Quasi-random number generators for multivariate
distributions based on generative neural networks

Marius Hofert\textsuperscript{1}, Avinash Prasad\textsuperscript{2}, Mu Zhu\textsuperscript{3}

2018-11-05

Abstract

Generative moment matching networks are introduced as quasi-random number genera-
tors for multivariate distributions. So far, quasi-random number generators for non-uniform multivariate distributions require a careful design, often need to exploit
specific properties of the distribution or quasi-random number sequence under con-
sideration, and are limited to few models. Utilizing generative neural networks, in
particular, generative moment matching networks, allows one to construct quasi-random
number generators for a much larger variety of multivariate distributions without such
restrictions. Once trained, the presented generators only require independent quasi-
random numbers as input and are thus fast in generating non-uniform multivariate
quasi-random number sequences from the target distribution. Various numerical ex-
amples are considered to demonstrate the approach, including applications inspired by
risk management practice.

Keywords
Generative moment matching networks, learning a distribution, quasi-random numbers,
copulas, distribution of sums of dependent random variables, expected shortfall.

MSC2010
62H99, 65C60, 60E05, 00A72, 65C10.

1 Introduction

We are interested in finding an answer to the following question from the realm of quasi-
Monte Carlo methods:

\textsuperscript{1}Department of Statistics and Actuarial Science, University of Waterloo, 200 University Avenue West, Waterloo, ON, N2L 3G1, marius.hofert@uwaterloo.ca. The author acknowledges support from NSERC (Grant RGPIN-5010-2015).

\textsuperscript{2}Department of Statistics and Actuarial Science, University of Waterloo, 200 University Avenue West, Waterloo, ON, N2L 3G1, a2prasad@uwaterloo.ca. The author acknowledges support from NSERC (PGS D Scholarship).

\textsuperscript{3}Department of Statistics and Actuarial Science, University of Waterloo, 200 University Avenue West, Waterloo, ON, N2L 3G1, mu.zhu@uwaterloo.ca. The author acknowledges support from NSERC (RGPIN-2016-03876).
1 Introduction

How can we construct a quasi-random number generator for a large variety of multivariate distributions?

To this end, let $X = (X_1, \ldots, X_d)$ be a $d$-dimensional random vector with distribution function $F_X$, which, by Sklar’s Theorem, allows for the representation

$$F_X(x) = C(F_{X_1}(x_1), \ldots, F_{X_d}(x_d)), \quad x = (x_1, \ldots, x_d) \in \mathbb{R}^d,$$

where $C : [0, 1]^d \to [0, 1]$ is a copula of $F_X$ and $F_{X_1}, \ldots, F_{X_d}$ are the margins of $F_X$; see Nelsen (2006) or Joe (2014) for an introduction to copulas. Although not necessary, alone for the sake of simplicity we will focus on the case where $F_X(x)$ is absolutely continuous, in which case $C$ is uniquely determined by (1) and $f_X$ denotes the density of $F_X$; examples presented later also cover the case where the latter assumptions is not fulfilled.

Existing $F_X$ can be partitioned into three categories. Of the first kind are those $F_X$ for which a quasi-random number generator is known. A trivial example would be if $C$ is the independence copula $C(u) = u_1 \cdot \ldots \cdot u_d$, so if $X$ has independent components; there are multivariate Sobol’ or generalized Halton sequences available, see, for example, Lemieux (2009, Chapters 5). Another set of examples consists of those $F_X$ for which there is a one-to-one transformation known for constructing $X$ from a random vector $Z = (Z_1, \ldots, Z_d)$ with independent components. The most prominent such transformation is the inverse Rosenblatt transform, see Rosenblatt (1952), and the corresponding conditional distribution method (CDM) for sampling; see Embrechts et al. (2003) or Hofert (2010, p. 45). In principle, this method can be used to construct quasi-random number generators; see Cambou et al. (2017) where this idea has recently been introduced to address the motivating question for specific $C$. Concrete examples include the multivariate normal or Student $t$ distribution, as well as all models $F_X$ with a Clayton copula $C$. Of the second kind are those $F_X$ for which there is no one-to-one transformation from $Z$ to $X$ known or for which such a transformation is not analytically available and thus not applicable. However, distributions $F_X$ of this kind at least allow for a stochastic representation of $X$ which can be exploited for constructing a quasi-random number generator. Concrete examples in this category include $F_X$ with Frank or Joe copulas; see Cambou et al. (2017). Such $F_X$ share that their quasi-random number generators are carefully designed, which typically requires both knowledge about the construction of uniform quasi-random number sequences as well as the (loosely speaking) relative importance of the component random variables which form the stochastic representation of $X$. Constructing quasi-random number generators is typically already a hard problem for $F_X$ of the second kind. The third kind of $F_X$ are those for which there is no quasi-random number generator known. This is the case for most $F_X$, including those based on whole classes of copulas such as grouped normal variance mixture copulas, Archimad copulas, nested Archimedean copulas or extreme-value copulas, but also distributions for which we do not know the underlying copula $C$, for example.

With models $F_X$ of the second or, especially, the third kind in mind, we introduce generative neural networks (GNNs) for constructing quasi-random number generators and thus providing an answer to our motivating question. GNNs form a class of models which
2 Quasi-random GMMN samples

2.1 Generative neural networks

Most GNNs attempt to learn a map $f$ from a sample from an arbitrary $p$-dimensional prior density $f_Z$ of a random vector $Z = (Z_1, \ldots, Z_p)$ to a sample from the $d$-dimensional target density $f_X$ of a random vector $X = (X_1, \ldots, X_d)$. Often $f$ is a deep GNN. Popular GNNs primarily differ in strategies to most accurately and efficiently learn $f$. To this end, GANs (see Goodfellow et al. (2014)) use a min-max game between two (deep) neural networks (NNs), the generator and discriminator, to learn $f$; VAEs (see Kingma and Welling (2014)) use a variational inference approach which involves two (deep) NNs, an encoder and a decoder, after which the decoder can be used as the map $f$; and GMMNs (see Li et al. (2015)) utilize a (kernel) maximum mean discrepancy statistic to ensure the distribution of the generated sample is close to the target density.

In what follows, we will focus on GMMNs to learn $f$ since this approach utilizes an objective function which most straightforwardly targets matching samples from the NN
output with samples from the target density. In contrast, GANs and VAEs are geared towards generating a (single) “realistic” observation from the target density and require more sophisticated learning mechanisms to learn $f$ effectively. Thus while GANs and VAEs are notably successful in generating “realistic looking” synthetic images or textual data, they are not particularly equipped to learn the underlying distribution of the data well; see Appendix A where we illustrate this point for a well-known dataset from the deep learning literature.

To describe these GNN models, we start by defining the deep NN representation that will be used throughout this paper.

### 2.1.1 Deep neural networks

The **multi-layer perceptron (MLP)** is the quintessential deep NN. It is a composition of multiple layers of **perceptrons** or **neurons**, each of which uses a non-linear function to aggregate its corresponding inputs. Since the MLP is the only type of NN architecture we will use in this paper, we will refer to it as NN in what follows.

To fix ideas, let $L$ be the number of (hidden) layers in the NN and, for each $l = 0, \ldots, L + 1$, let $d_l$ be the dimension of layer $l$, that is, the number of neurons in layer $l$. In this notation, layer $l = 0$ refers to the **input layer** which consists of the input $z \in \mathbb{R}^p$ for $p = d_0$ and layer $l = L + 1$ refers to the **output layer** which consists of the output $y \in \mathbb{R}^d$ for $d = d_{L+1}$. Layers $l = 1, \ldots, L + 1$ can be described in terms of the output $a_{l-1} \in \mathbb{R}^{d_{l-1}}$ of layer $l - 1$ via

\[
\begin{align*}
  a_0 &= z \in \mathbb{R}^{d_0}, \\
  a_l &= f_l(a_{l-1}) = \phi_l(W_l a_{l-1} + b_l) \in \mathbb{R}^{d_l}, \quad l = 1, \ldots, L + 1, \\
  y &= a_{L+1} \in \mathbb{R}^{d_{L+1}},
\end{align*}
\]

with **weight matrices** $W_l \in \mathbb{R}^{d_l \times d_{l-1}}$, **bias vectors** $b_l \in \mathbb{R}^{d_l}$ and **activation functions** $\phi_l$; note that for vector inputs the activation function $\phi_l$ is understood to be applied componentwise. Figure 1 visualizes this construction and the notation we use. The NN $f_\theta : \mathbb{R}^p \leftarrow \mathbb{R}^d$ can be written as the composition

\[
f_\theta = f_{L+1} \circ f_L \circ \cdots \circ f_2 \circ f_1,
\]

with its (flattened) parameter vector given by

\[
\theta = (W_1, \ldots, W_{L+1}, b_1, \ldots, b_{L+1}).
\]

To fit $\theta$, we use the backpropagation algorithm (a stochastic gradient descent) based on a **cost function** $E$ (also known as loss or objective function) between the target output $x = x(z) \in \mathbb{R}^{d_{L+1}}$ and the actual output $y \in \mathbb{R}^{d_{L+1}}$ predicted by the NN.

The expressive power of NNs is primarily characterized by the universal approximation theorem; see Goodfellow et al. 2016, Chapter 6). In particular, given suitable activation
Figure 1 Structure of an NN with input $z = (z_1, \ldots, z_{d_0})$, $L = 1$ hidden layer with output $a_1 = f_1(a_0) = \phi_1(W_1a_0 + b_1)$ and output layer with output $y = a_2 = f_2(a_1) = \phi_2(W_2a_1 + b_2)$; note that in the figure, $W_{l,j}$ denotes the $j$th row of $W_l$ and $b_{l,j}$ the $j$th row of $b_l$. 
functions, a single-layer neural network with a finite number of neurons can approximate any continuous function on a compact subset of the multidimensional Euclidean space; see Nielsen (2015, Chapter 4) for a visual and interactive presentation of the universal approximation theorem. Cybenko (1989) first proposed such universal approximation results for the sigmoid activation function, \( \phi(x) = 1/(1 + e^{-x}) \) and Hornik (1991) then generalized the results to include arbitrary bounded and non-constant activation functions.

In recent years, the rectified linear unit (ReLU)

\[ \phi(x) = \max\{0, x\} \]

has become the most popular activation function for efficiently training NNs. This unbounded activation function does not satisfy the assumptions of the universal approximation theorem in Hornik (1991). However, there have since been numerous theoretical investigations into the expressive power of deep NNs with ReLU activation functions; see, for example, Pascanu et al. (2013), Montufar et al. (2014) or Arora et al. (2016). In particular, for certain conditions on the number of layers and neurons in the NN, Arora et al. (2016) provide a similar universal approximation theorem for NNs with ReLU activation functions.

### 2.1.2 Generative moment matching networks

As is standard in the literature, we assume independence among the components of \( \mathbf{Z} = (Z_1, \ldots, Z_p) \) from the prior density, that is, the prior density factors into the product of its marginal densities. Typical choices for the latter are either \( U(0, 1) \) or \( \mathcal{N}(0, 1) \) distributions; the latter denotes the standard normal distribution. The objective is then to generate samples from the target density \( f_X \) of \( \mathbf{X} = (X_1, \ldots, X_d) \) via the learned NN \( f_\theta : \mathbb{R}^p \to \mathbb{R}^d \) which represents the deterministic mapping from a sample from the prior distribution to a sample from the target distribution.

To learn \( f_\theta \) we assume to have \( n_{\text{trn}} \) training data points \( \mathbf{X}_1, \ldots, \mathbf{X}_{n_{\text{trn}}} \) from \( \mathbf{X} \); as explained before, we assume this to be a random sample from \( \mathbf{X} \) but it could equally well be a sufficiently large sample from a real dataset without knowing \( f_X \). Based on an input sample \( \mathbf{Z}_1, \ldots, \mathbf{Z}_{n_{\text{trn}}} \) of the prior distribution, the NN generates the output sample \( \mathbf{Y}_1, \ldots, \mathbf{Y}_{n_{\text{trn}}} \), where \( \mathbf{Y}_i = f_\theta(\mathbf{Z}_i) \), \( i = 1, \ldots, n_{\text{trn}} \). We are thus interested in whether the two samples \( \mathbf{X} = (\mathbf{X}_1^\top, \ldots, \mathbf{X}_{n_{\text{trn}}}^\top)^\top \in \mathbb{R}^{n_{\text{trn}} \times d} \) and \( \mathbf{Y} = (\mathbf{Y}_1^\top, \ldots, \mathbf{Y}_{n_{\text{trn}}}^\top)^\top \in \mathbb{R}^{n_{\text{trn}} \times d} \) come from the same distribution, that is, whether \( f_Y = f_X \).

To answer this question we utilize as cost function \( E \) the maximum mean discrepancy (MMD) statistic (from the kernel two-sample test introduced in Gretton et al. (2007)) whose sample version is given by

\[
\text{MMD}(\mathbf{X}, \mathbf{Y}) = \sqrt{\frac{1}{n_{\text{trn}}} \sum_{i=1}^{n_{\text{trn}}} \sum_{i_2=1}^{n_{\text{trn}}} (K(\mathbf{X}_{i_1}, \mathbf{X}_{i_2}) - 2K(\mathbf{X}_{i_1}, \mathbf{Y}_{i_2}) + K(\mathbf{Y}_{i_1}, \mathbf{Y}_{i_2})),}
\]

(2)

where \( K(\cdot, \cdot) : \mathbb{R}^{d} \times \mathbb{R}^{d} \to \mathbb{R} \) denotes a kernel (similarity) function. When a cost function such as the MMD statistic is expressed in terms of a kernel function, the samples are
Quasi-random GMMN samples are implicitly mapped to an infinite-dimensional feature space which corresponds to a (universal) reproducing kernel Hilbert space for certain kernels. Based on this correspondence, Gretton et al. (2007) and Gretton et al. (2012a) show that the Maximum Mean Discrepancy (MMD) converges in probability to 0 for \( n_{\text{trn}} \rightarrow \infty \) if and only if \( f_Y = f_X \). Minimizing the MMD with a universal kernel can thus be interpreted as matching all the moments of \( f_X \) and \( f_Y \), hence making MMD an intuitive choice as cost function for training the deep NN \( f_\theta \); our choice of kernel \( K \) is addressed in Section 3.2.

As suggested by Li et al. (2015), we adopt a mini-batch optimization procedure. This is necessary because the MMD-criterion (2) would otherwise require the evaluation of all \( \binom{n_{\text{trn}}}{2} \) pairs of observations, which is memory-prohibitive for even moderately large \( n_{\text{trn}} \). Rather than directly optimizing the MMD for the entire training dataset, we partition it into \( b \)atches of size \( n_{\text{bat}} \) and use the batches sequentially to update the parameters of the NN using the Adam optimizer of Kingma and Ba (2014). Instead of following the gradient at each iterative step, the Adam optimizer essentially uses a “memory-sticking gradient” – a weighted combination of the current gradient and past gradient from earlier iterations. After all the training data is exhausted, so roughly after \( \frac{n_{\text{trn}}}{n_{\text{bat}}} \) many batches were considered, one epoch of the training of the GMMN is completed. The overall training procedure is completed after \( n_{\text{epo}} \) epochs. The training of the GMMN can thus be summarized as follows; see Li et al. (2015).

**Algorithm 2.1 (Training GMMNs)**

1) Fix the maximum number \( n_{\text{epo}} \) of epochs and the batch size \( n_{\text{bat}} < n_{\text{trn}} \) per epoch, where \( n_{\text{bat}} \) is assumed to divide \( n_{\text{trn}} \).

2) Initialize the epoch counter \( k = 0 \) and the GMMN’s parameter vector \( \theta = \theta^{(0)} \); we follow Glorot and Bengio (2010) according to which the components of \( \theta^{(0)} \) are initialized as \( W_l \sim U(-\sqrt{6}/(d_l + d_{l-1}), \sqrt{6}/(d_l + d_{l-1}))^{d_l \times d_{l-1}} \) and \( b_l = 0 \) for \( l = 1, \ldots, L + 1 \).

3) For epoch \( k = 1, \ldots, n_{\text{epo}} \), do:

   3.1) Randomly partition the prior distribution sample \( Z_1, \ldots, Z_{n_{\text{trn}}} \) and training sample \( X_1, \ldots, X_{n_{\text{trn}}} \) into corresponding \( \frac{n_{\text{trn}}}{n_{\text{bat}}} \) non-overlapping batches \( Z_{1,b}^{(b)}, \ldots, Z_{n_{\text{bat}},b}^{(b)} \) and \( X_{1,b}^{(b)}, \ldots, X_{n_{\text{bat}},b}^{(b)} \), \( b = 1, \ldots, n_{\text{trn}}/n_{\text{bat}} \), of size \( n_{\text{bat}} \) each.

3.2) For batch \( b = 1, \ldots, n_{\text{trn}}/n_{\text{bat}} \), do:

   3.2.1) Compute the NN output \( Y_i^{(b)} = f_{\theta^{(k-1)}}(Z_i^{(b)}) \), \( i = 1, \ldots, n_{\text{bat}} \).

   3.2.2) Compute the gradient \( \frac{\partial}{\partial \theta} \text{MMD}(X^{(b)}, Y^{(b)}) \) based on the samples \( X^{(b)} = (X_1^{(b)\top}, \ldots, X_{n_{\text{bat}},b}^{(b)\top})\top \) and \( Y^{(b)} = (Y_1^{(b)\top}, \ldots, Y_{n_{\text{bat}},b}^{(b)\top})\top \).

   3.2.3) Then, take a gradient step to update \( \theta^{(k-1)} \) to \( \theta^{(k)} \) according to Adam; see Kingma and Ba (2014, Algorithm 1).

4) Return \( \hat{\theta} = \theta^{(n_{\text{epo}})} \); the fitted GMMN is then \( f_{\hat{\theta}} \).
An algorithm for sampling \( Y \) from \( f_Y \) via the GMMN based on a pseudo-random number generator (PRNG) for the input \( Z \) is given as follows.

**Algorithm 2.2 (Sampling from GMMNs via PRNGs)**

1) Fix the number \( n_{\text{gen}} \) of samples to generate from \( f_Y \).
2) Draw \( Z_i \overset{\text{ind.}}{\sim} f_Z, i = 1, \ldots, n_{\text{gen}} \).
3) Return \( Y_i = f_Y^\hat{\theta}(Z_i), i = 1, \ldots, n_{\text{gen}} \).

As the components of \( Z \) can be taken independent, the inversion method implies that \( Z \) is in distribution equal to \( F_Z^{-1}(U') \) where \( U' \sim U(0, 1) \) and

\[
F_Z^{-1}(u) = (F_{Z_1}^{-1}(u_1), \ldots, F_{Z_p}^{-1}(u_p)), \quad u \in [0, 1]^p.
\]

Therefore, for sampling \( Z_1, \ldots, Z_{n_{\text{gen}}} \) in Step 2) of Algorithm 2.2, one can simply draw \( U'_1, \ldots, U'_{n_{\text{gen}}} \overset{\text{ind.}}{\sim} U(0, 1)^p \) and transform, for each \( j = 1, \ldots, p \), the \( j \)th component sample \( U'_j, \ldots, U'_{n_{\text{gen}}j} \) with the quantile function \( F_{Z_j}^{-1} \) of \( Z_j \) to the distribution of \( Z_j \) of interest.

The next step is to replace the pseudo-random numbers by quasi-random numbers.

### 2.2 Quasi-random numbers

The idea behind quasi-random number generators (QRNGs) is to replace pseudo-random numbers by a low-discrepancy point set \( P_{n_{\text{gen}}} = \{v_1, \ldots, v_{n_{\text{gen}}}\} \subseteq [0, 1]^p \) in order to produce a more homogeneous coverage of \( [0, 1]^p \) in comparison to PRNGs; to obtain unbiased estimators and estimates of their variances, one typically randomizes \( P_{n_{\text{gen}}} \) in applications; see later. That is, with respect to a certain discrepancy measure, the empirical distribution of the \( P_{n_{\text{gen}}} \) is closer to the uniform distribution \( U(0, 1)^p \) than samples drawn using PRNGs.

Established notions of the discrepancy of a point set \( P_{n_{\text{gen}}} \) are as follows. The **discrepancy function** of \( P_{n_{\text{gen}}} \) in an interval \( I = [0, b] = \prod_{j=1}^p [0, b_j), b_j \in (0, 1], j = 1, \ldots, p \), is defined by

\[
D(I; P_{n_{\text{gen}}}) = \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} \mathbf{1}_{\{v_i \in I\}} - \lambda(I),
\]

where \( \lambda(I) \) is the \( p \)-dimensional Lebesgue measure of \( I \). Thus the discrepancy function is the difference between the number of points of \( P_{n_{\text{gen}}} \) in \( I \) and the probability of a \( p \)-dimensional standard uniform random vector to fall in \( I \). For \( \mathcal{A} = \{[0, b) : b \in (0, 1]^p\} \), the **star discrepancy** of \( P_{n_{\text{gen}}} \) is defined by

\[
D^*(P_{n_{\text{gen}}}) = \sup_{I \in \mathcal{A}} |D(I; P_{n_{\text{gen}}})|.
\]

If \( P_{n_{\text{gen}}} \) satisfies the condition \( D^*(P_{n_{\text{gen}}}) \in O(n_{\text{gen}}^{-1} \log^p n_{\text{gen}}) \), it is called a low-discrepancy sequence. There are different approaches to construct low-discrepancy sequences; see
Cambou et al. (2017) and Lemieux (2009, Chapters 5–6). We will work with the Sobol’ sequence, see Sobol’ (1967), and its implementation in the R package qrng.

When replacing PRNGs by QRNGs in Monte Carlo applications, it is important to obtain unbiased estimators and variance estimates, neither of which is typically possible with directly using \( P_{\text{gen}} \) due to its deterministic nature. To solve this, one can randomize \( P_{\text{gen}} \) via a \( U' \sim U(0,1)^p \) to obtain a randomized point set \( \tilde{P}_{\text{gen}} = \tilde{P}_{\text{gen}}(U') = \{ \tilde{v}_1, \ldots, \tilde{v}_{n_{\text{gen}}} \} \), where \( \tilde{v}_i = r(U', v_i) \), \( i = 1, \ldots, n_{\text{gen}} \), for a certain randomization function \( r \); see, for example, Cranley and Patterson (1976) for simple random shifts. This randomization preserves the low-discrepancy property of \( P_{\text{gen}} \) and the estimator of interest obtained from each \( \tilde{P}_{\text{gen}} \) is unbiased. Computing the estimator based on \( B \) such randomized point sets and computing the sample variance of the resulting \( B \) estimates then leads to an estimate of the variance of the estimator of interest. The Sobol’ sequence we work with uses a digital analog of a random shift as implemented in \texttt{sobol(, randomize = TRUE)} of the R package qrng.

### 2.3 Quasi-random GMMN sampling

Applications of RQMC methods include computing expectations

\[
\mathbb{E}(\Psi(X)),
\]

where \( X \) is a \( d \)-dimensional random vector as before and \( \Psi: \mathbb{R}^d \to \mathbb{R} \) is a measurable function. In our context, when using a sample from \( f_Y \) as an approximate sample from \( f_X \) and evaluating \( \mathbb{E}(\Psi(Y)) \) as an approximation to \( \mathbb{E}(\Psi(X)) \) by RQMC based on a sample of size \( n_{\text{gen}} \), we obtain

\[
\mathbb{E}(\Psi(X)) \approx \mathbb{E}(\Psi(Y)) \approx \mu_{\text{RQMC}}^{n_{\text{gen}}} = \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} \Psi(g(\tilde{u}_i)) = \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} h(\tilde{u}_i),
\]

where \( g = f_\theta \circ F_Z^{-1} \) and \( h = \Psi \circ g = \Psi \circ f_\theta \circ F_Z^{-1} \). Note that the error in the first approximation in (3) is small if the GMMN is trained well and the error in the second approximation is small if the unbiased \( \mu_{\text{RQMC}}^{n_{\text{gen}}} \) has a small variance. The primary bottle-neck in this setup is the error in the first approximation which is determined by the size \( n_{\text{trn}} \) of the training dataset and, in particular, by the batch size \( n_{\text{bat}} \) which is the major factor determining training efficiency of the GMMN. Given a sufficiently large \( n_{\text{bat}} \) and, by extension, \( n_{\text{trn}} \), the GMMN is trained well, which renders the approximation error in the first approximation in (3) negligible. However, in practice the size of \( n_{\text{bat}} \) is constrained by the quadratically increasing memory demands to compute the MMD loss function of the GMMN. Finally, let us note that the task of GMMN training and generation are separate steps which ensures that, once trained, generating quasi-random GMMN samples is fast.

We are now able to formulate a QRNG variant of Algorithm 2.2 for generating quasi-random GMMN samples.
Algorithm 2.3 (Sampling from GMMNs via QRNGs)

1) Fix the number $n_{\text{gen}}$ of samples to generate from $f_Y$.
2) Draw from a randomized Sobol’ sequence $\tilde{P}_{n_{\text{gen}}} = \{\tilde{v}_1, \ldots, \tilde{v}_{n_{\text{gen}}}\}$.
3) Compute $Z_i = F_{Z}^{-1}(\tilde{v}_i), i = 1, \ldots, n_{\text{gen}}$.
4) Return $Y_i = f_\theta(Z_i), i = 1, \ldots, n_{\text{gen}}$.

2.4 Analyzing the QMC estimator

In what follows we utilize the Koksma–Hlawka inequality to derive conditions under which the (non-randomized) quasi-Monte Carlo (QMC) estimator

$$\mu_{n_{\text{gen}}}^{\text{QMC}} = \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} h(v_i)$$

has a chance to have a small error when approximating $E(\Psi(Y))$. The variance of the RQMC estimator $\mu_{n_{\text{gen}}}^{\text{RQMC}}$ can be analytically expressed using Cambou et al. (2017, Proposition 6), however, we will rather numerically investigate the variance-reduction properties of $\mu_{n_{\text{gen}}}^{\text{RQMC}}$ for various functions $\Psi$ and transforms $g$ in Section 4.

By the Koksma–Hlawka inequality, see, for example, Niederreiter (1992),

$$\left| \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} h(v_i) - E(h(U')) \right| \leq V(h)D^*(P_{n_{\text{gen}}})$$

where $U' \sim U(0, 1)^p$ and the variation $V(h)$ is understood in the sense of Hardy and Krause; we refer to the right-hand side of the inequality as Koksma–Hlawka bound. Note that for any $g$ such that $g(U') \sim f_Y$, we know that $Y$ is in distribution equal to $g(U')$ and thus $E(\Psi(Y)) = E(\Psi(g(U'))) = E(h(U'))$. Based on this property, Cambou et al. (2017, Proposition 2) provide a Koksma–Hlawka bound for the change of variable $h$, given by

$$\left| \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} \Psi(y_i) - E(\Psi(Y)) \right| \leq V(h)D^*(P_{n_{\text{gen}}})$$

where $y_i = g(v_i), i = 1, \ldots, n_{\text{gen}}$, and $Y \sim f_Y$. Following Lemieux (2009, Section 5.6.1), we can derive an expression for $V(h)$. To this end, let

$$V^{(j)}(h; \alpha) = \int_{[0,1]^j} \left| \frac{\partial^j h^{(\alpha)}(v_{\alpha_1}, \ldots, v_{\alpha_j})}{\partial v_{\alpha_j} \cdots \partial v_{\alpha_1}} \right| dv_{\alpha_1} \cdots dv_{\alpha_j},$$

where $h^{(\alpha)}(v_{\alpha_1}, \ldots, v_{\alpha_j}) = h(\tilde{v}_1, \ldots, \tilde{v}_p)$ for $\tilde{v}_k = v_k$ if $k \in \{\alpha_1, \ldots, \alpha_j\}$ and $\tilde{v}_k = 1$ otherwise. Then

$$V(h) = \sum_{j=1}^{P} \sum_{\alpha: |\alpha| = j} V^{(j)}(h; \alpha), \quad (4)$$
where the inner sum is taken over all $\alpha = (\alpha_1, \ldots, \alpha_j)$ with $\{\alpha_1, \ldots, \alpha_j\} \subseteq \{1, \ldots, p\}$. Following Hlawka and Mück (1972) and Constantine and Savits (1996) Theorem 2.1, we have that

$$
\left| \frac{\partial^j h^{(\alpha)}(v_{a_1}, \ldots, v_{a_j})}{\partial v_{a_j} \cdots \partial v_{a_1}} \right| = \sum_{1 \leq |\beta| \leq j} \frac{\partial|\beta| \Psi}{\partial^{|\beta|} y_1 \cdots \partial^{|\beta|} y_d} \sum_{i=1}^j \sum_{m=1}^i c_k \prod_{m=1}^i \frac{\partial^{\kappa_m} g_k^{(\alpha)}(v_{a_1}, \ldots, v_{a_j})}{\partial^{\kappa_m j} v_{a_j} \cdots \partial^{\kappa_m 1} v_{a_1}},
$$

(5)

where $\beta \in \mathbb{N}_0^d$, $|\beta| = \sum_{k=1}^d \beta_k$ and where $\pi_i(\kappa, k)$ denotes the set of pairs $(\kappa, k)$ such that $k = (k_1, \ldots, k_i) \in \{1, \ldots, d\}^i$ and $\kappa = (\kappa_1, \ldots, \kappa_i)$ with $\kappa_m = (\kappa_{m,1}, \ldots, \kappa_{mj}) \in \{0,1\}^j$, $m = 1, \ldots, i$, and $\sum_{m=1}^i \kappa_{mi} = 1$ for $i = 1, \ldots, j$; see Constantine and Savits (1996) for more details on $\pi_i(\kappa, k)$ and the constants $c_k$. Furthermore, for index $j = 1, \ldots, d$, $g_j^{(\alpha)}(v_{a_1}, \ldots, v_{a_j}) = g_j(\tilde{v}_1, \ldots, \tilde{v}_p)$ (with $\tilde{v}_k$, $k = 1, \ldots, p$, defined as before) and $g_j(\tilde{v}_1, \ldots, \tilde{v}_p) = \phi_{L+1}(\tilde{W}_{L+1} a_l + \hat{b}_{L+1})$, where $a_l = \phi_l(\tilde{W}_l a_{l-1} + \hat{b}_l)$ for $l = 1, \ldots, L$ with $a_0 = F_{Z}^{-1}(\tilde{v})$ and $\tilde{W}_{L+1}$ denotes the $j$th row of $W_{L+1}$.

Based on the decomposition in (5), a sufficient condition to ensure that $V(h) < \infty$ is that all products of the form

$$
\frac{\partial^{|\beta|} \Psi}{\partial^{|\beta|} y_1 \cdots \partial^{|\beta|} y_d} \prod_{m=1}^i \frac{\partial^{\kappa_m} g_k^{(\alpha)}(v_{a_1}, \ldots, v_{a_j})}{\partial^{\kappa_m j} v_{a_j} \cdots \partial^{\kappa_m 1} v_{a_1}}, \quad i = 1, \ldots, j,
$$

are integrable. With this sufficient condition, we can formulate the following proposition.

**Proposition 2.4 (Sufficient conditions for finiteness of Koksma–Hlawka bound)**

Assume all appearing partial derivatives of $g$ and $\Psi$ exist and are continuous. Consider $g = f_\theta \circ F_{Z}^{-1}$, the point set $P_{n_{gen}} = \{v_1, \ldots, v_{n_{gen}}\} \subseteq [0,1)^p$ and let $y_i = g(v_i)$, $i = 1, \ldots, n_{gen}$, denote the quasi-random GMMN sample. Suppose that

1) $\Psi(y) < \infty$ for all $y \in \mathbb{R}^d$ and

$$
\frac{\partial^{|\beta|} \Psi(y)}{\partial^{|\beta|} y_1 \cdots \partial^{|\beta|} y_d} < \infty, \quad y \in \mathbb{R}^d,
$$

for all $\beta = (\beta_1, \ldots, \beta_d) \subseteq \{0, \ldots, d\}^d$ and $|\beta| \leq d$;

2) there exists an $M > 0$ such that $|D^k F_{Z}^{-1}| \leq M$, for each $k, j = 1, \ldots, p$, where $D^k$ denotes the $k$-fold derivative of its argument;

3) there exists, for each layer $l = 1, \ldots, L+1$ of the NN $f_\theta$, an $N_l > 0$ such that $|D^k \phi_l| \leq N_l$ for all $k = 1, \ldots, p$; and

4) the parameter vector $\theta = (\tilde{W}_1, \ldots, \tilde{W}_{L+1}, \hat{b}_1, \ldots, \hat{b}_{L+1})$ of the fitted NN is bounded.

Then there exists a constant $V$ independent of $n_{gen}$, but possibly depending on $\Psi$, $\theta$, $M$ and $N_1, \ldots, N_{L+1}$, such that

$$
\left| \frac{1}{n_{gen}} \sum_{i=1}^{n_{gen}} \Psi(y_i) - \mathbb{E}(\Psi(Y)) \right| \leq V D^*(P_{n_{gen}}).
$$
2 Quasi-random GMMN samples

Proof. Assumptions 2)–4) imply that all mixed partial derivatives of \( g = f_\hat{\theta} \circ F_Z^{-1} \) are bounded. By the assumption of continuous partial derivatives of \( g \), this implies that finite products of the form

\[
\prod_{m=1}^{i} \frac{\partial^{j_m} g_{k_m}^{(\alpha)}(v_{\alpha_1}, \ldots, v_{\alpha_j})}{\partial^{m_j} v_{\alpha_j} \cdots \partial^{m_1} v_{\alpha_1}}, \quad i = 1, \ldots, j,
\]

are integrable. By Assumption 1), Decomposition (5) and Hölder’s inequality, the quantity in (4) is bounded. This implies that \( h \) has bounded variation, so that the Koksma–Hlawka bound is finite.

An important consequence of Proposition 2.4 is that for suitable functions \( \Psi \), the convergence rate of the QMC estimator of \( E(\Psi(Y)) \) is determined via \( D^*(P_{n_{gen}}) \in O(n_{gen}^{-1} \log^p n_{gen}) \). The following remark provides insights into Assumptions 2)–4) of Proposition 2.4.

Remark 2.5

\( U(a,b)^p \) for \( a < b \), which is a popular choice for the prior distribution clearly satisfies Assumption 2) in Proposition 2.4. Assumption 3) is satisfied for various commonly used activation functions, such as:

1) Sigmoid. If \( \phi_l(x) = 1/(1 + e^{-x}) \) for layer \( l \), then \( N_l = 1 \).
2) ReLU. If \( \phi_l(x) = \max 0, x \) for layer \( l \), then \( N_l = 1 \). In this case, only the first derivative is (partly) non-zero. Additionally, note that the ReLU activation function is not differentiable at \( x = 0 \). However, even if \( \phi_l = \max \{0, x\} \) for all \( l = 1, \ldots, L + 1 \), the set of all pointwise discontinuities of the mixed partial derivatives of \( g \) is a null set. Hence, the discontinuities do not jeopardize the proof of Proposition 2.4.
3) Linear. If \( \phi_l(x) = x \) for layer \( l \), then \( N_l = 1 \). Only the first derivative is non-zero.
4) Tanh. If \( \phi_l(x) = \tanh(x) \) for layer \( l \), then \( N_l = 1 \).
5) Scaled exponential linear unit (SELU); see Klambauer et al. (2017). If, for layer \( l \),

\[
\phi_l(x) = \begin{cases} 
\lambda \alpha (\exp(-x) - 1), & \text{if } x < 0, \\
\lambda x, & \text{if } x \geq 0,
\end{cases}
\]

where \( \lambda \) and \( \alpha \) are prespecified constants, then \( N_l = \max \{\lambda, \lambda \alpha, 1\} \). The same argument about discontinuities made with the ReLU activation function applies equally well to the case of the SELU activation function.

Assumption 4) of Proposition 2.4 is satisfied in practice because NNs are always trained with regularization on the parameters, which means \( \hat{\theta} \) always lies in a compact set. Additionally note that in the general case where \( g \) is characterized by a composition of NN layers and \( F_Z^{-1} \) with a different (but standard) activation function in each layer, all partial derivatives of \( g \) exist and are continuous. Moreover, for the activation functions and prior distributions listed above, all mixed partial derivatives of \( g \) are bounded.
3 GMMN sampling for copula models

We now investigate the construction of QRNGs for non-uniform multivariate distributions through the lens of copula models. Conceptually, assuming $F_X$ admits a density $f_X$, it implies that

$$f_X(x) = c(F_{X_1}(x_1), \ldots, F_{X_d}(x_d)) \prod_{i=1}^{d} f_{X_i}(x_i), \quad x \in \mathbb{R}^d,$$

where $c$ denotes the density of the copula $C$ and $f_{X_1}, \ldots, f_{X_d}$ denote the marginal densities of $F_X$. Our goal is to learn the copula $C$ using a GNN. To this end, we utilize GMMNs with target density $c$. We thus learn a NN $f_\theta : \mathbb{R}^p \to [0,1]^d$ corresponding to a map which turns a sample from the prior distribution with density $f_Z$ into a sample from the copula $C$ with density $c$. We start by addressing key implementation details of Algorithm 2.1 and by utilizing this algorithm to train $f_\theta$ for a wide variety of copula families.

3.1 Training setup of GMMNs

To obtain a training dataset, generate $U_1, \ldots, U_{n_{\text{trn}}} \sim C$ using a suitable copula PRNG; see, for example, the function `rCopula()` of the R package `copula`. In our experiments, we set $n_{\text{trn}} = 60\,000$ for all examples and found this to be a sufficiently large sample size for the training dataset.

Concerning the dimensionality of the prior distribution, we choose $p = d$, that is, the NN $f_\theta$ is set to be a $d$-to-$d$ transformation. In our experiments, we observed that when we choose $p < d$, the accuracy of the GMMN approximation suffers. Alternatively, if $p > d$, we observed that the low discrepancy of the point set $\tilde{P}_n$ is preserved to a lesser degree upon transformation. The latter observation corroborates intuition since RQMC methods are known to work better in lower dimensions.

For the prior distribution itself, we worked with $d$ independent $\mathcal{N}(0,1)$ distributions throughout, that is, $Z \sim \mathcal{N}(\mathbf{0}, I_d)$, where $I_d$ denotes the $d$ dimensional identity matrix. Empirically, this choice worked better than the uniform distribution despite the fact that $\mathcal{N}(\mathbf{0}, I_d)$ does not satisfy the assumptions in Proposition 2.4.

Unless otherwise specified, we use the same setting across all examples in the rest of the paper.

3.2 Choice of kernel, neural network architecture and training details

A common choice of kernel function in the MMD statistic in (2) is the universal Gaussian kernel

$$K(x, y; \sigma) = \exp(-\|x - y\|^2/(2\sigma^2)),$$

where $\sigma$ is the bandwidth parameter. As described in Gretton et al. (2012b), the choice of $\sigma$ influences the statistical efficiency of the MMD statistic. Instead of fine-tuning the
bandwidth parameter to achieve optimal statistical efficiency, we follow Li et al. (2015) and utilize a mixture of Gaussian kernels with different bandwidth parameters as a suitable proxy in what follows, that is, we utilize the kernel

\[
K(x, y) = \sum_{i=1}^{n_{kern}} K(x, y; \sigma_i),
\]

where \(K\) denotes the number of mixture components and \(\sigma_i\) denotes the bandwidth parameter for the Gaussian Kernel of the \(i\)th mixture component for \(i = 1, \ldots, n_{kern}\). By using different bandwidth parameters, each component of the mixture can span a different range. In our context, since we are working with copula samples where \(x, y \in [0, 1]^d\), choosing the \(n_{kern}\) bandwidth parameter is more straightforward. After some initial experimentation, we fix \(n_{kern} = 6\) and choose \((\sigma_1, \ldots, \sigma_6) = (0.001, 0.01, 0.15, 0.25, 0.50, 0.75)\).

Following the notation introduced in Section 2.1.1, the depth and width of the NN architecture we employ are chosen as \(L = 1\) and \(d_1 = 300\), respectively. In particular, we found a one-layer NN to be sufficient for the examples considered which is in line with the universal approximation theorem. Also the depth was found sufficient by experimentation. As activation functions, we chose ReLU as \(\phi_1\) for the hidden layer which offers computational efficiency via non-expensive and non-vanishing gradients. For the output layer, \(\phi_2\) is chosen to be the sigmoid activation to obtain output values in the range required for the NN, so \([0, 1]^d\).

Now let us consider the training which involves various choices for the implementation of Algorithm 2.1 including key tuning parameters, software and hardware aspects. First, the batch size is important for the proper training of GMMNs. A too small \(n_{bat}\) will lead to poor estimates of the population MMD cost function which is based on \(n_{trn}\) samples. A too large \(n_{bat}\) will incur quadratically growing memory requirements due to the structure of kernel similarity functions. After experimentation, our choice for this trade-off was \(n_{bat} = 5000\) across all examples considered. Second, as number of epochs we chose \(n_{epo} = 300\). This was generally sufficient in our experiments to obtain accurate results. Third, the tuning parameters of the Adam optimizer were set to the default values reported in Kingma and Ba (2014). Last, all our experiments and examples were implemented in R using the packages keras and tensorflow, which serve as R interfaces to the corresponding namesake Python libraries. Furthermore, all experiments were run on four NVIDIA Tesla P100 GPUs with 16GB RAM each. Put together, we can obtain a trained NN \(f_\theta\) for a copula \(C\).

3.3 Visualizing GMMN samples

We equip Algorithms 2.2 and 2.3 with a minor post-processing step to generate and visualize pseudo- and quasi-random GMMN samples, respectively, for various copulas. The post-processing step involves computing pseudo-observations of the NN output in order to remove any residual marginal non-uniformity from the GMMN samples. Pseudo-observations of
the NN output can be obtained via

\[ \hat{U}_{ij} = \frac{R_{ij}}{n_{\text{gen}} + 1}, \quad i \in \{1, \ldots, n_{\text{gen}}\}, \quad j \in \{1, \ldots, d\}, \]

where \( R_{ij} \) denotes the rank of \( Y_{ij} \) among \( Y_{1j}, \ldots, Y_{n_{\text{gen}}j} \).

Concerning copula dimensionality, we focus on bivariate copula samples in this section, for higher-dimensional copulas, see Sections 3.4 and 4. Furthermore, for one-parameter copulas, the parameters of the copulas will be chosen such that Kendall’s tau, denoted by \( \tau \), is equal to 0.25 (weak dependence), 0.50 (moderate dependence) or 0.75 (strong dependence); clearly, this only applies to copula families where there is a one-to-one mapping between the copula parameter and \( \tau \).

### 3.3.1 t copulas

The \( t \) copula model is a prominent member of the elliptical class of copulas. A \( d \)-dimensional \( t \)-copula is given by

\[ C(u) = t_{\nu,P}(t_{\nu}^{-1}(u_1), \ldots, t_{\nu}^{-1}(u_d)), \quad u \in [0, 1]^d, \]

where \( t_{\nu,P} \) denotes the distribution function of the \( d \)-dimensional \( t \) distribution with \( \nu \) degrees of freedom, location vector \( 0 \) and scale matrix \( P \) (a correlation matrix here), and \( t_{\nu}^{-1} \) denotes the quantile function of the univariate \( t \) distribution with \( \nu \) degrees of freedom. For all \( t \) copulas considered in this work, we fix \( \nu = 4 \), which is a common choice.

Figure 2 displays pseudo-random samples from bivariate \( t \) copulas with \( \tau = 0.25, 0.50 \) and \( 0.75 \) (left column), along with pseudo-random GMMN samples (middle column; see Algorithm 2.2) and quasi-random GMMN samples (right column; see Algorithm 2.3). Visually comparing the first two columns, the GMMN PRNG produces samples very similar to realizations directly obtained from the corresponding \( t \) copula thus indicating that the three GMMNs (for the different \( \tau \)) have been trained sufficiently well. As the last column of plots indicates, the low-discrepancy property of \( \tilde{P}_{n_{\text{gen}}} \) has been preserved well upon transformation. Clearly, this is only an visual indication yet and does not imply that such generated quasi-random GMMN samples indeed are able to reduce the variance of an estimator based on such samples in comparison to pseudo-random number sequences from \( t \) copulas; this will be investigated in Section 4. Furthermore, note that \( t \) copulas are copulas of the first kind as introduced in the introduction, see Cambou et al. (2017) for the explicit form of the inverse Rosenblatt transform for \( t \) copulas.

### 3.3.2 Archimedean copulas

Archimedean copulas are copulas of the form

\[ C(u) = \psi(\psi^{-1}(u_1), \ldots, \psi^{-1}(u_d)), \quad u \in [0, 1]^d, \]
Figure 2 Pseudo-random (left column), pseudo-random GMMN (middle column), and quasi-random GMMN (right column) samples from $t_4$ copulas with $\tau = 0.25$ (top row), 0.50 (middle row) and 0.75 (bottom row), each with a sample size of $n_{\text{gen}} = 1000$. 
for an Archimedean generator \( \psi \) which is a continuous, decreasing function \( \psi : [0, \infty] \rightarrow [0, 1] \) which satisfies \( \psi(0) = 1 \), \( \psi(\infty) = \lim_{t \rightarrow \infty} \psi(t) = 0 \) and which is strictly decreasing on \([0, \inf t : \psi(t) = 0]\). Archimedean generators which indeed generate proper copulas of the Archimedean class are, for example, \( \psi_C(t) = (1 + t)^{-1/\theta} \) (for \( \theta > 0 \)) and \( \psi_G(t) = \exp(-t^{1/\theta}) \) (for \( \theta \geq 1 \)), generating Clayton and Gumbel copulas, respectively.

Figure 3 displays pseudo-random samples from bivariate Clayton copulas with \( \theta \) chosen such that \( \tau = 0.25, 0.50 \), and \( 0.75 \) (left column), along with pseudo-random GMMN samples (middle column) and quasi-random GMMN samples (right column). Figure 4 displays the same set of plots for Gumbel copulas. As in the case of \( t \) copulas in Figure 2, we observe that the middle column plots in each of the three figures are fairly similar to the respective left column plots, suggesting that the underlying GMMNs were sufficiently trained. Moreover, the right column plots in these figures visually confirm the low discrepancy for all nine copulas presented. Note that Clayton copulas are copulas of the first kind (see Cambou et al. (2017) for the inverse Rosenblatt transform) and Gumbel copulas are copulas of the second kind (see Cambou et al. (2017) for the idea to exploit the Marshall–Olkin sampling algorithm of these copulas to construct a QRNG).

### 3.3.3 Nested Archimedean copulas

*Nested Archimedean copulas (NACs)* are Archimedean copulas with arguments possibly replaced by other NACs; see McNeil (2008) or Hofert (2012). In particular, this class of copulas allows to construct asymmetric extensions of Archimedean copulas. Important to note here is that NACs are copulas of the third kind as described in the introduction, so there is no QRNG for this class of copulas known yet. To demonstrate the ability of GMMNs to capture such dependence structures, we consider the simplest three-dimensional copula for visualization and investigate higher dimensional NACs in Sections 3.4 and 4.

The three-dimensional NAC we consider here is

\[
C(u) = C_0(C_1(u_1, u_2), u_3), \quad u \in [0, 1]^3, \tag{6}
\]

where \( C_0 \) is a Clayton copula with \( \tau_0 = 0.25 \) and \( C_1 \) is a Clayton copula with \( \tau_1 = 0.50 \). In Sections 3.4.2 and 4 we will present examples of five- and ten-dimensional NACs.

Figure 5 displays pseudo-random samples from \( C \) (left), along with pseudo-random GMMN samples (middle) and quasi-random GMMN samples (right). As before, the similarity indicates that (at least the bivariate margins of) the copula \( C \) was (were) learned sufficiently well by the GMMN. Furthermore, the right plot showcases that the low discrepancy of \( \tilde{P}_{gen} \) is visually preserved under the GMMN transform associated with \( C \); note that the low discrepancy could have even been preserved if the right-most plot had not visually shown it as we are considering projections here.
Figure 3 Pseudo-random (left column), pseudo-random GMMN (middle column) and quasi-random GMMN (right column) samples from Clayton copulas with $\tau = 0.25$ (top row), 0.50 (middle row) and 0.75 (bottom row), each with a sample size of $n_{\text{gen}} = 1000$. 
Figure 4 Pseudo-random (left column), pseudo-random GMMN (middle column) and quasi-random GMMN (right column) samples from Gumbel copulas with \( \tau = 0.25 \) (top row), 0.50 (middle row) and 0.75 (bottom row), each with a sample size of \( n_{\text{gen}} = 1000 \).
3 GMMN sampling for copula models

Figure 5 Pseudo-random (left), pseudo-random GMMN (middle) and quasi-random GMMN (right) samples from a NAC copula $C$ in (6) with $\tau_0 = 0.25$ and $\tau_1 = 0.50$, each with a sample size of $n_{\text{gen}} = 1000$.

3.3.4 Marshall–Olkin copulas

Bivariate Marshall–Olkin copulas are of the form

$$C(u_1, u_2) = \min\{u_1^{\alpha_1}u_2, u_1u_2^{1-\alpha_2}\}, \quad u_1, u_2 \in [0, 1],$$

where $\alpha_1, \alpha_2 \in [0, 1]$. A notable feature of Marshall–Olkin copulas is that they have both an absolutely continuous component and a singular component. In particular, the singular component is determined by all points which satisfy $u_1^{\alpha_1} = u_2^{\alpha_2}$. Accurately capturing this singular component presents a different challenge for the GMMN, which is why we included this copula of the first kind.

Figure 6 depicts pseudo-random samples from a Marshall–Olkin copula with $\alpha_1 = 0.75$ and $\alpha_2 = 0.60$, along with pseudo-random GMMN samples (middle) and quasi-random GMMN samples (right). The middle plot highlights that the approximate GMMN sample is fairly similar to the pseudo-random plot directly from $C$ on the left. Additionally, as was the case in the previous examples, the plot on the right again visually confirms that GMMN samples generated via QRNG can preserve the low-discrepancy property.

3.3.5 Mixture copulas

We now consider three models, all of which are equally weighted two-component mixture copulas with one component being a 90-degree-rotated $t_4$ copula with $\tau = 0.5$. The other component is a Clayton copula ($\tau = 0.50$) in the first model, a Gumbel copula ($\tau = 0.50$) in the second model and a Marshall-Olkin copula ($\alpha_1 = 0.75$ and $\alpha_2 = 0.60$) in the third model. The three mixture copula models are referred to as the Clayton-$t(90)$, Gumbel-$t(90)$ and MO-$t(90)$ copulas, respectively.

Figure 7 depicts pseudo-random samples from these three mixture copulas along with the corresponding pseudo-random and quasi-random GMMN samples. As in the previous
3 GMMN sampling for copula models

![Figure 6](image)

Figure 6 Pseudo-random (left), pseudo-random GMMN (middle) and quasi-random GMMN (right) samples from a Marshall–Olkin copula with \( \alpha_1 = 0.75 \) and \( \alpha_2 = 0.60 \), each with a sample size of \( n_{\text{gen}} = 1000 \).

examples, the middle column plots closely resemble the realizations from the copulas in the left column and the right column plots visually confirm that the quasi-random GMMN samples seem to preserve the low-discrepancy of \( \hat{P}_{n_{\text{gen}}} \). While there exists a blurred singular component in the GMMN samples of the MO-\( t(90) \) mixture, the main features of the mixture have still been captured. This is rather remarkable given how hard of a problem it is to even visually closely learn a distribution, see Appendix A to realize this, let alone the rather complicated distributions we consider here. Furthermore, our approach does not require any particular knowledge about the design of QRNGs or the copula models considered.

3.4 Assessing the accuracy of GMMN samples

After a purely visual inspection of the generated samples, let us now address the accuracy of the GMMN approximation more formally with the help of goodness-of-fit assessments. Formally, we aim to test the null hypothesis

\[
H_0 : C = C^{\text{null}} \quad \text{vs} \quad H_1 : C \neq C^{\text{null}},
\]

for the post-processed GMMN samples \( \hat{U}_1, \ldots, \hat{U}_{n_{\text{gen}}} \) with copula \( C \) and the given target copula \( C^{\text{null}} \). Note that one typically tests the null hypothesis that \( C \) belongs to a whole parametric family (whose parameters are estimated based on the data available), but we indeed need to consider the specific copula \( C^{\text{null}} \) here in the context of random number generation. For all bivariate copulas considered, we compute Rosenblatt’s transform under \( H_0 \) and check for non-uniformity (which would indicate departure from \( H_0 \)). For the higher-dimensional copulas considered, we present box plots of Cramér–von Mises statistics for pseudo-random GMMN samples in comparison to those computed directly from a PRNG of \( C^{\text{null}} \).
Figure 7 Pseudo-random (left column), pseudo-random GMMN (middle column) and quasi-random GMMN (right column) samples from a Clayton-$t(90)$ (top row), Gumbel-$t(90)$ (middle row) and a MO-$t(90)$ mixture (bottom row) copula, each with a sample size of $n_{gen} = 1000$. 

3 GMMN sampling for copula models
3 GMMN sampling for copula models

3.4.1 Bivariate copulas

The Rosenblatt transform $R$, see Rosenblatt (1952), for the bivariate copula $C_{\text{null}}$ maps $(U_1, U_2) \sim C_{\text{null}}$ to $(R_1, R_2) = (U_1, C_{\text{null}}(U_2 | U_1))$, where $C_{\text{null}}(u_2 | u_1)$ denotes the conditional distribution function of $U_2$ given $U_1 = u_1$ under $C_{\text{null}}$. We exploit the fact that $(U_1, U_2) \sim C_{\text{null}}$ if and only if $(R_1, R_2) \sim U(0, 1)^2$. To this end, we apply $R$ to the post-processed pseudo-random GMMN samples $\{(\hat{U}_{i1}^1, \hat{U}_{i2}^1), \ldots, (\hat{U}_{i\text{gen}1}, \hat{U}_{i\text{gen}2})\}$ and display the corresponding Rosenblatt-transformed GMMN pseudo-samples $\{(R_{11}^1, R_{12}^1), \ldots, (R_{\text{gen}11}, R_{\text{gen}2})\}$ for each copula under consideration. Any departure from $U(0, 1)^2$ is an indication of a deficiency of the GMMN to have learned the right distribution.

Figure 8 shows the Rosenblatt-transformed samples of the previously considered $t_4$ copulas (top row), Clayton copulas (middle row) and Gumbel copulas (bottom row) such that $\tau = 0.25$ (left column), $\tau = 0.5$ (middle column) and $\tau = 0.75$ (right column). Figure 9 shows the previously considered Clayton-$t(90)$ (left) and Gumbel-$t(90)$ (right) mixture copulas. We can conclude from all these plots that no significant deviation from uniformity is visible. This indicates that for these examples our GMMN approximation is good.

3.4.2 Higher-dimensional copulas

As Cramér–von Mises statistic, see Genest et al. (2009), we employ

$$S_{\text{gen}} = \int_{[0,1]^d} n_{\text{gen}}(C_{\text{gen}}(u) - C_{\text{null}}(u))^2 \, dC_{\text{gen}}(u),$$

where the empirical copula

$$C_{\text{gen}}(u) = \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} \mathbb{1}\{\hat{U}_{i1} \leq u_1, \ldots, \hat{U}_{id} \leq u_d\}, \quad u \in [0,1]^d,$$

is the distribution function of pseudo-observations. For each setup, we compute realizations of $S_{\text{gen}}$ twice, once for the case where $\hat{U}_i$, $i = 1, \ldots, n_{\text{gen}}$, are the (post-processed) GMMN pseudo-samples and once where they are the pseudo-observation obtained from a sample of the true copula under consideration. Also here, let us note that, under a usual goodness-of-fit scenario based on real data, $C_{\text{null}}$ is a fitted copula from a parametric family. However, in our case, $C_{\text{null}}$ is an a priori fixed copula. Nonetheless we can still use the statistic $S_{\text{gen}}$ to assess the accuracy of pseudo-random GMMN samples in this way.

For each copula considered, we fix $n_{\text{gen}} = 1000$ and compute $B = 100$ realizations of $S_{\text{gen}}$ based on pseudo-observations from pseudo-random GMMN samples and pseudo-samples directly from the copula under consideration. We then use box plots to depict the distribution of $S_{\text{gen}}$ in each case. Figure 10 displays these box plots for $t_4$ (top row), Clayton (middle row) and Gumbel (bottom row) copulas of dimensions $d = 5$ (left column) and $d = 10$ (right column) and $\tau = 0.50$. Similarly, Figure 11 displays such box plots for $d$-dimensional nested Clayton (left column) and nested Gumbel (right column) copulas for $d = 3$ (top row), $d = 5$ (middle row) and $d = 10$ (bottom row). The three-dimensional
Figure 8 Rosenblatt-transformed pseudo-random GMMN samples corresponding to the $t_4$ (top row), Clayton (middle row) and Gumbel (bottom row) copulas with $\tau = 0.25$ (left column), $\tau = 0.5$ (middle column) and $\tau = 0.75$ (right column).
4 Application of quasi-random GMMN sampling

NACs have a structure as in \( [6] \) with \( \tau_0 = 0.25 \) and \( \tau_1 = 0.50 \), the five-dimensional NACs have structure \( C_0(C_1(u_1, u_2), C_2(u_3, u_4, u_5)) \) with corresponding \( \tau_0 = 0.25, \tau_1 = 0.50 \) and \( \tau_2 = 0.75 \), and the ten-dimensional NACs have structure \( C_0(C_1(u_1, \ldots, u_5), C_2(u_6, \ldots, u_{10})) \) with corresponding \( \tau_0 = 0.25, \tau_1 = 0.50 \) and \( \tau_2 = 0.75 \).

We can observe from both figures that the distribution of \( S_{n_{\text{gen}}} \) based on pseudo-random GMMN samples and pseudo-random copula samples are similar, with slightly higher \( S_{n_{\text{gen}}} \) values for the pseudo-random GMMN samples, especially for \( d = 10 \). Overall, the (pseudo-observations of the) pseudo-random GMMN samples approximate the (pseudo-observations of the) pseudo-random copula samples quite well according to these assessments.

4 Application of quasi-random GMMN sampling

In this section we numerically investigate the variance of the RQMC estimator \( \mu_{n_{\text{gen}}}^{\text{RQMC}} \) for various functions \( \Psi \) and transforms \( g = f_\theta \circ \Phi^{-1} \) corresponding to different copulas \( C \). Where possible, we compare \( \mu_{n_{\text{gen}}}^{\text{RQMC}} \) with estimators arising from standard copula PRNGs and QRNGs; see Cambou et al. (2017). However, note that the QRNGs in the latter reference are only applicable to some of the copulas we consider here.

We consider two categories of functions \( \Psi \). The first category consists of two test functions which are primarily used to test the performance of \( \mu_{n_{\text{gen}}}^{\text{RQMC}} \) in terms of its ability to preserve the low discrepancy of \( \tilde{P}_{n_{\text{gen}}} \). For this category of functions, we plot absolute error estimates of the RQMC and other estimators to compare their convergence rate; note here that the theoretical convergence rate of the Monte Carlo (MC) estimator’s absolute error is of the order \( O(n_{\text{gen}}^{-0.5}) \). The second category of functions \( \Psi \) we consider are motivated from practical applications in risk management. For this category of functions, variance estimates will be computed to compare their convergence rate; note that the MC estimator’s variance has a theoretical convergence rate of the order \( O(n_{\text{gen}}^{-1}) \).

Variance and absolute error estimates will be computed based on \( B = 25 \) randomized
Figure 10 Box plots based on $B = 100$ realization of $S_{n_{\text{gen}}}$ computed from pseudo-observations of a pseudo-sample of size $n_{\text{gen}} = 1000$ of the true underlying copula (denoted by PRNG) and from pseudo-observations of pseudo-random GMMN samples (denoted by GMMN PRNG) for a $t_4$ (top row), Clayton (middle row) and Gumbel (bottom row) copula with $\tau = 0.5$ as well as $d = 5$ (left column) and $d = 10$ (right column).
Figure 11 As in Figure 10 but for nested Clayton (left column) and nested Gumbel (right column) copulas and for $d = 3$ (top row), $d = 5$ (middle row) and $d = 10$ (bottom row).
point sets $\tilde{P}_{n_{\text{gen}}}$ for each of $n_{\text{gen}} \in \{2^9, 2^{9.5}, \ldots, 2^{18}\}$ to help roughly gauge the convergence rate for all estimators. Furthermore, regression coefficients $\alpha$ (obtained by regressing the logarithm of the measure of interest on the logarithm of $n_{\text{gen}}$) are computed and displayed to allow an easy comparison of the corresponding convergence rates $O(n_{\text{gen}}^{-\alpha})$.

4.1 Test functions

The first test function we consider is

$$\Psi_1(U) = \frac{3}{d} \sum_{j=1}^{d} U_j^2.$$ 

Notice that $E(\Psi_1(U)) = 1$ for $U \sim C$ from any copula $C$. For experiments involving this test function we thus do not perform the post-processing step of building pseudo-observations of the NN output.

Figure 12 shows plots of absolute error estimates for estimating $E(\Psi_1(U))$ for $t_4$ (first row), Clayton (second row) and Gumbel (third row) copulas in dimensions $d = 2$ (left column), $d = 5$ (middle column) and $d = 10$ (right column), as well as nested Gumbel copulas (fourth row) in dimension $d = 3$ (left column), $d = 5$ (middle column) and $d = 10$ (right column), with hierarchical structure and parameterization as described in Section 3.4.2. For the $t_4$ and Clayton copulas we numerically compare the efficiency of the RQMC estimator based on quasi-random GMMN samples (with legend label “GMMN Sobol”) with the RQMC estimator based on the CDM transform (with legend label “CDM Sobol”) and the MC estimator (with legend label “PRNG”). For the Gumbel and nested Gumbel copulas, we can only compare the GMMN-based RQMC estimator with the MC estimator. The legend of each plot also provides the regression coefficients $\alpha$ which approximate the convergence rate for each estimator.

From Figure 12 we observe that the GMMN-based RQMC estimator clearly outperforms the MC estimator. Naturally, so does the CDM-based RQMC estimator for the copulas for which it is available. On the one hand, the rate of convergence of the GMMN-based RQMC estimator reduces with increasing copula dimensions; see also the regression coefficients $\alpha$ which decreases from 1 to approximately 0.7 when we comparing the two- and ten-dimensional copulas. On the other hand, GMMN-based RQMC estimator still outperforms the MC estimator.

The second test function we consider involves the Rosenblatt transform and is given by

$$\Psi_2(U) = \prod_{j=1}^{d} \frac{|4R_j - 1| + j}{1 + j},$$

where $R_1 = U_1$ and, for $j = 2, \ldots, d$ and if $U \sim C$, $R_j = C(j_1, \ldots, j_{j-1}|U_j, U_{j-1}, \ldots, U_1)$ denotes the conditional distribution function of $U_j$ given $U_1 = u_1, \ldots, U_{j-1} = u_{j-1}$.

Figure 13 shows plots of absolute error estimates for estimating $E(\Psi_2(U))$ for $t_4$ copulas (top row), Clayton (middle row) and Gumbel (bottom row) copulas in dimensions $d = 2$.
### 4 Application of quasi-random GMMN sampling

| Sampling Method | PRNG | CDM Sobol | GMMN Sobol | NAC Gumbel |
|----------------|------|-----------|------------|------------|
| d = 2          | α = 0.5 | α = 0.98  | α = 0.99   |             |
| d = 10         | α = 0.78 | α = 0.94  | α = 0.76   | α = 0.77   |
| d = 5          | α = 0.49 | α = 0.93  | α = 0.94   |             |

**Figure 12** Absolute error estimates based on $B = 25$ replications for estimating $\mathbb{E}(\Psi_1(U))$ via MC based on a PRNG, via the RQMC estimator based on the CDM (whenever available; rows 1–2 only) and via the GMMN-based RQMC estimator. Note that in rows 1–3, $d \in \{2, 5, 10\}$, whereas in row 4, $d \in \{3, 5, 10\}$. 

29
Figure 13: Absolute error estimates based on $B = 25$ replications for estimating $\mathbb{E}(\Psi_2(U))$ via MC based on a PRNG, via the RQMC estimator based on the CDM (whenever available; rows 1–2 only) and via the GMMN-based RQMC estimator. Note that in rows 1–3, $d \in \{2, 5, 10\}$. 
4 Application of quasi-random GMMN sampling

(left column), $d = 5$ (middle column) and $d = 10$ (right column). Note that we cannot apply the transform $\Psi_2(U)$ for $U$ following nested Gumbel copulas in this case (which are of the third kind). Overall, we observe the same pattern as for the results obtained for $\Psi_1$; see Figure 12. Although the reduction in convergence rate as the copula dimension increases is more severe for $\Psi_2$ than for $\Psi_1$, the GMMN-based RQMC estimator still outperforms the MC estimator.

4.2 Examples from risk management practice

Consider modeling the dependence of $d$ risk-factor changes (for example, logarithmic returns) of a portfolio; see McNeil et al. (2015, Chapters 2, 6 and 7). We will demonstrate the efficiency of our GMMN-based RQMC estimator by considering two functionals: the joint exceedance probability of all $d$ risk-factor changes over a threshold and the expected shortfall of the aggregate loss, that is, computing a risk measure based on the sum of all risk-factor changes.

For the first example, we focus solely on dependence models for the random vector $U = (U_1, \ldots, U_d)$ following the underlying copula. The joint exceedance probability of $(0.99, \ldots, 0.99)$ is given by

$$P(U_1 > 0.99, \ldots, U_d > 0.99) = E(\{U_1 > 0.99, \ldots, U_d > 0.99\}) = E(\Psi_3(U)),$$

for $\Psi_3(u) = \{u_1 > 0.99, \ldots, u_d > 0.99\}$. Computing such exceedance probabilities is of interest, for example, for quantifying the probability of joint losses of the $d$ (dependent) components of a portfolio.

Figure 14 shows plots of variance estimates for estimating $E(\Psi_3(U))$ for $t_4$ (first row), Clayton (second row) and Gumbel (third row) copulas in dimensions $d = 2$ (left column), $d = 5$ (middle column) and $d = 10$ (right column), as well as nested Gumbel copulas (fourth row) in dimension $d = 3$ (left column), $d = 5$ (middle column) and $d = 10$ (right column) with hierarchical structure and parameterization as described in Section 3.4.2. The setup here is thus the same as in Section 4.1 except for a different function to be estimated and the fact that we investigate variance estimates instead of absolute error estimates. We can observe from the plots that the GMMN-based RQMC estimator outperforms the MC estimator. Similar to $E(\Psi_1(U))$ and $E(\Psi_2(U))$, we see a decrease in the convergence rate of the GMMN-based RQMC estimator as the copula dimension increases, although the latter still outperforms the MC estimator.

Now consider expected shortfall, a popular risk measure in quantitative risk management practice. If $X = (X_1, \ldots, X_d)$ denotes a random vector of risk-factor changes with $N(0,1)$ margins, $S = \sum_{j=1}^d X_j$ is the aggregate loss. Expected shortfall at level 0.99 of $S$ is given by

$$\frac{1}{1 - 0.99} \int_{0.99}^1 F^{-1}_S(x) \, dx = E(S \mid S > F^{-1}_S(0.99)) = E(\Psi_4(X)),$$
Figure 14 Variance estimates based on $B = 25$ replications for estimating $\mathbb{E}(\Psi_3(U))$ via MC based on a PRNG, via the RQMC estimator based on the CDM (whenever available; rows 1–2 only) and via the GMMN-based RQMC estimator. Note that in rows 1–3, $d \in \{2, 5, 10\}$, whereas in row 4, $d \in \{3, 5, 10\}$. 
Figure 15 Variance estimates based on $B = 25$ replications for estimating $E(\Psi(X))$ via MC based on a PRNG, via the RQMC estimator based on the CDM (whenever available; rows 1–2 only) and via the GMMN-based RQMC estimator. Note that in rows 1–3, $d \in \{2, 5, 10\}$, whereas in row 4, $d \in \{3, 5, 10\}$. 
where $F^{-1}_S$ denotes the quantile function of $S$. As done previously, various copulas will be used to model the dependence between the components of $X$.

Figure 15 shows plots of variance estimates for estimating $E(\Psi_4(X))$ for the same copula models as considered before. We can observe from the plots that the GMMN-based RQMC estimator outperforms the MC estimator. Similar as before, we see a decrease in the convergence rate of the GMMN-based RQMC estimator as the copula dimension increases, although the latter still outperforms the MC estimator.

5 Discussion

This work was inspired by the simple question of how one can construct a quasi-random number generator for a large variety of multivariate distributions. Until recently, this was only possible for multivariate distributions of the first or second kind as described in the introduction. In general, for the vast majority of multivariate distributions, constructing quasi-random number generators is a hard problem, even for those of the second kind; see also Cambou et al. (2017). Our approach based on GNNs provides a first, universal approach for building quasi-random number generators for such distributions.

However, this does not mean that the problem is completely solved. Our approach depends on first learning a generator $f_\theta$ such that, given $Z$ (from some known distribution such as the multivariate uniform or standard normal), $f_\theta(Z)$ follows the targeted multivariate distribution. Conditional on this first step being successful, we can then replace $Z$ with $F^{-1}_Z(\tilde{v})$ to generate quasi-random samples.

In high dimensions, however, learning an entire distribution is a hard problem, and so is learning the generator $f_\theta$. At a superficial level, the literature on GNNs – and the many headlines covering them – may give the impression that GNNs are now capable of generating samples from very high-dimensional distributions. Our research tells us that this is most certainly not the case, and the GNN community itself is beginning to become aware of this pitfall as well; see, for example, Arjovsky et al. (2017), Arora et al. (2018), and Tolstikhin et al. (2017). In particular, while available evidence is convincing that any specific generated sample $f_\theta(Z_0)$, typically an image, can be very realistic in the sense that it looks just like a typical training sample, this is not the same as saying that the entire collection of generated samples $\{f_\theta(Z_1), f_\theta(Z_2), \ldots\}$ will have the same distribution as the training sample. In Appendix A, we demonstrate more specifically what this means.

After extensive experimentation, our conclusion is that widely cited GNNs such as VAEs and GANs are not yet usable for the purpose of learning an entire distribution, and that GMMNs are the only viable option currently. By and large, this is because GMMNs rely on the MMD-criterion rather than some other criteria, for example, mean squared error, that do not measure the discrepancy between entire distributions. Even so, this still does not mean GMMNs are practical for very high dimensions yet, simply because the fundamental curse of dimensionality cannot be avoided easily. It is simply not realistic to hope that one can learn an entire distribution in high dimensions from a training sample of only moderate
size, but, as we have noted earlier, the effectiveness of GMMNs depends critically on the batch size $n_{\text{bat}}$ used to train them. That is, increasing $n_{\text{bat}}$ will lead to better training of the GMMNs at the expense of quadratically increasing memory requirements.

Going forward there are two primary impediments to the construction of quasi-random number generators for higher dimensional copulas and distributions. Firstly, the problem of distribution learning via GNNs remains a challenging task in higher dimensions. For further investigation in this direction, we may consider using other goodness-of-fit statistics for multivariate distributions rather than the MMD-criterion as the loss function provided that the statistic is differentiable in order to train a GNN. Secondly, we discovered from our empirical investigation in Section 4 that the convergence rates of GMMN-based RQMC estimators decrease with increasing dimension. Preserving the low-discrepancy of quasi-random number sequences upon transformations in high dimensions remains an open challenge in this regard.

In order to effectively evaluate the GMMN’s performance, we focused on dependence structures described via known and fixed parametric copulas (or mixtures thereof) to construct the training dataset. The natural subsequent setup for some of the applications described in Section 4 would be to start directly with real datasets. In this scenario, two different approaches can be adopted for the construction of appropriate RQMC estimators. First one could directly model the dependence structure, that is, pseudo-observations, of the real dataset via GMMNs. This approach would be recommended for large datasets and datasets with dependence structures not well described by parametric copulas. Alternatively, one could first fit an appropriate copula model to the real dataset and then use the fitted copula model to construct an arbitrarily large training dataset for the GMMN. This approach would be more suitable for smaller datasets and datasets well described by tractable parametric copulas (at the expense of requiring a two-step estimation approach).

A Misconception regarding popular generative neural networks

Popular GNNs such as GANs (see Goodfellow et al. (2014)) and VAEs (see Kingma and Welling (2014)) have reported some “impressive” results for generating “realistic” observations (for example, images, videos and text) from a target density as described by a training dataset. However, it is important to note that these widely acclaimed results do not imply that the generative models have actually learned the underlying probability distribution of the data. In fact, there is sufficient evidence to suggest that GANs and VAEs do not accurately learn the target density. We now briefly illustrate this point with two generative models with standard network architectures trained on the MNIST dataset (see LeCun et al. (1998)) for the purpose of generating handwritten digits. The MNIST dataset consists of a training set of 60,000 handwritten digits. Each example in the MNIST dataset is represented by a $28 \times 28$ grayscale image which, when flattened, is a vector in $\mathbb{R}^{784}$.

The first model is a VAE consisting of a convolutional/encoding NN and a deconvolutional/decoding NN. The R script `variational_autoencoder_deconv.R` for this VAE model
A Misconception regarding popular generative neural networks

is available as an example in the R package keras and was used without modification. The second model is a GAN consisting of two different convolutional NNs acting, respectively, as a generator and a discriminator. The R script acgan.R for this GAN model is also available as an example in keras and was used with minor modifications – specifically, we converted the auxiliary classifier GAN into a standard GAN by withholding the additional label information, that is, the type of digit (0–9) associated with each image of a handwritten digit.

For a quick visual assessment of whether samples generated by a GNN (here: VAE or GAN) match the target distribution represented by the training dataset, we use 2-dimensional (2D) principal component projections. We project the 60 000 training samples and samples generated by each GNN onto the two leading principal axes of the training data; see Figure 16. We can clearly see that the GNN-generated samples (middle and right plot, respectively) do not have the same distribution as the projected training data (left plot). The VAE sample appears to contain residual artifacts of the NN. Neither the VAE sample nor the GAN sample appears to disperse as widely in this 2-dimensional subspace as does the original training sample. This suggests that these GNNs are undersampling certain regions of the target density and oversampling others, meaning that certain types of digits are under-represented while other types are over-represented. The GAN sample appears to suffer more severely from this particular problem than the VAE.

![PCA 2D projection for MNIST training digits](image1)
![PCA 2D projection for MNIST VAE generated digits](image2)
![PCA 2D projection for MNIST GAN generated digits](image3)

**Figure 16** Projections on the first two principal axes of the 60 000 MNIST training digits (left), VAE generated digits (middle) and GAN generated digits (right).

The shortcoming of GNNs (here: VAEs or GANs) for the task of density estimation has been studied in the deep learning literature particularly for GAN models. One pitfall of GANs for learning multi-modal distributions is the so-called *mode collapse* problem, according to which GANs collapse on a few or even a single mode; see, for example, Arjovsky et al. (2017) and Tolstikhin et al. (2017). This results in a lack of diversity in the observations generated by the GAN; for the MNIST dataset, generated handwritten digits would be more similar to each other than they should be, even in large amounts of generated digits such as 60 000. This pitfall corroborates our previous observations from
Figure 16. Arjovsky et al. (2017) and Tolstikhin et al. (2017) have proposed some techniques to alleviate this mode collapse problem – although capturing all modes correctly still does not imply that the entire distribution has been learned correctly; see also the recent Arora et al. (2018) on whether GANs actually learn the underlying distribution.

References

Arjovsky, M., Chintala, S., and Bottou, L. (2017), Wasserstein GAN, https://arxiv.org/abs/1701.07875 (2018-11-01).

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. (2016), Understanding deep neural networks with rectified linear units, https://arxiv.org/abs/1611.01491 (2018-11-01).

Arora, S., Risteski, A., and Zhang, Y. (2018), Do GANs learn the distribution? Some Theory and Empirics, International Conference on Learning Representations, ICLR.

Cambou, M., Hofert, M., and Lemieux, C. (2017), Quasi-random numbers for copula models, Statistics and Computing, 27(5), 1307–1329.

Constantine, G. and Savits, T. (1996), A multivariate Faa di Bruno formula with applications, Transactions of the American Mathematical Society, 348(2), 503–520.

Cranley, R. and Patterson, T. N. L. (1976), Randomization of Number Theoretic Methods for Multiple Integration, SIAM Journal on Numerical Analysis, 13(6), 904–914.

Cybenko, G. (1989), Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems, 2(4), 303–314.

Embrechts, P., Lindskog, F., and McNeil, A. J. (2003), Modelling Dependence with Copulas and Applications to Risk Management, Handbook of Heavy Tailed Distributions in Finance, ed. by S. Rachev, Elsevier, 329–384.

Genest, C., Rémillard, B., and Beaudoin, D. (2009), Goodness-of-fit tests for copulas: A review and a power study, Insurance: Mathematics and economics, 44(2), 199–213.

Glorot, X. and Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks, Proceedings of the thirteenth international conference on artificial intelligence and statistics, 249–256.

Goodfellow, I. et al. (2014), Generative adversarial nets, Advances in neural information processing systems, 2672–2680.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016), Deep learning, vol. 1, MIT press Cambridge.

Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., and Smola, A. J. (2007), A kernel method for the two-sample-problem, Advances in neural information processing systems, 513–520.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012a), A kernel two-sample test, Journal of Machine Learning Research, 13(Mar), 723–773.

Gretton, A. et al. (2012b), Optimal kernel choice for large-scale two-sample tests, Advances in neural information processing systems, 1205–1213.
Hlawka, E. and Mück, R. (1972), Über eine Transformation von gleichverteilten Folgen II, *Computing*, 9(2), 127–138.

Hofert, M. (2010), Sampling Nested Archimedean Copulas with Applications to CDO Pricing, PhD thesis, Südwestdeutscher Verlag für Hochschulschriften AG & Co. KG, ISBN 978-3-8381-1656-3.

Hofert, M. (2012), A stochastic representation and sampling algorithm for nested Archimedean copulas, *Journal of Statistical Computation and Simulation*, 82(9), 1239–1255, doi:10.1080/00949655.2011.574632.

Hornik, K. (1991), Approximation capabilities of multilayer feedforward networks, *Neural networks*, 4(2), 251–257.

Joe, H. (2014), Dependence modeling with copulas, Chapman and Hall/CRC.

Kingma, D. P. and Ba, J. (2014), Adam: A method for stochastic optimization, *https://arxiv.org/abs/1412.6980* (2018-11-01).

Kingma, D. P. and Welling, M. (2014), Stochastic gradient VB and the variational auto-encoder, *Second International Conference on Learning Representations, ICLR*.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017), Self-normalizing neural networks, *Advances in Neural Information Processing Systems*, 971–980.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998), Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, 86(11), 2278–2324.

Lemieux, C. (2009), Monte Carlo and Quasi–Monte Carlo Sampling, Springer.

Li, Y., Swersky, K., and Zemel, R. (2015), Generative moment matching networks, *International Conference on Machine Learning*, 1718–1727.

McNeil, A. J. (2008), Sampling nested Archimedean copulas, *Journal of Statistical Computation and Simulation*, 78(6), 567–581.

McNeil, A. J., Frey, R., and Embrechts, P. (2015), Quantitative Risk Management: Concepts, Techniques, Tools, 2nd ed., Princeton University Press.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014), On the number of linear regions of deep neural networks, *Advances in neural information processing systems*, 2924–2932.

Nelsen, R. B. (2006), An Introduction to Copulas, Springer-Verlag.

Niederreiter, H. (1992), Random number generation and quasi-Monte Carlo methods, vol. 63, Siam.

Nielson, M. A. (2015), Neural Networks and Deep Learning, Determination Press, *http://neuralnetworksanddeeplearning.com* (2018-09-18).

Pascanu, R., Mikolov, T., and Bengio, Y. (2013), On the difficulty of training recurrent neural networks, *International Conference on Machine Learning*, 1310–1318.

Rosenblatt, M. (1952), Remarks on a multivariate transformation, *The annals of mathematical statistics*, 23(3), 470–472.

Sobol’, I. M. (1967), On the distribution of points in a cube and the approximate evaluation of integrals, *Zhurnal Vychislitel’noi Matematiki i Matematicheskoi Fiziki*, 7(4), 784–802.
Tolstikhin, I. O., Gelly, S., Bousquet, O., Simon-Gabriel, C.-J., and Schölkopf, B. (2017), Adagan: Boosting generative models, *Advances in Neural Information Processing Systems*, 5424–5433.