Generative neural networks for characteristic functions

Florian Brück

Abstract

In this work, we provide a simulation algorithm to simulate from a (multivariate) characteristic function, which is only accessible in a black-box format. We construct a generative neural network, whose loss function exploits a specific representation of the Maximum-Mean-Discrepancy metric to directly incorporate the targeted characteristic function. The construction is universal in the sense that it is independent of the dimension and that it does not require any assumptions on the given characteristic function. Furthermore, finite sample guarantees on the approximation quality in terms of the Maximum-Mean Discrepancy metric are derived. The method is illustrated in a short simulation study.

KEYWORDS: Characteristic function, generative modeling, simulation algorithm

1 Introduction

The characteristic function is one of the fundamental objects in probability theory, since it uniquely characterizes the distribution of a real-valued random vector in a concise way. Its properties often allow to simplify theoretical derivations, especially when sums of independent random variables are investigated. Further, it also allows to easily derive certain properties of the underlying random vector, such as its moments. A disadvantage of working with characteristic functions is that simulation from the corresponding random vector is not straightforward when there is no further information about the underlying random vector. As Devroye comments on the simulation from a (univariate) characteristic function in [12]: *"If the characteristic function is known in black-box format, very little can be done in a universal manner"*. This poses major challenges in applications, since simulation from the corresponding random vector is often essential to asses certain quantities of interest.

Several approaches to simulate from a random vector that corresponds to a given characteristic function seem to naturally come to mind. There are various ways of "inverting" the characteristic function to obtain its corresponding (Lebesgue) density or distribution function, such as the Fourier inversion formula, Lévy's characterization theorem and several other variants thereof. However, even though the theory provides clear constructive ways of recovering a

corresponding density or distribution function, it is still a major challenge to follow them in practice. The main reason for this difficulty is that all these ways of recovering the corresponding density or distribution function require the evaluation of many integrals. More specifically, for every point at which the density or distribution function should be recovered, one integral has to be evaluated. Usually, one needs to resort to numerical integration routines to evaluate these integrals, which become computationally intensive in already moderate dimensions, since they require an exponential growth of function evaluations to keep the error stable across different dimensions [5, Chapter IX.2].

After recovering the density or distribution function on a grid, one has to extend these objects to globally valid densities or distribution functions. Assuming that this non-trivial task can be achieved, simulation from a given density or distribution function without further information on the underlying distribution can only be considered rather simple in the univariate case. For example, [11, 13, 3, 7] consider the univariate case and essentially apply Fourier inversion to obtain the density and a dominating density of the corresponding random variable to apply acceptance-rejection simulation techniques. Similarly, [14, 9] use Fourier inversion techniques to obtain the distribution function and density corresponding to a univariate characteristic function along with error guarantees on Monte Carlo estimates from the approximated distribution function. An extension of these approaches to the multivariate setting seems challenging and the literature on this problem is scarce. Often, the authors seem to focus on the bivariate case and on inverting the related, but numerically more convenient, Laplace transform, not the characteristic function, see e.g. [10, 1, 2, 8]. Instead of "inverting" the characteristic function to an object suitable for simulation, this paper proposes a novel machine learning based simulation algorithm that directly incorporates the given characteristic function into its loss function.

The algorithm is inspired by generative machine learning models. Such models usually take an input dataset and try to generate samples that are as close as possible to the input data, where "closeness" is measured in terms of a certain "distance criterion". In contrast to these scenarios, we are not provided with an input dataset that we would like to imitate, but we would like to learn a specific target distribution that is solely available in terms of its characteristic function. To achieve this, we use a generative neural network, whose loss function is based on a specific representation of the Maximum-Mean-Discrepancy metric (MMD) derived from a translation invariant kernel [24]. The advantage of this choice of loss function is that it allows to exploit a representation of the MMD as a weighted L_2 distance of characteristic functions to directly incorporate the characteristic function into the loss function of the model, without having to simulate from the targeted distribution. Furthermore, an evaluation of the loss function only requires the ability to evaluate the given characteristic function at every possible argument, which allows to consider characteristic functions which are only accessible in a "black-box" format. The contribution of this paper can be summarized as follows.

Contribution: We provide a universal machine learning based method for

simulation from a characteristic function, which is only assumed to be given in a black-box format. We solely require the ability to evaluate the characteristic function at every argument. Moreover, the suggested method is independent of the dimension of the underlying random vector, which makes multivariate applications as easy as univariate applications.

The paper is organized as follows. Section 2 describes the construction of a generative neural network that generates samples from a given characteristic function. Section 3 provides theoretical guarantees on the approximation quality of such a generative neural network in terms of the MMD metric. Section 4 illustrates the algorithm in a short simulation study and Section 5 summarizes the results and sketches open research questions. Proofs can be found in Appendix A. Further information and results concerning the simulation study can be found in Appendices B and C.

2 Construction of the generator

Assume that we are given the characteristic function of a probability measure P on $\mathbb{R}^d$, i.e.

$$\Phi_P : \mathbb{R}^d \to \mathbb{C};\ \boldsymbol{z} \mapsto \Phi_P(\boldsymbol{z}) = \mathbb{E}\left[e^{i\boldsymbol{z}^\intercal \boldsymbol{X}}\right] = \int e^{i\boldsymbol{z}^\intercal \boldsymbol{x}} P(\mathrm{d}\boldsymbol{x}),$$

Recall that we assume that $\Phi_P(\cdot)$ is the only information about $\boldsymbol{X}/P$ that we have access to. The goal of this section is to construct a generative neural network, called generator, that can (approximately) create samples from the probability measure P.

The general idea can be described as follows: Choose your favourite neural network architecture $N_{\boldsymbol{\theta}} : \mathbb{R}^{d'} \to \mathbb{R}^d$, where $\boldsymbol{\theta} \in \Theta := \mathbb{R}^p$ denotes the parameter vector of a neural network with p parameters. Feed the neural network $N_{\boldsymbol{\theta}}(\cdot)$ random input vectors $(\boldsymbol{Z}_i)_{1\leq i\leq n}$ from a distribution $P_{\boldsymbol{Z}}$ which can be easily sampled. Take the output vectors $(\boldsymbol{Y}_i)_{1\leq i\leq n}$ of the neural network and "compare" how close they are to the target distribution P via a suitable loss function, where we use the notation

$$\boldsymbol{Y}_i := \boldsymbol{Y}_i(\boldsymbol{\theta}) := N_{\boldsymbol{\theta}}(\boldsymbol{Z}_i).$$

Update the neural network parameters $\boldsymbol{\theta}$ by taking a stochastic gradient descent step towards the minimum of the loss function. Iterate this procedure until convergence of $\boldsymbol{\theta}$ to the minimizer $\boldsymbol{\theta}^\star$ of the loss function is achieved. The resulting neural network $N_{\boldsymbol{\theta}^\star}$ should then approximately satisfy $N_{\boldsymbol{\theta}^\star}(\boldsymbol{Z}) \sim P$.

This procedure is very well known in Machine Learning under the term generative modeling. The key difference here is that we are not given a sample dataset $(\boldsymbol{X}_i)_{i\in\mathbb{N}}$ from P which can be used during training of the model. Instead, we target the theoretical representation of P in terms of Φ_P directly in the loss function of our model.

2.1 MMD metrics based on translation invariant kernels

Our choice for the loss function is based on specific representatives of the MMD metrics [24], namely those which are constructed from a translation invariant kernel. Before introducing our loss function explicitly, let us recall these special representatives of the MMD metrics. In general, every symmetric positive definite function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, called kernel, defines a semi-metric on the space of probability measures on $\mathbb{R}^d$ by

$$\mathrm{MMD}_k(P_1, P_2) := \left(\mathbb{E}\left[k(\boldsymbol{X}, \boldsymbol{X}')\right] - \mathbb{E}\left[k(\boldsymbol{X}, \boldsymbol{Y}')\right] - \mathbb{E}\left[k(\boldsymbol{X}', \boldsymbol{Y})\right] + \mathbb{E}\left[k(\boldsymbol{Y}, \boldsymbol{Y}')\right]\right)^{1/2},$$

where the random vectors $\boldsymbol{X}, \boldsymbol{X}' \sim P_1$ and $\boldsymbol{Y}, \boldsymbol{Y}' \sim P_2$ are mutually independent. If $k(\boldsymbol{x}, \boldsymbol{y}) = \psi_k(\boldsymbol{x} - \boldsymbol{y})$ for some continuous and positive definite function $\psi_k : \mathbb{R}^d \to \mathbb{R}$, k is called translation invariant kernel. Moreover, when $\psi_k(0) = 1$, Bochner's theorem implies that k has the representation

$$k(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}\left[\exp\left(i(\boldsymbol{x} - \boldsymbol{y})^\intercal \boldsymbol{W}\right)\right],$$

where $\boldsymbol{W}$ denotes a unique symmetric random variable with values in $\mathbb{R}^d$. Exploiting this representation of k, [24, Corollary 4] shows that $\mathrm{MMD}_k(P_1, P_2)$ can be expressed as

$$\mathrm{MMD}_k(P_1, P_2) = \left(\mathbb{E}\left[\|\Phi_{P_1}(\boldsymbol{W}) - \Phi_{P_2}(\boldsymbol{W})\|_2^2\right]\right)^{1/2}. \tag{1}$$

Therefore, when k is a translation invariant kernel with $k(\boldsymbol{0}, \boldsymbol{0}) = 1$, $\mathrm{MMD}_k(P_1, P_2)$ can be interpreted as a the expected distance of the characteristic functions Φ_{P_1} and Φ_{P_2} at the random location $\boldsymbol{W}$. Further, if the support[1] of $\boldsymbol{W}$ is equal to $\mathbb{R}^d$, [24, Theorem 9] shows that $\mathrm{MMD}_k(P_1, P_2)$ defines a proper metric on the space of probability measures on $\mathbb{R}^d$. Thus, from now on we will make the following assumption, which is satisfied for most of the commonly used translation invariant kernels.

Assumption 1. *The kernel k satisfies $k(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}\left[\exp\left(i\boldsymbol{W}^\intercal(\boldsymbol{x} - \boldsymbol{y})\right)\right]$ for some random vector $\boldsymbol{W}$ with support $\mathbb{R}^d$.*

Remark 1. *It is rather easy to find a kernel k for which $\boldsymbol{W}$ is known and has support $\mathbb{R}^d$. For example, when $k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(\|\boldsymbol{x} - \boldsymbol{y}\|_2^2/\sigma\right)$, the random vector $\boldsymbol{W}$ has independent components which follow a Gaussian distribution with mean 0 and variance $2/\sigma$ and when $k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(\|\boldsymbol{x} - \boldsymbol{y}\|_1/\sigma\right)$ the random vector $\boldsymbol{W}$ has independent components which follow a Cauchy distribution with location parameter 0 and scale parameter $1/\sigma$. In both cases, $\boldsymbol{W}$ has support $\mathbb{R}^d$. The kernels are called the Gaussian and Laplace kernel with bandwidth parameter σ, respectively.*

2.2 Construction of the loss function

We would like to construct a loss function which measures the distance of the distribution $P_{\boldsymbol{\theta}}$ of $N_{\boldsymbol{\theta}}(\boldsymbol{Z})$ and our target probability distribution P, which is

[1] The smallest closed set A s.t. $\mathbb{P}(\boldsymbol{W} \in A) = 1$

solely represented in terms of Φ_P. However, $P_{\boldsymbol{\theta}}$ is usually inaccessible and therefore we have to resort to empirical approximations of $P_{\boldsymbol{\theta}}$. Here, this is done in terms of sampling i.i.d. observations $(\boldsymbol{Y}_i)_{1\leq i\leq n} = (N_{\boldsymbol{\theta}}(\boldsymbol{Z}_i))_{1\leq i\leq n}$ from $P_{\boldsymbol{\theta}}$ and approximating $P_{\boldsymbol{\theta}}$ by its empirical measure $P_{\boldsymbol{\theta},n} := n^{-1}\sum_{i=1}^n \delta_{\boldsymbol{Y}_i}$. By virtue of (1), this allows to estimate $\mathrm{MMD}_k(P_{\boldsymbol{\theta}}, P)$ via

$$\mathrm{MMD}_k(P_{\boldsymbol{\theta},n}, P) = \left(\mathbb{E}_{\boldsymbol{W}}\left[\left\|n^{-1}\sum_{i=1}^n \exp(i\boldsymbol{W}^\intercal \boldsymbol{Y}_i) - \Phi_P(\boldsymbol{W})\right\|_2^2\right]\right)^{1/2}.$$

Simple calculations show that

$$(\mathrm{MMD}_k(P_{\boldsymbol{\theta},n}, P))^2 = \frac{1}{n^2}\mathbb{E}_{\boldsymbol{W}}\left[\sum_{i,j=1}^n \exp(i\boldsymbol{W}^\intercal(\boldsymbol{Y}_i - \boldsymbol{Y}_j))\right] - \frac{2}{n}\mathbb{E}_{\boldsymbol{W}}\left[\sum_{i=1}^n \exp(-i\boldsymbol{W}^\intercal \boldsymbol{Y}_i)\Phi_P(\boldsymbol{W})\right] + C_P,$$

where $C_P := \mathbb{E}_{\boldsymbol{W}}\left[\Phi_P(\boldsymbol{W})\overline{\Phi_P(\boldsymbol{W})}\right]$ is a constant that solely depends on P. Note that $\mathbb{E}_{\boldsymbol{W}}\left[\sum_{i=1}^n \exp(-i\boldsymbol{W}^\intercal \boldsymbol{Y}_i)\Phi_P(\boldsymbol{W})\right]$ is real-valued, since it can be expressed as $n\mathbb{E}_{\boldsymbol{Y}\sim P_{\boldsymbol{\theta},n},\boldsymbol{X}\sim P}\left[k(\boldsymbol{X},\boldsymbol{Y})\right]$. Usually, it is not realistic to assume that $\mathbb{E}_{\boldsymbol{W}}\Big[\sum_{i=1}^n \exp(-i\boldsymbol{W}^\intercal \boldsymbol{Y}_i)\Phi_P(\boldsymbol{W})\Big]$ can be computed in closed form. Thus, we further approximate $\mathrm{MMD}_k(P_{\boldsymbol{\theta}}, P)$ using the approximations

$$\mathbb{E}_{\boldsymbol{W}}\left[\sum_{i=1}^n \exp(-i\boldsymbol{W}^\intercal \boldsymbol{Y}_i)\Phi_P(\boldsymbol{W})\right] \approx \frac{1}{m}\sum_{i=1}^n\sum_{l=1}^m \Re\left(\exp(-i\boldsymbol{W}_l^\intercal \boldsymbol{Y}_i)\Phi_P(\boldsymbol{W}_l)\right), \quad (2)$$

and

$$\mathbb{E}_{\boldsymbol{W}}\Big[n^{-2}\sum_{i,j=1}^n \exp(i\boldsymbol{W}^\intercal(\boldsymbol{Y}_i - \boldsymbol{Y}_j))\Big] \approx \frac{1}{mn(n-1)}\sum_{\substack{i,j=1\\ i\neq j}}^n\sum_{l=1}^m \exp(i\boldsymbol{W}_l^\intercal(\boldsymbol{Y}_i - \boldsymbol{Y}_j)), \quad (3)$$

where $(\boldsymbol{W}_l)_{1\leq l\leq m}$ denote i.i.d. copies of $\boldsymbol{W}$ and $\Re(z)$ denotes the real part of a complex number z. Note that we do not take into account the observations with indices $i = j$ in (3), since they would introduce a bias in the estimate of $\mathrm{MMD}_k(P_\theta, P)$. Further, it is necessary to only consider the real part of the right hand side of (2), since, even though $\mathbb{E}_{\boldsymbol{W}}\left[\sum_{i=1}^n \exp(-i\boldsymbol{W}^\intercal \boldsymbol{Y}_i)\Phi_P(\boldsymbol{W})\right]$ is real-valued, $m^{-1}\sum_{l=1}^m\sum_{i=1}^n \exp(-i\boldsymbol{W}_l^\intercal \boldsymbol{Y}_i)\Phi_P(\boldsymbol{W}_l)$ does not need to be real-valued anymore. Taking the real part in (3) is not necessary, since every term is summed with its complex conjugate, which gives a real-valued approximation.

Combining the approximations above, this allows to define our loss function for the training of $N_{\boldsymbol{\theta}}(\cdot)$ as

$$\begin{aligned} L(\boldsymbol{\theta}) &:= L\left((\boldsymbol{Y}_i)_{1\leq i\leq n}, (\boldsymbol{W}_l)_{1\leq l\leq m}, \Phi_P\right) \\ &:= \frac{1}{mn(n-1)}\sum_{\substack{i,j=1\\ i\neq j}}^n\sum_{l=1}^m \exp(i\boldsymbol{W}_l^\intercal(\boldsymbol{Y}_i - \boldsymbol{Y}_j)) \end{aligned}$$

$$-2\Re\left(\frac{1}{nm}\sum_{l=1}^{m}\sum_{i=1}^{n}\exp(-i\boldsymbol{W}_l^\intercal Y_i)\Phi_P(\boldsymbol{W}_l)\right), \tag{4}$$

which is an approximation of the unknown quantity $\mathrm{MMD}(P_{\boldsymbol{\theta}},P)^2 - C_P$. Obviously, a $\boldsymbol{\theta}$ which minimizes $L\left((\boldsymbol{Y}_i)_{1\le i\le n},(\boldsymbol{W}_l)_{1\le l\le m},\Phi_P\right)$ is an approximation of $\operatorname{argmin}_{\boldsymbol{\theta}\in\Theta}\mathrm{MMD}_k(P_{\boldsymbol{\theta}},P)$. The quality of the approximation of the loss function is discussed in further detail in the next section.

Based on the loss function in (4), we may construct a generator of the probability distribution P as follows: Choose $\boldsymbol{Z}\sim P_{\boldsymbol{Z}}$, k (resp. $\boldsymbol{W}$) and a neural network $(N_{\boldsymbol{\theta}}(\cdot))_{\boldsymbol{\theta}\in\Theta}$. Define the loss function $L(\boldsymbol{\theta})$ as in (4) and find $\boldsymbol{\theta}^\star := \operatorname{argmin}_{\boldsymbol{\theta}\in\Theta} L(\boldsymbol{\theta})$. Then, $N_{\boldsymbol{\theta}^\star}(\boldsymbol{Z})$ should approximately be distributed according to P. The pseudo-code of the algorithm is schematically summarized in Algorithm 1.

Algorithm 1: Learning the generator of the probability distribution P which is solely parameterized in terms of Φ_P

Input: Characteristic function Φ_P of target probability distribution P
Requires: Kernel k, generator of $\boldsymbol{W}/\boldsymbol{Z}$, number of epochs e, and batch sizes $n,m\in\mathbb{N}$

```
Initialize a neural network N_θ : R^{d'} → R^d ;
for 1 ≤ k ≤ e do
    Simulate i.i.d. random vectors (W_l)_{1≤l≤m} and (Z_i)_{1≤i≤n};
    Compute (Y_i)_{1≤i≤n} = (N_θ(Z_i))_{1≤i≤n};
    Calculate L(θ) = L((Y_i)_{1≤i≤n}, (W_l)_{1≤l≤m}, Φ_P);
    Update θ by taking a gradient step towards the minimizer of L(θ).
end
```

Result: Generator $N_{\boldsymbol{\theta}}(\cdot)$ of the probability distribution P.

Remark 2. *An alternative loss function could be defined via*

$$\int_{\mathbb{R}^d}\|\hat{\Phi}_n(\boldsymbol{z})-\Phi_P(\boldsymbol{z})\|^2 w(\boldsymbol{z})\mathrm{d}\boldsymbol{z}, \tag{5}$$

where $\hat{\Phi}_n(\boldsymbol{z}) = n^{-1}\sum_{i=1}^n\exp(i\boldsymbol{z}^\intercal\boldsymbol{Y}_i)$ *denotes the empirical characteristic function of the sample* $(\boldsymbol{Y}_i)_{1\le i\le n}$ *and* $w:\mathbb{R}^d\to[0,\infty)$ *denotes a non-negative Lebesgue-integrable "weighting" function. A loss function of the form (5) has been used for estimation purposes and goodness-of-fit testing, see [28, 20] for reviews of the topic. It has also been used in generative machine learning to learn the distribution of a given dataset [4, 18, 17]. However, it has not been used for simulation purposes yet. A simple calculation shows that a simulation approach based on (4) is essentially equivalent to a simulation approach based on a discretized version of (5), when we assume that* $\boldsymbol{W}$ *has a density w.r.t. the Lebesgue measure or, equivalently, that* $w(\cdot)$ *integrates to* 1.

Remark 3. *In the definition of our loss function (4) we approximate* $\mathbb{E}_{\boldsymbol{W}}\left[\exp\left(i\boldsymbol{W}^\intercal(\boldsymbol{Y}_i-\boldsymbol{Y}_j)\right)\right]$ *according to (3). Mathematically, this is not necessary, since we know that* $\mathbb{E}_{\boldsymbol{W}}\left[\exp\left(i\boldsymbol{W}^\intercal(\boldsymbol{Y}_i-\boldsymbol{Y}_j)\right)\right]=k(\boldsymbol{Y}_i,\boldsymbol{Y}_j)$*. However, we refrain from using the exact expression, since, due to the equivalence of our loss function with a discretized version of (5), one can easily see that the loss function in (4) defines a pseudo-metric for every sample* $(\boldsymbol{W}_l)_{1\leq l\leq m}$ *(up to a multiplication of one term with* $(n-1)/n$*). This property would be lost if we used the exact mathematical expression instead. Thus, to essentially work with a pseudo-metric in every step of the algorithm, we use the approximation (3) instead of the mathematically exact expression in our loss function.*

Remark 4. *Since the loss function* $L\left((\boldsymbol{Y}_i)_{1\leq i\leq n},(\boldsymbol{W}_l)_{1\leq l\leq m},\Phi_P\right)$ *is an approximation of* $\mathrm{MMD}_k(P_{\boldsymbol{\theta}},P)^2-C_P$*, its realizations cannot be interpreted as a "large" or "small" loss in absolute terms. They may only be compared relatively to each other. However,* $C_P=\mathbb{E}_{\boldsymbol{W}}\left[\Phi_P(\boldsymbol{W})\overline{\Phi_P(\boldsymbol{W})}\right]$ *can be approximated by* $m^{-1}\sum_{1\leq l\leq m}\Phi_P(\boldsymbol{W}_l)\Phi_P(-\boldsymbol{W}_l)=:\hat{C}_P$*. Thus, to estimate* $\mathrm{MMD}_k(P_{\boldsymbol{\theta}},P)^2$ *directly, one can use* $L\left(\boldsymbol{Y}_i\right)_{1\leq i\leq n},(\boldsymbol{W}_l)_{1\leq l\leq m},\Phi_P)+\hat{C}_P$*, which then allows to interpret the loss of network as "large" or "small". The term* $\hat{C}_P$ *could obviously be included in every calculation of the loss, but to speed up the computation of the loss we have refrained from doing so.*

3 Theoretical guarantees

In general, providing a formal recipe of a neural network architecture that is reasonable for a problem at hand and comes with theoretical guarantees is known to be a difficult task. It is still an active area of research to provide theoretical guarantees for the approximation quality of neural networks with random input and general loss functions, see e.g. [6, 16, 19]. In this work, the goal is to choose a suitable architecture of the neural network which is compatible with the loss function in (4), in the sense that it allows to approximately generate samples of an arbitrary probability distribution P.

Two approaches might be considered to obtain results about the approximation quality. First, one could try to quantify the quality of the approximation of a function $G:\mathbb{R}^{d'}\to\mathbb{R}^d$ which satisfies $G(\boldsymbol{Z})\sim P$. When $P_{\boldsymbol{Z}}$ is absolutely continuous w.r.t. the Lebesgue measure, the existence of such a function G is ensured by Rosenblatt's transform [22]. Since the Rosenblatt transform is known to not be unique, it is clear that there are many functions G which satisfy $G(\boldsymbol{Z})\sim P$. Thus, it could be the case that $N_{\boldsymbol{\theta}}$ and a chosen G are very different, even though $N_{\boldsymbol{\theta}}(\boldsymbol{Z})$ is still a good approximation of a generator of P. Furthermore, it happens frequently that neither representative of G obeys any regularity conditions such as continuity or differentiability and $G\in L_1(P_{\boldsymbol{Z}})$ if and only if $\mathbb{E}_{\boldsymbol{X}\sim P}\left[|X|\right]$ is finite. Thus, an analysis based on a specific representation of G seems to be difficult.

On the other hand, a neater approach is to directly target the approximation in terms of the distance of the distribution of $N_{\boldsymbol{\theta}}(\boldsymbol{Z})$ and P. In general, one would hope for an approximation result in terms of a metric that metrizes weak convergence of probability measures on $\mathbb{R}^d$. In our framework, this distance is naturally given by the MMD metric and an analysis of the approximation capabilities in terms of the MMD metric of a feedforward neural network with ReLu activation function has been recently conducted in [27]. Their results can be summarized as follows: Consider a bounded, translation invariant and characteristic kernel $|k| \leq 1$ on $\mathbb{R}^d$. Then, if $P_{\boldsymbol{Z}}$ is an absolutely continuous probability distribution w.r.t. the Lebesgue measure, there exists a fully connected feedforward neural network $N_{\boldsymbol{\theta}^\star} : \mathbb{R}^{d'} \to \mathbb{R}^d$ with ReLu activation function, depth $h = 2$ and widths $(w_i)_{1\leq i\leq h} \geq 7d + 1$ such that $\mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star}, P) \leq 160\sqrt{d}\,(\max_{1\leq i\leq h} w_i)^{-1}\, h^{-1/2}$.

Since we assume that our kernel k is bounded, translation invariant and continuous, [23, Theorem 3.2] implies that the considered MMD metrics metrize weak convergence, i.e. $\mathrm{MMD}_k(P_n, P) \to 0$ if and only if P_n weakly converges to P. An immediate corollary is that there exists a sequence of neural networks whose distribution weakly converges to the target distribution.

Corollary 3.1. *Let $\boldsymbol{X}$ denote an arbirary random vector. Assume that $P_{\boldsymbol{Z}}$ is an absolutely continuous probability distribution w.r.t. the Lebesgue measure. Then there exists a sequence of fully connected neural networks $N^{(n)}_{\boldsymbol{\theta}}(\cdot)$ of depth 2 with ReLu activation function such that*

$$N^{(n)}_{\boldsymbol{\theta}}(\boldsymbol{Z}) \to \boldsymbol{X} \text{ in distribution.}$$

Thus, the right architecture of a neural network allows to built a generator which is "close" to the target distribution not only in the MMD_k-metric, but also in terms of the topology of weak convergence of probability measures.

[27] in combination with Corollary 3.1 explains how to choose the hyperparameters of a fully connected feedforward neural network with ReLu activation function such that there exists a specific parametrization of the network, which is a good generator for P. However, to this point, it is not clear that our approximation of the loss function in (4) allows to find such an approximation and whether the impact of the additional hyperparameters n and m can be quantified.

In the following, we will show that it is possible to obtain explicit bounds (in probability) on the approximation quality of a neural network with the loss function in (4). To this purpose, we need to introduce some notation. Let $P_{\boldsymbol{\theta},n} = n^{-1}\sum_{i=1}^n \delta_{N_{\boldsymbol{\theta}}(\boldsymbol{Z}_i)}$ denote the empirical measure of an i.i.d. sample from $P_{\boldsymbol{\theta}}$. Moreover, denote

$$\boldsymbol{\theta}_n = \mathrm{argmin}_{\boldsymbol{\theta}\in\Theta}\, \mathrm{MMD}_k(P_{\boldsymbol{\theta},n}, P)$$

and

$$\boldsymbol{\theta}_{n,m} = \mathrm{argmin}_{\boldsymbol{\theta}\in\Theta}\, L\left((\boldsymbol{Y}_i)_{1\leq i\leq n}, (\boldsymbol{W}_l)_{1\leq l\leq m}, \Phi_P\right).$$

With the notation at hand, we are ready to state explicit bounds on the approximation quality of a fully connected feedforward neural network that only depends on hyperparameters that can be chosen by the user.

Theorem 3.2. *Assume that k satisfies Assumption 1, $P_{\boldsymbol{Z}}$ is an absolutely continuous probability distribution w.r.t. the Lebesgue measure, $(w_i)_{1\leq i\leq h} \geq 7d+1$ and $h \geq 2$. Then the following is true:*

1. *Assume that $\mathbb{E}_{\boldsymbol{W}}\left[\exp(i\boldsymbol{W}^\intercal \boldsymbol{x})\Phi_P(-\boldsymbol{W})\right]$ can be calculated exactly for every $\boldsymbol{x} \in \mathbb{R}^d$. Then*

$$\mathrm{MMD}_k(P_{\boldsymbol{\theta}_n}, P) \leq \frac{160\sqrt{d}}{\left(\max_{1\leq i\leq h} w_i\right)\sqrt{h}} + \frac{4+6\sqrt{2\tau}}{\sqrt{n}}$$

   with probability at least $1-4\exp(-\tau)$.

2. *For every $\tau > 0$*

$$\begin{aligned}&\mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m}}, P)\\ &\leq \left(\left(\frac{160\sqrt{d}}{\max_{1\leq i\leq h} w_i\sqrt{h}}\right)^2 + \left(\frac{4+6\sqrt{2\tau}}{\sqrt{2n}}\right)^2 + \frac{\sqrt{(288m+128n)\tau}}{\sqrt{nm}} + \frac{1}{n-1}\right)^{1/2}\end{aligned}$$

   with probability at least $1-6\exp(-\tau)$.

Thus, if we assume that our optimization procedure is able to find $\boldsymbol{\theta}_{m,n}$, we can choose parameters $n, m, h, (w_i)_{1\leq i\leq h}$ such that $P_{\boldsymbol{\theta}_{m,n}}$ is arbitrarily close to P in terms of the MMD_k-metric with high probability. In other words, it is reasonable to assume that a small empirical loss in (4) means that the law of $N_{\boldsymbol{\theta}_{n,m}}(\boldsymbol{Z})$ is close to P in the MMD metric. Moreover, as the MMD_k-metric metrizes weak convergence, we can also assume that the law of $N_{\boldsymbol{\theta}_{n,m}}(\boldsymbol{Z})$ is close to P in the topology of weak convergence of probability measures.

Unfortunately, the bounds in Theorem 3.2 are rather loose, since they are based on inequalities that are valid for every probability measure P. Thus, it is expected that the bounds could be tightened significantly if additional assumptions on P are imposed. However, the author of this article is not aware of such results for the MMD metric.

4 Simulation results

This section illustrates our proposed simulation algorithm for a random vector that corresponds to a given characteristic function. The purpose is to illustrate that the method works in a rather universal manner, which is why we have chosen the same hyperparameters for all experiments that we conducted. According to the results of Section 3, we chose a standard feedforward neural network architecture with ReLU activation function and two hidden layers. The input layer has width $2d$, the first hidden layer has width 300 and the second

hidden layer has width 50 and the activation function for the last layer is chosen as a linear mapping to a d-dimensional output vector. The input random vector $\boldsymbol{Z}$ was chosen to have independent standard normal distributed margins. For all our experiments we used a convex combination of Gaussian kernels with bandwidths $b \in \{0.02, 0.5, 1, 5, 100\} =: B$, which leads to the corresponding random variable

$$\boldsymbol{W} \sim \sum_{b \in B} \mathbf{1}_{\{\eta = b\}} \sqrt{2/b} \tilde{\boldsymbol{W}},$$

where η is uniformly distributed on B and $\tilde{\boldsymbol{W}}$ follows a multivariate normal distribution with independent standard normal distributed margins. The very small and large bandwidths 0.02 and 100 were necessary to keep the neural network from learning a large constant. This can occur since $\mathrm{MMD}_k(c, P)$ is approximately constant for large c, which then mistakenly can be considered as a local minimum of the loss function. The batch sizes n, m have been fixed to 6000 as well as the number of epochs e. As a little modification to Algorithm 1, we have kept the sample $(\boldsymbol{W}_l)_{1 \leq l \leq m}$ constant across 20 epochs to avoid resampling in every iteration. To update the parameters by a gradient descent step, we chose to use the automatic differentiation routines of PyTorch [21] in combination with the ADAM optimizer introduced in [15]. We started with an initial learning rate 0.01 and adapted the learning rate after 2000 and 4000 epochs by multiplying the current learning rate with 0.1.

We have conducted three different experiments. First, we simulated from the characteristic function of two well-known distributions to compare the simulations results with the exact numerical simulation routines that are available in Python. As the first distribution we chose the multivariate normal distribution with characteristic function

$$\Phi^{(1)}_{\mu,\Sigma}(\boldsymbol{z}) = \exp\left(i\mu^\intercal \boldsymbol{z} - \boldsymbol{z}^\intercal \Sigma \boldsymbol{z}/2\right),$$

which has slim tails. As the second distribution we chose a heavy-tailed multivariate t-distribution with 3 degrees of freedom and corresponding characteristic function

$$\Phi^{(2)}_{\mu,\Sigma} = \frac{\sqrt{\pi}\Gamma(2)}{4\Gamma(3/2)} \exp\left(i\mu^\intercal \boldsymbol{z} - \sqrt{3}\sqrt{\boldsymbol{z}^\intercal \Sigma \boldsymbol{z}}\right)\left(2 + 2\sqrt{3}\sqrt{\boldsymbol{z}^\intercal \Sigma \boldsymbol{z}}\right),$$

which has only moments up to order 3. The purpose of these two examples is to enable a neutral assessment of the approximation quality of the generator, which requires access to functionals of the true underlying distribution, such as level sets of densities/distributions functions. Such quantities are only easily accessible in a multivariate setting for well-known distributions such as the normal and t-distributions. Moreover, a comparison of our method against the "gold standard" simulation algorithms that are implemented in the SciPy library of Python become possible.

As a second experiment, we have chosen a characteristic function, which does not correspond to a well-known distribution for which simulation routines are available. The purpose of this examples is to illustrate that the algorithm

works for distributions which are only accessible in a black-box format via their characteristic function. As such an example, we chose the characteristic function

$$\Phi^{(3)}_{\mu,\alpha} = \exp\left(i\mu^\intercal \boldsymbol{z}\right)\left(1+\frac{\|\boldsymbol{z}\|_2^2}{2\alpha}\right)^{-\alpha},$$

which is derived from the positive definite rational quadratic kernel/covariance function $\Phi^{(3)}_\alpha(\boldsymbol{z}) = \left(1+\|\boldsymbol{z}\|_2^2/2\alpha\right)^{-\alpha}$. Since $\Phi^{(3)}_\alpha(\boldsymbol{0}) = 1$, Bochner's theorem implies that $\Phi^{(3)}_{\mu,\alpha}$ is a valid characteristic function of some random random vector $\boldsymbol{X}$. Even though the distribution corresponding to $\Phi^{(3)}_\alpha$ has a radially symmetric density with the integral representation

$$\Gamma(\alpha)^{-1}\alpha^\alpha(2\pi)^{d/2}\int_0^\infty x^{d+\alpha-1}\exp\left(-2\pi^2x^2\|\boldsymbol{s}\|_2^2-\alpha x\right)\mathrm{d}x,$$

it is non-trivial to simulate from $\boldsymbol{X}$ in dimension $d \geq 2$, since no standard simulation method seems easily applicable. Thus, to the best of our knowledge, there is no known simulation algorithm for $\Phi^{(3)}_{\mu,\alpha}$ available in the literature. To keep the simulation study as concise as possible we always set $\alpha = 2$ in following.

As a third experiment, we have chosen a binary Gaussian mixture model, where the two multivariate Gaussian distributions are chosen with equal probability. The corresponding characteristic function is given by

$$\Phi^{(4)}_{\mu_1,\mu_2,\Sigma_1,\Sigma_2} = \frac{\exp\left(i\mu_1^\intercal\boldsymbol{z}-\boldsymbol{z}^\intercal\Sigma_1\boldsymbol{z}/2\right)+\exp\left(i\mu_2^\intercal\boldsymbol{z}-\boldsymbol{z}^\intercal\Sigma_2\boldsymbol{z}/2\right)}{2}.$$

This example is particularly interesting, because the margins follows a bimodal distribution and the density and distribution function are still tractable, which allows for comparison with the ground-truth and exact simulation schemes.

We generated the random vectors corresponding to all three above-mentioned experiments in dimensions $d = 2, 5, 10$. The location parameters μ and dependence parameters Σ (where applicable) were generated randomly and for the exact specification one can consult Appendix B. We report plots of the estimated marginal densities for a sample from the generator. Further, we report bivariate contour plots of the estimated bivariate densities of the last two margins of the generator and from the exact simulation algorithms that are available in Python. The estimates were obtained using the *gaussian_kde* function from the Scipy package in Python. When possible, we have also plotted the true marginal densities and contour levels of the underlying distributions. The number of samples used to generate these plots was set to 10^6, where we have fixed the contour levels to $\{0.0005, 0.001, 0.0025, 0.005, 0.01, 0.05, 0.1\}$ across all examples. This section contains only representative plots for each of the experiments. The additional plots which are not shown in this section can be found in Figures 5, 6, 7 and 8 in Appendix C.

A common observation that was noticed during training is that an adequate representation of the corresponding random vectors is only reached when the loss $L(\boldsymbol{\theta}) + \hat{C}_P$ is very small. In our experiments, we needed a loss smaller than

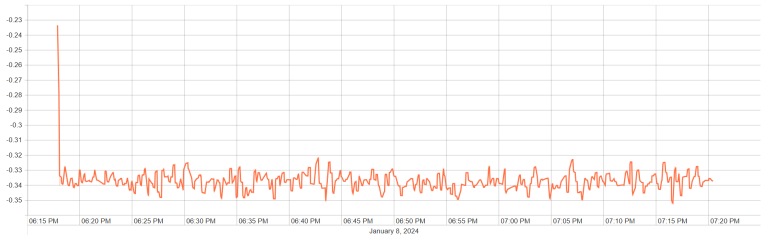


Figure 1: Exemplary visualization of the training loss during 6000 epochs

0.01 to obtain adequate results, which is probably due to the fact that our visual comparison of the distributions in terms of marginal densities and contour plots requires that the generator generates samples that are close to the target distribution in the topology of weak convergence as well. It is reasonable to assume that this is the case when the loss is very small, since we know that the MMD metric with our choice of kernel metrizes weak convergence. We should also mention that, even though the loss seems to stabilize only after a few hundred iterations (see Figure 1), further training after this stabilization improved the results. This is probably due to the fact that every sample of $(\boldsymbol{W}_l)_{1\leq l\leq m}$ specifies a new pseudo-metric which is used for comparison and some meaningful pseudo-metrics to learn the distribution are only sampled occasionally.

Figure 2 shows the estimated and true marginal and bivariate densities for the Gaussian and t-distribution in dimensions 2, 5 and 10. For both distributions, the margins are fitted almost perfectly across all dimensions. For the bivariate densities, one can observe that the contour levels of the estimated densities from the generator, the exact simulations algorithm and the true densities are almost indistinguishable towards the center of the distribution. Towards the tails, the estimates still seem to be close to the truth, but become a bit noisier. In particular, the contour levels of the estimated densities from the generator become noisier than those from the exact simulation algorithm. An exception is the Gaussian distribution in dimension 2. Here, the estimated bivariate densities from the generator do not seem to provide the correct contour levels below the mean on the y-axis. We could not find an explanation for this behavior, which only occured in the 2-dimensional case.

Figure 3 shows the estimated densities for the characteristic function derived from the rational quadratic kernel. Since there is no exact simulation algorithm and no simple closed form expression for the marginal/bivariate densities available, the results are plotted without comparison. The results reflect features that can be inferred from the characteristic function and its corresponding density, such as radial symmetry around the mean and slim tails, but judgment of the results is limited, since there is not much available information concerning the corresponding random vector. Generally speaking, one can say that the simulated random vectors seem to be adequate representations of the true underlying random vector.

Figure 4 shows the estimated densities of the Gaussian mixture distribution. The estimated marginal densities represent the location of the modes correctly, but are slightly under-pronounced. The same can be seen in the bivariate density plots, where the estimated contour lines from the estimated densities of the generator do not fully separate the two Gaussian distributions, in contrast to the estimated bivariate densities from the exact simulation algorithm. However, both the estimated bivariate densities from the generator and exact simulation algorithm do not provide a perfect fit, since their contour levels resemble the general shape of the density, but are a bit separated from the true contour levels towards the tails[2].

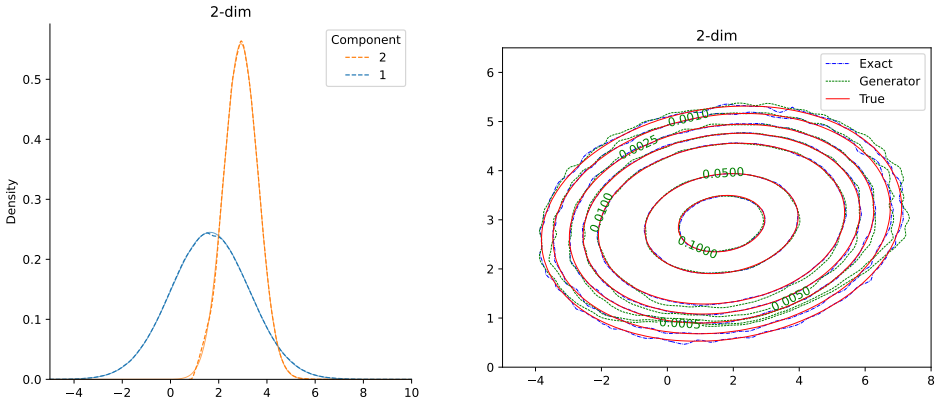
The code that was used to train the models and generate the plots can be found on `https://github.com/Flo771994/charfctgen/`. It was written in such a way that your own Pytorch implementation of a given characteristic function can be copy-pasted into the code to train a generator with the same hyperparameters as in this section.

5 Conclusion and Discussion

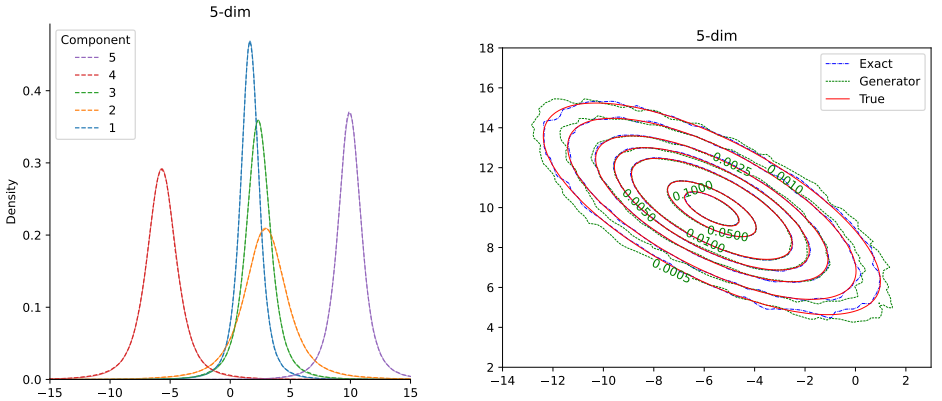
5.1 Conclusion

We have provided a simulation algorithm for the random vector corresponding to a given characteristic function, which is only accessible in a black-box format. The algorithm comes with statistical guarantees in terms of the MMD metric. To the best of the authors knowledge, the proposed algorithm is the only algorithm for the simulation from a characteristic function than can be applied in every dimension without any assumption on the underlying characteristic function. Furthermore, we have demonstrated that the algorithm seems to work reasonably well in practice, even though exact simulation routines should be preferred when they are available. The main observation is that key features of the distributions are reflected in the generated random vectors, even in higher dimensional examples, which suggest that the proposed simulation algorithm is a viable method for the simulation from a given characteristic function. Finally, generating random vectors with the proposed method is much simpler than simulation algorithms based on Fourier inversion techniques, since the method was demonstrated to work in a rather universal manner, independent of the particular example and dimension.

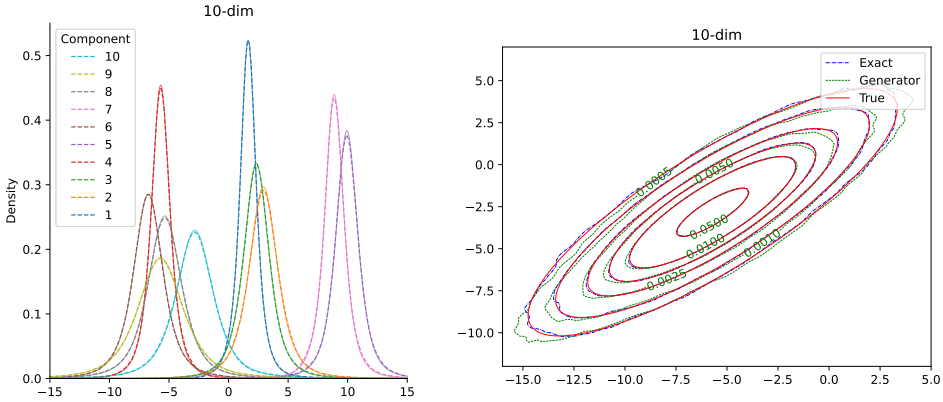
[2]This indicates that the number of simulated random vector is too low, as the estimated densities from the exact simulation algorithm should recover the true contour lines for a large enough sample size. Increasing the number of simulated random vectors from 10^6 to 10^7 only slightly improved the results, but further increasing the number of simulated random vectors was not possible under the constraint of conducting the computations in a reasonable amount of time on the available hardware.



(a) Gaussian distribution 2-dim

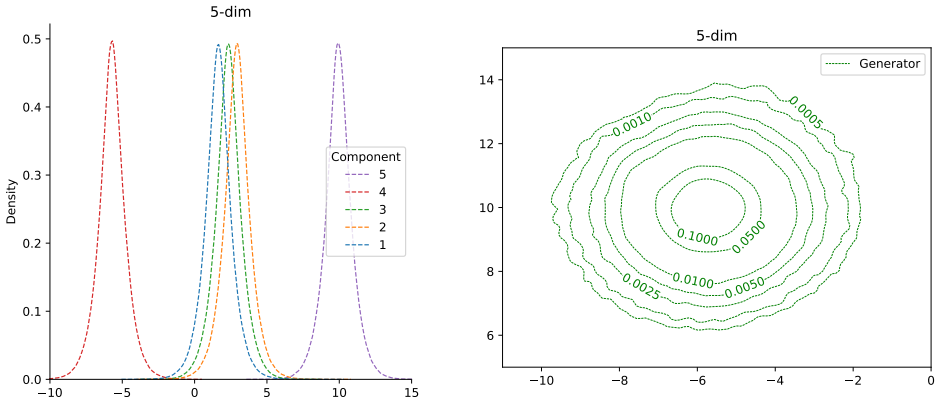


(b) t-distributionn 5-dim



(c) t-distribution 10-dim

Figure 2: Estimated marginal (left) and bivariate (right) densities of the Gaussian distribution and t-distribution in dimensions $2, 5, 10$. The bivariate densities correspond to the last two components of the corresponding random vector. The solid lines show the true densities and the (dash-)dotted lines correspond to estimated densities from the generator and the exact simulation algorithm (right,blue).



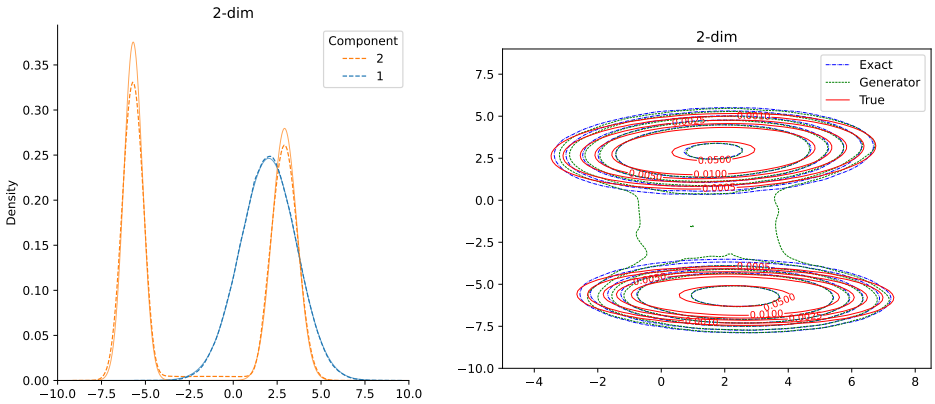
(a) Distribution corr. to rational quadratic kernel 5-dim

Figure 3: Estimated marginal (left) and bivariate (right) densities of the distribution corresponding to the rational quadratic kernel in dimensions 5. The bivariate densities correspond to the last two components of the corresponding random vector.

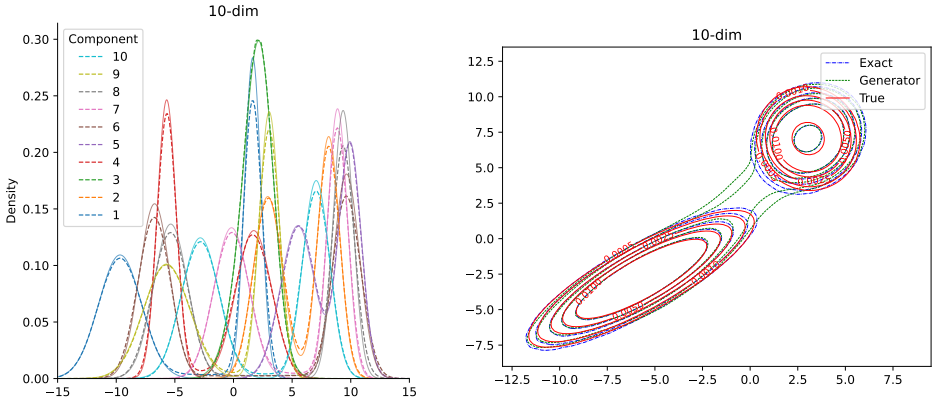
5.2 Discussion

One the one hand, the universality of the algorithm is an attractive feature. On the other hand, there are many statistical problems where one knows much more about a random vector than its corresponding characteristic function, such as tail-behavior, moments or even that it belongs to a certain family of distributions. In such cases, the proposed simulation algorithm cannot be considered as the optimal solution, since the only adjustable parameter with significant influence on such quantities is the distribution of the input random vector $\boldsymbol{Z}$ and the activation function of the neural network. For example, when it is known that the corresponding random vector does not have a first moment, this feature cannot be modeled by a Neural Network with ReLu activation function and Gaussian random vectors $\boldsymbol{Z}$, since the order of the tails is preserved by any ReLu neural network. To appropriately model the tails of a random vector one would need to allow for activation functions that can adapt to certain tail behavior or use the ReLu activation function in combination with an input random vector that has the correct tail behavior. Unfortunately, replacing the ReLu activation function with a different activation function makes us lose the statistical guarantees and choosing an input random vectors with the correct tail behavior can only be achieved if one assumes certain knowledge of the corresponding random vector. Thus, it is an open research question if a more flexible network architecture can be found that is still universally applicable, comes with statistical guarantees and can adapt to certain stylized features of the target distribution.

Another example of a subject where the characteristic function of a random vector is the central object of study and additional information on its struc-



(a) Gaussian mixture distribution 2-dim



(b) Gaussian mixture distribution 10-dim

Figure 4: Estimated marginal (left) and bivariate (right) densities of the Gaussian mixture distribution in dimensions $2, 5, 10$. The bivariate densities correspond to the last two components of the corresponding random vector. The solid lines show the true densities and the (dash-)dotted lines correspond to estimated densities from the generator and the exact simulation algorithm (right,blue).

ture is available are Lévy processes. Lévy processes are usually characterized in terms of their characteristic function at time $t = 1$, which follows a specific parametrization in terms of a Lévy measure and a Gaussian part. Most statistical models for Lévy processes consider the Gaussian part as given and are then parameterized solely in terms of the Lévy measure. This parametrization in terms of the Lévy measure is convenient, since it nicely encodes the properties of the associated Lévy process, such as tail behavior and path properties. A serious challenge for practical applications is however the fact that it is usually impossible to generate exact samples from a Lévy process with a given Lévy measure, since one does not have direct access to the corresponding random vector. This limits the modeling freedom of a statistician, since the modeling process usually can only be conducted under the constraint that at least an approximate simulation algorithm for the Lévy process needs to be available. In principle, the algorithm proposed in this paper can be employed to simulate from a random vector corresponding to a given Lévy measure, but exploiting the special structure of the characteristic function of a Lévy process allows for a more fine-tuned network architecture. Unfortunately, the required architecture is incompatible with the universality of the approach presented in this paper. Therefore, we leave this important application for future research.

Another open question is whether it is possible to obtain guarantees on the approximation of functionals of the distribution corresponding to a given characteristic function, such as expectations of certain functions of the random vector and quantiles. Theorem 3.2 only provides a bound on the MMD metric, which is equivalent to a bound on $\sup_{f \in \mathcal{H}_k} |\mathbb{E}_{\boldsymbol{Z} \sim P_{\boldsymbol{Z}}} \left[f(N_{\theta_{m,n}}(\boldsymbol{Z})) \right] - \mathbb{E}_{\boldsymbol{X} \sim P} \left[f(\boldsymbol{X}) \right]|$, where $\mathcal{H}_k$ denotes the unit ball of the reproducing kernel Hilbert space associated to the kernel k. However, even though $\mathcal{H}_k$ can be explicitly constructed from a given kernel k, it is not clear how useful these bounds are in practice, since many functionals of interest cannot be represented by elements of $\mathcal{H}_k$. A thorough analysis of this problem seems challenging, see e.g. [25, 26], and we leave such an analysis for future research.

References

[1] Joseph Abate, Gagan L. Choudhury, and Ward Whitt. "Numerical inversion of multidimensional Laplace transforms by the Laguerre method". In: *Performance Evaluation* 31.3 (1998), pp. 229–243. ISSN: 0166-5316. DOI: https://doi.org/10.1016/S0166-5316(97)00002-3.

[2] Joseph Abate and Ward Whitt. "A unified framework for numerically inverting Laplace transforms". In: *INFORMS Journal on Computing* 18.4 (2006), pp. 408–421. DOI: https://doi.org/10.1287/ijoc.1050.0137.

[3] Joseph Abate and Ward Whitt. "The Fourier-series method for inverting transforms of probability distributions". In: *Queueing systems* 10 (1992), pp. 5–87. DOI: https://doi.org/10.1007/BF01158520.

[4] Abdul Fatir Ansari, Jonathan Scarlett, and Harold Soh. "A characteristic function approach to deep implicit generative modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7478–7487. URL: https://openaccess.thecvf.com/content_CVPR_2020/papers/Ansari_A_Characteristic_Function_Approach_to_Deep_Implicit_Generative_Modeling_CVPR_2020_paper.pdf.

[5] Søren Asmussen and Peter W Glynn. *Stochastic Simulation: Algorithms and Analysis*. Vol. 57. Stochastic Modelling and Applied Probability. Springer, 2007. DOI: https://doi.org/10.1007/978-0-387-69033-9.

[6] Bolton Bailey and Matus Telgarsky. "Size-noise tradeoffs in generative networks". In: *Advances in Neural Information Processing Systems* 31 (2018). URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/9bd5ee6fe55aaeb673025dbcb8f939c1-Paper.pdf.

[7] L Barabesi and L Pratelli. "Universal methods for generating random variables with a given characteristic function". In: *Journal of Statistical Computation and Simulation* 85.8 (2015), pp. 1679–1691. DOI: http://dx.doi.org/10.1080/00949655.2014.892108.

[8] Lorenzo Cappello and Stephen G. Walker. "A Bayesian motivated Laplace inversion for multivariate probability distributions". In: *Methodology and Computing in Applied Probability* 20 (2018), pp. 777–797. DOI: 10.1007/s11009-017-9587-y.

[9] Zisheng Chen, Liming Feng, and Xiong Lin. "Simulating Lévy processes from their characteristic functions and financial applications". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 22.3 (2012), pp. 1–26. DOI: https://doi.org/10.1145/2331140.2331142.

[10] Gagan L. Choudhury, David M. Lucantoni, and Ward Whitt. "Multidimensional Transform Inversion with Applications to the Transient M/G/1 Queue". In: *The Annals of Applied Probability* 4.3 (1994), pp. 719–740. ISSN: 10505164. URL: http://www.jstor.org/stable/2245060.

[11] Luc Devroye. "An Automatic Method for Generating Random Variates with a Given Characteristic Function". In: *SIAM Journal on Applied Mathematics* 46.4 (1986), pp. 698–719. URL: http://www.jstor.org/stable/2101740.

[12] Luc Devroye. "Nonuniform Random Variate Generation". In: vol. 13. Handbooks in Operations Research and Management Science. Elsevier, 2006, pp. 83–121. DOI: https://doi.org/10.1016/S0927-0507(06)13004-2.

[13] Luc Devroye. "On the computer generation of random variables with a given characteristic function". In: *Computers & Mathematics with Applications* 7.6 (1981), pp. 547–552. ISSN: 0898-1221. DOI: https://doi.org/10.1016/0898-1221(81)90038-9.

[14] Paul Glasserman and Zongjian Liu. "Sensitivity Estimates from Characteristic Functions". In: *Operations Research* 58.6 (2010), pp. 1611–1623. DOI: 10.1287/opre.1100.0837.

[15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1412.6980.

[16] Holden Lee et al. "On the Ability of Neural Nets to Express Distributions". In: vol. 65. Proceedings of Machine Learning Research. PMLR, July 2017, pp. 1271–1296. URL: https://proceedings.mlr.press/v65/lee17a.html.

[17] Shengxi Li et al. "Neural Characteristic Function Learning for Conditional Image Generation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2023, pp. 7204–7214. URL: https://openaccess.thecvf.com/content/ICCV2023/papers/Li_Neural_Characteristic_Function_Learning_for_Conditional_Image_Generation_ICCV_2023_paper.pdf.

[18] Shengxi Li et al. "Reciprocal Adversarial Learning via Characteristic Functions". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 217–228. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/021f6dd88a11ca489936ae770e4634ad-Paper.pdf.

[19] Yulong Lu and Jianfeng Lu. "A Universal Approximation Theorem of Deep Neural Networks for Expressing Probability Distributions". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 3094–3105. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/2000f6325dfc4fc3201fc45ed01c7a5d-Paper.pdf.

[20] Simos G Meintanis. "A review of testing procedures based on the empirical characteristic function". In: *South African Statistical Journal* 50.1 (2016), pp. 1–14. DOI: https://hdl.handle.net/10520/EJC186846.

[21] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[22] Murray Rosenblatt. "Remarks on a Multivariate Transformation". In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 470–472. ISSN: 00034851. URL: http://www.jstor.org/stable/2236692.

[23] Bharath Sriperumbudur. "On the optimal estimation of probability measures in weak and strong topologies". In: *Bernoulli* 22.3 (2016), pp. 1839–1893. DOI: 10.3150/15-BEJ713.

[24] Bharath K. Sriperumbudur et al. "Hilbert space embeddings and metrics on probability measures". In: *Journal of Machine Learning Research* 11 (2010), pp. 1517–1561. DOI: https://www.jmlr.org/papers/volume11/sriperumbudur10a/sriperumbudur10a.pdf.

[25] Clément Dombry Thibault Modeste. "Characterization of translation invariant MMD on $\mathbb{R}^d$ and connections with Wasserstein distances (preprint)". In: *HAL* ffhal-03855093v1 (2022).

[26] Titouan Vayer and Rémi Gribonval. "Controlling Wasserstein distances by Kernel norms with application to Compressive Statistical Learning". In: *Journal of Machine Learning Research* 24.149 (2023), pp. 1–51.

[27] Yunfei Yang, Zhen Li, and Yang Wang. "On the capacity of deep generative networks for approximating distributions". In: *Neural Networks* 145 (2022), pp. 144–154. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2021.10.012.

[28] Jun Yu. "Empirical Characteristic Function Estimation and Its Applications". In: *Econometric Reviews* 23.2 (2004), pp. 93–123. DOI: 10.1081/ETC-120039605.

A Proof of Theorem 3.2

Let $\boldsymbol{\theta}^\star := \operatorname{argmin}_{\boldsymbol{\theta}\in\Theta} \mathrm{MMD}_k(P_{\boldsymbol{\theta}}, P)$.

1.

$$\begin{aligned}\mathrm{MMD}_k(P_{\boldsymbol{\theta}_n}, P) &\leq \mathrm{MMD}_k(P_{\boldsymbol{\theta}_n}, P_{\boldsymbol{\theta}_n,n}) + \mathrm{MMD}_k(P_{\boldsymbol{\theta}_n,n}, P)\\ &\leq \mathrm{MMD}_k(P_{\boldsymbol{\theta}_n}, P_{\boldsymbol{\theta}_n,n}) + \mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star,n}, P)\\ &\leq \mathrm{MMD}_k(P_{\boldsymbol{\theta}_n}, P_{\boldsymbol{\theta}_n,n}) + \mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star,n}, P_{\boldsymbol{\theta}^\star}) + \mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star}, P),\end{aligned}$$

where we have used that MMD_k satisfies the triangle inequality and $\boldsymbol{\theta}_n = \operatorname{argmin}_{\boldsymbol{\theta}\in\Theta} \mathrm{MMD}_k(P_{\boldsymbol{\theta},n}, P)$. Now $\mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star}, P) \leq 160\sqrt{d}\left(\max_{1\leq i\leq h} w_i\right)^{-1} h^{-1/2}$ by [27, Theorem 2.8]. Moreover,

$$\begin{aligned}&P\left(\mathrm{MMD}_k(P_{\boldsymbol{\theta}_n}, P_{\boldsymbol{\theta}_n,n}) + \mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star,n}, P_{\boldsymbol{\theta}^\star}) \geq \frac{4}{\sqrt{n}} + \frac{6\sqrt{2\tau}}{\sqrt{n}}\right)\\ &\leq P\left(\mathrm{MMD}_k(P_{\boldsymbol{\theta}_n}, P_{\boldsymbol{\theta}_n,n}) \geq \frac{2}{\sqrt{n}} + \frac{3\sqrt{2\tau}}{\sqrt{n}}\right) +\\ &P\left(\mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star,n}, P_{\boldsymbol{\theta}^\star}) \geq \frac{2}{\sqrt{n}} + \frac{3\sqrt{2\tau}}{\sqrt{n}}\right)\\ &\leq 2\exp(-\tau) + 2\exp(-\tau) = 4\exp(-\tau)\end{aligned}$$

by [19, Propositon 3.2], where we use $P(f(\boldsymbol{\theta}_n, (\boldsymbol{Z}_i)_{1\leq i\leq n}) \in A) = \mathbb{E}\left[P(f(\boldsymbol{\theta}_n, (\boldsymbol{Z}_i)_{1\leq i\leq n}) \in A|\boldsymbol{\theta}_n)\right]$ for every measurable function f and measurable set A to account for the

randomness of $\boldsymbol{\theta}_n$. Thus, we have

$$\mathrm{MMD}_k(P_{\boldsymbol{\theta}_n}, P) \leq 160\sqrt{d}\left(\max_{1\leq i\leq h} w_i\right)^{-1} h^{-1/2} + \frac{4+6\sqrt{2\tau}}{\sqrt{n}}$$

with probability at least $1-4\exp(-\tau)$

2. Define

$$\begin{aligned}\tilde{\mathrm{MMD}}_{k,m}(P_1,P_2)^2 := \frac{n^2}{n(n-1)}\mathbb{E}_{\boldsymbol{Y}_1,\boldsymbol{Y}_2\sim P_1}\left[\frac{1}{m}\sum_{l=1}^{m}\Re\left(\exp\left(i\boldsymbol{W}_l^\intercal(\boldsymbol{Y}_1-\boldsymbol{Y}_2)\right)\right)\right]\\ + \mathbb{E}_{X_1,X_2\sim P_2}\left[k(\boldsymbol{X}_1,\boldsymbol{X}_2)\right]\\ - 2\Re\left(\frac{1}{m}\sum_{l=1}^{m}\mathbb{E}_{\boldsymbol{Y}\sim P_1}\left[\exp\left(i\boldsymbol{W}_l^\intercal\boldsymbol{Y}\right)\Phi_{P_2}(-\boldsymbol{W}_l)\right]\right),\end{aligned}$$

for arbitrary probability measures P_1, P_2, where $(\boldsymbol{W}_l)_{1\leq l\leq m}$ denotes an i.i.d. sample of $\boldsymbol{W}$. Observe that

$$\begin{aligned}\boldsymbol{\theta}_{n,m} &= \mathrm{argmin}_{\boldsymbol{\theta}\in\Theta} L\left((\boldsymbol{Y}_i)_{1\leq i\leq n},(\boldsymbol{W}_l)_{1\leq l\leq m},\Phi_P\right))\\ &= \mathrm{argmin}_{\boldsymbol{\theta}\in\Theta}\tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta},n},P),\end{aligned}$$

since $\tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta},n},P)^2 = L\left((\boldsymbol{Y}_i)_{1\leq i\leq n},(\boldsymbol{W}_l)_{1\leq l\leq m},\Phi_P)\right) + 1/(n-1) + C_P$, because $k(\boldsymbol{x},\boldsymbol{x})=1$ for all $\boldsymbol{x}\in\mathbb{R}^d$. Now, using that the square of a metric remains a metric, we get

$$\begin{aligned}\mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m}},P)^2 &\leq \mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m}},P_{\boldsymbol{\theta}_{n,m},n})^2 + \mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m},n},P)^2\\ &= \mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m}},P_{\boldsymbol{\theta}_{n,m},n})^2 + \mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m},n},P)^2 - \tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta}_{n,m},n},P)^2\\ &\quad + \tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta}_{n,m},n},P)^2\\ &\leq \mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m}},P_{\boldsymbol{\theta}_{n,m},n})^2 + \mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m},n},P)^2 - \tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta}_{n,m},n},P)^2\\ &\quad + \tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta}^\star,n},P)^2\\ &\leq \mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m},n},P)^2 - \tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta}_{n,m},n},P)^2 \pm C(\boldsymbol{\theta}_{m,n})\\ &\quad + \tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta}^\star,n},P)^2 - \mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star,n},P)^2 \pm C(\boldsymbol{\theta}^\star)\\ &\quad + \mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m}},P_{\boldsymbol{\theta}_{n,m},n})^2 + \mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star,n},P_{\boldsymbol{\theta}^\star})^2 + \mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star},P)^2.\end{aligned}$$

where we have used that $\boldsymbol{\theta}_{n,m} = \mathrm{argmin}_{\boldsymbol{\theta}\in\Theta}\tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta},n},P)^2$ and the appropriate constants are provided by Lemma 1 below and are of the form $C(\boldsymbol{\theta}_{m,n}) := \left(-\frac{1}{n}C_1(\boldsymbol{\theta}_{m,n}) - \frac{1}{n(n-1)}\right)$ and $C(\boldsymbol{\theta}^\star) := \left(-\frac{1}{n}C_1(\boldsymbol{\theta}^\star) - \frac{1}{n(n-1)}\right)$, where $C_1(\boldsymbol{\theta}^\star), C_1(\boldsymbol{\theta}_{m,n}) \in [0,1]$. As above, $\mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star},P)^2 \leq \left(160\sqrt{d}\left(\max_{1\leq i\leq h} w_i\right)^{-1} h^{-1/2}\right)^2$ and $\mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m}},P_{\boldsymbol{\theta}_{n,m},n})^2 + \mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star,n},P_{\boldsymbol{\theta}^\star})^2 \leq \left(\frac{4}{\sqrt{n}} + \frac{6\sqrt{2\tau}}{\sqrt{n}}\right)^2/2$ with probability at least $1-4\exp(-\tau)$. Thus, it remains to bound

$$\mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m},n},P_{\boldsymbol{\theta}^\star})^2 - \tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta}_{n,m},n},P_{\boldsymbol{\theta}^\star})^2 \pm C(\boldsymbol{\theta}_{m,n})$$

and

$$\tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta}^\star,n}, P_{\boldsymbol{\theta}^\star})^2 - \mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star,n}, P_{\boldsymbol{\theta}^\star})^2 \pm C(\boldsymbol{\theta}^\star)$$

An application of Lemma 1 yields for every $c > 0$

$$\begin{aligned}
&\mathbb{P}\left(\mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m},n}, P_{\boldsymbol{\theta}^\star})^2 - \tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta}_{n,m},n}, P_{\boldsymbol{\theta}^\star})^2 - C(\boldsymbol{\theta}_{m,n}) > c\right) \\
&= \mathbb{E}\left[\mathbb{P}\left(\mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m},n}, P_{\boldsymbol{\theta}^\star})^2 - \tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta}_{n,m},n}, P_{\boldsymbol{\theta}^\star})^2 - C(\boldsymbol{\theta}_{m,n}) > c \;\middle|\; \boldsymbol{\theta}_{n,m}\right)\right] \\
&\leq \exp\left(-\frac{nmc^2}{72m+32n}\right).
\end{aligned}$$

A similar result applies to $\tilde{\mathrm{MMD}}_{k,m}(P_{\boldsymbol{\theta}^\star,n}, P_{\boldsymbol{\theta}^\star})^2 - \mathrm{MMD}_k(P_{\boldsymbol{\theta}^\star,n}, P_{\boldsymbol{\theta}^\star})^2 + C(\boldsymbol{\theta}^\star)$.

Thus, since $C(\boldsymbol{\theta}_{m,n}) < 0$ and $-C(\boldsymbol{\theta}^\star) \leq \frac{1}{n-1}$, setting $\tau = \frac{nmc^2}{72m+32n}$ and combining this with the derivations above gives

$$\begin{aligned}
\mathrm{MMD}_k(P_{\boldsymbol{\theta}_{n,m}}, P)^2 &\leq \left(\frac{160\sqrt{d}}{\max_{1\leq i\leq h} w_i\sqrt{h}}\right)^2 + \left(\frac{4+6\sqrt{2\tau}}{\sqrt{n}}\right)^2/2 + 2\frac{\sqrt{(72m+32n)\tau}}{\sqrt{nm}} \\
&\quad + C(\boldsymbol{\theta}_{m,n}) - C(\boldsymbol{\theta}^\star) \\
&\leq \left(\frac{160\sqrt{d}}{\max_{1\leq i\leq h} w_i\sqrt{h}}\right)^2 + \left(\frac{4+6\sqrt{2\tau}}{\sqrt{n}}\right)^2/2 + 2\frac{\sqrt{(72m+32n)\tau}}{\sqrt{nm}} + \frac{1}{n-1}
\end{aligned}$$

with at least probability $1 - 6\exp(-\tau)$.

Lemma 1. *Let $n, m \in \mathbb{N}$ and let $P_n = n^{-1}\sum_{i=1}^n \delta_{\boldsymbol{V}_i}$ denote the empirical measure of i.i.d. observations $(\boldsymbol{V}_i)_{1\leq i\leq n}$ from a probability measure P and let Q denote an arbitrary probability measure. Moreover, let $k(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}\left[\exp(i\boldsymbol{W}^\intercal(\boldsymbol{x}-\boldsymbol{y})\right]$ denote a symmetric, continuous and translation invariant kernel. Then*

$$\begin{aligned}
&\mathbb{P}\left(\mathrm{MMD}_k\left(P_n, Q\right)^2 - \tilde{\mathrm{MMD}}_{k,m}(P_n, Q)^2 - \left(-\frac{1}{n}C_1 - \frac{1}{n(n-1)}\right) > \epsilon\right) \\
&\leq \exp\left(-\frac{nm\epsilon^2}{72m+32n}\right)
\end{aligned}$$

and

$$\begin{aligned}
&\mathbb{P}\left(\mathrm{MMD}_k\left(P_n, Q\right)^2 - \tilde{\mathrm{MMD}}_{k,m}(P_n, Q)^2 - \left(-\frac{1}{n}C_1 - \frac{1}{n(n-1)}\right) < -\epsilon\right) \\
&\leq \exp\left(-\frac{nm\epsilon^2}{72m+32n}\right),
\end{aligned}$$

where $C_1 := \mathbb{E}_{\boldsymbol{V}_1,\boldsymbol{V}_2\sim P}\left[k(\boldsymbol{V}_1, \boldsymbol{V}_2)\right] \in [0, 1]$.

Proof. We will use McDiarmids inequality to prove the claim. Observe that

$$\mathrm{MMD}_k\left(P_n, Q\right)^2 - \tilde{\mathrm{MMD}}_{k,m}(P_n, Q)^2 = \frac{1}{n^2}\sum_{i,j=1}^{n}\left(\frac{1}{m}\sum_{l=1}^{m} k(\boldsymbol{V}_i, \boldsymbol{V}_j) - \frac{n^2}{n(n-1)}\Re\left(\exp\left(i\boldsymbol{W}_l^\intercal(\boldsymbol{V}_i - \boldsymbol{V}_j)\right)\right)\right)$$
$$+ \frac{2}{n}\sum_{i=1}^{n}\left(\frac{1}{m}\sum_{l=1}^{m}\Re\left(\exp\left(i\boldsymbol{W}_l^\intercal\boldsymbol{V}_i\right)\Phi_Q(-\boldsymbol{W}_l)\right) - \mathbb{E}_{\boldsymbol{Y}\sim Q}\left[k(\boldsymbol{V}_i, \boldsymbol{Y})\right]\right).$$

where $\mathbb{E}_{\boldsymbol{W}_l}\left[\Re\left(\exp\left(i\boldsymbol{W}_l^\intercal(\boldsymbol{V}_i - \boldsymbol{V}_j\right)\right)\right)\right] = k(\boldsymbol{V}_i, \boldsymbol{V}_j)$ and $\mathbb{E}_{\boldsymbol{W}_l}\left[\Re\left(\exp\left(i\boldsymbol{W}_l^\intercal\boldsymbol{V}_i\right)\Phi_Q(-\boldsymbol{W}_l)\right)\right] = \mathbb{E}_{\boldsymbol{Y}\sim Q}\left[k(\boldsymbol{V}_i, \boldsymbol{Y})\right]$. Thus, $\mathbb{E}\left[\mathrm{MMD}_k\left(P_n, Q\right) - \tilde{\mathrm{MMD}}_{k,m}(P_n, Q)\right] = -\frac{1}{n}C_1 - \frac{1}{n(n-1)}$, where $C_1 := \mathbb{E}_{\boldsymbol{V}_1,\boldsymbol{V}_2\sim P}\left[k(\boldsymbol{V}_1, \boldsymbol{V}_2)\right]$. Note that $C_1 \in [0,1]$, since $k(\boldsymbol{x}, \boldsymbol{y}) \in [0,1]$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ due to its positive definiteness. Moreover, note that

$$f(\boldsymbol{V}_1, \ldots, \boldsymbol{V}_n, \boldsymbol{W}_1, \ldots, \boldsymbol{W}_m) := \mathrm{MMD}_k\left(P_n, Q\right)^2 - \tilde{\mathrm{MMD}}_{k,m}(P_n, Q)^2$$

is a function that satisfies the bounded differences property of McDiarmid's inequality with $c_i = 12/n$ when $\boldsymbol{V}_i$ is replaced by some $\tilde{\boldsymbol{V}}_i$ and with $c_k \leq 8/m$ when $\boldsymbol{W}_l$ is replaced by some $\tilde{\boldsymbol{W}}_l$. Thus, we get

$$P\left(f(\boldsymbol{V}_1, \ldots, \boldsymbol{V}_n, \boldsymbol{W}_1, \ldots, \boldsymbol{W}_m) - \left(-\frac{1}{n}C_1 - \frac{1}{n(n-1)}\right) > \epsilon\right)$$
$$\leq \exp\left(-\frac{2\epsilon^2}{144/n + 64/m}\right) = \exp\left(-\frac{nm\epsilon^2}{72m + 32n}\right)$$

and

$$P\left(f(\boldsymbol{V}_1, \ldots, \boldsymbol{V}_n, \boldsymbol{W}_1, \ldots, \boldsymbol{W}_m) - \left(-\frac{1}{n}C_1 - \frac{1}{n(n-1)}\right) < -\epsilon\right)$$
$$\leq \exp\left(-\frac{nm\epsilon^2}{72m + 32n}\right)$$

which proves the claim. □

B Parameters for the simulation study

For $\Phi^{(1)}_{\mu,\Sigma}, \Phi^{(2)}_{\mu,\Sigma}, \Phi^{(3)}_{\mu,1}$ we chose the location parameters

$$\mu_1 = \begin{pmatrix}1.64\\2.92\end{pmatrix}, \quad \begin{pmatrix}1.64\\2.92\\2.33\\-5.70\\9.94\end{pmatrix}, \quad \begin{pmatrix}1.64\\2.92\\2.33\\-5.70\\9.94\\-6.73\\8.87\\-5.35\\-5.71\\-2.83\end{pmatrix}$$

and dependence parameters

$$\Sigma_1 = \begin{pmatrix} 2.65 & 0.16 \\ 0.16 & 0.51 \end{pmatrix}, \quad \begin{pmatrix} 0.61 & -0.59 & -0.20 & -0.32 & 0.17 \\ -0.59 & 3.07 & 1.30 & 1.50 & -1.36 \\ -0.20 & 1.30 & 1.05 & 0.85 & -0.68 \\ -0.32 & 1.50 & 0.85 & 1.58 & -0.86 \\ 0.17 & -1.36 & -0.68 & -0.86 & 0.99 \end{pmatrix},$$

$$\begin{pmatrix} 0.49 & -0.28 & -0.16 & -0.00 & 0.14 & 0.42 & -0.15 & 0.51 & 0.58 & 0.46 \\ -0.28 & 1.54 & 0.77 & -0.07 & -0.44 & -1.08 & 0.37 & -1.25 & -1.79 & -1.37 \\ -0.16 & 0.77 & 1.22 & -0.11 & -0.35 & -0.76 & 0.26 & -0.90 & -1.41 & -1.08 \\ -0.00 & -0.07 & -0.11 & 0.65 & 0.10 & 0.23 & -0.07 & 0.20 & 0.22 & 0.21 \\ 0.14 & -0.44 & -0.35 & 0.10 & 0.92 & 0.54 & -0.15 & 0.64 & 0.94 & 0.68 \\ 0.42 & -1.08 & -0.76 & 0.23 & 0.54 & 1.67 & -0.23 & 1.28 & 1.96 & 1.48 \\ -0.15 & 0.37 & 0.26 & -0.07 & -0.15 & -0.23 & 0.70 & -0.25 & -0.43 & -0.36 \\ 0.51 & -1.25 & -0.90 & 0.20 & 0.64 & 1.28 & -0.25 & 2.13 & 2.23 & 1.81 \\ 0.58 & -1.79 & -1.41 & 0.22 & 0.94 & 1.96 & -0.43 & 2.23 & 3.88 & 2.63 \\ 0.46 & -1.37 & -1.08 & 0.21 & 0.68 & 1.48 & -0.36 & 1.81 & 2.63 & 2.57 \end{pmatrix}$$

according to the appropriate dimension. For the binary Gaussian mixture distribution with characteristic function $\Phi^{(4)}_{\mu_1,\mu_2,\Sigma_1,\Sigma_2}$ we chose the scale parameters μ_1 and Σ_1 for the first Gaussian distribution, whereas for the second Gaussain distribution we have used the location parameters

$$\mu_2 = \begin{pmatrix} 2.33 \\ -5.70 \end{pmatrix}, \quad \begin{pmatrix} -6.73 \\ 8.87 \\ -5.35 \\ -5.71 \\ -2.83 \end{pmatrix}, \quad \begin{pmatrix} -9.67 \\ 8.12 \\ 1.58 \\ 1.73 \\ 5.48 \\ 9.64 \\ -0.17 \\ 9.35 \\ 3.05 \\ 7.05 \end{pmatrix}$$

and dependence parameters

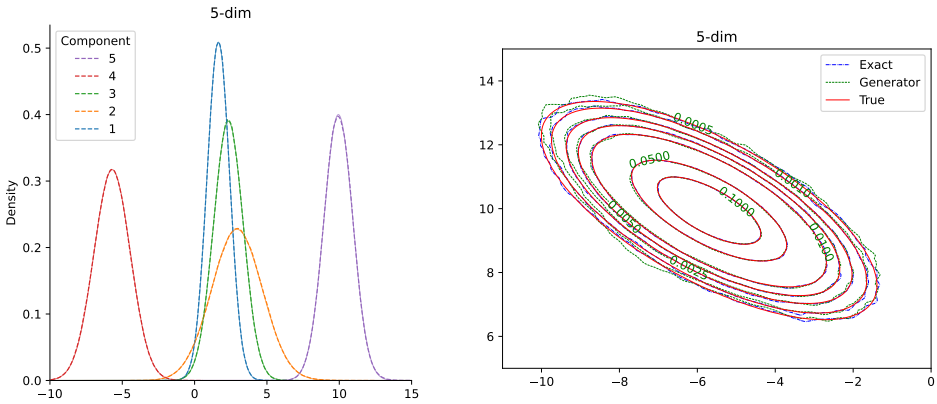
$$\Sigma_2 = \begin{pmatrix} 2.36 & -0.05 \\ -0.05 & 0.28 \end{pmatrix}, \quad \begin{pmatrix} 1.79 & 0.22 & -0.26 & 1.61 & -1.58 \\ 0.22 & 0.61 & 0.03 & 0.33 & -0.31 \\ -0.26 & 0.03 & 0.42 & -0.35 & 0.29 \\ 1.61 & 0.33 & -0.35 & 1.98 & -1.68 \\ -1.58 & -0.31 & 0.29 & -1.68 & 2.50 \end{pmatrix},$$

$$\begin{pmatrix}
3.33 & 0.60 & 2.09 & -2.03 & 1.91 & -1.21 & 1.94 & -0.25 & 0.12 & -1.11 \\
0.60 & 0.87 & 0.36 & -0.39 & 0.43 & -0.17 & 0.33 & -0.08 & 0.12 & -0.27 \\
2.09 & 0.36 & 2.40 & -1.59 & 1.59 & -0.85 & 1.53 & -0.26 & 0.30 & -0.97 \\
-2.03 & -0.39 & -1.59 & 2.32 & -1.54 & 1.04 & -1.49 & 0.29 & -0.20 & 1.00 \\
1.91 & 0.43 & 1.59 & -1.54 & 2.18 & -0.86 & 1.57 & -0.25 & 0.10 & -0.93 \\
-1.21 & -0.17 & -0.85 & 1.04 & -0.86 & 1.22 & -0.83 & 0.07 & -0.09 & 0.47 \\
1.94 & 0.33 & 1.53 & -1.49 & 1.57 & -0.83 & 2.24 & -0.11 & 0.14 & -0.87 \\
-0.25 & -0.08 & -0.26 & 0.29 & -0.25 & 0.07 & -0.11 & 0.71 & -0.01 & 0.22 \\
0.12 & 0.12 & 0.30 & -0.20 & 0.10 & -0.09 & 0.14 & -0.01 & 0.71 & -0.04 \\
-1.11 & -0.27 & -0.97 & 1.00 & -0.93 & 0.47 & -0.87 & 0.22 & -0.04 & 1.30
\end{pmatrix}$$

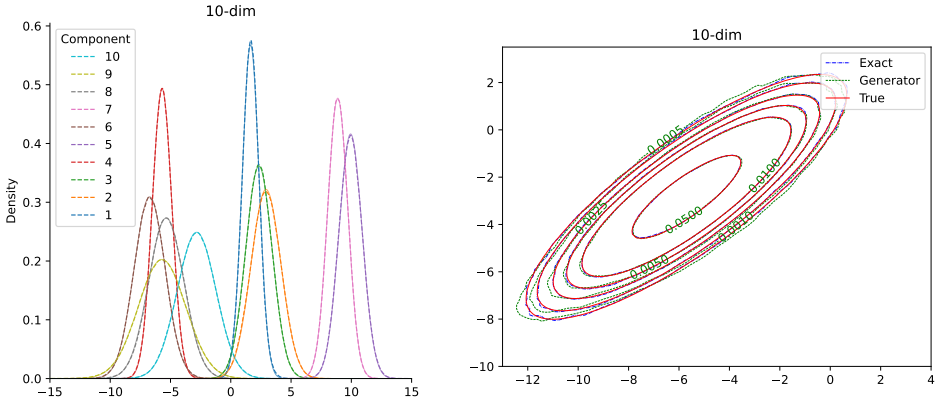
according to the appropriate dimension.

For both sets of parameters, the location parameters were randomly drawn according to the uniform distribution on $[-10, 10]$, whereas the positive definite dependence parameters were generated by *spd_matrix* function from sklearn.

C Additional simulation results

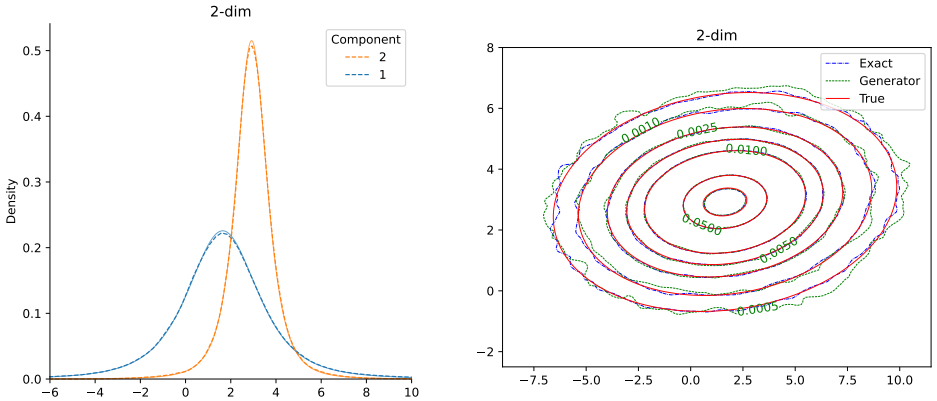


(a) Gaussian distribution 5-dim



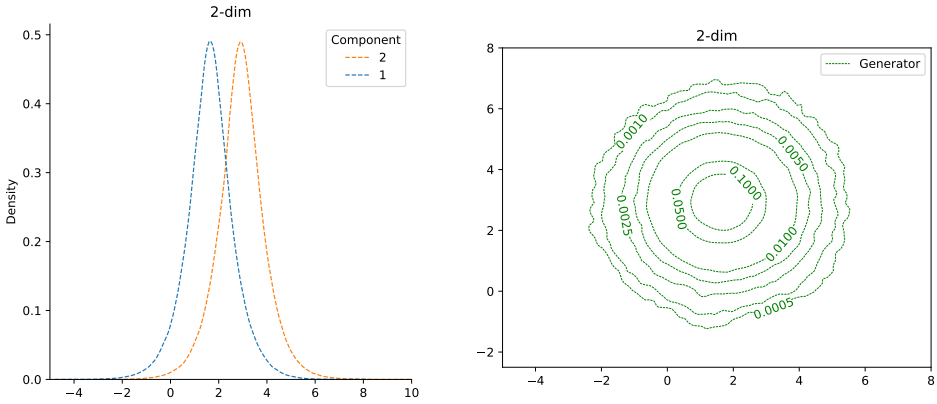
(b) Gaussian distribution 10-dim

Figure 5: Estimated marginal (left) and bivariate (right) densities of the Gaussian distribution in dimensions $5, 10$. The bivariate densities correspond to the last two components of the corresponding random vector. The solid lines show the true densities and the (dash-)dotted lines correspond to estimated densities from the generator and the exact simulation algorithm (right,blue).

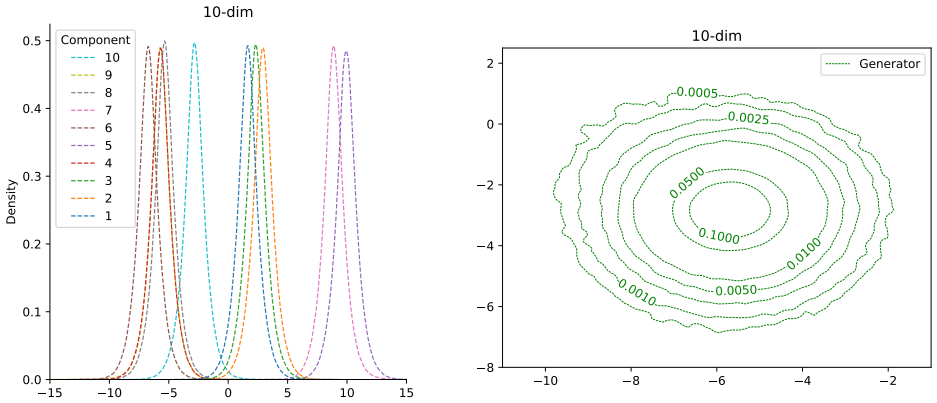


(a) t-distribution 2-dim

Figure 6: Estimated marginal (left) and bivariate (right) densities of t-distribution in dimensions 2. The bivariate densities correspond to the last two components of the corresponding random vector. The solid lines show the true densities and the (dash-)dotted lines correspond to estimated densities from the generator and the exact simulation algorithm (right,blue).

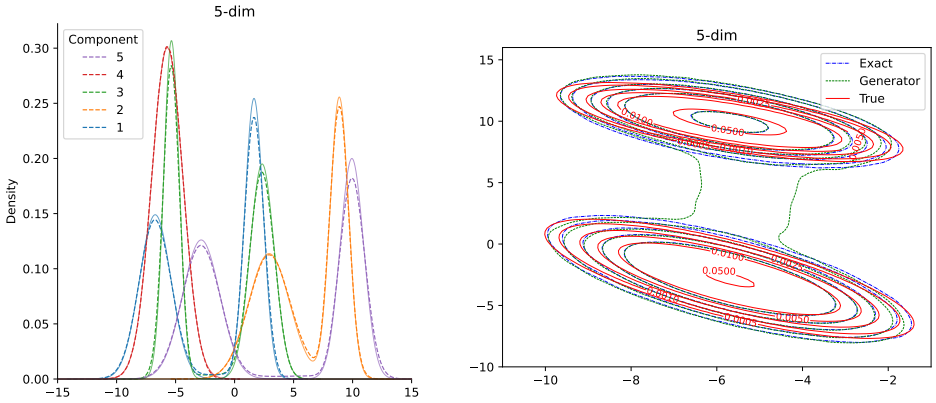


(a) Distribution corr. to rational quadratic kernel 2-dim



(b) Distribution corr. to rational quadratic kernel 10-dim

Figure 7: Estimated marginal (left) and bivariate (right) densities of the distribution corresponding to the rational quadratic kernel in dimensions $2, 10$. The bivariate densities correspond to the last two components of the corresponding random vector.



(a) Gaussian mixture distribution 5-dim

Figure 8: Estimated marginal (left) and bivariate (right) densities of the Gaussian mixture distribution in dimensions 5. The bivariate densities correspond to the last two components of the corresponding random vector. The solid lines show the true densities and the (dash-)dotted lines correspond to estimated densities from the generator and the exact simulation algorithm (right,blue).