Segmented Generative Networks: Data Generation in the Uniform Probability Space

Nunzio A. Letizia, *Graduate Student Member, IEEE*, and Andrea M. Tonello, *Senior Member, IEEE*

***Abstract*— Recent advancements in generative networks have shown that it is possible to produce real-world-like data using deep neural networks. Some implicit probabilistic models that follow a stochastic procedure to directly generate data have been introduced to overcome the intractability of the posterior distribution. However, the ability to model data requires deep knowledge and understanding of its statistical dependence—which can be preserved and studied in appropriate latent spaces. In this article, we present a segmented generation process through linear and nonlinear manipulations in the same-dimensional latent space where data are projected to. Inspired by the known stochastic method to generate correlated data, we develop a segmented approach for the generation of dependent data, exploiting the concept of copula. The generation process is split into two frames: one embedding the covariance or copula information in the uniform probability space, and the other embedding the marginal distribution information in the sample domain. The proposed network structure, referred to as a segmented generative network (SGN), also provides an empirical method to sample directly from implicit copulas. To show its generality, we evaluate the presented approach in three application scenarios: a toy example, handwritten digits, and face image generation.**

***Index Terms*— Copula, correlation, data analytics, dependence, distribution, explainable machine learning (ML), generative adversarial networks (GANs), generation, generative networks, machine learning.**

I. Introduction

DEEP learning literature has grown significantly over the last years. The key to its success lies on the extended availability of labeled data, increased computational power, and GPUs. Given such a background, deep generative networks that are able to produce images, videos, text, and music have been widely investigated. However, most of the techniques and architectures proposed so far are heavily applications and data dependent, making impossible to adopt them across different domains. To develop new modeling methodologies, a full knowledge and control of the steps involved during the learning process is required in order to extract meaningful information from the collected data.

Manuscript received December 19, 2019; revised June 19, 2020 and October 2, 2020; accepted November 25, 2020. Date of publication December 17, 2020; date of current version March 1, 2022. *(Corresponding author: Nunzio A. Letizia.)*

The authors are with the Chair of Embedded Communication Systems, Institute of Networked and Embedded Systems, University of Klagenfurt, 9020 Klagenfurt, Austria (e-mail: nunzio.letizia@aau.at; andrea.tonello@aau.at).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2020.3042380.

Digital Object Identifier 10.1109/TNNLS.2020.3042380

Observations can be studied and described essentially in two different ways: a deterministic or a probabilistic way. Given a collection of observed data $\mathbf{x}$ (discrete or continuous valued), one may try to fit a deterministic model with parameters θ that best reproduce the relationship between a given function $\mathbf{f}(\mathbf{z};\theta)$, whose input is $\mathbf{z}$ and the desired output is $\mathbf{x}$. The lack of a real-world complete observation makes such correspondence inexact so that it is useful to introduce a cost function C that expresses the goodness of fitting $\mathbf{x}$ through $\mathbf{f}$ by changing its parameters θ. A standard procedure to determine the best parameters tries to solve the following optimization problem:

$$\theta_{\min} = \arg\min_{\theta} C(\mathbf{f}(\mathbf{z};\theta), \mathbf{x}). \tag{1}$$

Such methodology can be implemented through a simple neural network that attempts to express $\mathbf{f}$, with input $\mathbf{z}$, as a superposition of nonlinear weighted (with parameter θ) functions with the aim of minimizing a particular objective function C using both backpropagation and gradient descent techniques. Depending on the architecture of the artificial neural network, the approximation of the mere output $\mathbf{x}$ can be arbitrarily good. The universal approximation capability renders deep neural networks excellent in finding patterns and in performing tasks very close to human's accuracy, e.g., for classification [1], [2].

A different objective, such as the generation of new "reasonable" data, cannot be treated with the same deterministic fitting approach. Indeed, $\mathbf{f}$ can perfectly map $\mathbf{z}$ into the observed data $\mathbf{x}$, but it does not tell how to pick a new input $\hat{\mathbf{z}}$ so that $\hat{\mathbf{x}} = \mathbf{f}(\hat{\mathbf{z}};\theta)$ is a statistically representative new sample. The statistical validation is intrinsically missing. The right way to proceed is to describe the process in a probabilistic manner. Let $\mathbf{x}$ be a training set consisting of n independent samples drawn from a distribution with probability density function (PDF) $p_{\text{data}}(\mathbf{x})$; then, the idea is to learn an estimate $p_{\text{model}}(\mathbf{x};\theta)$ of the real distribution $p_{\text{data}}(\mathbf{x})$. Compared with the former approach, which can be described as a pure deterministic fitting of drawn samples with distribution p_{data}, the latter approach consists of fitting in distribution, that is, learning to capture the statistical distribution of the training data.

Most of the generative models work with the maximum likelihood (ML) principle [3]; given a probability distribution parameterized by θ, the ML estimator for θ is defined as

$$\theta_{\text{ML}} = \arg\max_{\theta} p_{\text{model}}(\mathbf{x};\theta), \quad \mathbf{x} \sim p_{\text{data}}(\mathbf{x}). \tag{2}$$

Since we do not have access to the real distribution p_{data}, we use the empirical one, $\hat{p}_{\text{data}}$, which considers only the n training points. In this way, (2) can be rewritten using the Kullback–Leibler (KL) divergence

$$\theta_{\text{ML}} = \arg\min_{\theta} D_{\text{KL}}(\hat{p}_{\text{data}}(\mathbf{x})||p_{\text{model}}(\mathbf{x};\theta)) \tag{3}$$

with the useful interpretation that minimizing the KL divergence is equal to minimizing the cross entropy between the distributions.

Although the log-likelihood criterion has been largely used for generative networks, it is important to remark that having good quality samples is neither a sufficient nor a necessary condition for results with high log likelihood [4], [5].

In this article, we want to show how both linear and nonlinear dependence in collected data can be exploited in a different latent space, in detail, the uniform probability space (UPS).[1] This enables a deeper understanding of the underlying true distribution. Linear transformation schemes, copulas, and noncomplex neural network architectures are good candidates for such a purpose. Furthermore, we propose a combined generative-like model that splits and segments the data generation process into two frames: one embedding the covariance or copula information in the uniform space, and the other embedding the marginal distribution information in the sample domain. We refer to such methodology as segmented generative network (SGN), in particular, SGN-C when the process targets the generation of data with the same correlation of the samples in the data set, SGN-D instead, when the generation process targets the attainment of the full statistical dependence among data. In principle, such a segmentation procedure can be beneficial not only for a better global understanding but also for a robust generation process, regardless of the semantics of the observed data. Indeed, the proposed approach is general and can be adopted to study multimodality in data coming from images, audio, text, or heterogeneous signals.

This article is organized as follows. In Section II, we briefly present the main related work with the respective different approaches. In Section III, we present the key points of our methodology. In Section IV, we report numerical results and comparisons. Finally, Section V reports the conclusions.

A. Definitions and Notation

$\mathbf{X}$ denotes a multivariate random variable of dimension d whose components are X_i with $i = 1, \ldots, d$, whereas $\mathbf{x}^{(j)}$ for $j = 1, \ldots, n$ denotes the jth realization of $\mathbf{X}$ among n independent observations. $\mathbf{x}_i$ denotes a column vector of n realizations of X_i. Furthermore, $x_i^{(j)}$ denotes the ith entry (out of d) of the jth collected sample (out of n observations). In a compact notation, $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d]$ denotes an $n \times d$ matrix, sometimes referred to as the training data. Σ_x and $p_{\mathbf{X}}(\mathbf{x})$ denote the sample covariance matrix and probability density function of $\mathbf{X}$, respectively. $F_{\mathbf{X}}(\mathbf{x}) = P(X_1 \leq x_1, \ldots, X_d \leq x_d)$ and $F_{\mathbf{X}}^{-1}(\mathbf{x})$ denote the cumulative distribution function and quantile function, respectively. The expected value of $\mathbf{X}$ is denoted with the expectation operator $\mathbb{E}_{\mathbf{x}\sim p_{\mathbf{X}}(\mathbf{x})}[\mathbf{X}]$. Finally, $D_{\text{KL}}(p||q)$ is the KL divergence from q to p and it is defined as $\mathbb{E}_{\mathbf{x}\sim p_{\mathbf{X}}(\mathbf{x})}[\log(p_{\mathbf{X}}(\mathbf{x})/q_{\mathbf{X}}(\mathbf{x}))]$.

[1]UPS: space of uniform distributions.

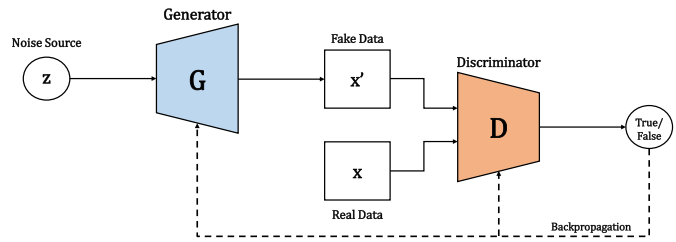

Fig. 1. GAN framework in which generator and discriminator are learned during the training process.

II. Related Work

One of the most promising data generation techniques is represented by generative adversarial networks (GANs), proposed by Goodfellow *et al.* [6]. The main idea is to train a pair of networks in competition with each other: a generator network G that captures the data distribution and a discriminator network D that distinguishes if a sample is an original coming from real data rather than a fake coming from data generated by G (see Fig. 1). The training procedure for G is to maximize the probability of D making a mistake. GANs can be thought as a minimax two-player game that will end when a Nash equilibrium point is reached. Given an input noise vector $\mathbf{y}$ with distribution $p_{\text{noise}}(\mathbf{y})$, the map to the data space is achieved through $G(\mathbf{y};\theta_{\text{gen}})$. Defining the value function $V(G, D)$ as

$$V(G, D) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{y}\sim p_{\text{noise}}(\mathbf{y})}[\log(1 - D(G(\mathbf{y})))] \tag{4}$$

it has been proved that the generator implicitly learns the true distribution. Indeed, $p_{\text{gen}} = p_{\text{data}}$ holds when the equilibrium is reached. StyleGANs [7] are the most recent evolution of GANs since they produce state-of-the-art results in synthesizing high-resolution images. They achieve it by improving the generator architecture (a mapping network for the latent representation and a synthesis one with different resolution levels) for a better understanding of the output. However, minor efforts have been carried out in generating structured nonimages data using GANs. Some attempts to synthesize stochastic signals, such as audio, text from images, or even communication signals, have been investigated in [8]–[10].

Variational autoencoders [11] are another particular class of generative models based on variational inference. Taken $\mathbf{z}$ as the latent (hidden) variable of the observed value $\mathbf{x}$ for a parameter θ, then $p_\theta(\mathbf{z}|\mathbf{x})$ represents the intractable true posterior, which can be approximated by a tractable one, $q_\phi(\mathbf{z}|\mathbf{x})$, for a parameter ϕ. A probabilistic encoder produces $q_\phi(\mathbf{z}|\mathbf{x})$, whereas a probabilistic decoder produces $p_\theta(\mathbf{x}|\mathbf{z})$. Recalling that the KL divergence is a measure of difference between two distributions, the idea is to maximize a variational lower bound $\mathcal{L}$ on the marginal likelihood

$$\log p_\theta(\mathbf{x}^{(i)}) = D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) \tag{5}$$

where

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = -D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right]. \quad (6)$$

Rather than outputting the code $\mathbf{z}$, the encoder outputs parameters describing a distribution for each dimension of the latent space. In the case where the prior is assumed to be Gaussian, $\mathbf{z}$ will consist of mean and variance. Tuning in the latent space and processing the new latent samples through the decoder is a way to generate new data.

Following the idea that "a good representation is that in which the data have a distribution that is easy to model," Dinh *et al.* [12] proposed a nonlinear deterministic transformation of the data, which maps them into a latent space of independent variables where the probability density results tractable. The framework lies behind the change of variable rule

$$p_X(\mathbf{x}) = p_Z(g^{-1}(\mathbf{x})) \cdot \left|\det \frac{\partial g^{-1}(\mathbf{x})}{\partial \mathbf{x}}\right| \quad (7)$$

when both the determinants of the Jacobian and g^{-1} are easy to compute; in that case, it is straightforward to directly sample from $p_X(\mathbf{x})$ since $\mathbf{x} = g(\mathbf{z})$. NICE, Real NVP, and the most recent Glow [13] belong to flow-based generative models that are able to provide a good latent-variable inference and excellent log-likelihood evaluation.

The Gaussian process latent variable model (GP-LVM) [14] aims at inferring both the latent code $\mathbf{z}$ and the mapping function $\mathbf{f}$ that lead to the data set $\mathbf{x}$. The prior distribution over $\mathbf{z}$ is set as Gaussian, while $\mathbf{f}(\cdot)$ is described as a Gaussian process (GP) $\mathbf{f} \sim \mathcal{GP}(\mathbf{0}, \mathbf{K})$, where $K(\cdot, \cdot)$ is the covariance function, commonly referred to as squared exponential kernel. This approach ensures a smooth mapping from the latent to the sample space while providing a closed-form expression to approximate the true posterior distribution $p(\mathbf{z}|\mathbf{x})$ and to find a variational lower bound for a robust training procedure [15].

The methods presented so far have been mostly and successfully applied to images and they all share the ability to generate new samples in parallel. The synthesis of fully visible belief networks (FVBNs) (see [16], [17]) and autoregressive models [18] is difficult to parallelize; therefore, due to their sequential nature, they are relatively slow. Indeed, their core idea is to factorize the joint probability distribution of d dimensional inputs $\mathbf{x}$ into products of 1-D conditional distributions

$$p_{\mathrm{model}}(\mathbf{x}) = \prod_{i=1}^{d} p_{\mathrm{model}}(x_i|x_1, \ldots, x_{i-1}). \quad (8)$$

Generation is done by generating one dimension at a time leading to good quality of the samples (as WaveNet for human speech [19]) since they optimize the likelihood directly.

Boltzmann machines [20] and generative stochastic network [21] rely on estimating the transition operator of a Markov chain $p(x_i|x_{i-1})$ but are now less used since Markov chains fail to scale in high-dimensional spaces.

Bearing this overview in mind, our method consists in a different approach. It does not focus on maximizing the likelihood or minimizing the KL divergence between the model and the reference true distribution, but it aims to produce, in the linear case, samples belonging to an unknown distribution that pretends to be a linear estimator of the true one, whereas in the nonlinear framework, new samples are drawn from the same distribution of the real data exploiting the concept of copula and GANs. To the best of our knowledge, this is the first work that embeds the copula concept inside a GAN framework.

III. Proposed Approach

The statistical dependence between the components of a multivariate random variable $\mathbf{X}$ is described by the joint probability distribution $p_{\mathbf{X}}(x_1, x_2, \ldots, x_d)$. The correlation, instead, measures how the components are related on average, and it is expressed in terms of the expectation $\mathbb{E}[\mathbf{X} \cdot \mathbf{X}^T]$ or the covariance $\mathbb{E}[(\mathbf{X} - m_X) \cdot (\mathbf{X} - m_X)^T]$, where m_X denotes the expected value of $\mathbf{X}$. The correlation is often associated with the idea of linear dependence.

Let $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d]$ be a set of realizations of $\mathbf{X}$, also referred to as the collected multivariate data. We would like to generate new unseen samples $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_d]$ similar, in some way as we will discuss, to $\mathbf{x}$. In the following, we propose two different approaches.

1) The first one revisits the known stochastic generation process of correlated data (data that exhibit the same correlation as the collected one) and highlights the need for a segmentation and domain adaptation step. Such a method will be referred to as SGN targeting and modeling the correlation of data (SGN-C).
2) The second one, instead, studies the stochastic generation process of dependent data (data that exhibit the same joint distribution as the collected one) by applying the same domain adaptation of SGN-C but integrating the concept of copula in order to segment the generation process. Moreover, it provides a direct method to sample from copulas. Such an approach will be referred to as SGN targeting and modeling the statistical dependence of data (SGN-D).

A. Transform Sampling

The training data have been generated by some fixed unknown or difficult to construct probability distribution $p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{X}}(x_1, x_2, \ldots, x_d)$ with cumulative distribution function $F_{\mathbf{X}}(\mathbf{x}) = P(X_1 \leq x_1, \ldots, X_d \leq x_d)$. However, it is plausible to assume that the marginal density of each X_i is known or can be easily derived as $p_{X_i}(x_i)$ with cumulative $F_{X_i}(x_i)$. Hence, the data can be mapped into a latent space with the same dimension using the inverse transform sampling method. In particular, if U_i be a uniform random variable, then

$$X_i = F_{X_i}^{-1}(U_i) \quad (9)$$

is a random variable with cumulative distribution F_{X_i}. To project the data $\mathbf{x}$ into the latent space, it is enough to compute the transformation $u_i = F_{X_i}(x_i)\ \forall i = 1, \ldots, d$. This first step can be interpreted as a simple encoder, which tries to represent the data in a domain space where linear manipulations are easier to be implemented. One way to go back is to train a neural network, which, given $\mathbf{u}$ as input and $\mathbf{x}$ as output, finds

the inverse mapping between the two spaces, the latent, and the sample ones. This inverse transformation back to the sample space can be easily interpreted as a decoder. Fig. 2 shows the SGN framework.

The autoencoder described so far has no generative properties since it only replicates the data set $\mathbf{x}$. The way to generate new samples is to build a new encoded set $\hat{\mathbf{u}}$ and feed it into the already trained inverse network.

In general, X_i and X_j with $i \neq j \leq d$ are dependent random variables (let us consider, for example, the intensity of two consecutive pixels in an image or the amplitude of a waveform like the sound). A first order of approximation is the linear dependence; the idea is to choose the new encoded set $\hat{\mathbf{u}}$ as a set of d correlated uniform variables—correlation quantified by the sample covariance matrix Σ_u of the encoded initial set $\mathbf{u}$.

B. Correlated Uniforms (SGN-C)

Let Σ_x and Σ_u be the sample covariance matrices of the data set $\mathbf{x}$ and the latent uniform code $\mathbf{u}$, respectively. If $\hat{\mathbf{x}}$ has covariance matrix $\Sigma_{\hat{x}}$ equal to Σ_x, then we define $\hat{\mathbf{x}}$ *similar* to $\mathbf{x}$. Denoting with $\mathbf{F^{-1}}$ the nonlinear inverse cumulative distribution function, then if $\hat{\mathbf{u}}$ is similar to $\mathbf{u}$, it follows that $\hat{\mathbf{x}} = \mathbf{F^{-1}}(\hat{\mathbf{u}})$ is similar to $\mathbf{x}$.

To generate d correlated uniform distributed random variables, we use the NORTA method [22], in the following denoted as covariance method. The first step requires the generation of d correlated Gaussian distributed random variables. In particular, given a mean vector μ and a covariance matrix Σ, to generate a sample $Y \sim \mathcal{N}(\mu, \Sigma)$ from the multivariate normal distribution, we need to consider first a vector $\mathbf{z}$ of uncorrelated Gaussian random variables and then find a matrix $\mathbf{C}$ and square root of Σ such as $\mathbf{C} \cdot \mathbf{C}^T = \Sigma$ (for example, using the Cholesky decomposition). It follows that $\mathbf{y} = \mu + \mathbf{C} \cdot \mathbf{z}$ is a vector of d Gaussian random variables $\{Y_i\}_{i=1}^d$ with the desired properties. Applying the probability integral transform to each entry leads to d correlated uniform random variables

$$\hat{U}_i = \Phi(Y_i) \tag{10}$$

where Φ is the cumulative distribution function of the standard normal distribution. If we wanted to impose Σ_u as the covariance matrix of $\hat{\mathbf{u}}$, we could transform the latent code $\mathbf{u}$ into a new latent code $\mathbf{n}$ in the Gaussian space, compute the covariance matrix Σ_n, and use it to sample from the multivariate normal distribution $\mathcal{N}(\mu, \Sigma_n)$. Instead, if we only wanted to impose the correlation matrix $\mathbf{R_u}$, we would rather just sample from the multivariate normal distribution $\mathcal{N}(\mu, \mathbf{R_u})$ to get a good practical approximation as described in [23].

The generation of the encoded set can be easily implemented in parallel, which means that we do not need to wait for any information from previous samples. Once the encoded set $\hat{\mathbf{u}}$ is built, the transformation $\mathbf{F^{-1}}(\hat{\mathbf{u}})$ gives $\hat{\mathbf{x}}$ as a new generated sample, exploiting only the linear information inside the data set. Note that exploiting an artificial neural network for the inverse transform is not mandatory; one could use the inverse cumulative distribution function [quantile function, see (9)] implemented by step functions or kernel smoothing functions. Section IV will show some visual results.

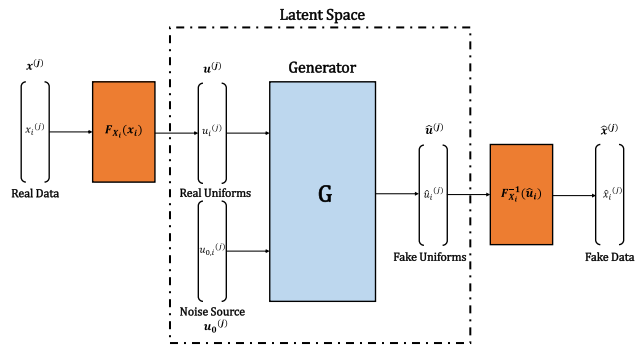

Fig. 2. Transform sampling: SGN approach where original data are projected into a latent space, and the new codes are generated and back-projected again.

C. Dependent Uniforms (SGN-D)

Working only with linear dependence (correlation) is not sufficient to fully reproduce the relationship between data. Thus, it is necessary to build a new encoded set $\hat{\mathbf{u}}$ of d dependent uniform variables.

When we discussed the procedure to build correlated uniforms, we were looking for a vector $\hat{\mathbf{u}}$ whose covariance matrix $\Sigma_{\hat{u}}$ was equal to the prescribed one Σ_u and built it by using the SGN-C algorithm described in Section III-B. Another approach could have been the following; given a family of functions S_θ that takes uncorrelated uniform variables $\mathbf{u_0}$ as input and transforms them into correlated ones $\hat{\mathbf{u}}$, one could solve the optimization problem

$$\theta_{\min} = \arg\min_{\theta} \delta(\Sigma_u, \Sigma_{S_\theta(\mathbf{u_0})}) \tag{11}$$

where δ is a measure of distance between the sample covariance matrices of $\mathbf{u}$ and $\hat{\mathbf{u}}$.

In the same way, given the latent codes $\mathbf{u}$ with probability density function $p_{\mathbf{u}}(\mathbf{u})$, the main idea to generate dependent uniform random variables is to train a neural network G_θ, which takes independent uniform variables $\mathbf{u_0}$ as input and maps them into new dependent ones, $\hat{\mathbf{u}}$, with distribution $q_{\hat{\mathbf{u}}}(\hat{\mathbf{u}})$. This is again a generative model whose objective function is

$$\theta_{\min} = \arg\min_{\theta} \delta(p_{\mathbf{u}}(\mathbf{u}), q_{\hat{\mathbf{u}}}(G_\theta(\mathbf{u_0})) \tag{12}$$

where now δ is a measure of discrepancy between the real distribution $p_{\mathbf{u}}$ and the generated one $q_{\hat{\mathbf{u}}}$.

Before introducing the approach chosen for solving problem (12), we focus on the particular properties that $p_{\mathbf{u}}$ and $q_{\hat{\mathbf{u}}}$ have, recalling the concept of copula.

Let $(U_1, U_2, \ldots, U_d)$ be uniform random variables, and then, their joint cumulative distribution function $F_{\mathbf{U}}(\mathbf{u}) = P(U_1 \leq u_1, \ldots, U_d \leq u_d)$ is a copula $C : [0,1]^d \rightarrow [0,1]$ (see [24] for an analytic description). Copulas are a useful tool to construct multivariate distributions and analyze data dependence. Indeed, Sklar's theorem [25] states that if $F_{\mathbf{X}}$ is a d-dimensional cumulative distribution function with

continuous marginals $F_{X_1}, \ldots, F_{X_d}$, then $F_{\mathbf{X}}$ has a unique copula representation

$$F_{\mathbf{X}}(x_1, \ldots, x_d) = C(F_{X_1}(x_1), \ldots, F_{X_d}(x_d)). \tag{13}$$

Moreover, when the multivariate distribution has a probability density function $f_{\mathbf{X}}$, it holds that

$$f_{\mathbf{X}}(x_1, \ldots, x_d) = c(F_{X_1}(x_1), \ldots, F_{X_d}(x_d)) \cdot \prod_{i=1}^{d} f_{X_i}(x_i) \tag{14}$$

where c is the density of the copula. This last relationship is rather interesting because it affirms that the dependence internal structure of $f_{\mathbf{X}}$ can be recovered using the density of the marginals f_{X_i} and the density c of the copula. Under this perspective, problem (12) can be reformulated as

$$\theta_{\min} = \arg\min_{\theta} \delta(c_{\mathbf{u}}(\mathbf{u}), c_{\hat{\mathbf{u}}}(G_\theta(\mathbf{u_0})) \tag{15}$$

where $c_{\mathbf{u}}$ and $c_{\hat{\mathbf{u}}}$ are the densities of the copulas related to $\mathbf{u}$ and $\hat{\mathbf{u}}$, respectively. The objective is to reach the equality $c_{\mathbf{u}} = c_{\hat{\mathbf{u}}}$ and to sample a new encoded set $\hat{\mathbf{u}}$ from $c_{\mathbf{u}}$, preserving the entire hidden dependence in $\mathbf{u}$ by construction. Due to the fact that we are working with a finite set of samples, we should build the empirical copula of the encoded given set $\mathbf{u}$, with expression

$$C_n(\mathbf{u}) = \frac{1}{n} \sum_{j=1}^{n} \mathbb{1}_{\left\{U_1^{(j)} < u_1, \ldots, U_d^{(j)} < u_d\right\}} \tag{16}$$

where n is the number of observations, $U_i^{(j)}$ denotes the jth realization of the ith random variable, with $i = 1, \ldots, d$, and $\mathbb{1}_A$ is the indicator function. When its dimensionality increases, this is not feasible anymore and one way to proceed is to choose a parametric family of multivariate copulas, like the multivariate Gaussian copula with correlation matrix Σ (equivalent to NORTA [26]) or the multivariate Student's t-copula with ν degrees of freedom and correlation matrix Σ, which is more suitable for data containing phenomena of extreme value dependence [27]. Archimedean copulas are a particular class of copulas that admit a closed formula and allow modeling dependence varying one parameter, they have the following representation:

$$C(u_1, \ldots, u_d; \theta) = \psi^{[-1]}(\psi(u_1, \theta) + \cdots + \psi(u_d, \theta); \theta) \tag{17}$$

where ψ is a continuous, strictly decreasing and convex generator function with pseudoinverse $\psi^{[-1]}$ [24] and θ is a parameter. Section IV compares some results using different copulas in specific applications. An interesting way to overcome the lack of parametric multivariate copulas is to take advantage of the huge number of parametric families of bivariate copulas through the concept of vine copulas [26], [28]. The idea is to model the copula density as the product of pairs of conditional copula bivariate densities, under a tree or vine decomposition, which leads to tractable and flexible probabilistic models. A recent attempt to generate data using vine copulas [29] exploited autoencoders and their ability to find lower dimensional representation.

We have argued about the parameters of the function δ that we are trying to minimize, but we never mentioned so far the type of distance/discrepancy that δ has to mime. Since we are elevating our approach to a general one that does not focalize on low dimension of data or specific type of distributions, we propose two different approaches. The first one is the maximum mean discrepancy (MMD) metric.

Let $\mathbf{x} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(l)}\}$ and $\mathbf{y} = \{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(m)}\}$ be observations taken independently of $p = c_{\mathbf{u}}$ and $q = c_{\hat{\mathbf{u}}}$, respectively. Let (χ, d) be a nonempty compact metric space in which p and q are defined. Then, the MMD is defined as

$$\mathrm{MMD}(\mathcal{F}, p, q) := \sup_{f \in \mathcal{F}} (\mathbb{E}_{\mathbf{x} \sim p}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{y} \sim q}[f(\mathbf{y})]) \tag{18}$$

where $\mathcal{F}$ is a class of functions $f : \chi \to \mathbb{R}$. Since $p = q$ if and only if $\mathbb{E}_{\mathbf{x} \sim p}[f(\mathbf{x})] = \mathbb{E}_{\mathbf{y} \sim q}[f(\mathbf{y})]\ \forall f \in \mathcal{F}$, MMD is a metric that measures the disparity between p and q (see [30]).

When $\mathcal{F}$ is a reproducing kernel Hilbert space (RKHS), f can be replaced by a kernel $k \in \mathcal{H}$ (i.e., Gaussian or Laplace kernels). In this case, Gretton *et al.* [31] showed that

$$\begin{aligned}\mathrm{MMD}^2(\mathcal{H}, p, q) = \mathbb{E}_{\mathbf{x}, \mathbf{x}' \sim p}[k(\mathbf{x}, \mathbf{x}')] - 2\mathbb{E}_{\mathbf{x} \sim p, \mathbf{y} \sim q}[k(\mathbf{x}, \mathbf{y})] \\ + \mathbb{E}_{\mathbf{y}, \mathbf{y}' \sim q}[k(\mathbf{y}, \mathbf{y}')]\end{aligned} \tag{19}$$

where $\mathbf{x}'$ is an independent copy of $\mathbf{x}$ with the same distribution and $\mathbf{y}'$ is an independent copy of $\mathbf{y}$. For practical implementation, an unbiased empirical estimate is given by

$$\begin{aligned}\mathrm{MMD}_u^2(\mathcal{H}, \mathbf{x}, \mathbf{y}) = &\frac{1}{l(l-1)} \sum_{i \neq i'} k(\mathbf{x}^{(i)}, \mathbf{x}^{(i')}) \\ &+ \frac{1}{m(m-1)} \sum_{j \neq j'} k(\mathbf{y}^{(j)}, \mathbf{y}^{(j')}) \\ &- \frac{2}{lm} \sum_{i=1}^{l} \sum_{j=1}^{m} k(\mathbf{x}^{(i)}, \mathbf{y}^{(j)}).\end{aligned} \tag{20}$$

Finally, we can define the type of discrepancy δ as the MMD_u^2 estimator, in particular

$$\delta(c_{\mathbf{u}}(\mathbf{u}), c_{\hat{\mathbf{u}}}(G_\theta(\mathbf{u_0})) = \mathrm{MMD}_u^2(\mathcal{H}, \mathbf{u}, G_\theta(\mathbf{u_0})) \tag{21}$$

and proceed with its minimization by the exploitation of the chain rule and the gradient descent method as described in [32]. We found this methodology not effective for embedding the copula dependence structure. Therefore, we decided to use it during the evaluation phase (see Section IV) and, instead, adopt a GAN framework to reproduce the copula dependence.

The core idea is to identify the copula with a first generator G_u. The generator G_u receives an equal-dimensional independent uniform noise source $\mathbf{z}$ as input and internally creates the copula dependence structure by opposing a discriminator D_u that tries to distinguish between real and fake dependent uniform samples, $\mathbf{u}$ and $\hat{\mathbf{u}}$, respectively. The corresponding value function reads as follows:

$$\begin{aligned}V(G_u, D_u) = &\mathbb{E}_{\mathbf{u} \sim c_{\mathbf{u}}(\mathbf{u})}[\log D_u(\mathbf{u})] \\ &+ \mathbb{E}_{\mathbf{z} \sim \mathcal{U}(0,1)}[\log(1 - D_u(G_u(\mathbf{z})))].\end{aligned} \tag{22}$$

At the same time, in order to strengthen the copula generation process, another GAN (G_x, D_x) mimes the relationship (the quantile function $F_{X_i}^{-1}$ for $i = 1, \ldots, d$) between the latent code $\mathbf{u}$ and the sample space $\mathbf{x}$. To do so, it takes the generated uniforms $\hat{\mathbf{u}}$ and statistically transforms them into

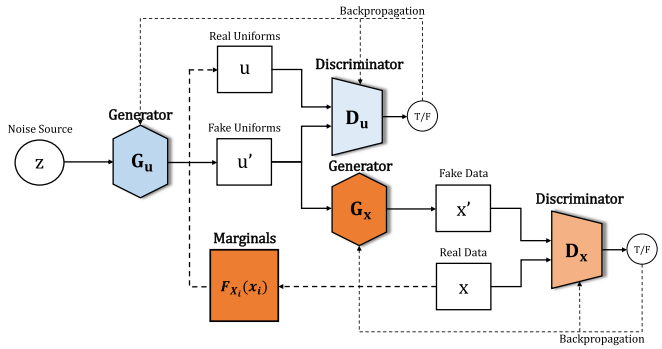

Fig. 3. SGN modeling the dependence: a first network builds the dependent uniforms and a second network converts them into new data.

new samples $\hat{\mathbf{x}}$, checking again the statistical significance with another discriminator D_x. The second value function is defined as

$$V(G_x, D_x) = \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}(\mathbf{x})}[\log D_x(\mathbf{x})] + \mathbb{E}_{\hat{\mathbf{u}}\sim c_{\hat{\mathbf{u}}}}[\log(1 - D_x(G_x(\hat{\mathbf{u}})))]. \quad (23)$$

We denote the full generation process as SGNs modeling the dependence (SGN-D). Succinctly, the first generator G_u embeds the copula density structure $c_{\mathbf{u}}(\mathbf{u})$ and produces dependent uniform samples $\hat{\mathbf{u}}$. The second generator G_x embeds the marginals structure (the quantile $F_{X_i}^{-1}$ for $i = 1, \ldots, d$) and statistically maps $\hat{\mathbf{u}}$ into new samples $\hat{\mathbf{x}}$. Fig. 3 graphically summarizes the entire followed methodology.

To identify the marginal cumulative distribution function F_{X_i}, for $i = 1, \ldots, d$, it is sufficient to cyclically repeat the aforementioned segmentation process by concatenating another GAN, which takes as input the last generated samples $\hat{\mathbf{x}}$ and projects them into a new encoded set $\tilde{\mathbf{u}}$.

Section IV presents some graphical and numerical results, comparing the different methodologies.

IV. Evaluation of Results

This section discusses and compares the SGN-C and SGN-D approaches to generate new samples in three different case studies. We consider a 2-D toy data set and two higher dimensional data sets, MNIST [33] and CelebA [34]. For each of them, we qualitatively evaluate the generation performance of the SGN-C approach (e.g., covariance, Gaussian, and t-copula) and the SGN-D approach (with GANs). Finally, for the last data set scenario, we compare some quantitative results using three different metrics.

A. Qualitative Evaluation

We used Keras with TensorFlow [35] as back end to implement the proposed model. The code has been tested on a Windows-based operating system provided with Python 3.6, TensorFlow 1.13.1, Intel core i7-3820 CPU, and one GPU GTX1080. Two different types of neural network architectures have been deployed according to the specific data set under analysis. To allow the reproducibility of the work presented, Tables I and II report all the implementation aspects. The details of the network and the chosen parameters for the first toy example are reported in Table I. For the two data set containing images, the established DCGAN [36] architecture has been used as a foundation for the proposed SGN-D structure with all details reported in Table II. To overcome numerical issues in the cases of the images, the definition interval of the uniform distribution is transformed from [0, 1] to $[-1, 1]$.

TABLE I
SGN-D Based on GAN Architecture for Synthetic Toy Model

Operation	Feature maps	Activation
Generator G_u		
$G_u(\mathbf{z}) : \mathbf{z} \sim \mathcal{U}(0, 1)$	2	
Fully connected	500	ReLU
Dropout	0.5	
Fully connected	2	Sigmoid
Generator G_x		
$G_x(\mathbf{u}) : \mathbf{u} \sim c_u$	2	
Fully connected	500	ReLU
Dropout	0.5	
Fully connected	2	Sigmoid
Discriminator D_u		
$D_u(\mathbf{u}) : \mathbf{u} \sim c_u$	2	
Fully connected	500	LeakyReLU
Dropout	0.4	
Fully connected	10	LeakyReLU
Fully connected	1	Sigmoid
Discriminator D_x		
$D_x(\mathbf{x}) : \mathbf{x} \sim p_x(\mathbf{x})$	2	
Fully connected	500	LeakyReLU
Dropout	0.4	
Fully connected	10	LeakyReLU
Fully connected	1	Sigmoid
Number of generators	2	
Batch size	64	
Number of iterations	50000	
Leaky ReLU slope	0.2	
Learning rate	0.0002	
Optimizer	Adam (β_1 = 0.5, β_2 = 0.9999)	

1) 2-D Toy Database: Consider a set of two random variables whose statistics is computed from a collection of 2000 observations. Thus, given $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$, we wish to generate a new sample $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2]$. In order to impose a nonlinear statistical dependence structure, we build $\mathbf{x}$ as follows:

$$\mathbf{x} = [\sin(t), t\cos(t)] + \mathbf{n} \quad (24)$$

where $t \sim \mathcal{N}(0, 1)$ and $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbb{I})$ with $\sigma = 0.01$.

Proceeding as explained in Section III-B leads to a set of correlated uniforms and correlated samples, $\hat{\mathbf{u}}$ and $\hat{\mathbf{x}}$, respectively. Despite having the same covariance matrix and the same marginals, the linear transformation is not capable to account for the full dependence structure, as shown in the top row of Fig. 4. What is missing is indeed the copula density component in (14). Since, in general, there is no closed-form expression for the copula density $c(\mathbf{u})$, the multivariate Gaussian copula and the Student's t-copula are possible choices that contain information regarding correlation coefficients and, thus, linear properties. Nevertheless, both of them are not enough to cover or at least approximate the dependence in $\mathbf{x}$. Therefore, we considered both Clayton [37] and Frank Archimedean copulas. From the central row of Fig. 4, we can immediately understand that there exists a set of parameters θ, which approximates the dependence around the mean values but is not able to do the same around the tails.

TABLE II
SGN-D BASED ON DCGAN ARCHITECTURE FOR IMAGES

Operation	Feature maps	Activation
Generator G_u		
$G_u(\mathbf{z}) : \mathbf{z} \sim \mathcal{U}(-1, 1)$	1024	
Fully connected	4096	ReLU
Reshape and upsampling	$(8, 8, 64)$	
Convolution and BatchNorm	64	ReLU
Upsampling		
Convolution and BatchNorm	32	ReLU
Convolution	3	Tanh
Generator G_x		
$G_x(\mathbf{u}) : \mathbf{u} \sim c_u$	$(32, 32, 3)$	
Convolution and BatchNorm	128	ReLU
Upsampling		
Convolution and BatchNorm	64	ReLU
Convolution	3	Tanh
Discriminator D_u		
$D_u(\mathbf{u}) : \mathbf{u} \sim c_u$	$(32, 32, 3)$	
Convolution	32	LeakyReLU
Dropout	0.25	
Convolution and BatchNorm	64	LeakyReLU
Dropout	0.25	
Convolution and BatchNorm	128	LeakyReLU
Dropout	0.25	
Convolution and BatchNorm	256	LeakyReLU
Dropout	0.25	
Flatten and dense	1	Sigmoid
Discriminator D_x		
$D_x(\mathbf{x}) : \mathbf{x} \sim p_x(\mathbf{x})$	$(32, 32, 3)$	
Convolution	32	LeakyReLU
Dropout	0.25	
Convolution and BatchNorm	64	LeakyReLU
Dropout	0.25	
Convolution and BatchNorm	128	LeakyReLU
Dropout	0.25	
Convolution and BatchNorm	256	LeakyReLU
Dropout	0.25	
Flatten and dense	1	Sigmoid
Number of generators	2	
Batch size	64	
Number of iterations	50000	
Leaky ReLU slope	0.2	
Learning rate	0.0002	
Optimizer	Adam (β_1 = 0.5, β_2 = 0.9999)	

In the low-dimensional space, i.e., 2-D, it is still feasible to calculate the empirical copula for a certain bins resolution. In such a case, sampling from copula results in a trivial task and provides good quality of the generated samples (bottom-left corner of Fig. 4). Whenever the space dimension increases, such an approach cannot be easily followed anymore due to numerical problems, the reason why an SGN-D-based methodology, which implicitly estimates the distribution, can be exploited. Bottom subplots with orange samples of Fig. 4 show the generated samples under the SGN-D framework for both G_u (back-projected with quantiles in sample domain) and G_x output samples.

2) Handwritten Digit Database: Now, consider a set 28×28 pixels representing images of digit 4. This results in 784 dependent random variables whose statistics is computed from a collection of 2000 observations. Thus, given $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{784}]$, we wish to generate a new sample $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_{784}]$. The nonlinear dependence structure is an intrinsic property of images, so this example fits the purpose.

Again, following the steps presented in Sections III-B and III-C for both SGN-C and SGN-D methods leads to a set of correlated uniforms and correlated samples, $\hat{\mathbf{u}}$ and $\hat{\mathbf{x}}$, respectively. The covariance method is not enough to obtain a smooth detailed picture [see the generated digits in Fig. 5(b)]; nevertheless, it is able to capture and reproduce in most of the cases the essence of the picture, i.e., digit 4. On the other hand, it is interesting to note that the digits generated by the generator G_x [see Fig. 5(f)] cannot be distinguished from the real, whereas the digits marginally backprojected from dependent uniforms coming from generator G_u [see Fig. 5(e)] are rather blurry. Such an effect is due to the discrete nature of the handwritten digit data set distribution. Indeed, the joint distribution is mostly nonzero around small spheres centered in 0 and 1; therefore, the approximation of the marginals $F_{X_i}(x_i)$ results poor in the intermediate values (due to the steepness of the cumulative function). At the same time, the output of G_u is a set of dependent uniforms with values uniformly distributed from 0 to 1 which have to be transformed into the sample space, using the poorly estimated $1D$ inverse cumulative distribution function. The second GAN G_x solves this numerical issue by mapping the data from the uniform space into the sample one through highly nonlinear smooth transformations.

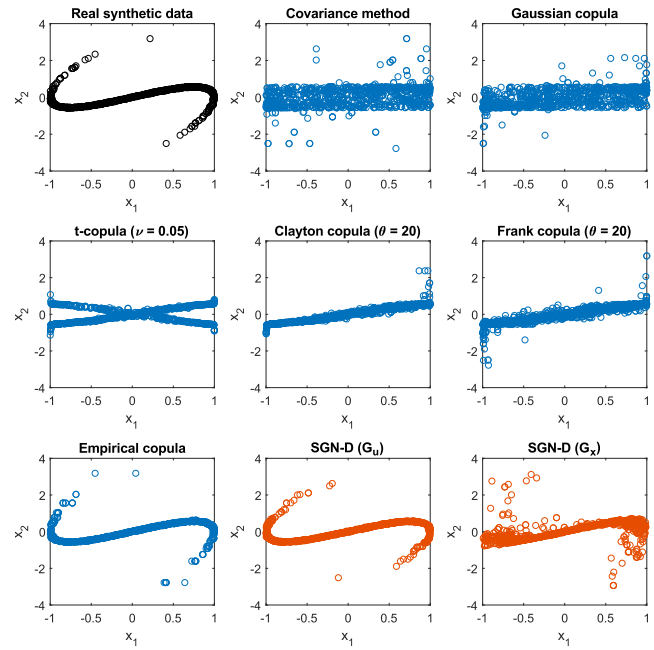

Fig. 4. Comparison of 2-D samples generated using SGN-C, Archimedean and empirical copulas (blue samples), and SGN-D (orange samples) approaches.

3) Celebrity Faces Database: As the last example, we propose a set of (cropped) 32×32 color images representing faces of celebrities covering some pose variations and including different backgrounds. The motivation resides in the intrinsic continuous property of the distribution since all the colors are feasible, yielding to robust cumulative marginals. The CelebA data set [34] contains more than 200k faces. To be coherent with previous examples and to focus more on the correct identification of the block components charged to generate both correlated and dependent uniforms rather than to their quality, we considered only the first 20 000 samples of the data set. Fig. 6 shows the results.

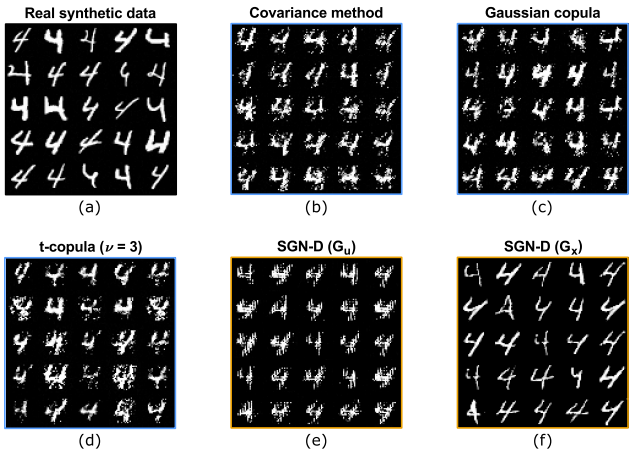

Fig. 5. Comparison of high-dimensional samples (digits) generated using (b)–(d) SGN-C and (e) and (f) SGN-D approaches. Original data (a).

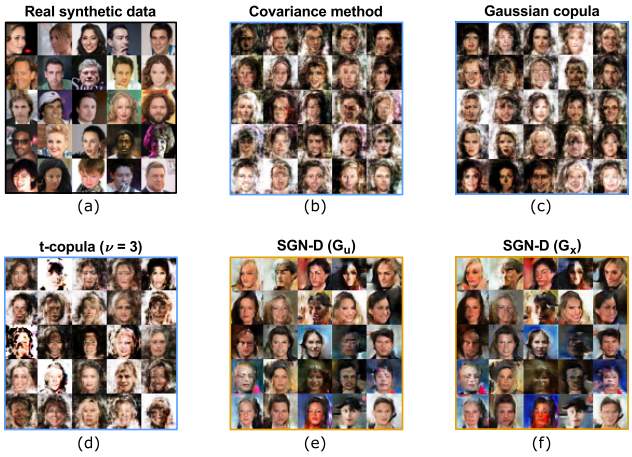

Fig. 6. Comparison of high-dimensional samples (faces) generated using (b)–(d) SGN-C and (e) and (f) SGN-D approaches. Original data (a).

The methods involving covariance and parametric copulas perform poorly in terms of quality of the details but capture the relevant information of the image and replicate it in new blurry faces. On the contrary, GANs introduce higher nonlinear dependence and thus harmonic details, the more the network trains itself.

The most interesting part is that, conversely to the MNIST case where the dependent uniforms were correctly generated but erroneously backprojected to sample domain due to poor quantiles, this time the estimated marginals do not have steep gradients. Therefore, the projected generated samples are smooth, as shown in Fig. 6(e). Moreover, these samples are extremely similar to the output of the generator G_x [see Fig. 6(f)] accordingly to the interpretation that G_x mimes the quantile functions $F_{X_i}^{-1}(x_i)$ $\forall i \in \{1, \ldots, 1024\}$. Fig. 7(a) and (b) shows an example of data representation and data generation in the uniform space, respectively.

B. Quantitative Evaluation

While several measures have been introduced so far to assess the performance of a specific generative model, there is no consensus as to which measure is the most appropriate [38]. Nevertheless, lately, measures that deal with embedding layers and feature space have found extensive use. For the purpose of this article, we decided to adopt three of them, in particular, the inception score (IS) [39], the Frèchet inception distance (FID) [40], and the kernel inception distance (KID) [41] metrics.

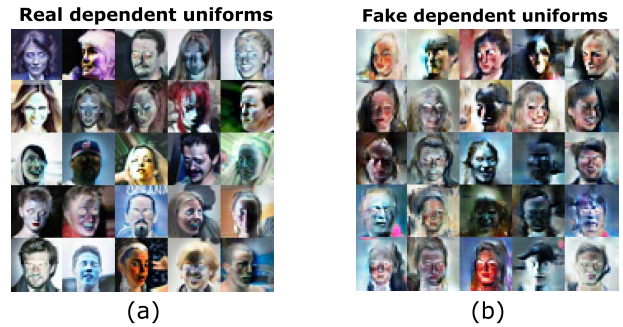

Fig. 7. Dependent uniforms (a) obtained from transform sampling of original data and (b) obtained as output of G_u.

The IS computes the average KL divergence between the conditional distribution of the images label $p(y|\mathbf{x})$ and the marginal distribution $p(y) = \mathbb{E}_{\mathbf{x}}[p(y|\mathbf{x})]$, on the pretrained Inception Net [42]. It is defined as $\exp(\mathbb{E}_{\mathbf{x}}[D_{\mathrm{KL}}(p(y|\mathbf{x})||p(y))])$ and assumes high values for a low entropy of $p(y|\mathbf{x})$, achieved when samples are easily classifiable, and for a high entropy of $p(y)$, to favor diversity.

FID compares the statistics of generated samples to real ones by computing the Frèchet distance between two multivariate Gaussian distributions. Indeed, both data are projected into a feature space (Inception representations) in which a Gaussian distribution fits them

$$\mathrm{FID}(r, g) = ||\mu_r - \mu_g||_2^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{\frac{1}{2}}\right) \quad (25)$$

where $X_r \sim \mathcal{N}(\mu_r, \Sigma_r)$ and $X_g \sim \mathcal{N}(\mu_g, \Sigma_g)$ are outputs of a pool layer in the Inception Net [42] for real and generated samples, respectively. Low FID values correspond to better similarity in distribution.

However, the Gaussianity of the Inception representations is often not guaranteed. As a result of the ReLU activations, the representations are not negative with some components equal to zero [41]. To overcome such limitation, we decided to use the KID metric [41]. KID is the squared MMD (see Section III-C) between the inception representations. In particular, if $\phi(\cdot)$ is the function mapping the real ($\mathbf{x}_r$) and generated ($\mathbf{x}_g$) samples into the inception representation, then

$$\mathrm{KID}(r, g) = \mathrm{MMD}^2(\mathcal{H}, \phi(\mathbf{x}_r), \phi(\mathbf{x}_g)). \quad (26)$$

We used the polynomial kernel $k(\mathbf{x}, \mathbf{y}) = ((1/d)\mathbf{x} \cdot \mathbf{y}^T + 1)^3$, where d is the representation dimension. Compared with FID, KID has the advantage of being independent of the distribution of the latent representation.

We evaluated the scores only for the generated samples in the CelebA data set scenario. The inception network is a deep CNN (convolutional neural network) pretrained on the

TABLE III
PERFORMANCE OF SGN-C AND SGN-D USING THE IS, FID, AND KID MEASURES ON THE CELEBA DATA SET

	CelebA Dataset (32×32)					
Space	**Uniform**			**Sample**		
Method	**IS**	**FID**	**KID**	**IS**	**FID**	**KID**
Real synthetic data	3.71	25.6	0.00	2.77	18.0	0.00
Covariance	3.43	210	0.19	2.13	136	0.11
Gaussian copula	3.30	208	0.19	2.16	136	0.11
t-copula	3.50	195	0.16	2.13	125	0.10
SGN-D (G_u)	3.27	76.6	0.03			
SGN-D (G_x)				2.61	70.9	0.03

ImageNet data set. Hence, data sets that are too semantically different from ImageNet would lead to poor ISs. Since we are not interested in getting the best performance out of our architecture, but rather in a fair comparison between the different methodologies, we look at relative results. In particular, Table III reports the scores for the different methodologies adopted.

The IS, FID, and KID scores are consistent with human visual perceptions. Indeed, as depicted from visual intuition, there is almost no difference between the scores obtained from covariance and parametric copulas methods, while there is a significant gap between the achieved scores with the linear (SGN-C) and nonlinear dependence (SGN-D) approaches.

V. CONCLUSION

This article has first discussed some known procedures to generate correlated variables from a sample set, highlighting the need of a domain transformation (from the sample to the uniform variables domain). The same domain adaptation has been exploited for the generation of statistically dependent variables, recalling the concept of copula. This mathematical tool enables the partitioning and segmentation of the dependence structure generation into two well-defined steps, an initial step that creates the data dependence between uniform random variables (copula) and a second step that projects the uniform random variables back into the sample domain (inverse transform sampling), leading to new data. The former case has been analyzed through the aid of different copula structures, whereas the latter case has been analyzed through an estimation of the marginal cumulative distribution (and its inverse), where the more samples are available, the better the approximation is. Such segmentation totally disregards the semantics of the input data and, in principle, can be applied to any type of data/signal.

This procedure has led to the design of an SGN architecture, based on GANs, successfully implemented as proved by several qualitative and quantitative results. This goes in the direction of explainable machine learning, i.e., a full comprehension of the design of a neural network through a mathematical segmentation of the problem.

REFERENCES

[1] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[2] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[4] H. E. Gerhard, F. A. Wichmann, and M. Bethge, "How sensitive is the human visual system to the local statistics of natural images," *PLoS Comput. Biol.*, vol. 9, no. 1, Jan. 2013, Art. no. e1002873.

[5] A. van den Oord and J. Dambre, "Locally-connected transformations for deep gmms," in *Proc. Int. Conf. Mach. Learn. (ICML) Deep Learn. Workshop Abstr.*, 2015, pp. 1–8.

[6] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, Dec. 2014, pp. 2672–2680.

[7] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4401–4410.

[8] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–17.

[9] H. Zhang *et al.*, "StackGAN++: Realistic image synthesis with stacked generative adversarial networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1947–1962, Aug. 2019.

[10] A. M. Tonello, N. A. Letizia, D. Righini, and F. Marcuzzi, "Machine learning tips and tricks for power line communications," *IEEE Access*, vol. 7, pp. 82434–82452, Jun. 2019.

[11] D. P Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114

[12] L. Dinh, D. Krueger, and Y. Bengio, "NICE: Non-linear independent components estimation," 2014, *arXiv:1410.8516*. [Online]. Available: http://arxiv.org/abs/1410.8516

[13] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1×1 convolutions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10215–10224.

[14] N. Lawrence, "Probabilistic non-linear principal component analysis with Gaussian process latent variable models," *J. Mach. Learn. Res.*, vol. 6, pp. 1783–1816, Nov. 2005.

[15] M. Titsias and N. D. Lawrence, "Bayesian Gaussian process latent variable model," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, May 2010, pp. 844–851.

[16] B. J. Frey, G. E. Hinton, and P. Dayan, "Does the wake-sleep algorithm produce good density estimators," in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 661–667.

[17] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 219–230, Feb. 1998.

[18] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra, "Deep autoregressive networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 1242–1250.

[19] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with PixelCNN decoders," *CoRR*, vols. abs/1606.05328, pp. 1–13, Jun. 2016.

[20] G. E. Hinton and S. Osindero, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7 pp. 1527–1554, 2006.

[21] G. Alain *et al.*, "GSNs: Generative stochastic networks," *Inf. Inference*, vol. 5, no. 2, pp. 210–249, Jun. 2016.

[22] M. C. Cario and B. L. Nelson, "Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix," Dept. Ind. Eng. Manage. Sci., Northwestern Univ., Evanston, IL, USA, Tech. Rep., 1997.

[23] M. Falk, "A simple approach to the generation of uniformly distributed random variables with prescribed correlations," *Commun. Statist. Simul. Comput.*, vol. 28, no. 3, pp. 785–791, Jan. 1999.

[24] R. B. Nelsen, *An Introduction to Copulas* (Springer Series in Statistics). Berlin, Germany: Springer-Verlag, 2006.

[25] A. Sklar, "Fonctions de répartition à n dimensions et leurs marges" *Publications de l'Institut de Statistique de l'Université de Paris*, vol. 8, pp. 229–231, 1959.

[26] T. Bedford, A. Daneshkhah, and K. J. Wilson, "Approximate uncertainty modeling in risk analysis with vine copulas," *Risk Anal.*, vol. 36, no. 4, pp. 792–815, Apr. 2016.

[27] S. Demarta and A. J. McNeil, "The t copula and related copulas," *Int. Stat. Rev.*, vol. 73, no. 1, pp. 111–129, Jan. 2007.

[28] T. Bedford and R. Cooke, "Probability density decomposition for conditionally dependent random variables modeled by vines," *Ann. Math. Artif. Intell.*, vol. 32, no. 1–4, pp. 245–268, 2001.
[29] N. Tagasovska, D. Ackerer, and T. Vatter, "Copulas as high-dimensional generative models: Vine copula autoencoders," 2019, *arXiv:1906.05423*. [Online]. Available: http://arxiv.org/abs/1906.05423
[30] R. Fortet and E. Mourier, "Convergence de la répartition empirique vers la répartition théorique," *Annales Scientifiques de l'École normale supérieure*, vol. 70, no. 3, pp. 267–285, 1953.
[31] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *J. Mach. Learn. Res.*, vol. 13, pp. 723–773, Mar. 2012.
[32] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani, "Training generative neural networks via maximum mean discrepancy optimization," in *Proc. 31st Conf. Uncertainty Artif. Intell.*, Amsterdam, The Netherlands, 2015, pp. 258–267.
[33] Y. LeCun and C. Cortes. (2010). *MNIST Handwritten Digit Database*. Accessed: Apr. 10, 2019. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[34] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3730–3738.
[35] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vols. abs/1603.04467, pp. 1–19, Mar. 2016.
[36] J. Li, J. Jia, and D. Xu, "Unsupervised representation learning of image-based plant disease with deep convolutional generative adversarial networks," in *Proc. 37th Chin. Control Conf. (CCC)*, Jul. 2018, pp. 9159–9163.
[37] D. G. Clayton, "A model for association in bivariate life tables and its application in epidemiological studies of familial tendency in chronic disease incidence," *Biometrika*, vol. 65, no. 1, pp. 141–151, 1978.
[38] A. Borji, "Pros and cons of GAN evaluation measures," *Comput. Vis. Image Understand.*, vol. 179, pp. 41–65, Feb. 2019.
[39] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *CoRR*, vols. abs/1606.03498, pp. 1–10, Jun. 2016.
[40] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6626–6637.
[41] M Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–36.
[42] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vols. abs/1512.00567, pp. 1–10, Dec. 2015.

Nunzio A. Letizia (Graduate Student Member, IEEE) received the Laurea Magistrale degree in electrical engineering *(summa cum laude)* from the University of Udine, Udine, Italy, in 2018. He is currently pursuing the Ph.D. degree with the University of Klagenfurt, Klagenfurt, Austria.

He is currently a University Assistant with the University of Klagenfurt. During his studies, he was awarded with a full scholarship at Scuola Superiore dell'Universitá degli studi di Udine. His research interests spread from mathematics and physics to communication engineering. His research areas are in the field of signal processing and statistical learning.

Andrea M. Tonello (Senior Member, IEEE) received the Dr.Eng. degree (Hons.) in electronics and the Dr.Res. degree in electronics and telecommunications from the University of Padova, Padua, Italy, in 1996 and 2002, respectively.

From 1997 to 2002, he was with Bell Labs-Lucent Technologies, Whippany, NJ, USA, as a Member of the Technical Staff. Then, he was promoted to the Technical Manager and appointed to the Managing Director of the Bell Labs Italy Division. In 2003, he joined the University of Udine, Udine, Italy, where he became an Aggregate Professor in 2005 and an Associate Professor in 2014. He is currently a Professor of embedded communication systems with the University of Klagenfurt, Klagenfurt, Austria. He is also the Founder of the spinoff company, WiTiKee, Basagliapenta, Italy.

Dr. Tonello received several awards, including the Distinguished Visiting Fellowship from the Royal Academy of Engineering, U.K., in 2010, the IEEE VTS and COMSOC Distinguished Lecturer Awards in 2011, 2015, and 2018, the UC3M Chair of Excellence for the term 2019–2020, and nine best paper awards. He served/serves as an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE ACCESS, and *IET Smart Grid*. He was the Chair of the IEEE ComSoc Technical Committee on Power Line Communications from 2014 to 2018. He is also the Chair of the IEEE ComSoc Technical Committee on Smart Grid Communications. He has been appointed as the Director of Industry Outreach of the IEEE ComSoc for the term 2020–2021.