Implicit Generative Copulas

Tim Janke, Mohamed Ghanmi, Florian Steinke
Energy Information Networks and Systems
TU Darmstadt, Germany
{tim.janke},{florian.steinke}@tu-darmstadt.de

Abstract

Copulas are a powerful tool for modeling multivariate distributions as they allow to separately estimate the univariate marginal distributions and the joint dependency structure. However, known parametric copulas offer limited flexibility especially in high dimensions, while commonly used non-parametric methods suffer from the curse of dimensionality. A popular remedy is to construct a tree-based hierarchy of conditional bivariate copulas. In this paper, we propose a flexible, yet conceptually simple alternative based on implicit generative neural networks. The key challenge is to ensure marginal uniformity of the estimated copula distribution. We achieve this by learning a multivariate latent distribution with unspecified marginals but the desired dependency structure. By applying the probability integral transform, we can then obtain samples from the high-dimensional copula distribution without relying on parametric assumptions or the need to find a suitable tree structure. Experiments on synthetic and real data from finance, physics, and image generation demonstrate the performance of this approach.

1 Introduction

Many approaches are available to reliably model univariate distributions of real-valued variables. One can use various parametric distributions or employ flexible, non-parametric methods, e.g., by using empirical distribution functions, kernel density estimation (KDE), or quantile regression [22]. Modeling distributions in higher dimensions is a much harder task. Parametric approaches using, e.g., Gaussian distributions, are common but inflexible. Many non-parametric approaches become infeasible in higher dimensions due to the curse of dimensionality [18].

Recent successes in modeling multivariate distributions have been achieved with deep neural network-based likelihood-free implicit generative models [28], such as Generative Adversarial Networks [13] (GANs) and Generative Moment Matching Networks [24; 10].

Copulas are a tool to decouple the modeling of the univariate marginal distributions from modeling the high-dimensional joint dependency structure [19]. This allows to obtain high-quality marginal models with the mentioned univariate techniques. Moreover, the (conditional) marginals can be tailored to the problem at hand and many fields have developed sophisticated domain-specific univariate models for this task, e.g., for probabilistic forecasts in finance [4], weather [36], or energy [17]. These models can then be combined with a suitable copula structure to enable the simulation of multivariate quantities, e.g., [34; 29; 41]. In the machine learning literature copulas have been used to increase the flexibility of Bayesian networks [11], to model multi-agent coordination in reinforcement learning [43], or for image generation via the Vine Copula Autoencoder [40]. Another popular application of copulas is synthetic tabular data generation [33; 27]. Note that in many of these applications, the main goal is to sample from a multivariate distribution and the copula is used as a building block for a generative model.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), virtual.

High-dimensional copula distributions are commonly modeled via parametric structures such as the Gaussian or the student-t copula [8]. [25] propose to enhance the subclass of Archimedean copulas by learning the generator functions via deep neural networks. However, the family of Archimedean copulas is of limited use in higher dimension as it makes restrictive symmetry assumptions. A more flexible, semi-parametric state-of-the art approach is to use vine copulas which are built from trees of bivariate pair-copulas [1]. These pair-copulas can again be parametric, such as the Gumbel or Clayton copula, or non-parametric, e.g., by using bivariate kernel density estimation [32].

Directly applying implicit generative modeling to the task of estimating copula distributions is not straight-forward since copula distributions are required to have uniform marginals. Such an approach has been proposed by [23; 16], but without ensuring the uniformity property, at least not for finite sample sizes when the model is not expected to exactly fit the true copula distribution. Ensuring the marginal uniformity of the learned copula distribution is important since deviations from uniformity will result in unwanted alterations of the marginal distributions in the data space.

In this paper we show how to design and train implicit generative copula (IGC) models to match the dependency structure of given data. A key challenge is to ensure the marginal uniformity of the estimated distribution. We achieve this through first learning a latent distribution with unspecified marginals but the same dependency structure as the training data. We then obtain the desired copula model by applying the probability integral transform component-wise to the latent distribution. The IGC model and the data distribution are matched in copula space using the energy distance [39]. During training, the probability integral transform is approximated through a differentiable softrank layer which allows to use gradient-based methods.

Our contributions

- We propose the first universal, non-parametric model for estimating high-dimensional copula distributions with guaranteed uniformity of the marginal distributions.
- We show how flexible likelihood-free implicit generative models based on deep neural networks can be trained for this task through the use of a differentiable softrank layer.
- Compared with the state-of-the-art semi-parametric vine copula approach, we demonstrate similar or improved performance for different tasks.

We present the proposed copula model in Section 2. Model training is described in Section 3. Section 4 shows experimental results for synthetic and real data from finance, physics, and image generation. We conclude in Section 5.

2 Proposed IGC Model

We introduce our IGC model following the schematic shown in Figure 1. Figure 2 provides an exemplary application for a bivariate two-component Gaussian mixture data distribution.

Copula Basics The vector-valued continuous random variable $\mathbf{X} \in \mathbb{R}^D$ represents the data source of our task. We denote its distribution by P and the cumulative distribution function (cdf) of $\mathbf{X}$ by $F_{\mathbf{X}}(\mathbf{x})$. Moreover, let $F_{X_d}(x_d)$ be the cdf of the marginal distribution of X_d, the d-th component of $\mathbf{X}$, for $d = 1, \ldots, D$.

Each sample vector $\mathbf{x} \in \mathbb{R}^D$ can be mapped to a vector $\mathbf{u} \in [0,1]^D$ by defining as u_d the value of x_d with respect to the corresponding marginal cdf of X_d, i.e., $u_d = F_{X_d}(x_d)$ for all $d = 1, \ldots, D$. This operation is called the probability integral transform (PIT). In the copula literature, values obtained by the PIT are called pseudo-observations [19]. The random variable $\mathbf{U}$ follows a so-called *copula distribution* P^C by construction, i.e., the distribution is defined on the *unit space* $[0,1]^D$ and its marginals are uniform. The joint cdf of $\mathbf{U}$ is typically called the *copula*, which we denote by $F_{\mathbf{U}}(\mathbf{u})$ here. Sklar's theorem states that such such a copula function exists for all random variables $\mathbf{X}$ and it holds that $F_{\mathbf{X}}(\mathbf{X}) = F_{\mathbf{U}}(F_{X_1}(X_1), \ldots, F_{X_d}(X_D))$, i.e., any multivariate distribution can be expressed in terms of its marginals and its copula [38].

IGC Model In this work, we aim at defining a flexible, non-parametric model for copula distributions $Q^C_{\boldsymbol{\theta}}$ in high dimensions. Such a model can either be used to (approximately) represent the

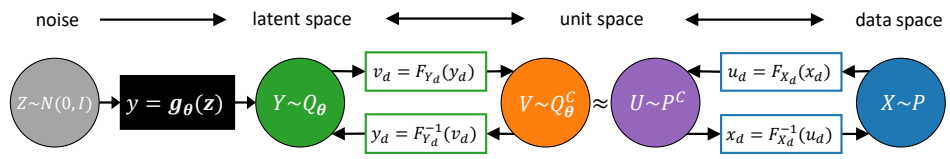
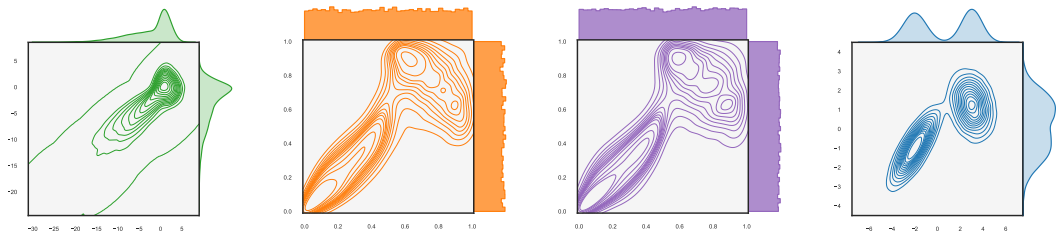


Figure 1: Illustration of the implicit generative copula (IGC) concept: We use a generative neural network $\mathbf{g}_{\boldsymbol{\theta}}$ with learnable parameters $\boldsymbol{\theta}$ to map samples from a multivariate standard Normal distribution to samples of the latent distribution $Q_{\boldsymbol{\theta}}$. The marginal cdfs F_{Y_d} of this distribution are then used to generate samples of the copula distribution $Q_{\boldsymbol{\theta}}^C$. The model is trained to match the copula distribution of the real data P^C using the energy distance.



(a) $Q_{\boldsymbol{\theta}}$ (b) $Q_{\boldsymbol{\theta}}^C$ (c) P^C (d) P

Figure 2: Contour plots and marginal densities of all data and model distributions mentioned in Figure 1 for the example of a two-dimensional mixture of Gaussians data distribution P. Contour plots are derived via KDE from a set of 10000 samples.

copula distribution P^C of the unit space vectors $\mathbf{U}$ or the distribution P of the original data vectors $\mathbf{X}$, in which case the $\mathbf{U}$ vectors have to be transformed component-wise with the inverse cdfs of the components X_d, i.e., $x_d = F_{X_d}^{-1}(u_d)$.

A flexible model class for learning high-dimensional probability distributions is the class of implicit generative models [28]. Here, latent random variables with a simple and known distribution are transformed via a parameterizable mapping, typically a deep neural network, to a complex distribution that approximates the distribution of the training data. A straight-forward application of this framework to model copula distributions, as done in [16] and [23], is difficult. The output of the generative model can be guaranteed to lie in $[0, 1]^D$ by applying an appropriate output layer, e.g., using a sigmoid function. However, guaranteeing the desired uniformity of the marginal distributions is not straight-forward. We have first experimented with additional training losses that penalize deviations from marginal uniformity. However, a simpler approach without any hyper-parameters for weighting additional cost terms and with a guarantee of marginal uniformity is the following two-step procedure.

In the first step, we model the distribution of the vector-valued latent random variable $\mathbf{Y} \in \mathbb{R}^D$. To this end, we start with random variables $\mathbf{Z} \in \mathbb{R}^K$ from a simple base distribution that we choose as zero mean, unit variance Gaussian here. The tuneable map $\mathbf{g}_{\boldsymbol{\theta}} : \mathbb{R}^K \to \mathbb{R}^D$ with parameters $\boldsymbol{\theta}$ is then used to transform the noise samples $\mathbf{z}$ to the latent samples $\mathbf{y}$ as

$$\mathbf{y} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}). \tag{1}$$

We denote the resulting probability distribution of $\mathbf{Y}$ by $Q_{\boldsymbol{\theta}}$ and the marginal cdfs by $F_{Y_d}(y_d)$.

In a second step, we transform the latent variables $\mathbf{Y}$ into unit space vectors $\mathbf{V}$. To this end, we define component-wise for $d = 1, \ldots, D$

$$v_d = F_{Y_d}(y_d). \tag{2}$$

This transformation step is analogous to going form $\mathbf{X}$ to $\mathbf{U}$. It guarantees both that the sample values v_d from the model lie in the unit interval $[0, 1]$ and that they are uniformly distributed, regardless of the distribution $Q_{\boldsymbol{\theta}}$. We refer to the distribution of $\mathbf{V}$ as the model copula distribution and denote it by $Q_{\boldsymbol{\theta}}^C$.

Interpretation Note that the distributions $Q_{\boldsymbol{\theta}}$ and P might be very different. More precisely, the mapping (1) can result in arbitrary marginal distributions but an optimal model of the distribution P would have

$$P^C = Q^C_{\boldsymbol{\theta}}. \tag{3}$$

Given a sufficiently rich function class for $\mathbf{g}_{\boldsymbol{\theta}}$ any distribution of $\mathbf{Y}$ can be modeled with small error [26]. This statement naturally extends to universal approximation properties for the dependency structure of $\mathbf{Y}$, i.e., the unit space random vectors $\mathbf{V}$. The approximation of the data copula distribution P^C with our model copula distribution $Q^C_{\boldsymbol{\theta}}$ will thus be arbitrarily close if the generator function $\mathbf{g}_\theta$ is flexible enough and if the optimal $\boldsymbol{\theta}$ is selected.

3 Training Procedure

Problem statement We aim at empirically estimating an IGC model $Q^C_{\boldsymbol{\theta}}$ from data, i.e., we want to determine the optimal parameter vector $\boldsymbol{\theta}$ given a set of N training samples $\mathbf{x}^1, \dots, \mathbf{x}^N$ from $\mathbf{X}$. The underlying target is to optimally match the copula model $Q^C_{\boldsymbol{\theta}}$ to the (typically unknown) copula distribution P^C from which the samples were generated.

Parameter estimation We first transform the given data samples into unit space vectors in $[0,1]^D$, before starting the actual training procedure. Since we typically do not know the exact marginal cdfs F_{X_d} of the components X_d, we estimate them from the given data. To this end, we use the empirical cdfs separately for each component and define for $n = 1, \dots, N$ and $d = 1, \dots, D$,

$$u^n_d = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[x^i_d \leq x^n_d]. \tag{4}$$

Here, $\mathbf{1}[A]$ is the indicator function of event A, i.e., it is 1 iff A is true. Note that this is an unbiased and consistent estimator for the true cdf [42].

We then aim at matching the copula distribution $Q^C_{\boldsymbol{\theta}}$ to the unit space vector samples $\mathbf{u}^1, \dots, \mathbf{u}^N$. We base the inference of our likelihood-free model on the energy distance ED^2 [39] between distributions P^C and $Q^C_{\boldsymbol{\theta}}$,

$$ED^2(P^C, Q^C_{\boldsymbol{\theta}}) = 2\mathbb{E}\|\mathbf{U} - \mathbf{V}\| - \mathbb{E}\|\mathbf{U} - \mathbf{U}'\| - \mathbb{E}\|\mathbf{V} - \mathbf{V}'\|. \tag{5}$$

Here, $\mathbf{U}, \mathbf{U}'$ and $\mathbf{V}, \mathbf{V}'$ are independent copies of a random vector with distribution P^C and $Q^C_{\boldsymbol{\theta}}$, respectively. It holds that $ED^2(P^C, Q^C_{\boldsymbol{\theta}}) = 0$ if and only if $P^C = Q^C_{\boldsymbol{\theta}}$ [39].

In practice, we use a sample-based approximation of the energy distance. We draw M samples $\mathbf{v}^1(\boldsymbol{\theta}), \dots, \mathbf{v}^M(\boldsymbol{\theta})$ from $Q^C_{\boldsymbol{\theta}}$ and minimize the loss function

$$\mathcal{L}_{ED^2}(\boldsymbol{\theta}) = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \|\mathbf{u}^n - \mathbf{v}^m(\boldsymbol{\theta})\|_2 - \frac{1}{2M(M-1)} \sum_{m=1}^{M} \sum_{m'=1}^{M} \|\mathbf{v}^m(\boldsymbol{\theta}) - \mathbf{v}^{m'}(\boldsymbol{\theta})\|_2. \tag{6}$$

Since P^C is independent of $\boldsymbol{\theta}$, the second term of (5) does not need to be included. Note that the energy distance is an instance of the more general maximum mean discrepancy (MMD) measure [15; 37]. We also experimented with MMD losses based on the Gaussian kernel but found the energy distance to be more robust as it does not require to tune the kernel bandwidth.

To generate samples from our copula model $Q^C_{\boldsymbol{\theta}}$, we first draw samples $\mathbf{y}^1, \dots, \mathbf{y}^M$ from the implicit generative model $Q_{\boldsymbol{\theta}}$. To obtain vectors in unit space, we then again require the cdfs $F_{Y_d}(y_d)$ of the marginal distributions of the components Y_d of $\mathbf{Y}$, $d = 1, \dots, D$. However, these functions are not known during model training and will likely change after each gradient step. Hence, we again use the empirical distribution functions to define for $m = 1, \dots, M$ and $d = 1, \dots, D$,

$$v^m_d(\boldsymbol{\theta}) = \frac{1}{M} \sum_{j=1}^{M} \mathbf{1}[y^j_d(\boldsymbol{\theta}) \leq y^m_d(\boldsymbol{\theta})]. \tag{7}$$

This provides us with an unbiased estimate of the current cdf during training. However, the indicator operation in (7) is not differentiable and hence will not allow the gradients to flow through this

operation. Therefore, during training, we replace (7) with a *softrank* layer based on a scaled sigmoid function

$$v_d^m(\boldsymbol{\theta}) = \frac{1}{M}\left(0.5 + \sum_{j=1}^{M} \frac{1}{1+\exp(\alpha(y_d^m(\boldsymbol{\theta}) - y_d^j(\boldsymbol{\theta})))}\right), \tag{8}$$

where α is a scaling constant [35]. For sufficiently large α, (8) provides a close approximation to the empirical marginal cdfs at the current training step. A larger M will result in a finer approximation of the true marginal cdfs but comes with the increased cost for computing the ranks which requires $\mathcal{O}(M^2)$ operations for all samples.

Sampling from the trained model Once we have completed training and thus have fixed $\boldsymbol{\theta}$, we have also fixed the component-wise cdfs $F_{Y_d}(y_d)$. Now, the operation to estimate $F_{Y_d}(y_d)$ does not need to be differentiable anymore and we can chose any available method to estimate the univariate marginal distributions based on samples from $Q_{\boldsymbol{\theta}}$. We choose to simply draw a very large set of samples from $Q_{\boldsymbol{\theta}}$ and then use the empirical marginal cdfs, i.e., for each sampled value y_d^t we store the corresponding value u_d^t according to (4), $t = 1, \ldots, T$. Given a new realization of Y_d, we obtain its approximate cdf value $\hat{F}_{Y_d}(y_d)$ by interpolation of the stored values. Other more compact approximations like univariate kernel density estimation would also be feasible.

In sum, we obtain a sample $\mathbf{v}^i$ from $Q_{\boldsymbol{\theta}}^C$ at test time by first sampling a realization $\mathbf{z}^i$ from $N(\mathbf{0}, \mathbf{I})$, transform it via $\mathbf{y}^i = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}^i)$, and then apply the component-wise PIT approximation to obtain $\mathbf{v}^i = [\hat{F}_{Y_1}(y_1^i), \ldots, \hat{F}_{Y_d}(y_D^i)]$. Given estimates for the component-wise marginal cdfs of the data $\hat{F}_{X_1}, \ldots, \hat{F}_{X_d}$ we can derive a sample $\mathbf{s}^i$ in data space via $\mathbf{s}^i = [\hat{F}_{X_1}^{-1}(v_1^i), \ldots, \hat{F}_{X_d}^{-1}(v_D^i)]$.

4 Experiments

In the following we empirically demonstrate the capabilities of IGC models on a series of experiments on synthetic and real data with increasing complexity. Additional experiments can be found Appendix B.

Implementation For all experiments except the image generation task, we use a fully connected neural network with two layers, 100 units per layer, ReLU activation functions, and train for 500 epochs. For the image generation experiment we use a three layer, fully connected neural network with 200 neurons in each layer, and train for 100 epochs. In all cases we train with a batch size of $N_{batch} = 100$ and generate $M = 200$ samples from the model per batch. The number of noise distributions is set as $K = 3D$ and $T = 10^6$. We use the Adam optimizer [20] with default parameters.

All experiments besides the training of the autoencoder models were carried out on a desktop PC with a Intel Core i7-7700 3.60Ghz CPU and 8GB RAM. For the training of the autoencoders we used Google Colab [14]. Training times for all experiments are in the range of a few minutes except for the image generation task. Details are provided in Appendix B. We use tensorflow with the Keras API for the neural networks [2; 7]. For copula modeling, we use pyvinecopulib [31] in Python and the R package kdecopula [30]. Our code is available from `https://github.com/TimCJanke/igc`.

Evaluation We evaluate the models by comparing the distance between the learned and the true copula. More specifically, we use the integrated squared error (ISE) in unit space, i.e.,

$$ISE = \int_{[0,1]^D} (F_{\mathbf{U}}(\mathbf{w}) - F_{\mathbf{V}}(\mathbf{w}))^2 d\mathbf{w} \approx \frac{1}{N}\sum_{n=1}^{N} (F_{\mathbf{U}}(\mathbf{u}^n) - F_{\mathbf{V}}(\mathbf{u}^n))^2, \tag{9}$$

where $F_{\mathbf{U}}$ is the joint cdf of $\mathbf{U}$, the true copula, and $F_{\mathbf{V}}$ the joint cdf of the learned model. Since analytical integration is not possible, we approximate the integral with an empirical sum and use the data vectors $\mathbf{u}^1, ..., \mathbf{u}^N$ for this purpose.

The copula function $F_{\mathbf{U}}$ is not known for real data. Instead, we use the empirical cdf of the unit space data vectors in (9), i.e., for $\mathbf{w} \in [0,1]^D$ we set

$$\hat{F}_{\mathbf{U}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\prod_{d=1}^{D} \mathbf{1}[u_d^n \leq w_d]. \tag{10}$$

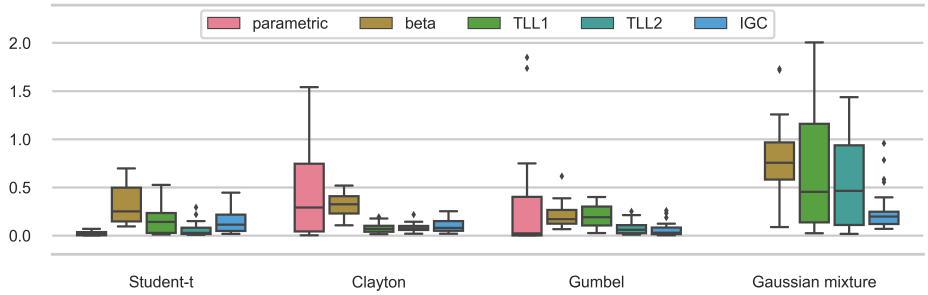


Figure 3: Box plot for the integrated squared error (lower is better) for 25 runs of fitting bivariate copulas to training data from a Student-t, Gumbel, Clayton, and Gaussian mixture copula.

Due to the curse of dimensionality this approximation of the joint cdf will have a coarse structure in high dimensions. This is why we only use the values at the data vectors $\mathbf{u}^1, ..., \mathbf{u}^N$ in (9).

For parametric copula models the distribution function $F_{\mathbf{V}}$ is analytic. For the non-parametric models, including IGC, we again resort to approximation. We draw 10^5 samples from the copula distribution model and use their empirical cdf in unit space in (9). We found that this procedure provided reliable and stable estimates for the underlying copula in dimensions $D \leq 5$.

4.1 Synthetic data

Learning bivariate parametric copulas We first show that IGC models are able to emulate bivariate parametric copulas. We compare our IGC approach to two non-parametric copula density estimation techniques, kernel density estimation with beta kernels (BETA) [6] and the transformation local likelihood estimator (TLL) [12]. These are state-of-the art non-parametric estimators for pair-copulas [30]. The compact support of the beta distribution on $[0, 1]$ is convenient for copula density estimation. The idea of the TLL approach is to first transform the data using the inverse normal CDF such that the data is supported on $\mathbb{R}^2$. Then the density is estimated via local regression using linear (TLL1) or cubic polynomials (TLL2) on a fixed grid of points. Finally the estimated density is transformed back to unit space. Additionally we report the performance of a parametric model. In this approach a copula family is selected from a set of copulas based on the Bayesian information criterion (BIC). This set comprises the Independence, Gaussian, Student-t, Clayton, Gumbel, Frank, Joe, BB1, and BB7 copula. Parameters are then estimated via maximum likelihood.

We run experiments for data sets generated from a Student-t, a Gumbel, and a Clayton copula as well as the copula resulting from a two-component Gaussian mixture distribution as shown in Figure 2. For each one of these, we sampled a random parameter set and then generated 1000 training samples from the resulting model. We repeated each experiment 25 times. Appendix B contains the details on the considered parameter ranges and the exact sampling procedure.

Figure 2 shows the data and the model with intermediate steps for one instance of the Gaussian mixture case. The fitted copula matches the data distribution in unit space very well. Figure 3 presents aggregated ISE results for the different setups. The IGC model shows comparable performance scores to TLL1 and TLL2 for the Student-t, Clayton, and Gumbel copulas, and better performance than the BETA approach. For the more complex Gaussian mixture test case, IGC shows superior performance and a much lower variance than all baseline methods. Notably, the parametric approach with BIC-based model selection results in large variance in accuracy for the Clayton and Gumbel data, most likely because the wrong copula family is selected in some cases.

Learning multivariate vine copulas We now turn to the problem of estimating copulas in dimensions $D > 2$. To this end, we conduct a similar experiment as before using 5-dimensional vine copulas as data generating distribution. For each of 25 repetitions, we first sample a random tree structure using pyvinecopulib's `RVineStructure.Simulate()` method. Then, we randomly assign parametric pair-copulas with random parameters to each edge in the tree. The considered families of bivariate copulas are Independence, Gaussian, Student-t, Clayton, Gumbel, Frank, Joe, BB1, and BB7. Further details are available in Appendix B. We generate 5000 samples from the resulting vine copula

model and use these to train our IGC model as well as the benchmark models. As benchmarks, we use a vine copula model with $TLL2$ pair-copulas only and a vine copula model which can select all parametric copula families named above as well as the $TLL2$. Additionally, we estimate a parametric Gaussian copula. The parameters are estimated via maximum likelihood and the copula families are selected using the BIC. The selection of the vine structure is based on the Dissmann algorithm [9]. We evaluate all models with the ISE at 10000 data points sampled from the true model.

The results of the simulation study are presented in Figure 4. The IGC model has the lowest mean ISE (0.0697) followed by the TLL2-Vine model (0.2093), while the latter has a lower median ISE (0.0339 vs. 0.045). This is the result of some large outliers of the ISE for the vine models that do not occur for the IGC model. These are most likely caused by a poorly selected vine structure. Interestingly, the results specifically imply that the data is on average better modeled by the IGC model than by the vine approach although this model class entails the true data generating model. However, recovering the true model does not seem to be an easy task.

4.2 Real data

Exchange rates Copulas are a popular tool in financial risk management as they can be used to estimate the distribution of the multivariate returns over different assets [34] under the assumption of a stationary copula. We consider a data set of size $N = 5844$ that contains 15 years of daily exchange rates between the US-Dollar and the Canadian Dollar, the Euro, the British Pound, the Swiss Franc, and the Japanese Yen. The data was obtained from the *R* package *qrm_data*. We preprocess the data to filter out the effects of temporal dependencies. To this end, we fit an AR(1)-GARCH(1,1) [4] process with Student-t innovations to the time series of the daily returns. We then obtain the standardized residuals from the AR-GARCH models and transform these observations to the unit space using the empirical cdfs. Figure 5a shows a kernel density estimate for two selected dimensions of the resulting data set in unit space, namely the US-Dollar/Euro and US-Dollar/Pound exchange rates. While being non-trivial and multi-modal, the distribution is strongly concentrated along the diagonal. We then estimate a Gaussian copula, a vine copula, a vine copula with only TLL2 pair-copulas, a GMMN model with a sigmoid output layer as proposed by [16], and our IGC model for this data. To ensure a test data set of appropriate size, we use a 5-fold cross-validation scheme where 20% of the data is used for training and 80% for testing. For the IGC and the GMMN model we additionally use five different random initializations per fold for the neural network weights, i.e. we report means and standard deviations from 25 values for the IGC and the GMMN model and 5 values for the other methods. The results are presented in Table 1. The IGC model clearly outperforms the Gaussian baseline and also the GMMN model. However, the vine copula models show lower average ISEs. This might be due to the symmetric nature of the data which can be approximated well by Student-t and TLL pair-copulas.

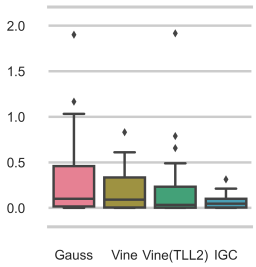
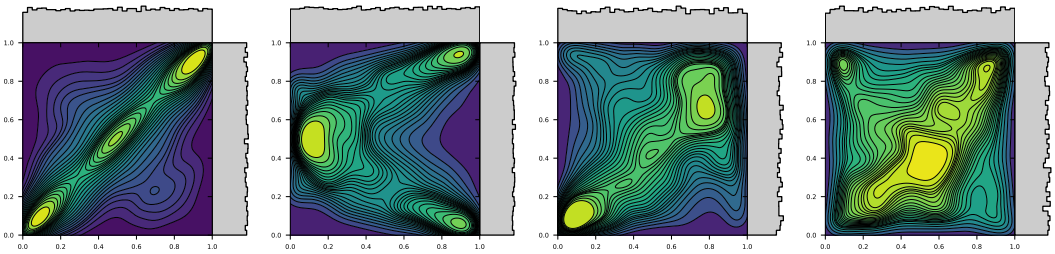


Figure 4: Box plot of the ISE results for 25 runs of fitting data from a 5D vine copula

MAGIC Gamma Telescopes data In order to test our approach on a more complex dependency structure, we consider the MAGIC (Major Atmospheric Gamma-ray Imaging Cherenkov) Telescopes data set available from the UCI repository (`https://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope`). This data was also used for benchmarking non-parametric copula estimation techniques in [32]. We only consider the observations classified as *gamma* and the 5 variables *fLength, Width, fConc, fM3Long, fM3Trans*. The size of the data set is $N = 12332$. We use the empirical cdfs to transform the observations to the unit space. Figure 5b showcases the dependency structure of the data for two selected data dimensions. The observed structures are highly asymmetric. We again fit the data with

Table 1: ISE mean and standard deviation for exchange rate and magic data from 5 fold CV and 5 random NN weight initializations per fold (lower is better).

	exchange rates	magic
Gauss	0.9196 ± 0.1024	0.2415 ± 0.0167
Vine	0.2791 ± 0.0290	0.0588 ± 0.0055
Vine-TLL2	$\mathbf{0.2457} \pm 0.0383$	0.0588 ± 0.0055
GMMN [16]	0.5791 ± 0.2638	0.08065 ± 0.0286
IGC (ours)	0.3829 ± 0.1385	$\mathbf{0.0345} \pm 0.0105$



(a) (b) (c) (d)

Figure 5: Exemplary bivariate KDE estimates of the copula densities from the real data sets: (a) Exchange rates EUR/USD-GBP/USD, (b) MAGIC fLength-fM3Trans, (c) and (d): FashionMNIST Autoencoder latent space dimensions 1-2 and 2-3

a Gaussian copula, a vine copula with all available pair-copulas as above, a vine copula with TLL2 pair-copulas, and a GMMN model with a sigmoid output layer as benchmarks and use the same 5-fold CV strategy as before, i.e., in each fold we use 20% as training data, 80% as test data, and report the average ISE. The results are presented in Table 1. The IGC model achieves the lowest average ISE. Again the GMMN model shows a substantially larger mean ISE with a much larger standard deviation. As only TLL2 pair-copulas were selected by the BIC criterion, the scores for both vine models are the same. The Gaussian copula is clearly not suited for such complex data as the average ISE is 5 times larger than for the other approaches.

Copula Autoencoders [40] introduced the Vine Copula Autoencoder, a generative model which uses vine copulas for ex-post density estimation of the latent space of a trained autoencoder. In a first step, an autoencoder is trained on a data set to learn a low dimensional representation of the data. After training, the encoder part of the network is used to map the training data to the autoencoder's latent space representation. Next, the uni-variate marginal distributions are estimated, e.g., by using the empirical cdfs or kernel density estimation, and the compressed data is mapped to unit space. After fitting a copula model to the observations in unit space, one can sample from the fitted copula model, apply the inverse marginal cdfs, and map the simulated data back to the image space using the decoder network of the autoencoder in order to generate new data samples.

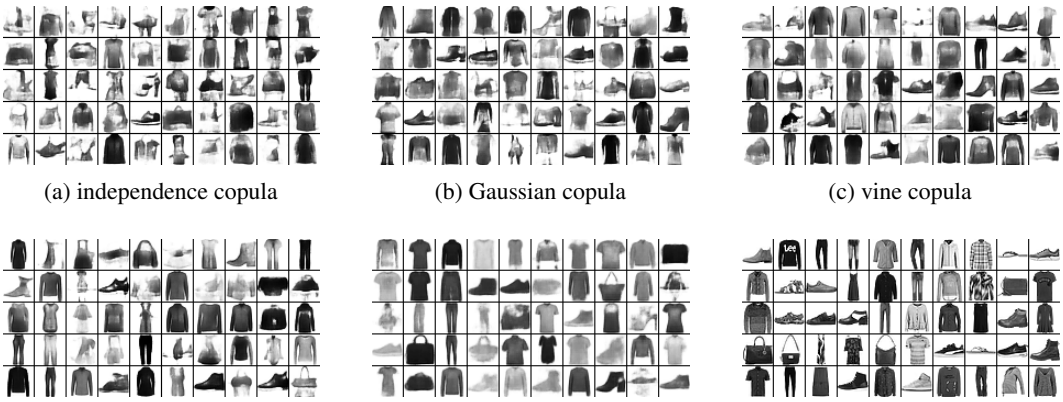
In the following, we present results for the FashionMNIST [44] data set. We train a convolutional autoencoder on the entire training data of 60000 samples with a latent space of dimension 25. Details on the architecture are found in Appendix B. After training the autoencoder, we estimate the empirical cdfs of the compressed data using the training data. See Figures 5c and 5d for exemplary visualizations of the resulting bivariate data densities in unit space. Notably, these distributions are much more complex than the ones from the previous experiments. We fit this data with a Gaussian copula, a vine copula with TLL2 pair-copulas, a GMMN with sigmoid output layer, and an IGC model. We also test an independence copula, i.e., we assume independence over the latent space. Additionally we report results for a standard variational autoencoder (VAE) [21] with the same architecture.

Table 2: MMD scores for the FashionMNIST test set (lower is better)

	image	latent	copula
Indep	0.01912	0.00722	0.00373
Gauss	0.00619	0.00209	0.00087
Vine	0.00674	0.00131	0.00079
GMMN	**0.00392**	0.00341	**0.00073**
IGC	**0.00426**	**0.00114**	**0.00069**
VAE	0.01316		

Exemplary samples from all models and the test set are presented in Figure 6. Sampling with the independence copula, i.e., assuming no dependency structure, leads to images with many artifacts. The Gaussian copula produces better images, but still produces some artifacts. Samples from the vine copula or the IGC model are comparable in quality. They show smaller details and few implausible artifacts. The images generated by the VAE are blurry and show much less variation in pixel intensity than the test data.

In 25 dimensions it is not feasible anymore to compute and compare empirical cdfs as required for ISE scoring. We thus resort to the MMD [15] for the numerical evaluation since it is commonly used



(a) independence copula (b) Gaussian copula (c) vine copula

(d) IGC (ours) (e) VAE (f) test data

Figure 6: Generated image samples from different copula models on top of a pre-trained autoencoder as well as from a standard variational autoencoder. The latent space has 25 dimensions.

as a simple and robust measure for evaluating image generation models [46]. We generate 10000 images from each model and compare those to the 10000 test set images by computing the MMD with a Gaussian kernel. We use the bandwidths 1.5, 350, and 8 for the latent unit space, the latent data space, and the image space, respectively. The bandwidths were selected based on the median heuristic proposed in [15].

The results are found in Table 2. The IGC model performs better than all other models for the latent space distributions. Interestingly the GMMN model shows slightly lower MMD values for samples in the image space while the MMD value in the latent data space is quite large. We provide p-values from the test proposed by [5] in Appendix B.

5 Conclusion

We have proposed the first fully non-parametric model framework for copulas in higher dimensions that guarantees uniformity of the marginal distributions. The proposed IGC approach that is based on an implicit generative step and a ranking transformation is structurally simple and entirely differentiable. Yet, if the generator class is sufficiently complex, any copula dependency structure can be modeled. The model can be well implemented with standard deep learning frameworks. For various test data sets we have shown a modeling performance on par or above other state-of-the-art approaches, especially, the vine copula approach.

IGC models should be further investigated in various ways. First, it would be straight-forward to condition the copula models on external factors, either by including such values into the input of the generator network or by adapting the noise distributions, e.g., scaling their variance. This would allow to describe context-dependent changes of the dependency structure. Second, many more applications for copulas should be examined, e.g., synthetic tabular data generation [45]. Third, the ranking operation could probably be sped up from $\mathcal{O}(M^2)$ to $\mathcal{O}(M \log M)$ using ideas from [3]. Finally, the use of adversarial training schemes could also be investigated.

Acknowledgments and Disclosure of Funding

This work has been performed in the context of the LOEWE center emergenCITY.

References

[1] K. Aas, C. Czado, A. Frigessi, and H. Bakken. Pair-copula constructions of multiple dependence. *Insurance: Mathematics and economics*, 44(2):182–198, 2009.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[3] M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, pages 950–959. PMLR, 2020.

[4] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.

[5] W. Bounliphone, E. Belilovsky, M. Blaschko, I. Antonoglou, and A. Gretton. A test of relative similarity for model selection in generative models. 2016.

[6] A. Charpentier, J.-D. Fermanian, and O. Scaillet. The estimation of copulas: Theory and practice. *Copulas: From theory to application in finance*, pages 35–64, 2007.

[7] F. Chollet et al. Keras. `https://keras.io`, 2015.

[8] S. Demarta and A. J. McNeil. The t copula and related copulas. *International Statistical Review*, 73(1):111–129, 2005.

[9] J. Dissmann, E. C. Brechmann, C. Czado, and D. Kurowicka. Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis*, 59: 52–69, 2013.

[10] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 258–267, 2015.

[11] G. Elidan. Copula bayesian networks. In *Advances in Neural Information Processing Systems*, volume 23, 2010.

[12] G. Geenens, A. Charpentier, D. Paindaveine, et al. Probit transformation for nonparametric kernel estimation of the copula density. *Bernoulli*, 23(3):1848–1873, 2017.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

[14] Google Colaboratory. URL `https://colab.research.google.com/`.

[15] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[16] M. Hofert, A. Prasad, and M. Zhu. Quasi-random sampling for multivariate distributions via generative neural networks. *Journal of Computational and Graphical Statistics*, pages 1–24, feb 2021.

[17] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, R. J. Hyndman, et al. Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond. *International Journal of Forecasting*, 32(3):896–913, 2016.

[18] J.-N. Hwang, S.-R. Lay, and A. Lippman. Nonparametric multivariate density estimation: a comparative study. *IEEE Transactions on Signal Processing*, 42(10):2795–2810, 1994.

[19] H. Joe. *Dependence modeling with copulas*. CRC press, 2014.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[22] R. Koenker and G. Bassett Jr. Regression quantiles. *Econometrica: Journal of the Econometric Society*, pages 33–50, 1978.

[23] N. A. Letizia and A. M. Tonello. Segmented generative networks: Data generation in the uniform probability space. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[24] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727. PMLR, 2015.

[25] C. K. Ling, F. Fang, and J. Z. Kolter. Deep archimedean copulas. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[26] Y. Lu and J. Lu. A universal approximation theorem of deep neural networks for expressing probability distributions. In *Advances in Neural Information Processing Systems*, pages 3094–3105, 2020.

[27] D. Meyer, T. Nagler, and R. J. Hogan. Copula-based synthetic data generation for machine learning emulators in weather and climate: application to a simple radiation model. *Geoscientific Model Development Discussions*, pages 1–21, 2021.

[28] S. Mohamed and B. Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

[29] A. Möller, A. Lenkoski, and T. L. Thorarinsdottir. Multivariate probabilistic forecasting using ensemble bayesian model averaging and copulas. *Quarterly Journal of the Royal Meteorological Society*, 139(673):982–991, 2013.

[30] T. Nagler. kdecopula: An R package for the kernel estimation of bivariate copula densities. *Journal of Statistical Software*, 84(7):1–22, 2018.

[31] T. Nagler and T. Vatter. pyvinecopulib. URL `https://github.com/vinecopulib/pyvinecopulib/`.

[32] T. Nagler, C. Schellhase, and C. Czado. Nonparametric estimation of simplified vine copula models: comparison of methods. *Dependence Modeling*, 5(1):99–120, 2017.

[33] N. Patki, R. Wedge, and K. Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410. IEEE, 2016.

[34] A. J. Patton. A review of copula models for economic time series. *Journal of Multivariate Analysis*, 110:4–18, 2012.

[35] T. Qin, T.-Y. Liu, and H. Li. A general approximation framework for direct optimization of information retrieval measures. *Information retrieval*, 13(4):375–397, 2010.

[36] A. E. Raftery, T. Gneiting, F. Balabdaoui, and M. Polakowski. Using bayesian model averaging to calibrate forecast ensembles. *Monthly weather review*, 133(5):1155–1174, 2005.

[37] D. Sejdinovic, B. Sriperumbudur, A. Gretton, and K. Fukumizu. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The Annals of Statistics*, pages 2263–2291, 2013.

[38] M. Sklar. Fonctions de repartition an dimensions et leurs marges. *Publ. inst. statist. univ. Paris*, 8:229–231, 1959.

[39] G. J. Székely, M. L. Rizzo, et al. Testing for equal distributions in high dimension. *InterStat*, 5 (16.10):1249–1272, 2004.

[40] N. Tagasovska, D. Ackerer, and T. Vatter. Copulas as high-dimensional generative models: Vine copula autoencoders. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[41] J. Tastu, P. Pinson, and H. Madsen. Space-time trajectories of wind power generation: Parametrized precision matrices under a gaussian copula approach. In *Modeling and stochastic learning for forecasting in high dimensions*, pages 267–296. Springer, 2015.

[42] A. W. Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

[43] H. Wang, L. Yu, Z. Cao, and S. Ermon. Multi-agent imitation learning with copulas. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 139–156. Springer, 2021.

[44] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[45] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. Modeling Tabular data using Conditional GAN. In *Advances in Neural Information Processing Systems*, 2019.

[46] Q. Xu, G. Huang, Y. Yuan, C. Guo, Y. Sun, F. Wu, and K. Weinberger. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018.

Appendix

A Algorithm

The algorithm for training an IGC model is described in Algorithm 1.

Algorithm 1: Training algorithm

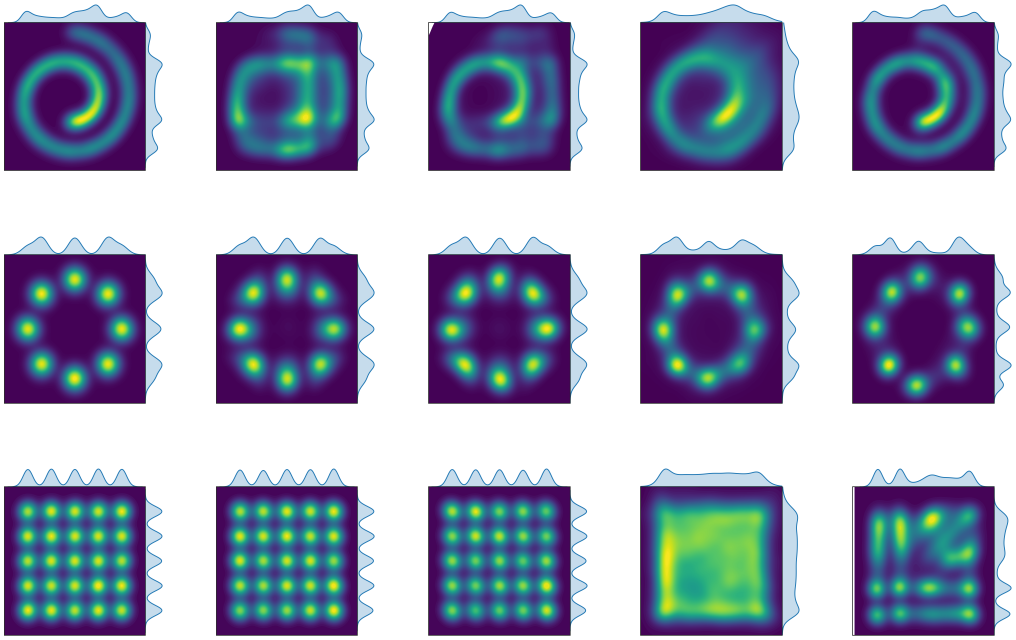
Input : data $\{\mathbf{u}_n\}_{n=1}^N$, initial model parameters $\boldsymbol{\theta}_0$, initial learning rate η, number of batches B, number of random noise samples M, number of epochs N_{epochs}
Output : Model parameters $\boldsymbol{\theta}^*$
Partition data into B mini batches $\{\mathbf{U}_b\}_{b=1}^B$
For each batch b, generate a set of M random noise samples $\mathbf{Z}_b = [\mathbf{z}_b^1, ..., \mathbf{z}_b^M]$
for N_{epochs} **do**
  for $b = 1, ..., B$ **do**
    for $m = 1, ..., M$ **do**
      Get $\mathbf{y}_b^m \leftarrow g(\mathbf{z}_b^m, \boldsymbol{\theta})$
    end
    Map model output to unit space $\mathbf{V}_b \leftarrow Softrank(\mathbf{Y}_b)$
    Compute gradient of loss $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{U}_b, \mathbf{V}_b)$ over batch
    Update learning rate η (e.g. using ADAM)
    Update model parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}$
  end
end

B Experiments

B.1 Additional experiments on toy data sets

We provide additional experiments on three toy data sets commonly used in the literature on deep generative models, the "Swiss Roll", the "Grid of Gaussians", and the "Ring of Gaussians". We test four copula based models: a Gaussian copula, a TLL2 copula, an GMMN with sigmoid output layer [16], and our IGC model. For these models we use a linear interpolation of the ECDF of the training set as models for the marginals of the data distribution. We additionally report results for two implicit generative models that directly model the data distribution, a GMMN [24; 10] and a GAN [13]. Like the IGC model, the GMMN based models are trained by minimizing the energy distance. All neural network models use two layers, 100 neurons per layer, and are trained for 500 epochs. IGC and GMMN models are trained using the Adam optimizer with standard values. For the GAN we use a lower learning rate of $lr = 0.0002$ and a lower momentum $\beta_1 = 0.5$ as the model did not converge with the standard settings. We evaluate the models using the average negative log-likelihood of a test set based on kernel density estimates. We repeat each experiment 10 times with different training and test sets of size 5000 and random initializations for the neural networks.

Table 3 presents the results. The GAN achieves the best result for the Swiss Roll data. However, the scores for the GAN show a much higher standard deviation than all other methods. The IGC model has the second lowest score overall and the lowest score of all copula based methods. For the ring of Gaussians, the TLL2 copula shows the lowest NLL, closely followed by the IGC and GMMN copula. Here the GAN shows the worst performance of all models. The Grid of Gaussians is a trivial test case for the copula based models as it is sufficient to sample from the marginal distributions with an independence copula. Both the GMMN and the GAN show substantially worse NLL values and fail at properly approximating the true data distribution as can be seen from the bottom row of Figure 7.



(a) True density (b) TLL2 copula (c) IGC (d) GMMN (e) GAN

Figure 7: True and exemplary learned densities for the Swiss Roll (top), Ring of Gaussians (middle), and Grid of Gaussians (bottom).

Table 3: Means and standard deviations of the average negative log-likelihood (lower is better) from 10 repetitions with random parameter initializations and training and test sets.

	swiss roll	ring	grid
Gaussian copula	6.10 (0.01)	5.72 (0.02)	**4.47 (0.01)**
TLL2 copula	5.61 (0.01)	**5.11 (0.01)**	**4.47 (0.01)**
GMMN copula	5.45 (0.09)	5.24 (0.01)	**4.48 (0.01)**
IGC	5.26 (0.08)	5.19 (0.03)	**4.47 (0.01)**
GMMN	5.76 (0.09)	5.40 (0.05)	6.27 (0.05)
GAN	**4.82 (0.40)**	7.14 (0.52)	5.95 (0.26)

B.2 Learning bivariate copulas

For the Student-t, Gumbel, and Clayton copulas we sample the parameters and rotations uniformly from the ranges given in Table 4.

Samples form the Gaussian mixture copula are generated by sampling N times from one of the two components with equal probability and then transforming all samples to the unit space via the component-wise PIT. Samples for the component $j \in \{1, 2\}$ are drawn from the two-dimensional Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_j, \alpha_j \boldsymbol{\Sigma}_j)$, where $\boldsymbol{\mu}_j$ is sampled from $\mathcal{U}(-5, 5)^2$, $\sigma_{12,j}$ is sampled from $\mathcal{U}(-0.95, 0.95)$, α_j is sampled from $\mathcal{U}(0.8, 1.2)$, and $\boldsymbol{\Sigma}_j = \left(\begin{smallmatrix} 1 & \sigma_{12,j} \\ \sigma_{12,j} & 1 \end{smallmatrix}\right)$.

B.3 Learning multivariate vine copulas

The following table presents the parameter ranges of the pair-copulas for the vine copula experiments. We used uniform sampling for selecting the pair-copula family as well as the parameters and rotations.

Table 4: Parameter ranges for pair-copulas for learning vine copulas

	θ_1	θ_2	rotation
Independence	-	-	-
Gaussian	$\pm[0.5, 0.95]$	-	$0°$
Student-t	$\pm[0.5, 0.95]$	$[2, 10]$	$0°$
Clayton	$[2, 10]$	-	$\{0°, 90°, 180°, 270°\}$
Gumbel	$[2, 10]$	-	$\{0°, 90°, 180°, 270°\}$
Frank	$[10, 25]$	-	$0°$
Joe	$[2, 10]$	-	$\{0°, 90°, 180°, 270°\}$
BB1	$[1, 5]$	$[1, 5]$	$\{0°, 90°, 180°, 270°\}$
BB7	$[1, 6]$	$[2, 20]$	$\{0°, 90°, 180°, 270°\}$

B.4 Training times

Table 5 shows the average training times for the IGC model and a vine copula model with TLL2 pair-copulas for the different data sets. Note that the IGC model for the FashionMNIST experiment is a three-layer neural network with 200 neurons per layer which is trained for 100 epochs while in all other cases a two-layer network with 100 neurons per layer was trained for 500 epochs. Timings are for a desktop PC with a Intel Core i7-7700 3.60Ghz CPU and 8GB RAM.

Table 5: Average training times for IGC and vine copula (TLL2)

	IGC	vine copula (TLL2)	D	N_{train}
Learning bivariate copulas	49s	<1s	2	1000
Learning vine copulas	131s	5s	5	5000
Exchange rates	33s	4s	5	1169
Magic	61s	4s	5	2466
FashionMNIST AE	1472s	1743s	25	60000

B.5 Autoencoder architecture

We used the following architecture for the autoencoder and the VAE:

- Encoder:

$$\begin{aligned} x \in \mathbb{R}^{32\times32\times1} &\to Conv(32,4,2) \to BN \to ReLU \\ &\to Conv(64,4,2) \to BN \to ReLU \\ &\to Conv(128,4,2) \to BN \to ReLU \\ &\to FC(256) \to ReLU \\ &\to FC(25) \to z \in \mathbb{R}^{25} \;\; (AE) \\ &\to FC(50) \to \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}\mathbf{I}) \to z \in \mathbb{R}^{25} \;\; (VAE) \end{aligned}$$

- Decoder:

$$\begin{aligned} z \in \mathbb{R}^{25} &\to FC(4096) \to ReLU \to Reshape((4,4,256)) \\ &\to ConvT(128,4,2) \to BN \to ReLU \\ &\to ConvT(64,4,2) \to BN \to ReLU \\ &\to ConvT(32,4,2) \to BN \to ReLU \\ &\to ConvT(1,1,1) \to Sigmoid \to y \in \mathbb{R}^{32x32x1} \end{aligned}$$

$Conv(a, b, c)$ denotes a convolutional layer with a filters, a kernel of height and width b, and a stride with height and width c. $ConvT(a, b, c)$ denotes a deconvolutional layer. $FC(a)$ denotes a fully connected layer with a neurons. $Sigmoid$ and $ReLU$ denote the sigmoid and ReLU activation

functions and BN denotes batch normalization layers. We use a padding of 2 to achieve an image resolution of 32×32 pixels and normalize the inputs to the range $[0, 1]$. The models are trained with the binary cross entropy as reconstruction loss for 100 epochs using the Adam optimizer with default parameters.

B.6 Significance of FashionMNIST results

In the following tables we present the p-values from the test proposed in [5] for the experiments on the FashionMNIST data. High/low values in the row indicate significantly better/worse performance compared to model in the column.

Table 6: p-values for copula space samples

	IGC	GMMN	Vine	Gauss	Indep
IGC		0.5561	0.9871	0.9990	1.0000
GMMN	0.4439		0.8645	0.9689	1.0000
Vine	0.0129	0.1355		0.9510	1.0000
Gauss	0.0010	0.0311	0.0490		1.0000
Indep	0.0000	0.0000	0.0000	0.0000	

Table 7: p-values the latent space samples

	IGC	GMMN	Vine	Gauss	Indep
IGC		1.0000	0.9812	1.0000	1.0000
GMMN	0.0000		0.0000	0.0000	1.0000
Vine	0.0188	1.0000		1.0000	1.0000
Gauss	0.0000	1.0000	0.0000		1.0000
Indep	0.0000	0.0000	0.0000	0.0000	

Table 8: p-values image space samples

	IGC	GMMN	Vine	Gauss	Indep	VAE
IGC		0.1131	1.0000	1.0000	1.0000	1.0000
GMMN	0.8869		1.0000	1.0000	1.0000	1.0000
Vine	0.0000	0.0000		0.1152	1.0000	1.0000
Gauss	0.0000	0.0000	0.8848		1.0000	1.0000
Indep	0.0000	0.0000	0.0000	0.0000		0.0000
VAE	0.0000	0.0000	0.0000	0.0000	1.0000	