but with only categorical variables and without sampling weights. Additionally, they implemented the Gumbel softmax of (3) on models utilizing autoencoders. Depending on the data sets and metrics employed, it is not possible to identify the best method. However, the Wasserstein GAN using the softmax method emerges as one of the top-performing approaches.

3 WGAN with sampling weights

3.1 Wasserstein GAN

A well-known problem in the GAN literature is mode collapsing [23, 24]. It occurs when the generator produces data from a specific mode rather than from the entire original distribution. The Wasserstein GAN is a distinctive method within the GAN framework, initially developed by Arjovsky et al. [1]. One of the aims of this method is to counter this mode collapsing problem. This specific type of GAN utilizes the Wasserstein distance, also known as the Wasserstein-1 or Earth-Mover distance. Let$(\mathcal{X}, d)$ be a Polish space with metric d, and let $\mathbb{P}_r$ and $\mathbb{P}_\theta$ be two distributions in the space of probability measures defined on $\mathcal{X}$. Here, $\mathbb{P}_r$ (resp. $\mathbb{P}_\theta$) will denote the distribution of the original (resp. synthetic) data and we take $d(x, y) = \|x - y\|_2$. The Wasserstein distance between $\mathbb{P}_r$ and $\mathbb{P}_\theta$ is then given by

$$\begin{aligned} W\left(\mathbb{P}_r, \mathbb{P}_\theta\right) &= \inf_{\pi \in \Pi\left(\mathbb{P}_r, \mathbb{P}_\theta\right)} \mathbb{E}_{(X,Y)\sim\pi}\left[d\left(X, Y\right)\right] \\ &= \inf_{\pi \in \Pi\left(\mathbb{P}_r, \mathbb{P}_\theta\right)} \int_{\mathcal{X}\times\mathcal{X}} d(x, y)\mathrm{d}\pi(x, y), \end{aligned} \tag{5}$$

where $\Pi\left(\mathbb{P}_r, \mathbb{P}_\theta\right)$ denotes the set of all joint distributions $\pi\left(x, y\right)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_\theta$.

However, directly incorporating this distance into the GAN framework can pose challenges, particularly due to the computational and time-consuming nature of the Wasserstein distance. Yet, the following theorem can provide assistance.

Theorem 1 Let $\mathbb{P}_r$ be any distribution. Let $\mathbb{P}_\theta$ be the distribution of $g_\theta\left(Z\right)$, where Z is a random variable with density p and $g_\theta\left(\cdot\right)$ is a locally Lipschitz function such that, for a certain probability distribution p over $\mathcal{Z}$, there are local Lipschitz constants $L(\theta, z)$ such that $\mathbb{E}_{Z\sim p}\left[L\left(\theta, Z\right)\right] < +\infty$. Then, there is a solution $f : \mathcal{X} \to \mathbb{R}$ to the problem:

$$\max_{\|f\|_L \leq 1} \mathbb{E}_{X\sim\mathbb{P}_r}\left[f(X)\right] - \mathbb{E}_{X\sim\mathbb{P}_\theta}\left[f(X)\right] \tag{6}$$

and we have

$$\nabla_\theta W\left(\mathbb{P}_r, \mathbb{P}_\theta\right) = -\mathbb{E}_{Z\sim p}\left[\nabla_\theta f\left(g_\theta\left(Z\right)\right)\right] \tag{7}$$

when both terms are well-defined and where $\max_{\|f\|_L \leq 1}$ denotes the maximum on the class of 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$.

The proof Arjovsky et al. [1] of the theorem is grounded on three key points:

1. Kantorovich-Rubinstein duality that tell us that:

$$W\left(\mathbb{P}_r, \mathbb{P}_\theta\right) = \sup_{\|f\|_L \leq 1} \left(\mathbb{E}_{x\sim\mathbb{P}_r}\left[f(x)\right] - \mathbb{E}_{x\sim\mathbb{P}_\theta}\left[f(x)\right]\right). \tag{8}$$

   The proof of this duality can be found in Villani [25];
2. the envelope theorem;
3. the dominate convergence theorem.

The aim is to substitute the Wasserstein distance that needs to be minimized by the difference between the two expectations in (6). To accomplish this, it is imperative for our discriminator, commonly referred to as the critic in WGAN literature, to belong to the 1-Lipschitz class. It is worth noting that if f belongs to the K-Lipschitz class, the Kantorovich-Rubinstein duality becomes:

$$K \cdot W\left(\mathbb{P}_r, \mathbb{P}_\theta\right) = \sup_{\|f\|_L \leq K} \mathbb{E}_{x\sim\mathbb{P}_r}\left[f(x)\right] - \mathbb{E}_{x\sim\mathbb{P}_\theta}\left[f(x)\right], \tag{9}$$

and we approximate up to an irrelevant scaling factor.

To enforce this Lipschitz constraint in the literature, two techniques are available:

- the weight clipping, introduced for images by Arjovsky et al. [1];
- the gradient penalty, introduced for images by Gulrajani et al. [11].

3.2 WGAN with weight clipping

In this article, we adapt WGAN of Arjovsky et al. [1] by incorporating sampling weights, see Algorithm 1. The concept behind weight clipping is to confine the weights of neural networks to a restricted space$\mathcal{W}$, thereby ensuring that the critic f remains K-Lipschitz. In the following algorithms, $f_\eta\left(\cdot\right)$ denotes the critic neural network, sampling is made with replacement, w_i, $i = 1, \cdots, N$, is the sampling weight (such that their sum is equal to 1) and the chosen generalization of the stochastic gradient descent method is Adam algorithm [26]. ADAM is a first-order gradient-based optimization algorithm that uses exponentially smoothed gradient and gradient squared of the considered loss function. It combines the advantages of Root Mean Square Propagation (RMSProp), which stabilizes gradients across the iterations (by normalizing them) and Adaptive Gradient (AdaGrad),

which deals with sparse gradients by keeping sufficiently large learning rates for rare parameters (which especially appear in NLP where inputs are often sparse vectors/matrices). Unlike AdaGrad, ADAM deals with sparse gradients through their exponential smoothing. In our experiments, ADAM provided the best results.

Algorithm 1 WGAN with weight clipping and sampling weights

Require: α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic by generator iteration. β_1 and β_2 the parameters for the Adam algorithm.
Require: η_0, the initial parameters of the critic ($f_\eta(\cdot)$). θ_0, the initial parameters of the generator($g_\theta(\cdot)$). Chosen randomly.
while θ has not converged **do**
  for $t = 0, \cdots, n_{critic}$ **do**
    Sample $\left\{x^{(i)}\right\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the weighted (with w_i) empirical distribution.
    Sample $\left\{z^{(i)}\right\}_{i=1}^{m} \sim p(z)$ a batch of prior data.
    $\gamma_\eta \leftarrow \nabla_\eta \left[\frac{1}{m}\sum_{i=1}^{m} f_\eta\left(g_\theta\left(z^{(i)}\right)\right) - \frac{1}{m}\sum_{i=1}^{m} f_\eta\left(x^{(i)}\right)\right]$
    $\eta \leftarrow \text{Adam}\left(\eta, \gamma_\eta, \alpha, \beta_1, \beta_2\right)$
    $\eta \leftarrow \text{clip}\left(\eta, -c, c\right)$
  end for
  Sample $\left\{z^{(i)}\right\}_{i=1}^{m} \sim p(z)$ a batch of prior data.
  $\gamma_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_\eta\left(g_\theta\left(z^{(i)}\right)\right)$
  $\theta \leftarrow \text{Adam}\left(\theta, \gamma_\theta, \alpha, \beta_1, \beta_2\right)$
end while

Remark: Another solution is to draw samples from the unweighted emperical distributions of the real data and then use the weights (see Appendix A).

3.3 WGAN with gradient penalty

We implement the approach introduced by Gulrajani et al. [11]. The gradient penalty capitalizes on the fact that a differentiable function on a convex set is 1-Lipschitz if and only if its gradient has a norm smaller or equal to 1 everywhere. That explains the expression of$\Gamma^{(i)}$ in Algorithm 2 hereunder.

More details can be found in Gulrajani et al. [11]. One of the interests of this approach is that it allows for having fewer vanishing gradient problems[1]that occur when WGAN-CP has a small clipping parameter [1, 11]. It mitigates also the risk of exploding gradient[2][11].

[1] Vanishing gradient occurs when the gradients of the loss function (here in Algorithm 2) become extremely small during backpropagation, causing the neural weights to update minimally and hindering effective learning.

[2] Exploding gradient occurs when the gradients of the loss function (here in Algorithm 2) become extremely large.

Algorithm 2 WGAN with gradient penalty and sampling weights

Require: α, the learning rate. m, the batch size. n_{critic}, the number of iterations of the critic by generator iteration. λ, the gradient penalty parameter. β_1 and β_2 the parameters for the Adam algorithm.
Require: η_0, the initial parameters of the critic ($f_\eta(\cdot)$). θ_0, the initial parameters of the generator ($g_\theta(\cdot)$). Chosen randomly.
while θ has not converged **do**
  for $t = 0, \cdots, n_{critic}$ **do**
    Sample $\left\{x^{(i)}\right\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the weighted (with w_i) empirical distribution.
    Sample $\left\{z^{(i)}\right\}_{i=1}^{m} \sim p(z)$ a batch of prior data.
    for $i = 1, \cdots, m$ **do**
      Sample $\kappa \sim \text{U}(0, 1)$
      $\tilde{x}^{(i)} = g_\theta\left(z^{(i)}\right)$
      $\hat{x}^{(i)} = \kappa x^{(i)} + (1-\kappa)\,\tilde{x}^{(i)}$
      $\Gamma^{(i)} = f_\eta\left(\tilde{x}^{(i)}\right) - f_\eta\left(x^{(i)}\right) + \lambda\left(\left\|\nabla_{\hat{x}} f_\eta\left(\hat{x}^{(i)}\right)\right\|_2 - 1\right)^2$
    end for
    $\gamma_\eta \leftarrow \nabla_\eta\left[\frac{1}{m}\sum_{i=1}^{m}\Gamma^{(i)}\right]$
    $\eta \leftarrow \text{Adam}\left(\eta, \gamma_\eta, \alpha, \beta_1, \beta_2\right)$
  end for
  Sample $\left\{z^{(i)}\right\}_{i=1}^{m} \sim p(z)$ a batch of prior data.
  $\gamma_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_\eta\left(g_\theta\left(z^{(i)}\right)\right)$
  $\theta \leftarrow \text{Adam}\left(\theta, \gamma_\theta, \alpha, \beta_1, \beta_2\right)$
end while

3.4 Our GAN structure

First, all hidden layers in the generator utilize a LeakyReLU activation function with a specific ϵ hyperparameter. This activation function is defined as

$$h(x) = \begin{cases} x & \text{if } x > 0, \\ \epsilon x & \text{otherwise.} \end{cases} . \tag{10}$$

The use of LeakyReLU can help mitigate the vanishing gradient problems that can occur with sigmoidal or hyperbolic tangent activation functions [27]. Additionally, elastic net regularization is incorporated to ensure the generalization of the model and prevent overfitting. The output layer is tailored to accommodate both continuous and categorical variables. For continuous variables, the output activation function is LeakyReLU; for categorical variables, the softmax activation function in (4) is used, as proposed by Camino et al. [9]. Second, our architecture of the critic includes several key features. Similarly to the generator, all of its hidden layers employ a LeakyReLU activation function with a specificϵ parameter and elastic net regularization is applied to the model. Interestingly, unlike the conventional GAN where the output layer typically uses a sigmoid (standard logistic function) activation function, the output layer of our critic is a linear function.

Although dropout was experimented, it did not yield better results, so it was not included in the final architecture.

3.5 WGAN with quantile transformers

Others have employed quantile transformers within the framework of GANs [8] or VAEs [28]. In this article, we adapt the WGAN with gradient penalty to implement such a transformation. Our approach involves forcing all the continuous variables to adhere to a standard normal distribution by utilizing the quantile transformer formula$\Phi^{-1}\left(F\left(X_k\right)\right)$. Here, $F\left(X_k\right)$, the cumulative distribution of the continuous variable X_k, follows a uniform distribution, resulting in $\Phi^{-1}\left(F\left(X_k\right)\right)$ conforming to a normal distribution. In practice, for each point x_{kj}, we utilize the transformation:

$$\Phi^{-1}\left(\frac{\sum_{i=1}^{N}\left(w_i \mathbf{1}_{\text{rank}(x_{ki})\leq \text{rank}(x_{kj})}-0.5*w_j\right)}{\sum_{i=1}^{N} w_i}\right), \qquad (11)$$

where N is the size of the data set. Subsequently, we apply a min-max transformation to confine our variables within the range $[0, 1]$. We then employ a sigmoid function for the part of the output layer of the generator which handles the continuous variables. To distinguish this technique from the one outlined in Section 3.3, we will refer to it as WGAN-QT in the remainder of the article. The purpose is to make the neural networks more effective. Variables with more complex distributions (non-normal, skewed, long-tailed, etc.) can disrupt the balance of our training data, impacting the generalization and performance of our models. Additionally, multimodal distributions can exacerbate vanishing gradient problems, as pointed out by Pascanu et al. [29] and Jamotton and Hainaut [28].

3.6 Other GANs to compare the results

We compare the methods explained in Sections 3.2, 3.3 and 3.5 with CopulaGANs and CTGANs. CopulaGAN first transforms the data before applying CTGAN. CTGANs [21] use Wasserstein GAN with gradient penalty, Gumbel softmax functions (as in (3)), a conditional approach and Gaussian mixture models as explained in Section 2. The code of both (CTGAN and CopulaGAN) can be found on DataCebo Github of the Synthetic Data Vault Project.[3] CopulaGANs and CTGANs do not incorporate sampling weights. Additionally, we will assess the results by comparing them to those obtained from a vanilla GAN with a structure similar to the one described in Sections 3.4 and 5.2, except for the output layer of the discriminator. We will provide two versions: with and without sampling weights.

[3] https://github.com/sdv-dev/SDV

4 Methods of evaluation

In Section 4.1, we present the measures found in the literature that we have decided to use, and in Section 4.2, we put forward a new measure.

4.1 Literature review

In the studies by Borji [12] and Borji [13], various evaluation methods for GANs are outlined. From these papers, we have chosen the most suitable evaluation techniques for our specific task of generating survey data. We will also incorporate additional methods that appear relevant to our specific needs.

Among all these methods, we selected a subset based on their theoretical strengths and empirical performance. We include the Fréchet Inception Distance (FID) because it has demonstrated strong performance according to the meta-evaluation of GAN metrics in Borji [12]. We also report Density and Coverage, which, as noted by Borji [13], are more reliable than metrics such as Precision and Recall, while also being highly interpretable. The Kullback-Leibler divergence is included as it remains a de facto standard for evaluating generative models, as emphasized in Borji [12]. Finally, we provide results using the Wasserstein distance, which is a meaningful measure of discrepancy between multivariate distributions, and the Cramér-von Mises criterion, which we found useful for assessing how well the bivariate distributions are preserved during data generation.

4.1.1 Fréchet inception distance

Fréchet Inception Distance [30] is one of the widely used measures in the context of GANs, that consists in assuming that the distribution of the data is close to multivariate normal distributions$\mathcal{N}\left(m_1, \mathrm{C}_1\right)$ and $\mathcal{N}\left(m_2, \mathrm{C}_2\right)$, for the original and the generated data respectively. This leads to an easy calculable 2-Wasserstein distance:

$$FID = \|m_1 - m_2\|_2^2 + \mathrm{Tr}\left(\mathrm{C}_1 + \mathrm{C}_2 - 2\left(\mathrm{C}_1 \cdot \mathrm{C}_2\right)^{\frac{1}{2}}\right). \qquad (12)$$

Although widely used, assuming this normality is unrealistic in practice. In our case, we only employ this measure for continuous variables.

4.1.2 Density and coverage

Density and Coverage are proposed by Naeem et al. [31]:

$$\text{Density} = \frac{1}{k}\sum_{j=1}^{n}\sum_{i=1}^{n} w_i \mathbf{1}_{y_j \in B(x_i, NDD_k(x_i))}; \qquad (13)$$

$$\text{Coverage} = \sum_{i=1}^{n} w_i \mathbf{1}_{\exists j \text{ s.t.} y_j \in B(x_i, NDD_k(x_i))}; \tag{14}$$

where each x_i, $i = 1, \cdots, n$, represents a data point from a test data set (with weights w_i such that their sum is equal to 1), while each y_j, $j = 1, \cdots, n$, represents a synthetic data point. The notation $B(x, r)$ denotes a sphere in $\mathbb{R}^d$ (where d is the number of variables) centered at x with radius r, and $NDD_k(x_i)$ represents the distance from x_i to the k-th nearest neighbor among all x_t for $t \neq i$.

The fundamental concept involves constructing spheres around each real data point such that they contain the k real nearest neighbors. These spheres form a manifold around the original data set through their union. Then, for the density, we evaluate the average number of synthetic data points contained in each sphere divided by k. Meanwhile, for the coverage, we assess the proportion of spheres containing a point of the synthetic data set. The objective is to devise measures capable of assessing both the fidelity and diversity of the generated data set. These methods represent advancements over older techniques such as Precision and Recall, as proposed by Kynkäänniemi et al. [32], offering increased robustness against outliers and mode dropping.

4.1.3 Kullback-Leibler divergence

We also provide Kullback-Leibler divergence, computed utilizing a knn-tree method as described in Boltz et al. [33].

4.1.4 Wasserstein-1 distance

We also compute the Wasserstein-1 distance as defined in (5) using the method of Schuhmacher et al. [14]. The closer it is to zero, the more similar the two multivariate distributions are.

4.1.5 Cramér-von Mises

We employ the empirical Cramér-von Mises criterion introduced in Annoye et al. [6]:

$$\widetilde{\text{CVM}} = n \sum_{i=1}^{n} w_i \left[\hat{F}_n \left(x_i^1, x_i^2 \right) - \hat{G}_n \left(x_i^1, x_i^2 \right) \right]^2, \tag{15}$$

where $\hat{F}_n$ (resp. $\hat{G}_n$) is the weighted empirical bivariate cumulative distribution function of any pair of variables constructed with the original (resp. synthetic) data set and the sum is over their values in the original data set. We then average the $\widetilde{\text{CVM}}$s of all possible couples.

4.2 New measure using Support Vector Data Description

In this section, we propose to use Support Vector Data Description (SVDD, Tax and Duin [34]) to obtain a new similarity measure. First, we describe the SVDD methodology in Section 4.2.1. For implementary reasons, we resort to One-Class Support Vector Machine (OC-SVM) in Section 4.2.2 and its link with SVDD in Section 4.2.3. Next, we introduce the sampling weights in Section 4.2.4 and the proposed measure in Section 4.2.5.

4.2.1 Support Vector Data Description

At its core, SVDD operates on the assumption that the densest region of our data set is represented by a single hypersphere with unknown radius R and center ζ.

Mathematically, SVDD aims to solve the following problem [34]:

$$\min_{R \in \mathbb{R}, \zeta \in \mathcal{H}, \xi_1, \cdots \xi_n \in \mathbb{R}^+} R^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i \tag{16}$$

subject to

$$\|\phi(x_i) - \zeta\|_{\mathcal{H}}^2 \leq R^2 + \xi_i, \quad \xi_i \geq 0, \text{ for } i = 1, \cdots, n, \tag{17}$$

where ϕ is an implicit feature mapping (to apply the kernel trick) into an inner product space with norm $\|\cdot\|_{\mathcal{H}}$, ξ_is are slack variables which allow for some outliers (i.e. points out of the hypersphere) and $\nu \in]0, 1]$is a parameter controlling the influence of the slack variables. It is proved in [35] thatν is an upper bound on the fraction of outliers.

Constraints (17) can be incorporated into (16) by using Lagrange multipliers; the resulting dual problem becomes

$$\min_{\mathrm{a} \in \mathbb{R}^n} \sum_{i,j=1}^{n} a_i a_j k(x_i, x_j) - \sum_{i=1}^{n} a_i k(x_i, x_i) \tag{18}$$

subject to

$$\begin{aligned} &0 \leq a_i \leq \frac{1}{\nu n}, \quad i = 1, \cdots, n, \\ &\mathrm{e}^T \mathrm{a} = 1, \end{aligned} \tag{19}$$

where $\mathrm{e} = (1, \cdots, 1)^T$ and $\mathrm{a} = (a_1, \cdots, a_n)^T$ is a vector of Lagrange multipliers.

This leads to the following decision function [35]:

$$f(x) = R^2 - \|\phi(x) - \zeta\|_{\mathcal{H}}^2 = R^2 - k(x,x) + 2\sum_{i=1}^{n} a_i k(x, x_i) - \sum_{i,j=1}^{n} a_i a_j k(x_i, x_j). \quad (20)$$

In Fig. 2, we represent the resulting hypersphere (as if it were two-dimensional).

4.2.2 One-class support vector machine

OC-SVM is another algorithm for outlier detection. It consists in projecting the data in a hyperspace, where a hyperplane can be constructed to separate the normal points from the outliers. Given some vectors $x_i \in \mathbb{R}^n$, $i = 1, \cdots, n$, the primal problem of OC-SVM is [36]

$$\min_{\mathrm{v}\in\mathcal{H}, \xi_1, \cdots, \xi_n \in \mathbb{R}^+, \rho \in \mathbb{R}} \frac{1}{2}\mathrm{v}^T\mathrm{v} - \rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \quad (21)$$

subject to

$$\mathrm{v}^T\phi(x_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \cdots, n. \quad (22)$$

In this context, v denotes the vector of coefficients of the hyperplane, ρ indicates the offset of the hyperplane from the origin, ϕ represents a mapping in the hyperspace, and the ξ_i s stand for the slack variables.

The dual problem of (21) and (22) is

$$\min_{\mathrm{a}\in\mathbb{R}^n} \frac{1}{2}\mathrm{a}^T Q\mathrm{a} \quad (23)$$

subject to

$$0 \leq a_i \leq \frac{1}{\nu n}, \quad i = 1, \cdots, n, \quad \mathrm{e}^T\mathrm{a} = 1, \quad (24)$$

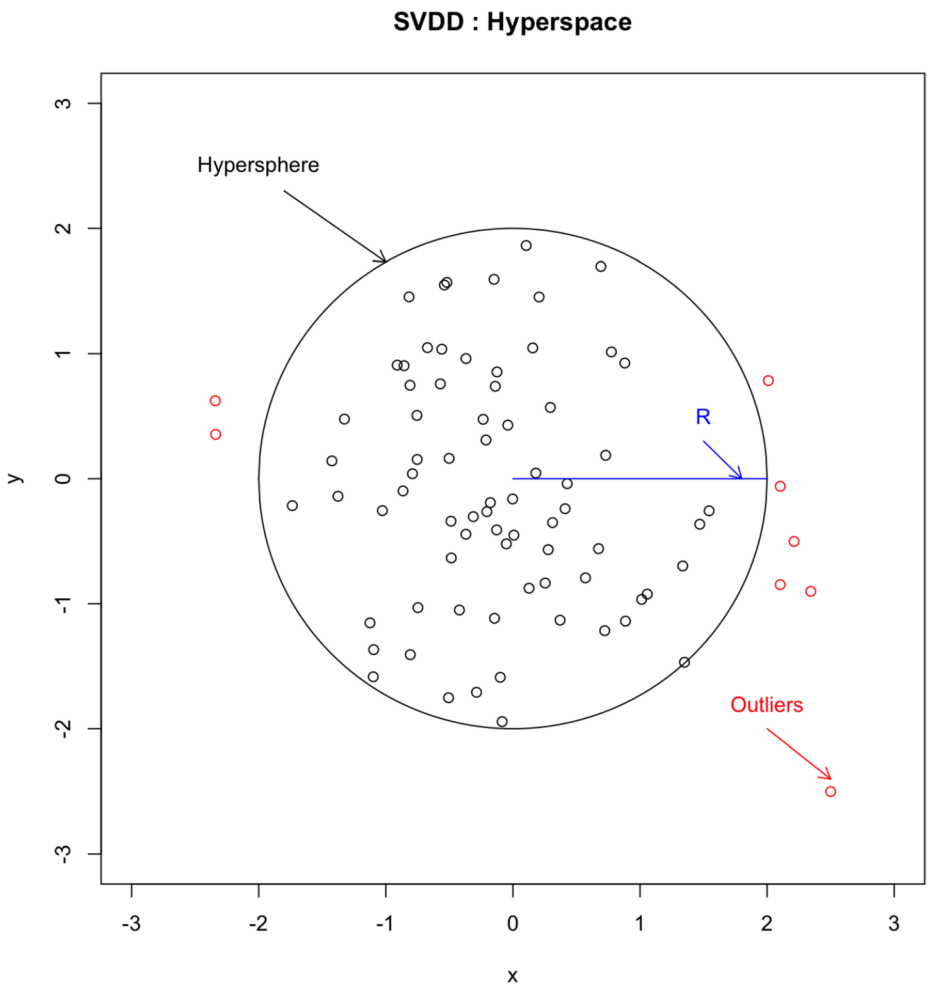

Fig. 2 Illustration ofHow an SVDD operates in a two-dimensional hyperspace. Although the projected data will live in a high-dimensional feature space induced by kernel, the geometric intuition is preserved in this 2 -D rendering

where Q_{ij}, the element (i, j) of the matrix Q is equal to $k(x_i, x_j)$. The decision function is then [36]

$$f(x) = \sum_{i=1}^{n} a_i k(x, x_i) - \rho. \tag{25}$$

4.2.3 Link between OC-SVM and SVDD

In practice, we opt for OC-SVM instead of SVDD, both algorithms having, with a Gaussian kernel, the same dual problem and thus the same a_is. For every $0 < a_j < \frac{1}{\nu n}$, ρ can be calculated from (25) with $x = x_j$. Next, from (20), we obtain

$$R^2 = 1 + \sum_{i,j=1}^{n} a_i a_j k(x_i, x_j) - 2\rho. \tag{26}$$

4.2.4 Sampling weights

If we introduce the sampling weights w_is, (16) and (21) become

$$\min_{R\in\mathbb{R},\zeta\in\mathcal{H},\xi_1,\cdots\xi_n\in\mathbb{R}^+} R^2 + \frac{1}{\nu}\sum_{i=1}^{n} w_i\xi_i \tag{27}$$

and

$$\min_{\mathrm{v}\in\mathcal{H},\xi_1,\cdots,\xi_n\in\mathbb{R}^+,\rho\in\mathbb{R}} \quad \frac{1}{2}\mathrm{v}^T\mathrm{v} - \rho + \frac{1}{\nu}\sum_{i=1}^{n} w_i\xi_i \tag{28}$$

respectively. This results in the same dual problems as in the unweighted case, except for the constraints in (19) that become

$$0 \le a_i \le \frac{w_i}{\nu}. \tag{29}$$

4.2.5 Proposed measure

Using SVDD, we calculate the radius in (26) for both the original and the generated data sets and compare them through the relative difference

$$\frac{R_1 - R_2}{R_1}, \tag{30}$$

where R_1 (resp. R_2) is the radius for the original (resp. generated) data set.

The significance of this measure lies in its expression as a ratio, offering a more effective basis for interpretation. Unlike the Wasserstein distance, which has a clear interpretation at 0 but lacks clarity for values such as 1 or 10, this measure provides a more intuitive and practical comparison because it represents the percentage of deviation between the two data sets.

5 Experiments and results

5.1 Implementation

We utilize a combination of R and Python softwares, incorporating Keras and Tensorflow packages, to execute our experiments. Python serves as the primary tool for machine learning tasks, while R is employed specifically for cross-validation and various analyses such as Wasserstein distance and SVDD.

We have autonomously developed GANs using Keras and Tensorflow, except for two variants based on conditional tabular GANs (CTGAN and CopulaGAN), for which we utilize the implementation provided by the SDV package in Python.

5.2 Experimental setup

We have divided the Labour Force Survey for Belgium 2019 (LFS, 146 variables) into two parts: two-thirds of the data for training and validation ($N = 24266$) and one-third for the test set ($n = 12133$).

On the first part, we employ a validation procedure to select the hyperparameters. This involves further dividing this set into two parts, with two-thirds for training and one-third for validation. The list of hyperparameters for the three Wasserstein GANs and the vanilla GANs can be found in Tables 1. Notice that, in the vanilla GANs, initial neural networks parameters are also chosen randomly, sampling is also made with replacement, $n_{critic} = 1$, a sigmoid function is used at the end of the discriminator, the loss function for both the generator and the discriminator is binary crossentropy, LeakyReLU and softmax functions are used as in the Wasserstein GANs and λ and c are not employed. To determine the optimal hyperparameters, we select the combination that minimizes the Wasserstein distance (5), employing the methodology developed in Schuhmacher et al. [14] for its calculation. Due to time limitations, we opt not to employ cross-validation in our process. Training neural networks of this nature is time-intensive, and incorporating cross-validation would extend the tuning period by 200%.

For the CTGAN and the CopulaGAN within the SDV package we have kept the default parameters.However,

Table 1 Hyperparameters for the Wasserstein and the vanilla GANs; final, final estimation (with N data) with optimal hyperparameters; one epoch corresponds to $\frac{N}{m}$ batches in the update part of the generator

Hyperparameter	Type of search
Number of hidden layers	Fixed: 2
Number of neurons by hidden layer generator	Random search
Number of neurons by hidden layer discriminator	Random search
ϵ, LeakyReLU	Random search
α, learning rate	Random search
λ_{1g}, elastic net penalty of the generator loss (lasso)	Random search
λ_{2g}, elastic net penalty of the generator loss (ridge)	Random search
λ_{1d}, elastic net penalty of the discriminator loss (lasso)	Random search
λ_{2d}, elastic net penalty of the discriminator loss (ridge)	Random search
Epoch	Fixed: 500 (validation) 2000 (final)
m, batch size	Fixed: 256 (validation) 128 (final)
β_1, Adam	Fixed: 0.5
β_2, Adam	Fixed: 0.9
n_{critic}	Fixed: 5
z	Fixed: $\mathcal{N}(0, \mathrm{I})$, with I the identity matrix of dimension 15.
λ, gradient penalty parameter	Random search
c, clipping parameter	Random search

Table 2 Hyperparameters for the CTGAN and the CopulaGAN

Hyperparameter	Value
Number of hidden layers	Fixed: 2
Number of neurons by hidden layer generator	Fixed: (256, 256)
Number of neurons by hidden layer discriminator	Fixed: (256, 256)
ϵ, LeakyReLU	Fixed: 0.2
α, learning rate	Fixed: $2 \cdot 10^{-4}$
λ_{2g}, ridge penalty of the generator loss	Fixed: $1 \cdot 10^{-6}$
λ_{2d}, ridge penalty of the discriminator loss	Fixed: $1 \cdot 10^{-6}$
Epoch	Fixed: 300
m, batch size	Fixed: 500
β_1, Adam	Fixed: 0.5
β_2, Adam	Fixed: 0.9
n_{critic}	Fixed: 1
z	Fixed: $\mathcal{N}(0, \mathrm{I})$, with I the identity matrix of dimension 128.
λ, gradient penalty parameter	Fixed: 10
τ, Gumble softmax parameter	Fixed: 0.2

these two methods take 13 times longer[4](than the above GANs) to be executed, making it challenging to fine-tune. These default values are used in Xu et al. [21] and can be found in Table 2.

Next, for each GAN model, we generate a synthetic data set that matches the size of the test set. We then compare these synthetic and original data to assess their similarities and differences.

5.3 Results

The SVDD measure is provided for all the variables and also exclusively for continuous variables, both with ν equal to 0.05 and a Gaussian kernel with γ equal to one divided by the number of variables:

$$k\left(x_i, x_j\right) = exp\left(\gamma \left\|x_i - x_j\right\|_2^2\right). \tag{31}$$

In Table 3, we compute all the quantities presented in Section 4. To prevent certain variables from dominating the measure, we have standardized all the continuous variables.

WGAN-GP appears to yield the most favorable results for Wasserstein distance, *FID* and $\widetilde{SVDD}/\widetilde{SVDD}_{cont}$ while WGAN-QT has the lowest average CVM and *KL*. Techniques employing sampling weights generally outperform CTGAN and CopulaGAN (except for the vanilla GAN with sampling weights). Among the three WGAN methods with sampling weights, those using gradient penalty perform better than the one using the clipping constraint. Unsurprisingly, the worst measures are obtained for the two vanilla GANs.

Both *SVDD* and $SVDD_{cont}$ measures appear quite interesting. Notably, some differences are substantial, whereas the variations in terms of Wasserstein distance are relatively small. Both vanilla GANs perform poorly for *SVDD* while the CTGAN and CopulaGAN have better results than WGAN-QT (WGAN-QT, however, surpasses CTGAN and CopulaGAN for all other measures). Observing the performances of WGAN-QT and WGAN-CP for *SVDD* and $SVDD_{cont}$, they appear to exhibit suboptimal results compared to WGAN-GP. Note that both methods (WGAN-CP and WGAN-QT) have negative *SVDD* and $SVDD_{cont}$, implying that the synthetic data set has a certain number of points outside the original data set support, which is not the case with the other techniques.

Next, we provide some graphs about Coverage and Density for our different data sets. In Fig. 3 (resp. Fig. 4), we illustrate Density (resp. Coverage) as a function of the

[4] See section 5.3

Table 3 Results: Wasserstein, Wasserstein-1 distance; *SVDD* (resp. $SVDD_{cont}$), relative difference of the radii (30) for all the variables (resp. the continuous variables); *KL*, Kullback-Leibler divergence; Av. $\widetilde{\text{CVM}}$, average of the $\widetilde{\text{CVM}}$; GAN weighted (resp. GAN), vanilla GAN with (resp. without) sampling weights

Measure	WGAN-GP	WGAN-CP	WGAN-QT	CTGAN	CopulaGAN	GAN weighted	GAN
Wasserstein	14.59	14.74	14.83	14.90	14.85	15.295	15.22
SVDD	0.001	−0.068	−0.035	0.020	0.017	0.195	0.250
$SVDD_{cont}$	0.001	−0.015	−0.006	0.001	−0.007	0.012	0.007
KL	6301	6243	6098	6317	6319	6501	6814
FID	0.17	1.50	2.28	6.55	7.45	29.06	26.50
Av. $\widetilde{\text{CVM}}$	0.005	0.007	0.002	0.015	0.012	0.048	0.061

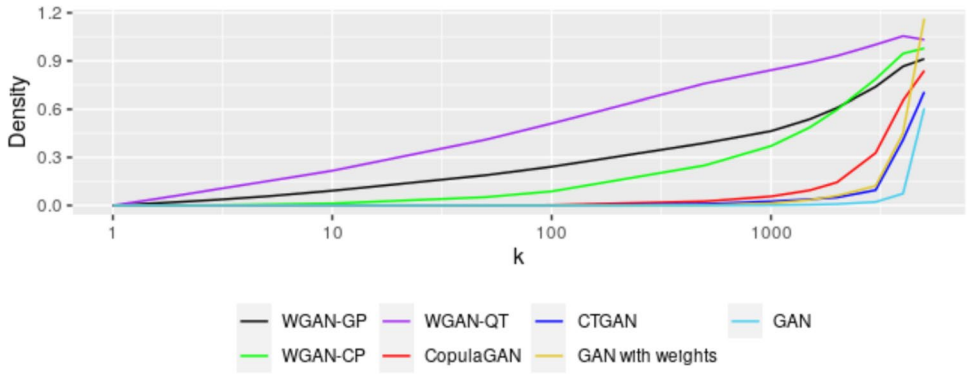

Fig. 3 Density of the different synthetic data sets with respect to the number of nearest neighbors k used to define local spheres

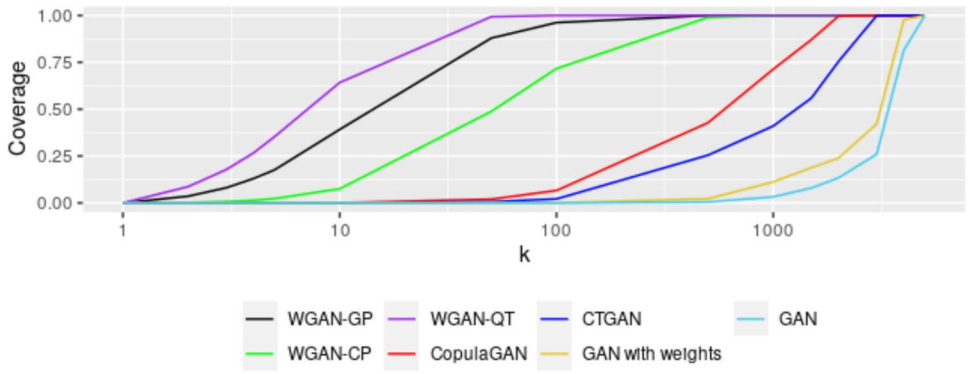

Fig. 4 Coverage of the different synthetic data sets with respect to the number of nearest neighbors k used to define local spheres

number of neighbors within each ball, ranging up to 5000 (on a logarithmic scale).

We observe that WGAN-QT and WGAN-GP provide the best results while the vanilla GANs perform poorly. CTGAN and CopulaGAN have lower Coverage and Density than the Wasserstein GANs with sampling weights but outperform vanilla GANs (mainly for Coverage).

In Fig. 4, WGAN-QT and WGAN-GP exhibit good Coverage for small balls with $k = 100$ while CTGAN and Copula GAN (resp. vanilla GANs) reach this Coverage level for k between 2000 and 3000 (resp. k close to 5000). As above, methods without sampling weights underperform compared to the weighted approaches (more precisely, Wassertein GANs (resp. vanilla GAN) with weights outperform CTGAN and CopulaGAN (resp. vanilla GAN) without weights).

Moreover, we also verified that none of the individuals within our synthetic data set were duplicates of individuals in our original data set and this was indeed confirmed. However, for privacy reasons, if such a duplicate appeared, a new synthetic individual would be generated and compared to the original ones.

Computational time *On a server equipped with 80 Intel(R) Xeon(R) Gold 6138 CPUs @ 2.00GHz, training WGAN-GP for 500 epochs over 16,178 records took approximately 50 minutes on average during the validation process. In comparison, WGAN-QT required around 1 hour. This poses a limitation, as the training time must be multiplied by the number of hyperparameter configurations being tested. For context, other methods such as CTGAN, using the default SDV settings (300 epochs), require approximately 13 hours to train. A potential improvement for all methods would be*

to utilize a GPU, which could significantly reduce training time.

Limitations *Another limitation is that WGANs may not be the most suitable choice for imputing certain variables when others are fixed. This is because the algorithm may downweight or ignore the influence of the conditioning variables, leading to generated data that does not align well with the specified conditions. Since the conditioning information is not explicitly embedded in the neural network architecture, the samples generated under a given condition may not resemble similar instances in the original dataset. In scenarios where conditional imputation is required, we believe that a variational autoencoder or a denoising diffusion model may provide a more effective solution.*

Sum-up *To sum-up, applied to our data, WGAN-QT produces results similar (sometimes better) to WGAN-GP and both outperform the other methods. However, WGAN-QT is more time-consuming due to the quantile transform. In Appendix B, we provide frequency (resp. estimated density) plots to compare original and synthetic data for some categorical variables (resp. one continuous variable). Graphs for other variables lead to similar observations. Finally, in Appendix C, we provide an additional study on the standard deviation of the results in Table 3.*

6 Conclusion

In this paper, we apply various data generation techniques using different innovations of the Wasserstein GAN. We have adjusted our algorithms to consider sampling weights and have also introduced new measures to evaluate quality of the generated data.

In our experiments, we have seen that the methods using gradient penalty (WGAN-GP and WGAN-QT) lead to the best results (using various metrics). Both algorithms use sampling weights and the Wasserstein distance, and outperform the other tested techniques that do not use them. WGAN-QT seems to have results that are comparable to WGAN-GP (slightly better or slightly worse, depending of the measures) but it takes more time. Furthermore, the SVDD measure we have computed appears promising. From the kernel trick, the SVDD algorithm provides a frontier that surrounds the data points more precisely than, for example, a hypersphere with a dimension equal to the number of variables. Next, as it reflects the relative difference between two radii (for the hyperspheres encompassing the original and synthetic data set, respectively), this measure can be relevant for the practitioner who has at disposal a percentage of deviation of the generated data with respect to the original data. It is more intuitive than distance measures (like, for example, the Wasserstein distance) which can make it challenging to determine whether the obtained quantity is substantial or not. In our experiments, WGAN-GP achieves the best SVDD (and the lowest Wasserstein distance).

Other GAN techniques remain unexplored in our investigation. For instance, Baowaly et al. [37] compare GAN, WGAN and BGAN (Boundary-Seeking GAN, where the generator is trained to produce data points lying on the decision boundary of the discriminator) using medical data. BGAN appears to outperform others in their study, although WGAN also performs well. Another example is provided by Bellemare et al. [38] and Mottini et al. [39], who utilize Cramér-GAN as an alternative to Wasserstein GAN. This technique employs the energy distance:

$\mathcal{E}(P,Q) := 2\mathbb{E}\|X-Y\|_2 - \mathbb{E}\|X-X'\|_2 - \mathbb{E}\|Y-Y'\|_2$

where X, X' (resp. Y, Y') are independent random variables distributed according to P (resp. Q). The interest of this distance lies in its unbiased sample gradient, unlike the Wasserstein distance.

Another limitation of our method is that it does not provide formal differential privacy guarantees. Some studies have explored incorporating differential privacy into GANs and WGANs using gradient clipping and noise addition [40, 41]. These works show that, in the case of WGAN-CP, one can ensure differential privacy by adding Gaussian noise of the form $\mathcal{N}\left(0, \sigma_n^2 c_g^2 I\right)$ at line 5 of Algorithm 1, where c_g and $\sigma_n^2$are constants as defined in [40]. Further research could extend this approach to models such as WGAN with gradient penalty. In that case, appropriate values forc_g and σ_n must be determined to suit the gradient penalty framework, allowing noise to be added (e.g., at line 11 of Algorithm 2) to ensure (ϵ, δ)-differential privacy.

An interesting addition to our analysis could involve comparing more our GAN algorithms with techniques based on Variational Autoencoders, which represent another prominent approach in the synthetic data generation literature. Especially in the case of conditional imputation to other variables. Another potential avenue for research involves improving the exploration of hyperparameters and enhancing the time efficiency of GANs, which currently represent a general bottleneck.

A Alternative algorithm

Here we provide an alternative to Algorithm 1:

Algorithm 3 WGAN-CP with weight: Method 2

Require: α the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic by generator iteration. β_1 and β_2 the parameters for the Adam algorithm.
Require: η_0, the initial parameters of the critic ($f_\eta(\cdot)$). θ_0, the initial parameters of the generator ($g_\theta(\cdot)$). Chosen randomly.
while θ has not converged **do**
  for $t = 0, \cdots, n_{critic}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the unweighted empirical distribution.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior data.
    $\gamma_\eta \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_\eta\left(g_\theta\left(z^{(i)}\right)\right) - \frac{1}{\sum_{i=1}^{m} w_i}\sum_{i=1}^{m} w_i f_\eta\left(x^{(i)}\right)\right]$
    $\eta \leftarrow \text{Adam}\left(\eta, \gamma_\eta, \alpha, \beta_1, \beta_2\right)$
    $\eta \leftarrow \text{clip}\left(\eta, -c, c\right)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior data.
  $\gamma_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_\eta\left(g_\theta\left(z^{(i)}\right)\right)$
  $\theta \leftarrow \text{Adam}\left(\theta, \gamma_\theta, \alpha, \beta_1, \beta_2\right)$
end while

B Graphs

In this section, we provide some graphs (Figs. 5, 6, 7, 8 and 9) comparing the original test set with a synthetic data set constructed using WGAN-GP. Plots with synthetic data generated with WGAN-QT are similar.

C Standard deviation

In this section, we generate one hundred synthetic data sets using each method proposed in this chapter. Then, we study the standard deviation of the measures in Table 3 across the different synthetic data sets, except for the average $\widetilde{\text{CVM}}$ due to time constraints. We observe that all methods have similar standard deviations for the Wasserstein distance as well as for FID. For $SVVD$ and $SVDD_{cont}$, the standard deviations are quite small, confirming that the vanilla GANs underperform compared to the other methods and that WGAN-GP has an $SVDD$ value close to 0. Finally, the Kullback-Leibler divergence, although it may initially appear larger, is actually quite small relative to the value of the measure.

These standard deviations show that the different predictions made with the same method all have similar distances to the original data Table 4.

D Variational autoencoder

We present here the same evaluation metric as in Section 5, this time using a Variational Autoencoder (VAE) [3]. The hyperparameters listed in Table 5 are selected using the same validation procedure based on the Wasserstein distance, described in Section 5.2.

We implement the approach described in Algorithm 4, incorporating sampling weights into the training process. For both the encoder and decoder, all hidden layers use the LeakyReLU activation function. The output layer of the decoder combines LeakyReLU and softmax activations, following the WGAN architecture.

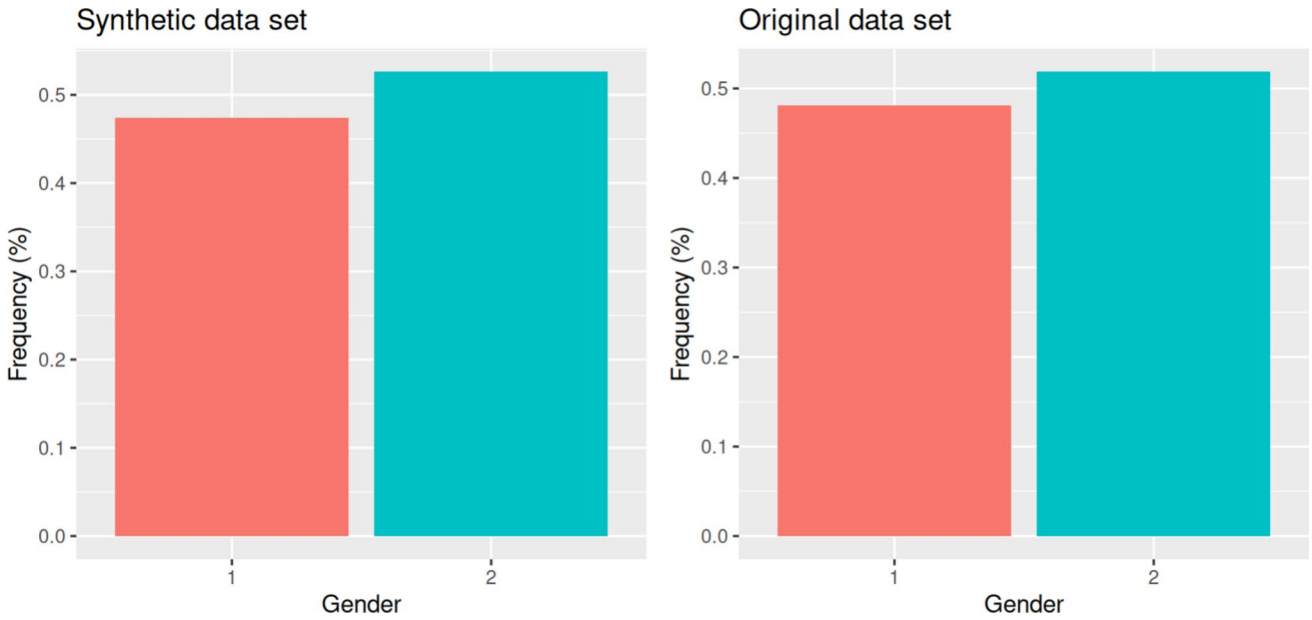

Fig. 5 Comparison of the f frequency of each gender in the synthetic (above) and original (below) data. *x*-axis: 1, male; 2, female

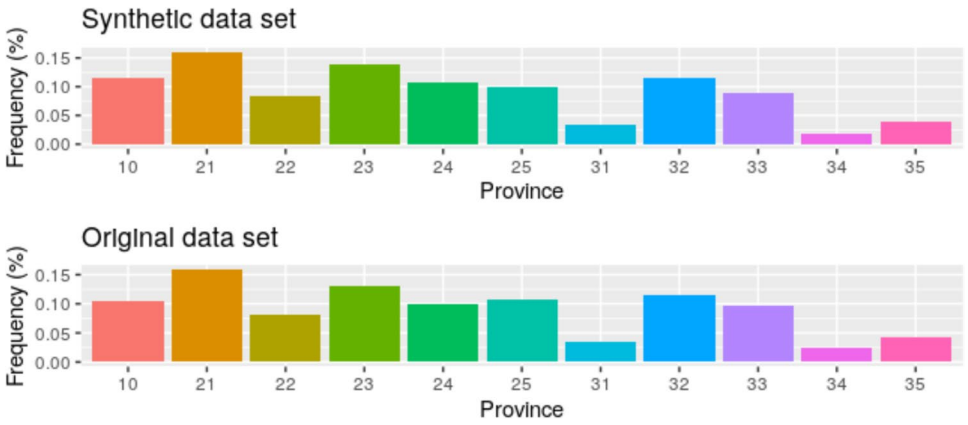

Fig. 6 Comparison of the f frequency of people in each province between synthetic (above) and original (below) data. *x*-axis, label of the province in LFS

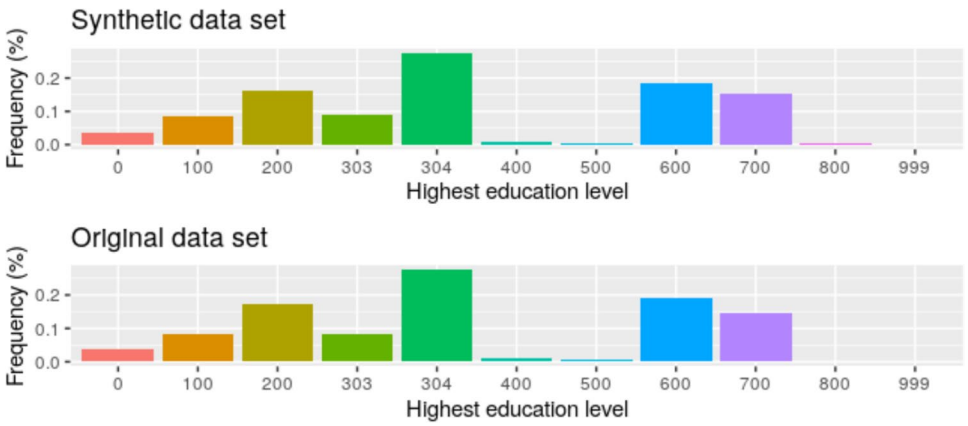

Fig. 7 Comparison of the frequency of people in each educational level between synthetic (above) and original (below) data. *x*-axis, label of the education level in LFS

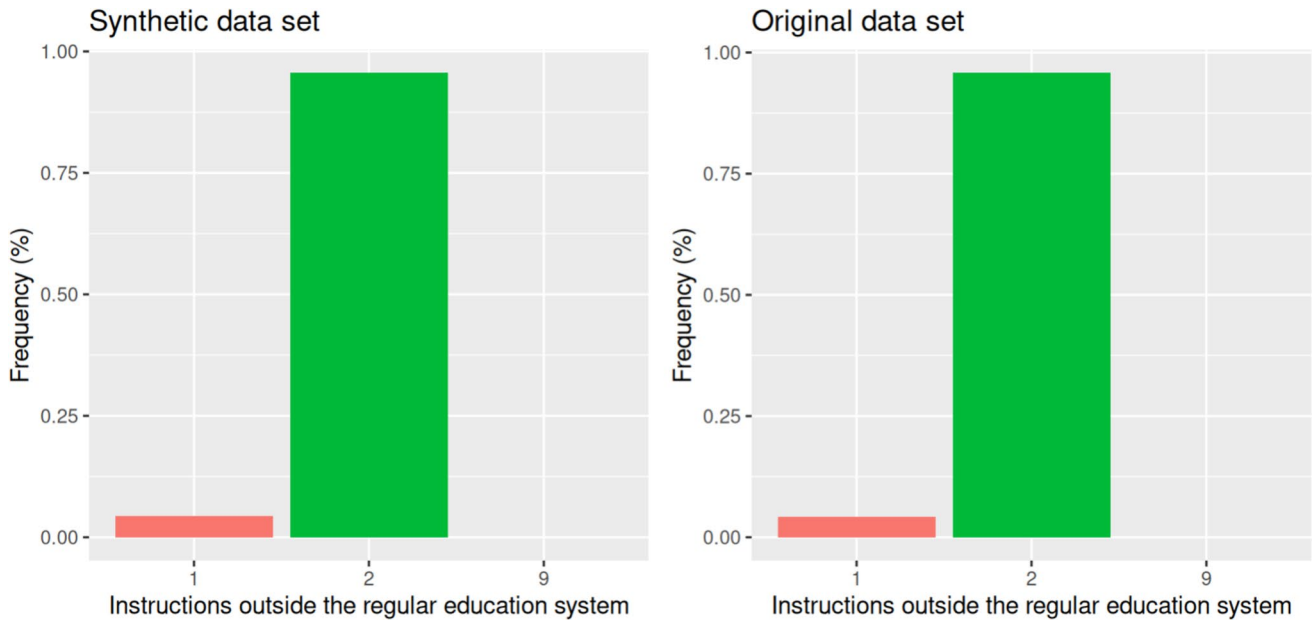

Fig. 8 Comparison of the frequency of people with instruction outside the regular education system between synthetic (above) and original (below) data. *x*-axis: 1, outside; 2, inside; 9, not applicable

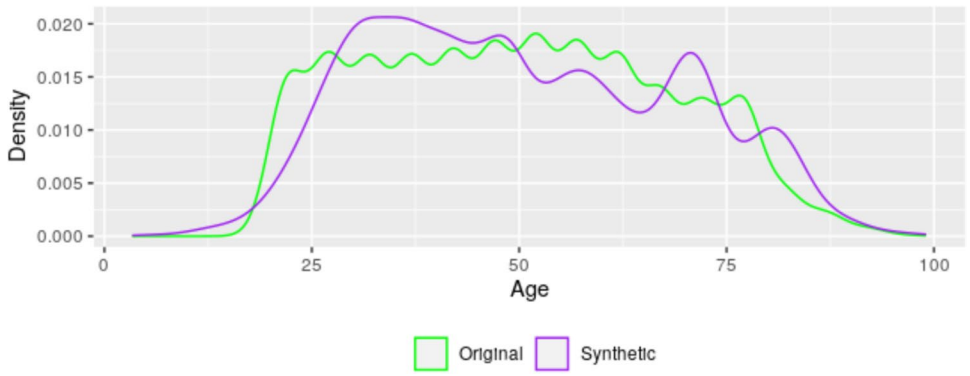

Fig. 9 Estimated density of age in the synthetic (purple) and original (green) data

Table 4 Standard deviation of the results over 100 generations: Wasserstein, Wasserstein-1 distance; FID, Fréchet Inception Distance; *SVDD* (resp. $SVDD_{cont}$), relative difference of the radii (30) for all the variables (resp. the continuous variables); *KL*, Kullback-Leibler divergence; GAN weighted (resp. GAN), vanilla GAN with (resp. without) sampling weights

Measure	WGAN-GP	WGAN-CP	WGAN-QT	CTGAN	CopulaGAN	GAN weighted	GAN
Wasserstein	0.0019	0.0016	0.0017	0.0019	0.0019	0.0010	0.0010
SVDD	0.0030	0.0045	0.0041	0.0033	0.0033	0.0040	0.0029
$SVDD_{cont}$	0.0006	0.0003	0.0003	0.0004	0.0004	0.0005	0.0004
KL	1.0216	1.4261	4.6379	0.02910	0.1161	0.0694	0.0180
FID	0.0061	0.0154	0.0336	0.0345	0.0340	0.0432	0.0384

Table 5 Hyperparameters for the VAE; final, final estimation (with N data) with optimal hyperparameters; one epoch corresponds to $\frac{N}{m}$ batches in the update part of the generator

Hyperparameter	Type of search
Number of hidden layers	Fixed: 2
Number of neurons by hidden layer decoder	Random search
Number of neurons by hidden layer encoder	Random search
ϵ, LeakyReLU	Random search
α, learning rate	Random search
λ_{1d}, elastic net penalty of the decoder loss (lasso)	Random search
λ_{2d}, elastic net penalty of the decoder loss (ridge)	Random search
λ_{1e}, elastic net penalty of the encoder loss (lasso)	Random search
λ_{2e}, elastic net penalty of the encoder loss (ridge)	Random search
Epoch	Fixed: 500 (validation) 2000 (final)
m, batch size	Fixed: 256 (validation) 128 (final)
β_1, Adam	Fixed: 0.5
β_2, Adam	Fixed: 0.9
ι	Fixed: $\mathcal{N}(0, \mathrm{I})$, with I the identity matrix of dimension 15.

As shown in Table 6, the results are promising, though they do not outperform those obtained using the WGAN-GP approach.

Table 6 Results: Wasserstein, Wasserstein-1 distance; *SVDD* (resp. $SVDD_{cont}$), relative difference of the radii of (30) for all the variables (resp. the continuous variables); *KL*, Kullback-Leibler divergence; Av. $\widehat{\mathrm{CVM}}$, average of the $\widehat{\mathrm{CVM}}$; VAE weighted, VAE with sampling weights

Measure	VAE weighted
Wasserstein	14.63
SVDD	-0.007
$SVDD_{cont}$	-0.002
KL	6305
FID	0.89

Algorithm 4 VAE with sampling weights

Require: α, the learning rate. m, the batch size. β_1 and β_2 the parameters for the Adam algorithm.
Require: η_0, the initial parameters of the encoder ($e_\eta(\cdot)$). θ_0, the initial parameters of the decoder ($d_\theta(\cdot)$). Chosen randomly.
while (θ, η) has not converged **do**
  Sample $\boldsymbol{x} = \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the weighted (with w_i) empirical distribution.
  Sample $\left\{\boldsymbol{\iota}^{(i)}\right\}_{i=1}^{m} \sim p(\iota)$ a batch of prior data.
  $\boldsymbol{\mu}, \log\left(\boldsymbol{\sigma}^2\right) = e_\eta(\mathbf{x})$
  $\boldsymbol{\sigma} = \exp(0.5 \times \log(\boldsymbol{\sigma}^2))$
  for $i = 1, \cdots, m$ **do**
    Using reparameterization trick: $\boldsymbol{z}^{(i)} = \boldsymbol{\mu} + \boldsymbol{\sigma} \times \boldsymbol{\iota}^{(i)}$
    $\hat{\boldsymbol{x}}^{(i)} = d_\theta(\boldsymbol{z}^{(i)})$
  end for
  $L_r = \frac{1}{m}\sum_{i=1}^{m}\left\|\boldsymbol{x}^{(i)} - \hat{\boldsymbol{x}}^{(i)}\right\|^2$
  $L_{KL} = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{d} -0.5\left(1 + \log\left(\sigma_{ij}^2\right) - \mu_{ij}^2 - \sigma_{ij}^2\right)$
  $L_{Total} = L_r + L_{KL}$
  $\Gamma_{(\theta,\eta)} = \nabla_{(\theta,\eta)} L_{Total}$
  $(\theta, \eta) \leftarrow \mathrm{Adam}\left((\theta, \eta), \Gamma_{(\theta,\eta)}, \alpha, \beta_1, \beta_2\right)$
end while

Table 7 Decomposition of FID, along with differences in SVDD hypersphere radii and center norms; GAN weighted (resp. GAN), vanilla GAN with (resp. without) sampling weights; VAE weighted, Variational autoencoder with sampling weights; FID_1 location part of *FID* as defined in (32), FID_2 scaling part of *FID* as defined in (33); R_1 (resp. R_2) and $\|\zeta_1\|$ (resp. $\|\zeta_2\|$) radius and norm of the center of the hypersphere of the SVDD for the original (resp. synthetic) data set; cont., for continuous variables

Measure	WGAN-GP	WGAN-CP	WGAN-QT	CTGAN	CopulaGAN	GAN weighted	GAN	VAE weighted
FID_1	7×10^{-25}	1×10^{-24}	7×10^{-25}	7×10^{-25}	7×10^{-25}	3×10^{-19}	1×10^{-17}	1×10^{-24}
FID_2	0.17	1.50	2.28	6.55	7.45	29.06	26.50	0.89
$\|\zeta_1\| - \|\zeta_2\|$	0.0050	0.0361	0.0248	0.0050	0.0077	-0.0357	-0.0513	0.0077
$R_1 - R_2$	0.0004	-0.0297	-0.0153	0.0088	0.0075	0.0859	0.1102	-0.0028

E Decompostition of some measures

E.1 Decomposition of the FID

As explained in [42], the first part of

$$FID_1 = \|m_1 - m_2\|_2^2 \tag{32}$$

is a location term, while the second part,

$$FID_2 = \mathrm{Tr}\left(\mathrm{C}_1 + \mathrm{C}_2 - 2\left(\mathrm{C}_1 \cdot \mathrm{C}_2\right)^{\frac{1}{2}}\right). \tag{33}$$

is a scaling term. In Table 7, we provide the decomposition of the FID computed on the continuous variables from Tables 3 and 6. We observe that the bias term is very small across all methods, with vanilla GANs showing a comparatively higher, though still negligible, bias. This indicates a very small bias for continuous variables, especially in comparison to the variance.

E.2 Difference of the norm of the centers

When performing SVDD, it can be difficult to compute the exact center but it is relatively straightforward to calculate its norm:

$$\|\zeta\|^2 = \sum_{i,j=1}^{n} a_i a_j k\left(x_i, x_j\right). \tag{34}$$

We compare the centers using the following formula:

$$\|\zeta_1\| - \|\zeta_2\|, \tag{35}$$

where ζ_1 (resp. ζ_2) is the center of the hypersphere for the original (resp. synthetic) data set.

In Table 7, we present the values of the quantity defined in (35) as well as the difference of the radii for the different methods. We observe that the difference of the center is small across all methods. The largest deviation, in absolute value, occurs with the GAN variant that does not incorporate sampling weights. This also indicates that the bias is quite small for both continuous and categorical variables.

Funding This work was partially funded by European Regional Development Fund (ERDF, project 2021BE16RFPR001-T-11-05).

Data Availability This article uses the 2019 Belgian Labour Force Survey. One of the purposes of the study is to develop methods for anonymizing confidential data sets; therefore, we use the research version of the microdata. Access to these data is available for scientific research purposes upon request, following the procedure described at: https://statbel.fgov.be/en/about-statbel/what-we-do/microdata-research.

References

1. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein generative adversarial networks. In: International conference on machine learning, pp 214–223. PMLR
2. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2020) Generative adversarial networks. Commun ACM 63(11):139–144
3. Kingma DP, Welling M (2022) Auto-encoding variational bayes. arXiv:1312.6114 [stat.ML]
4. Ho J, Jain A, Abbeel P (2020) Denoising diffusion probabilistic models. Advances in neural information processing systems, vol 33, pp 6840–6851
5. Kotelnikov A, Baranchuk D, Rubachev I, Babenko A (2023) Tabddpm: modelling tabular data with diffusion models. In: International conference on machine learning, pp 17564–17579. PMLR
6. Annoye H, Beretta A, Heuchenne C (2024) Statistical matching using kernel canonical correlation analysis and super-organizing map. Expert Syst Appl 123134. https://doi.org/10.1016/j.eswa.2023.123134
7. Cheng J, Yang Y, Tang X, Xiong N, Zhang Y, Lei F (2020) Generative adversarial networks: a literature review. KSII Trans Internet Inf Syst 14(12)
8. Xu L, Veeramachaneni K (2018) Synthesizing tabular data using generative adversarial networks. arXiv:1811.11264 [cs.LG]
9. Camino R, Hammerschmidt C, State R (2018) Generating multi-categorical samples with generative adversarial networks. arXiv:1807.01202 [stat.ML]
10. Walia M, Tierney B, McKeever S (2020) Synthesising tabular data using Wasserstein conditional gans with gradient penalty (wcgan-gp). In: AICS, pp 325–336
11. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A (2017) Improved training of Wasserstein gans. Advances in neural information processing systems, vol 30

12. Borji A (2019) Pros and cons of gan evaluation measures. Comput Vis Image Underst 179:41–65
13. Borji A (2022) Pros and cons of gan evaluation measures: new developments. Comput Vis Image Underst 215:103329. https://doi.org/10.1016/j.cviu.2021.103329
14. Schuhmacher D, Bähre B, Gottschlich C, Hartmann V, Heinemann F, Schmitzer B (2024) transport: computation of optimal transport plans and Wasserstein distances. R package version 0.15-0. https://cran.r-project.org/package=transport
15. Lu S, Yu L, Feng S, Zhu Y, Zhang W (2019) Cot: cooperative training for generative modeling of discrete data. In: International conference on machine learning, pp 4164–4172. PMLR
16. Fang J, Li A, Jiang Q (2019) Gdagan: an anonymization method for graph data publishing using generative adversarial network. In: 2019 6th International Conference on Information Science and Control Engineering (ICISCE), pp 309–313. IEEE
17. Feutry C, Piantanida P, Bengio Y, Duhamel P (2018) Learning anonymized representations with adversarial neural networks. arXiv:1802.09386 [stat.ML]
18. Bae H, Jung D, Choi H-S, Yoon S (2019) Anomigan: generative adversarial networks for anonymizing private medical data. In: Pacific symposium on biocomuting 2020, pp 563–574. World Scientific
19. Park N, Mohammadi M, Gorde K, Jajodia S, Park H, Kim Y (2018) Data synthesis based on generative adversarial networks. Proceedings of the VLDB endowment, vol 11, no 10, pp 1071–1083 https://doi.org/10.14778/3231751.3231757
20. Engelmann J, Lessmann S (2021) Conditional Wasserstein gan-based oversampling of tabular data for imbalanced learning. Expert Syst Appl 174:114582. https://doi.org/10.1016/j.eswa.2021.114582
21. Xu L, Skoularidou M, Cuesta-Infante A, Veeramachaneni K (2019) Modeling tabular data using conditional gan. Advances in neural information processing systems, vol 32
22. Vega-Márquez B, Rubio-Escudero C, Riquelme JC, Nepomuceno-Chamorro I (2020) Creation of synthetic data with conditional generative adversarial networks. In: Martínez Álvarez F, Troncoso Lora A, Sáez Muñoz JA, Quintián H, Corchado E (eds) 14th International conference on soft computing models in industrial and environmental applications (SOCO 2019), pp 231–240. Springer, Cham
23. Metz L, Poole B, Pfau D, Sohl-Dickstein J (2017) Unrolled generative adversarial networks. arXiv:1611.02163 [cs.LG]
24. Zhang Z, Li M, Yu J (2018) On the convergence and mode collapse of gan. In: SIGGRAPH Asia 2018 technical briefs, pp 1–4
25. Villani C (2009) Optimal transport: old and new. Grundlehren der mathematischen Wissenschaften a series of comprehensive studies in mathematics, vol 338. Springer, Heidelberg
26. Kingma DP, Ba J (2017) Adam: a method for stochastic optimization. arXiv:1412.6980 [cs.LG]
27. Maas AL, Hannun AY, Ng AY-T et al (2013) Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the international conference on machine learning, vol 30, p 3. Atlanta, GA
28. Jamotton C, Hainaut D (2024) Variational autoencoder for synthetic insurance data. Intell Syst Appl 24:200455. https://doi.org/10.1016/j.iswa.2024.200455
29. Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. In: International conference on machine learning, pp 1310–1318. Pmlr
30. Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S (2017) Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems, vol 30
31. Naeem MF, Oh SJ, Uh Y, Choi Y, Yoo J (2020) Reliable fidelity and diversity metrics for generative models. In: International conference on machine learning, pp 7176–7185. PMLR
32. Kynkäänniemi T, Karras T, Laine S, Lehtinen J, Aila T (2019) Improved precision and recall metric for assessing generative models. Advances in neural information processing systems, vol 32
33. Boltz S, Debreuve E, Barlaud M (2007) knn-based high-dimensional Kullback-Leibler distance for tracking. In: Eighth international Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'07), pp 16–16. IEEE
34. Tax DMJ, Duin RPW (2004) Support vector data description. Mach Learn 54:45–66
35. Schölkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC (2001) Estimating the support of a high-dimensional distribution. Neural Comput 13(7):1443–1471
36. Lampert CH et al (2009) Kernel methods in computer vision. Found Trends® Comput Graph Vision 4(3):193–285
37. Baowaly MK, Lin C-C, Liu C-L, Chen K-T (2019) Synthesizing electronic health records using improved generative adversarial networks. J Am Med Inform Assoc 26(3):228–241
38. Bellemare MG, Danihelka I, Dabney W, Mohamed S, Lakshminarayanan B, Hoyer S, Munos R (2017) The cramer distance as a solution to biased Wasserstein gradients. arXiv:1705.10743 [cs.LG]
39. Mottini A, Lheritier A, Acuna-Agost R (2018) Airline passenger name record generation using generative adversarial networks. arXiv:1807.06657 [cs.LG]
40. Xie L, Lin K, Wang S, Wang F, Zhou J (2018) Differentially private generative adversarial network. arXiv:1802.06739
41. Liu E, Chu Z, Zhang X (2025) Wasserstein gan for moving differential privacy protection. Sci Rep 15(1):19634
42. Dowson DC, Landau BV (1982) The fréchet distance between multivariate normal distributions. J Multivar Anal 12(3):450–455. https://doi.org/10.1016/0047-259X(82)90077-X