Sampling from the latent space in Autoencoders: A simple way towards generative models?

Anonymous authors
Paper under double-blind review

Abstract

By sampling from the latent space of an autoencoder and decoding the latent space samples to the original data space, any autoencoder can simply be turned into a generative model. For this to work, it is necessary to model the autoencoder's latent space with a distribution from which samples can be obtained. Several simple possibilities (kernel density estimates, Gaussian distribution) and more sophisticated ones (Gaussian mixture models, copula models, normalization flows) can be thought of and have been tried recently. This study aims to discuss, assess, and compare various techniques that can be used to capture the latent space so that an autoencoder can become a generative model, while striving for simplicity. Among them, a new copula-based method, the *Empirical Beta Copula Autoencoder*, is considered. Furthermore, we provide insights into further aspects of these methods, such as targeted sampling or synthesizing new data with specific features.

1 Introduction

Generating realistic sample points of various data formats has been of growing interest in recent years. Thus, new algorithms such as *Autoencoders (AEs)* and *Generative Adversarial Networks (GANs)* Goodfellow et al. (2014) have emerged. GANs use a discriminant model, penalizing the creation of unrealistic data from a generator and learning from this feedback. On the other hand, AEs try to find a low-dimensional representation of the high-dimensional input data and reconstruct from it the original data. To turn an AE into a generative model, the latent low-dimensional distribution is modeled, samples are drawn, and thereupon new data points in the original space are constructed with the decoder. Based on that, *Variational Autoencoders (VAEs)* have evolved, optimizing for a Gaussian distribution in the latent space Kingma & Welling (2014). Adversarial autoencoders (AAEs) utilize elements of both types of generative models, where a discriminant model penalizes the distance of the encoded data from a prior (Gaussian) distribution (Makhzani et al., 2016). However, such strong (and simplifying) distributional assumptions as in the VAE or AAE can have a negative impact on performance, leading to a rich literature coping with the challenge of reducing the gap between approximate and true posterior distributions (e.g., Rezende & Mohamed 2015; Tomczak & Welling 2018; Kingma et al. 2016; Gregor et al. 2015; Cremer et al. 2018; Marino et al. 2018; Takahashi et al. 2019). We argue that imposing restrictions on the distribution should be avoided and that more flexible approaches for modeling the latent space seem beneficial.

Recently, Tagasovska et al. 2019 presented the *Vine Copula Autoencoder (VCAE)* to overcome the mentioned problems. Their approach comprises two building blocks, an autoencoder and a vine copula which models the dependence structure in latent space. By that, they were able to create realistic, new images with samples from the fitted vine copula model in the latent space. In this work, we want to elaborate on this idea and compare various methods to model the latent space of an autoencoder to turn it into a generative model. To this end, we analyze, amongst others, the usage of *Gaussian mixture models (GMM)* as done by Ghosh et al. 2020, the vine copula approach by Tagasovska et al. 2019, and simple multivariate *Kernel Density Estimates*. Additionally, we introduce a new, non-parametric copula approach, the *Empirical Beta Copula Autoencoder (EBCAE)*. To assess the ability to turn a standard autoencoder into a powerful generative model, we inspect resulting images, check the models for their ability to generalize and compare additional features. We also check whether these methods may be a simple alternative to more complex models, such as

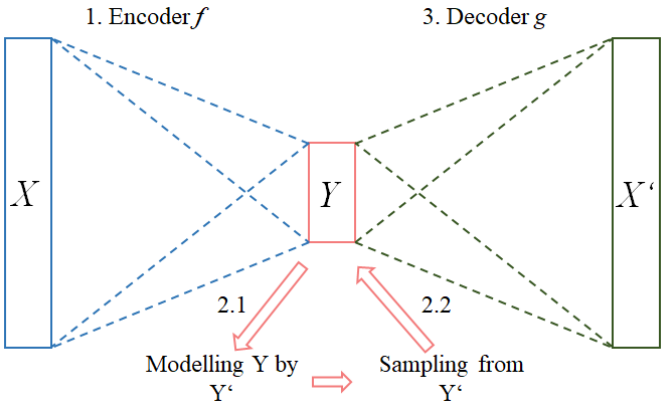

Figure 1: Function scheme of simple generative autoencoders. 1. An encoder f encodes the data X to a low dimensional representation Y. 2.1 Y is modeled by Y', 2.2 Generate new synthetic samples of the latent space by sampling from Y'. 3. Decode the new samples with the decoder g.

normalization flows Rezende & Mohamed (2015) or diffusion models (see, e.g., Rombach et al. 2022; Vahdat et al. 2021). More specifically, we use the well-known Real NVP (Dinh et al., 2017) as an example from these more sophisticated machine learning models but do not elaborate on these in detail. Note that in contrast to other methods (e.g., as proposed by Oring et al. 2021, Berthelot et al. 2019 or van den Oord et al. 2017), the investigated overall approach does not restrict or change the training of the autoencoder in any form. By doing so, we strive for simplicity and take an alternative route to more and more complex models. All models considered in this work are constructed in three steps, visualized in Figure 1. First, an autoencoder, consisting of an encoder f and a decoder g, is trained to find a low-dimensional representation of the data X. Second, the data in the latent space Y is used to learn the best fitting representation Y' of it. This is where the examined models differ from each other by using different methods to model the latent space. Finally, we sample from the learned representation of the latent space and feed the samples into the decoder part of the autoencoder, creating new synthetic data samples.

Generative models are a vivid part of the machine learning literature. For example, new GAN developments Varshney et al. (2021); Karras et al. (2021); Lee et al. (2021); Hudson & Zitnick (2021), developments in the field of autoencoders, Larsen et al. (2016); Yoon et al. (2021); Zhang et al. (2020); Shen et al. (2020) or developments in variational autoencoders Sohn et al. (2015); Havtorn et al. (2021); Masrani et al. (2019); Xu et al. (2019) are emerging. We again want to emphasize that for the models we consider, no prior is needed, nor the optimization approach is changed, i.e., the latent space is modeled after the training of the autoencoder post-hoc. Thus, the presented approach could be transferred to other, more sophisticated, state-of-the-art autoencoders, as hinted in Ghosh et al. 2020. The general idea of creating new data by sampling in the latent space of a generative model has already been used by, e.g., Tagasovska et al. 2019; Dai & Wipf 2019; Brehmer & Cranmer 2020 or Ghosh et al. 2020, but to the best of our knowledge, no analysis and comparison of such methods have been made so far. Closely related, more and more researchers specifically address the latent space of generative models Mishne et al. (2019); Fajtl et al. (2020); Moor et al. (2020); Oring et al. (2021); Hofert et al. (2021) in their work.

This work does not propose a new 'black-box algorithm' for generating data (although we present the new EBCAE) but analyses challenges and possible answers on how autoencoders can be turned into generative models by using well-understood tools of data modeling. We show that this idea generally works with various approaches but that it is hard to find a trade-off between out-of-bound sampling and creating new pictures. We debate further properties of the used methods as targeted sampling and synthesizing images. Our conclusion is intended to point out relevant aspects to the user and discusses the advantages and disadvantages of the models examined.

The remainder of the paper is structured as follows. Section 2 introduces various methods for modeling the latent space. Besides traditional approaches, copula-based methods are introduced. Section 3 describes the

implementation, evaluation, and results of the experiments carried out. In Section 4 we discuss the results and conclude the paper. Last, we provide additional experiments and insides for interested readers in the appendix.

2 Modeling the latent space

In this section, we want to introduce and reflect on different methods to model the latent space in an autoencoder. All methods aim to fit the low-dimensional data Y as best as possible to be able to create new sample points in the latent space, which leads to new realistic images after passing the decoder. We first recap more 'traditional' statistical tools, followed by copulas as an intuitive and flexible tool for modeling high-dimensional data. We briefly explain how each approach can be used to model data in the latent space and how to obtain samples thereof. Note that we do not introduce our benchmark models, namely the standard plain vanilla *VAE* and the *Real NVP*, and refer to the original papers instead (Kingma & Welling, 2014; Dinh et al., 2017).

2.1 Traditional modeling methods

We classify the *multivariate Gaussian distribution*, a *Kernel Density Estimation (KDE)*, and a *Gaussian Mixture Model (GMM)* as traditional modeling methods and give a rather short treatment of each below. They are well known and can be studied in various statistics textbooks such as Hastie et al. 2001 or Bishop 2006.

Multivariate Gaussian

The probably simplest method is to assume the data in the latent space to follow a multivariate Gaussian distribution. Thus, we estimate the covariance matrix $\hat{\Sigma}$ and mean vector $\hat{\mu}$ of Y. In the second step, we draw samples thereof and pass them through the decoder to generate new images. Note that this is similar to the sampling procedure in a VAE, but without forcing the latent space to be Gaussian during training.

GMM

The *Gaussian Mixture Model (GMM)* aims to model the density of the data by mixing M multivariate Gaussian distributions. Thus, the Gaussian mixture model has the form

$$f(x) = \sum_{m=1}^{M} \alpha_m \phi(x; \mu_m, \Sigma_m) \tag{1}$$

where α_m denotes the mixing parameter and ϕ the density of the multivariate normal distribution with mean vector μ_m and covariance matrix Σ_m. The model is usually fit by maximum likelihood using the EM algorithm. By combining several Gaussian distributions, it is more flexible than estimating only one Gaussian distribution as above. A GMM can be seen as some kind of kernel method (Hastie et al., 2001), having a rather wide kernel. In the extreme case, i.e., where m equals the number of points the density is estimated on, a Gaussian distribution with zero variance is centered over each point. Kernel density estimation is introduced in the following.

KDE

Kernel Density Estimation is a well-known non-parametric tool for density estimation. Put simply, a KDE places a density around each data point. The estimated density is constructed by

$$f(x) = \frac{1}{N\lambda} \sum_{i=1}^{N} K_\lambda(x_0, x_i) \tag{2}$$

with N being the total number of data points, λ the bandwidth, and K the used kernel. The kernel density estimation can be performed in univariate data as well as in multivariate data. Note that the choice

of bandwidth and kernel can affect the resulting estimated density. In this work, we rely on the most commonly used kernel, the Gaussian Kernel, and a bandwidth fitted via *Silverman's rule of thumb* (Silverman, 1986) for the univariate KDEs, while we use a grid search with 10-fold cross-validation in the multivariate case.

We use kernel density estimation in multiple fashions. First, we use a multivariate KDE to model the density of the data in the latent space itself. In the case of a Gaussian kernel, it can be written by

$$f(x) = \frac{1}{N\sqrt{\Sigma}2\pi} \sum_{i=1}^{N} e^{-1/2(x-x_i)'\Sigma^{-1}(x-x_i)} \tag{3}$$

where Σ represents the covariance matrix of the kernel, i.e., the matrix of bandwidths. Second, we ignore the dependence structure between margins and estimate the univariate densities of each dimension in the latent space by a KDE. In this way, we are able to find out whether explicitly modeling the dependence structure is necessary or not. We call that approach the *Independent modeling approach*. Last, we use univariate KDEs for modeling the marginal distributions of each dimension in the latent space and use them in the copula models described below.

2.2 Copula based models

In the following, we first introduce copulas as a tool for high-dimensional data, which allows us to model the latent space in our application. Then, we focus on the two copula-based methods to model the latent space of the autoencoder: the *vine copula* and the *empirical beta copula* approach. For detailed introductions to copulas, we refer the reader to Nelsen 2006; Joe 2014; Durante & Sempi 2015.

Copulas have been subject to an increasing interest in the *Machine Learning* community over the last decades, see, e.g., Dimitriev & Zhou 2021; Janke et al. 2021; Messoudi et al. 2021; Ma et al. 2021; Letizia & Tonello 2020; Liu 2019; Kulkarni et al. 2018; Tran et al. 2015. In a nutshell, copula theory enables us to decompose any d-variate distribution function into d marginal univariate distributions and their joint dependence structure, given by the copula function. Thus, copulas "couple" multiple univariate distributions into one joint multivariate distribution. More formally, a d-variate copula $C : [0,1]^d \to [0,1]$ is a d-dimensional joint distribution function whose margins are uniformly distributed on the unit interval. Decomposing and coupling distributions with copulas is formalized in Theorem 2.1 going back to Sklar 1959.

Theorem 2.1 (Sklar 1959)**.** *Consider a d-dimensional vector of random variables $\boldsymbol{Y_i} = (Y_{i,1}, \dots, Y_{i,d})$ with joint distribution function $F_{\boldsymbol{Y}}(y_i) = P(Y_1 \leq y_{i,1}, \dots, Y_d \leq y_{i,d})$ for $i = 1, \dots, n$. The marginal distribution functions F_j are defined by $F_j(y_{i,j}) = P(Y_j \leq y_{i,j})$ for $y_{i,j} \in \mathbb{R}$, $i = 1, \dots, n$ and $j = 1, \dots, d$. Then, there exists a copula C, such that*

$$F_{\boldsymbol{Y}}(y_1, .., y_d) = C(F_1(y_1), \dots, F_d(y_d))$$

for $(y_1, \dots, y_d) \in \mathbb{R}^d$. Vice versa, using any copula $\tilde{C}$, it follows that $\tilde{F}_{\boldsymbol{Y}}(y_1, .., y_d) := \tilde{C}(F_1(y_1), \dots, F_d(y_d))$ is a proper multivariate distribution function.

This allows us to construct multivariate distributions with the same dependence structure but different margins or multivariate distributions with the same margins but different couplings/pairings, i.e., dependence structures. The simplest estimator is given by the empirical copula. It can be estimated directly on the ranks of each marginal distribution by

$$\hat{C}(\mathbf{u}) = \frac{1}{n} \sum_{i=1}^{n} \prod_{j=1}^{d} \mathbf{1}\left\{ \frac{r_{i,j}^{(n)}}{n} \leq u_j \right\} \tag{4}$$

with $\mathbf{u} = (u_1, \dots, u_d) \in [0,1]^d$ and $r_{i,j}^{(n)}$ denoting the rank of each $y_{i,j}$ within $(y_{1,j}, \dots, y_{n,j})$, i.e.,

$$r_{i,j}^{(n)} = \sum_{k=1}^{n} \mathbf{1}\{y_{k,j} \leq y_{i,j}\}. \tag{5}$$

Note that $\mathbf{u} = (u_1, \ldots, u_d)$ represents a quantile level, hence a scaled rank. Simultaneously, the univariate margins can be estimated, e.g., using a KDE as earlier in the paper. Note that it is not possible to draw new samples from the empirical copula directly as no random process is involved.

Vine Copula Autoencoder

Although a variety of two-dimensional copula models exist, the amount of multivariate (parametric) copula models is somewhat limited. *Vine copulas* offer a solution to this problem and decompose the multivariate density as a cascade of bivariate building blocks organized in a hierarchical structure. This decomposition is not unique, and it influences the estimation procedure of the model. Here, we use *regular-vine (r-vine)* models Czado (2019); Joe (2014) to model the 10, 20 and 100 dimensional latent space of the autoencoders at hand. An r-vine is built of a sequence of linked trees $T_i = (V_i, E_i)$, with nodes V_i and edges E_i for $i = 1, \ldots, d-1$ and follows distinct construction rules which we present in Appendix A.

The d-dimensional copula density can then be written as the product of its bivariate building blocks:

$$c(u_1, \ldots, u_d) = \prod_{i=1}^{d-1} \prod_{e \in E_i} c_{a_e b_e; D_e}(u_{a_e|D_e}, u_{b_e|D_e}) \tag{6}$$

with conditioning set D_e and conditional probabilities, e.g., $u_{a_e|D_e} = \mathbb{P}(U_{a_e} \leq u_{a_e}|D_e)$. For each copula, encoding the dependence of two conditional variables, any bivariate copula model, including non-parametric modeling approaches (as done by Tagasovska et al. 2019) can be chosen. However, the construction and estimation of vine copulas is rather complicated. Hence, assuming independence for seemingly unimportant building blocks, so-called truncation, is regularly applied. Because of this, truncated vine copula models do not capture the complete dependence structure of the data, and their usage is not underpinned by asymptotic theory. We refer to Czado (2019); Czado & Nagler (2022); Aas (2016) for reviews of vine copula models.

Empirical Beta Copula Autoencoder

The *empirical beta copula* (Segers et al., 2017) avoids choosing a single, parametric multivariate copula model due to its non-parametric nature. Further, it offers an easy way to model the full, non-truncated multivariate distribution based on the univariate ranks of the joint distribution and, thus, seems to be a reasonable choice to model the latent space. The empirical beta copula is closely related to the empirical copula (see Formula 5) and is a crucial element of the Empirical-Beta-Copula Autoencoder. It is solely based on the ranks $r_{i,j}^{(n)}$ of the original data $\mathbf{Y}$ and can be interpreted as a continuous counterpart of the empirical copula. It is defined by

$$C^{\beta} = \frac{1}{n} \sum_{i=1}^{n} \prod_{j=1}^{d} F_{n, r_{i,j}^{(n)}}(u_j) \tag{7}$$

for $\mathbf{u} = (u_1, \ldots, u_d) \in [0,1]^d$, where

$$F_{n, r_{i,j}^{(n)}}(u_j) = P(U_{(r_{i,j}^{(n)})} \leq u_j) \tag{8}$$

$$= \sum_{p=r_{i,j}^{(n)}}^{n} \binom{n}{p} u_j^p (1-u_j)^{(n-p)} \tag{9}$$

is the cumulative distribution function of a *beta distribution*, i.e., $\mathbb{B}(r_{i,j}^{(n)}, r_{i,j}^{(n)} + n - 1)$. As $r_{i,j}$ is the rank of the i^{th} element in dimension j, $U_{(r_{i,j})}$ represents the ${r_{i,j}}^{\text{th}}$ order statistic of n i.i.d. uniformly distributed random variables on $[0,1]$. For example, if the rank of the i^{th} element in dimension j is 5, $U_{(r_{i,j})} = U_{(5)}$ denotes the 5^{th} order statistic on n i.i.d. uniformly distributed random variables.

The intuition behind the empirical beta copula is as follows: Recall that the marginal distributions of a copula are uniformly distributed on $[0,1]$ and, hence, the k^{th} smallest value of scaled ranks $r_{i,j}^{(n)}/n$ corresponds to the

k^{th} order statistic $U_{(k)}$. Such order statistics are known to follow a *beta distribution* $\mathbb{B}(k, k+n-1)$ (David & Nagaraja, 2003). Consequently, the mathematical idea of the empirical beta copula is to replace each indicator function of the empirical copula with the cumulative distribution function of the corresponding rank $r_{i,j}^{(n)}$.

We argue that the empirical beta copula can be seen as the naturally extended version of the empirical copula, thus, it seems to be a good choice for dependence modeling. Segers et al. 2017 further demonstrate that the empirical beta copula outperforms the empirical copula both in terms of bias and variance. A theorem stating the asymptotic behavior of the empirical copula is given in Appendix B.

Synthetic samples in the latent space y' are created by reversing the modeling path. First, random samples from the copula model $\mathbf{u} = (u_1, \ldots, u_d)$ are drawn. Then, the copula samples are transformed back to the natural scale of the data by the inverse probability integral transform of the marginal distributions, i.e., $y'_j = \hat{F}_j(u_j)$, where $\hat{F}_j$ is the estimated marginal distribution and u_j the jth element of the copula sample for $j \in \{1, \ldots, d\}$. Algorithm 1 summarizes the procedure.

Algorithm 1: Sampling from Empirical Beta Copula

Input: Sample $Y \subset \mathbb{R}^{n \times d}$, new sample size m
begin
  Compute rank matrix $R^{n \times d}$ out of Y
  Estimate marginals of Y with KDE, $\widehat{f}_1(y_1), \ldots, \widehat{f}_d(y_d)$.
  for $i \leq m$ **do**
    Draw random from $I \in [1, \ldots, n]$
    for $j \leq d$ **do**
      Draw $u_{I,j} \sim \mathbb{B}(R_{Ij}, n + 1 - R_{Ij})$
    Set $u_i = (u_{I1}, \ldots, u_{Id})$
    Rescale margins by $Y_i = \widehat{F}_1^{-1}(u_{i1}), \ldots, \widehat{F}_d^{-1}(u_{id})$.
Output: New sample Y' of size m

3 Experiments

In this section, we present the results of our experiments. We use the same architecture for the autoencoder in all experiments for one dataset but replace the modeling technique for the latent space for all algorithms. The architecture, as well as implementation details, are given in Appendix C. We further include a standard VAE and the Real NVP normalization flow approach modeling the latent space in our experiments to serve as a benchmark.

Methodology

We train an autoencoder consisting of two neural nets, an *encoder* f, and a *decoder* g. The encoder f maps data X from the original space to a lower-dimensional space, while the decoder g reconstructs this low-dimensional data Y from the low-dimensional latent space to the original space (see Fig. 1). We train both neural nets in a way that the reconstruction loss is minimized, i.e., that the reconstructed data $X' = g(f(X))$ is as similar to the original data X as possible. In the second step, we model the latent space Y data with a multivariate Gaussian distribution, a Gaussian mixture model, Kernel density estimates, the two presented copula methods and the Real NVP. Thus, we fit models with different flexibility and complexity while keeping the training process of the autoencoder untouched. Last, new samples are generated by decoding random samples from the learned model in the latent space. Note that such an approach is only reasonable when the underlying autoencoder has learned a relevant and interesting representation of the data and the latent space is smooth. We demonstrate this in Appendix D.

(a) MNIST samples

(b) CelebA samples

Figure 2: Comparison of synthetic samples of different Autoencoder models. **1**st **row:** Fitted normal distribution, **2**nd **row:** Independent margins, **3**rd **row:** KDE-AE, **4**th **row:** GMM, **5**th **row:** VCAE, **6**th **row:** EBCAE, **7**th **row:** VAE, **8**th **row:** Real NVP, **Last row:** original pictures.

Datasets

We conduct experiments on one small-scale, one medium, and one large-scale dataset. The small-scale *MNIST* dataset (LeCun et al., 2010) includes binary images of digits, while the medium-scale *SVHN* dataset (Netzer et al., 2011) contains images of house numbers in Google Street View pictures. The large-scale *CelebA* dataset (Liu et al., 2015) consists of celebrity images covering 40 different face attributes. We split data into a train set and a test set of 2000 samples which is a commonly used size for evaluation (Tagasovska et al., 2019; Xu et al., 2018). Note that the data sets cover different dimensionalities in the latent space, allowing for a throughout assessment of the methods under investigation.

Evaluation

Evaluation of results is performed in several ways. First, we visually compare random pictures generated by the models. Second, we evaluate the results with the framework proposed by Xu et al. 2018, since a log-likelihood evaluation is known to be incapable of assessing the quality (Theis et al., 2016) and unsuitable for non-parametric models. Based on their results, we choose five metrics in our experiments: The *earth mover distance (EMD)*, also known as *Wasserstein distance* (Vallender, 1974); the *mean maximum discrepancy (MMD)* (Gretton et al., 2007); the *1-nearest neighbor-based two-sample test (1NN)*, a special case of the classifier two-sample test (Lopez-Paz & Oquab, 2017); the *Inception Score* (Salimans et al., 2016); and the *Frêchet inception distance* (Heusel et al., 2017). In line with Tagasovska et al. 2019 and as proposed by Xu et al. 2018, we further apply the EMD, MMD, and 1NN over feature mappings in the convolution space over ResNet-34 features. For all metrics except the Inception Score, lower values are preferred. For more details on the metrics, we refer to Xu et al. 2018. Next, we evaluate the ability to generate new, realistic pictures by the different latent space modeling techniques. Therefore, we compare new samples with their nearest neighbor in the latent space stemming from the original data. This shows us whether the learned distribution covers the whole latent space, or stays too close to known examples, i.e., the model does not generalize enough. Finally, we compare other features of the tested models, such as their ability of targeted sampling and of recombining attributes.

Results

Figure 2a and Figure 2b show images generated from each method for MNIST and CelebA. The GMM model is composed of 10 elements, and the KDE is constructed using a Gaussian kernel with a bandwidth fitted via a grid search and 10-fold cross-validation. The specification of the Real NVPs are given in the Appendix.

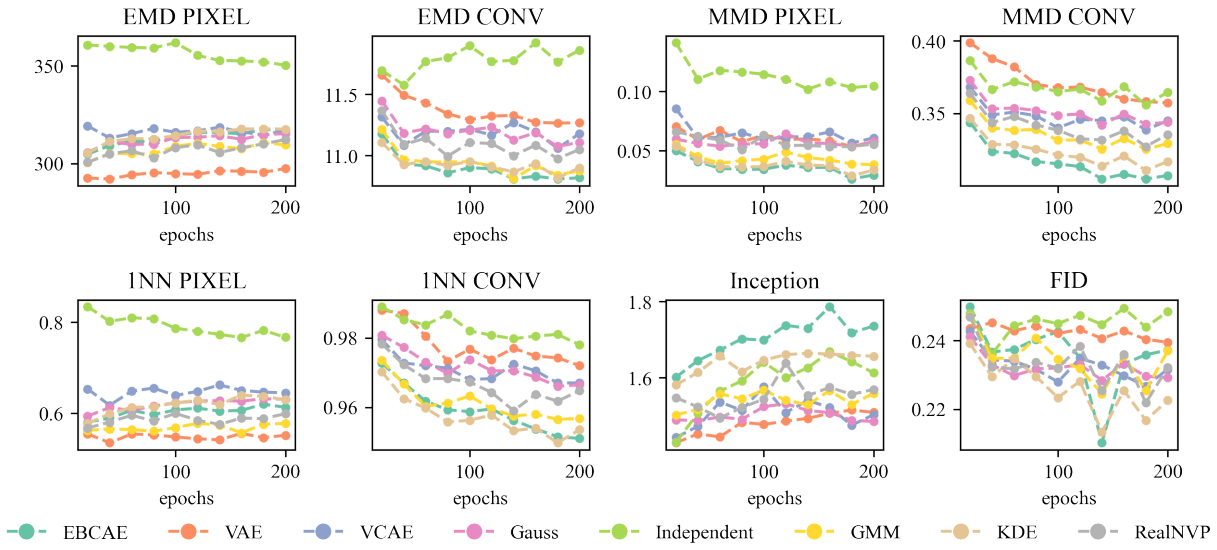

Figure 3: Performance metrics of generative models on **CelebA**, reported over epochs computed from 2000 random samples. Note that they only differ in the latent space sampling and share the same autoencoder.

For the MNIST dataset, we observe the best results for the EBCAE (row 6) and KDE (row 3), while the other methods seem to struggle a bit. For the CelebA, our visual observations are slightly different. All methods produce images that are clearly recognizable as faces. However, the Gaussian samples in row 1 and independent margins in row 2 create pictures with some unrealistic artefacts, blurry backgrounds, or odd colors. This is also the case for the GMM in row 4 and VCAE in row 5, but less severe. We believe that this comes from samples of an empty area in the latent space, i.e., where none of the original input pictures were projected to. In contrast to that, the samples in the latent space of the KDE, EBCAE, and Real NVP stay within these natural bounds, producing good results after passing the decoder (rows 3, 6, 8). Recall that all methods use the same autoencoder and only differ by means of sampling in the latent space. From our observations, we also conclude that the autoencoder for the CelebA dataset is less sensitive toward modeling errors in the latent space since all pictures are clearly recognizable as faces. In contrast, for the MNIST dataset, not all images clearly show numbers. Similar results for SVHN are presented in the Appendix.

The numerical results computed from 2000 random samples displayed in Figure 3 prove that dependence truly matters within the latent space. Simultaneously, the KDE, GMM, and EBCAE perform consistently well over all metrics, delivering comparable results to the more complex Real NVP. Especially the EBCAE outperforms the other methods, whereas the VCAE, Gauss model, and VAE usually cluster in the middle.

We further report results over the number of samples in the latent space in Figure 9 in the Appendix. This, at first sight, unusual perspective visualizes the capability to reach good performance even for small sample sizes in latent space. In a small-sample regime, it is crucial to assess how fast a method adapts to data in the latent space and models it correctly. We see that all methods perform well for small sample sizes, i.e., $n = 200$. Similar experiments for MNIST and SVHN can be found in Appendix E.

Next, we evaluate the different modeling techniques in their ability to generate new, realistic images. For this, we focus on pictures from the CelebA dataset in Figure 4. First, we create new, random samples with the respective method (top row) and then compare these with their decoded nearest neighbor in the latent space (middle row). The bottom row displays the latent space nearest neighbor in the original data space

before applying the autoencoder. By doing so, we are able to disentangle two effects. First, the effect from purely encoding-decoding an image and, second, the effect of modeling the latent space. Thus, we can check whether new images are significantly different from the input, i.e., whether the distribution modeling the latent space merely reproduces images or generalizes to some extent.

We observe that the samples from GMM, VCAE and the Real NVP substantially differ from their nearest neighbors. However, again they sometimes exhibit unrealistic colors and blurry backgrounds. The samples created from KDE and EBCAE look much more similar to their nearest neighbors in the latent space, indicating that these methods do not generalize to the extent of the other methods. However, their samples do not include unrealistic colors or features and seem to avoid sampling from areas where no data point of the original data is present. Thus, they stay in 'natural bounds'. Note that this effect apparently is not reflected in the numerical evaluation metrics. We, therefore, recommend that, in addition to a quantitative evaluation, a qualitative evaluation of the resulting images should always be performed.

(a) Gaussian (b) Independent (c) KDE (d) Real NVP

(e) GMM (f) VCAE (g) EBCAE

Figure 4: Resulting examples of the six investigated modeling methods after decoding. **Top row:** New examples. **Middle row:** Nearest neighbor of training data Y in latent space after decoding. **Bottom row:** Original input picture of nearest neighbor in latent space.

To further underpin this point, Figure 5 shows 2-dimensional TSNE-Embeddings (see, e.g.,van der Maaten & Hinton 2008) of the latent space for all six versions of the autoencoder (MNIST). Black points indicate original input data, and colored points are synthetic samples from the corresponding method. We see that the KDE, as well as the EBCAE, stay close to the original space. The samples from the GMM and Real NVP also seem to closely mimic the original data, whereas the other methods fail to do so. This visualization confirms our previous conjecture that some algorithms tend to sample from 'empty' areas in the latent space, leading to unrealistic results.

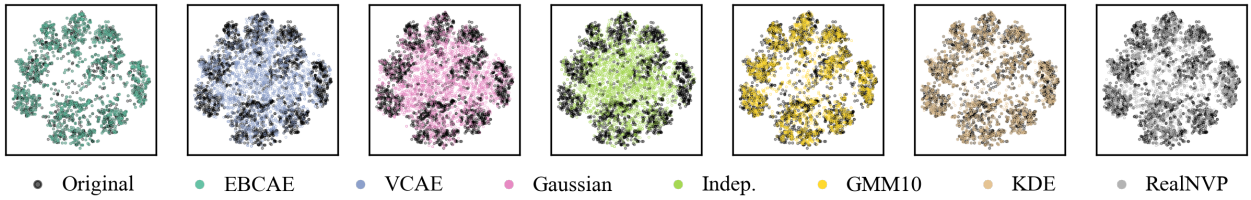

Figure 5: TSNE embeddings of samples in the latent space of the **MNIST** dataset. Points from the original input training data Y are given in black, whereas new, synthetic samples Y' stemming from the different modeling methods are colored.

We also report computing times for learning and sampling of the different models for MNIST and CelebA in Table 1. Unsurprisingly, the more straightforward methods such as Gauss, Independence, KDE, and GMM, exhibit the lowest sampling times. The Real NVP shows the highest learning time as a neural network is fitted. However, we expect the difference to be much smaller once trained on an appropriate GPU. The times also reflect the complexities of the methods in the latent space dimensions.

Table 1: Modeling and sampling time in the **CelebA** and **MNIST** dataset of 2000 artificial samples based on a latent space of size $n = 2000$ in [s].

	CelebA	**CelebA**	**MNIST**	**MNIST**
Method	Learn	Sample	Learn	Sample
Gauss	<0.01	0.01	0.002	0.002
Indep.	4.10	0.07	0.393	0.003
KDE	75.25	0.01	13.958	0.001
GMM	1.35	0.03	0.115	0.004
VCAE	306.97	148.48	10.345	4.590
EBCAE	3.41	59.36	0.328	5.738
Real NVP	2541.19	3.69	341.608	0.477

Last, we discuss other features of the tested methods, such as targeted sampling and recombination. In contrast to the other techniques, the KDE and EBCAE allow for targeted sampling. Thus, we can generate new images with any desired characteristic directly, e.g., only ones in a data set of images of numbers. In the case of the KDE, this simply works by sampling from the estimated density of the corresponding sub-group. In the case of the EBCAE, we randomly choose among rows in the rank matrix of original samples that share the desired specific attribute. Other approaches are also possible, however, they need further tweaks to the model, training, or sampling as the *conditional variational autoencoder* (Sohn et al., 2015).

The second feature we discuss is recombination. By using copula-based models (VCAE and EBCAE), we can facilitate the decomposition idea and split the latent space in its dependence structure and margins, i.e., we combine the dependence structure of images with a specific attribute with the marginal distributions of images with different attributes. Therefore, copula-based methods allow controlling the attributes of created samples to some extent. Our experiments suggest that the dependence structure provides the basic properties of an image, while the marginal distributions are responsible for details (see, e.g., Figure 6). However, we want to point out that it is not generally clear what information is embedded in the dependence structure and what information is in the marginal distributions of latent space. This might also depend on the autoencoder and the dataset at hand. Thad said, using such a decomposition enables higher flexibility and hopefully fuels new methodological developments in this field.

Figure 6: **Top row:** Samples from the EBCAE with dependence structure in latent space from samples with glasses, marginal distributions in latent space from samples without glasses. **Middle row:** Nearest neighbor of training data Y in latent space after decoding. **Bottom row:** Original input picture of nearest neighbor in latent space.

4 Discussion

In this section, we want to discuss the results of our experiments and want to express some further thoughts. So, is sampling from the latent space a simple way towards generative models? We observed that sampling from the latent space is indeed a viable approach to turn an autoencoder into a generative model. We also observe that the studied methods can achieve competitive results comparable to more complex approaches like the Real NVP and hence may be interesting for future research or application in more advanced autoencoders. Simultaneously, each modeling approach in this setting comes with its own restrictions, advantages, and problems.

We witness a trade-off between the ability to generalize, i.e., to create genuinely new pictures, and sample quality, i.e., to avoid unrealistic colors or artefacts. In cases where new data points are sampled in the neighborhood to existing points (as in the KDE or EBCAE), the newly generated data stays in somehow natural bounds and provides realistic, but not completely new, decoded samples. On the other hand, modeling the latent space too generically leads to bad-quality images. We believe this is similar to leaving the feasible set of an optimization problem or sampling from a wrong prior. While being close to actual points of the original latent space, new samples stay within the feasible set. By moving away from these points, the risk of sampling from an unfeasible region and thus creating unrealistic new samples increases. Recombination via a copula-based approach of marginal distributions and dependence structures offers the possibility to detect new feasible regions in the latent space for the creation of realistic images. Also, interpolating by building convex combinations of two points in the latent space seems reasonable. However, without further restrictions during training (see, e.g., discussion in Ghosh et al. 2020), we cannot principally guarantee proper interpolation results. Further, we observe that the mentioned trade-off is not reflected by the performance metrics. Therefore, we strongly recommend not only checking quantitative results but also finding and analyzing the nearest neighbor in the original data to detect the pure reproduction of pictures. This also reveals that the development of further evaluation metrics could be beneficial.

A closely related issue is the choice of a parametric vs. a non-parametric modeling method in the latent space. Parametric methods can place probability mass in the latent space, where no data point of the original input data was observed. Thus, parametric methods are able to generate (truly) new data, subject to their assumption. However, if the parametric assumption is wrong, the model creates samples from 'forbidden' areas in the latent space leading to unrealistic images. In spite of this, carefully chosen parametric models can be beneficial, and even a log-likelihood is computable and traceable (although we do not use it for training). Non-parametric methods avoid this human decision and possible source of error completely but are closely bound to the empirical distribution of the given input data. Consequently, such methods can miss important areas of the latent space but create more realistic images. Furthermore, adjusting parameters of the non-parametric models, such as increasing bandwidths or lowering truncation levels, offer possibilities to slowly overcome these limitations.

Besides the major points above, the EBCAE and KDE offer an easy way of targeted sampling without additional training effort. This can be beneficial for various applications and is not as straightforward with other methods. Lastly, the investigated methods differ in their runtime. While vine copula learning and sampling is very time-intensive for high dimensions, the EBCAE is much faster but still outperformed by the competitors. For the non-copula methods, the GMM is really fast in both datasets while still capturing the dependence structure to some extent. In contrast to that, the Real NVP needs more time for training but is rather quick in generating new samples.

To sum up, we can confirm that there are indeed simple methods to turn a plain autoencoder into a generative model. We conclude that the optimal method to do so depends on the goals of the user. Besides runtime considerations, the specific application of the autoencoder matters. For example, if one is interested in targeted sampling, EBCAE or KDE should be applied. Recombination experiments call for a copula-based approach, whereas in all cases, the trade-off between generalization and out-of-bound sampling should be considered. Lastly, during our research, we found that future, more theory-driven work most likely could establish the structural link between copulas and normalization flows via the Rosenblatt transformation Rosenblatt (1952).

References

Kjersti Aas. Pair-copula constructions for financial applications: A review. *Econometrics*, 4(4), 2016. ISSN 2225-1146.

Tim Bedford and Roger M. Cooke. Vines: A new graphical model for dependent random variables. *The Annals of Statistics*, 30(4):1031–1068, 2002. ISSN 00905364.

David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2019.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 442–453. Curran Associates, Inc., 2020.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders, 2018. URL `https://arxiv.org/abs/1801.03558`.

Claudia Czado. *Analyzing Dependent Data with Vine Copulas: A Practical Guide With R*. Springer International Publishing, 2019.

Claudia Czado and Thomas Nagler. Vine copula based modeling. *Annual Review of Statistics and Its Application*, 9(1):453–477, 2022. doi: 10.1146/annurev-statistics-040220-101153.

Bin Dai and David Wipf. Diagnosing and enhancing vae models, 2019. URL `https://arxiv.org/abs/1903.05789`.

H. A. David and H. N. Nagaraja. *Order Statistics*. John Wiley and Sons, 08 2003. doi: http://dx.doi.org/10.1002/0471722162.

Alek Dimitriev and Mingyuan Zhou. CARMS: Categorical-antithetic-REINFORCE multi-sample gradient estimator. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Fabrizio Durante and Carlo Sempi. *Principles of copula theory*. CRC Press LLC, 01 2015. doi: 10.1201/b18674.

Jiri Fajtl, Vasileios Argyriou, Dorothy Monekosso, and Paolo Remagnino. Latent bernoulli autoencoder. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2964–2974. PMLR, 13–18 Jul 2020.

Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael Black, and Bernhard Scholkopf. From variational to deterministic autoencoders. In *International Conference on Learning Representations*, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1462–1471, Lille, France, 07–09 Jul 2015. PMLR.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. In B. Schölkopf, J. Platt, and T. Hoffman (eds.), *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.

Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020. doi: 10.1038/s41586-020-2649-2.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Jakob D. Drachmann Havtorn, Jes Frellsen, Soren Hauberg, and Lars Maaløe. Hierarchical vaes know what they don't know. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4117–4128. PMLR, 18–24 Jul 2021.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Marius Hofert, Avinash Prasad, and Mu Zhu. Quasi-random sampling for multivariate distributions via generative neural networks. *Journal of Computational and Graphical Statistics*, 30(3):647–670, 2021. doi: 10.1080/10618600.2020.1868302.

Drew A Hudson and Larry Zitnick. Generative adversarial transformers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4487–4499. PMLR, 18–24 Jul 2021.

Tim Janke, Mohamed Ghanmi, and Florian Steinke. Implicit generative copulas. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

H. Joe. *Dependence modeling with copulas*. Chapman & Hall/CRC, 01 2014. doi: 10.1201/b17116.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Vaibhav Kulkarni, Natasa Tagasovska, Thibault Vatter, and Benoit Garbinato. Generative models for simulating mobility trajectories. *ArXiv*, 2018.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1558–1566, New York, New York, USA, 20–22 Jun 2016. PMLR.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Jinhee Lee, Haeri Kim, Youngkyu Hong, and Hye Won Chung. Self-diagnosing GAN: Diagnosing underrepresented samples in generative adversarial networks. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Nunzio A. Letizia and Andrea M. Tonello. Segmented generative networks: Data generation in the uniform probability space. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2020. doi: 10.1109/TNNLS.2020.3042380.

Weiwei Liu. Copula multi-label learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Jiaqi Ma, Bo Chang, Xuefei Zhang, and Qiaozhu Mei. CopulaGNN: Towards integrating representational and correlational roles of graphs in graph neural networks. In *International Conference on Learning Representations*, 2021.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders, 2016.

Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3403–3412. PMLR, 10–15 Jul 2018.

Vaden Masrani, Tuan Anh Le, and Frank Wood. The thermodynamic variational objective. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Soundouss Messoudi, Sébastien Destercke, and Sylvain Rousseau. Copula-based conformal prediction for multi-target regression. *Pattern Recognit.*, 120:108101, 2021.

Gal Mishne, Uri Shaham, Alexander Cloninger, and Israel Cohen. Diffusion nets. *Applied and Computational Harmonic Analysis*, 47(2):259–285, 2019. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2017.08.007.

Michael Moor, Max Horn, Bastian Rieck, and Karsten Borgwardt. Topological autoencoders. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7045–7054. PMLR, 13–18 Jul 2020.

Roger B. Nelsen. *An Introduction to Copulas*. Springer Science+Business Media, Inc., 2006. ISBN 0-387-28659-4.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL `http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf`.

Alon Oring, Zohar Yakhini, and Yacov Hel-Or. Autoencoder image interpolation by shaping the latent space. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8281–8290. PMLR, 18–24 Jul 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and BjÃ¶rn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Murray Rosenblatt. Remarks on a Multivariate Transformation. *The Annals of Mathematical Statistics*, 23 (3):470 – 472, 1952. doi: 10.1214/aoms/1177729394.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Johan Segers, Masaaki Sibuya, and Hideatsu Tsukahara. The empirical beta copula. *Journal of Multivariate Analysis*, 155:35–51, 2017. ISSN 0047-259X. doi: doi.org/10.1016/j.jmva.2016.11.010.

Tianxiao Shen, Jonas Mueller, Regina Barzilay, and Tommi S. Jaakkola. Educating text autoencoders: Latent representation guidance via denoising. In *Proceedings of The thirty-seventh International Conference on Machine Learning*, pp. 8719–8729, 2020.

B. W. Silverman. *Density estimation for statistics and data analysis / B.W. Silverman.* Chapman and Hall London ; New York, 1986. ISBN 0412246201.

Abe Sklar. Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8:229–231, 1959.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Natasa Tagasovska, Damien Ackerer, and Thibault Vatter. Copulas as high-dimensional generative models: Vine copula autoencoders. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi. Variational autoencoder with implicit optimal priors. AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33015066.

L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, Apr 2016.

Jakub Tomczak and Max Welling. Vae with a vampprior. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1214–1223. PMLR, 09–11 Apr 2018.

Dustin Tran, David Blei, and Edo M Airoldi. Copula variational inference. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In *Neural Information Processing Systems (NeurIPS)*, 2021.

S. S. Vallender. Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability and Its Applications*, 18:435–435, 1974.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6309–6318, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Sakshi Varshney, Vinay Kumar Verma, Srijith P K, Lawrence Carin, and Piyush Rai. CAM-GAN: Continual adaptation modules for generative adversarial networks. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Haowen Xu, Wenxiao Chen, Jinlin Lai, Zhihan Li, Youjian Zhao, and Dan Pei. On the necessity and effectiveness of learning the prior of variational auto-encoder, 2019. URL `https://arxiv.org/abs/1905.13452`.

Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Q. Weinberger. An empirical study on evaluation metrics of generative adversarial networks. *ArXiv*, abs/1806.07755, 2018.

Sangwoong Yoon, Yung-Kyun Noh, and Frank Park. Autoencoding under normalization constraints. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12087–12097. PMLR, 18–24 Jul 2021.

Zijun Zhang, Ruixiang Zhang, Zongpeng Li, Yoshua Bengio, and Liam Paull. Perceptual generative autoencoders. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11298–11306. PMLR, 13–18 Jul 2020.

Appendix

A Details on the Vine Copula

In the vine copula autoencoder Tagasovska et al. (2019) use *regular-vine (r-vines)*. A r-vine is built of a sequence of linked trees $T_i = (V_i, E_i)$, with nodes V_i and edges E_i for $i = 1, \ldots, d-1$. A $d-$dimensional vine tree structure $V = (T_1, ..., T_{d-1})$ is a sequence of $T-1$ trees if (see Czado 2019):

1. Each tree $T_j = (N_i, E_i)$ is connected, i.e. for all nodes $a, b \in T_i, i = 1, ..., d-1$, there exists a path $n_1, ..., n_k \subset N_j$ with $a = n_1, b = n_k$.
2. T_1 is a tree with node set $N_1 = \{1, ..., d\}$ and edge set E_1.
3. For $i \geq 2$, T_j is a tree with node set $N_i = E_{i-1}$ and edge set E_i .
4. For $i = 2, ..., d-1$ and $\{a, b\} \in E_i$ it must hold that $|a \cap b| = 1$.

An example of a five-dimensional vine tree structure is given below in Figure 7. Note that the structure has to be estimated and multiple structures are possible. For details on vine copula estimation, see Czado (2019); Joe (2014); Bedford & Cooke (2002).

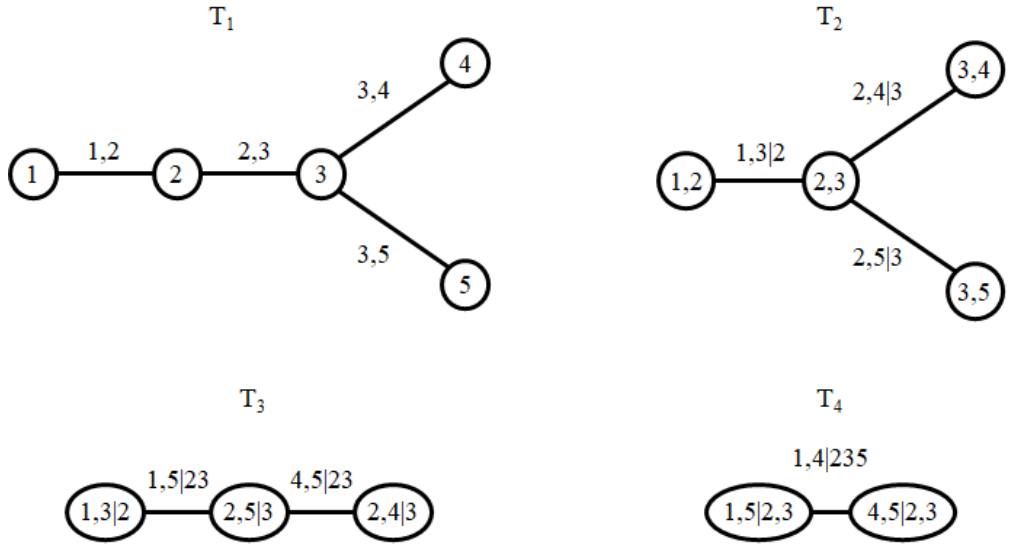

Figure 7: Example of a vine copula tree structure $T_1 - T_4$ for five dimensions.

B Asymptotics of the Empirical Beta Copula

Theorem B.1 gives the asymptotic behavior of the empirical beta copula.

Theorem B.1 (Asymptotics of the empirical beta copula)**.** *Let the copula C have continuous first-order partial derivatives $\dot{C}_j = \delta C(\mathbf{u})/\delta u_j$ for each $j \in \{1, \ldots, d\}$ on the set $I_j = \{\mathbf{u} \in [0,1]^d : 0 < u_j < 1\}$. The corresponding empirical copula is denoted as $\mathbb{C}_n$, with empirical copula process $\mathbb{G}_n = \sqrt{n}\Big(\mathbb{C}_n(\mathbf{u}) - C(\mathbf{u})\Big)$ and empirical beta copula $\mathbb{C}_n^\beta$ with empirical beta copula process $\mathbb{G}_n^\beta = \sqrt{n}\Big(\mathbb{C}_n^\beta(\mathbf{u}) - C(\mathbf{u})\Big)$. Suppose $\mathbb{G}_n \rightsquigarrow \mathbb{G}$ for $n \to \infty$ to $\mathbb{G}$ in $l^\infty([0,1]^d)$, where $\mathbb{G}$ is a limiting process having continuous trajectories almost surely. Then, in $l^\infty([0,1]^d)$*

$$\mathbb{G}_n^\beta = \mathbb{G}_n + o_p(1), n \longrightarrow \infty.$$

Proof. See Segers et al. 2017 Section 3. □

In short, Theorem B.1 states that the empirical beta copula has the same large-sample distribution as the empirical copula and, thus, converges to the true copula. However, the empirical beta copula performs better for small samples. Segers et al. 2017 demonstrate that the empirical beta copula outperforms the empirical copula both in terms of bias and variance.

C Implementation

C.1 Implementation of the Autoencoder

We implemented the experiments in Python 3.8 Van Rossum & Drake Jr (1995) using `numpy 1.22.0, scipy 1.7.1„ scikit-learn 1.1.0 and pytorch 1.10.1` Harris et al. (2020); Virtanen et al. (2020); Pedregosa et al. (2011); Paszke et al. (2019). The AEs were trained using the Adam optimizer with learning rate 0.001 for MNIST and 0.0005 for SVHN and CelebA. A weight decay of 0.001 was used in all cases. Batch sizes were fixed to 128 (MNIST), 32 (SVHN) and 100 (CelebA) samples for training, while the size of the latent space was set to 10 (MNIST), 20 (SVHN) and 100 (CelebA) according to the data sets size and complexity. Training was executed on a separate train set and evaluated on a hold-out test set of 2000 samples, similar to Tagasovska et al. 2019. For comparison with the VCAE and performance metrics, we have resorted to the implementation from Tagasovska et al. 2019 and Xu et al. 2018. The architectures for all networks are described in Appendix C.2. We trained the autoencoders on an NVIDIA Tesla V100 GPU with 10 Intel Xeon Gold 6248 CPUs. The experiments are executed afterward on a PC with an Intel i7-6600U CPU and 20GB RAM.

C.2 Architectures of Autoencoders and VAE

We use the same architecture for EBCAE, VCAE, and VAE as described below. All models were trained by minimizing the Binary Cross Entropy loss.

MNIST

Encoder:

$$\begin{aligned} x \in R^{32\times 32} &\rightarrow Conv_{32} && \rightarrow BN \rightarrow ReLu \\ &\rightarrow Conv_{64} && \rightarrow BN \rightarrow ReLu \\ &\rightarrow Conv_{128} && \rightarrow BN \rightarrow ReLu \\ & && \rightarrow FC_{10} \end{aligned}$$

Decoder:

$$\begin{aligned} y \in R^{10} \rightarrow FC_{100} &\rightarrow ConvT_{128} && \rightarrow BN \rightarrow ReLu \\ &\rightarrow ConvT_{64} && \rightarrow BN \rightarrow ReLu \\ &\rightarrow ConvT_{128} && \rightarrow BN \rightarrow ReLu \\ & && \rightarrow FC_{1} \end{aligned}$$

For all (de)convolutional layers, we used 4×4 filters, a stride of 2, and a padding of 1. BN denotes batch normalization, $ReLU$ rectified linear units, and FC fully connected layers. Last, $Conv_k$ denotes the convolution with k filters.

SVHN

In contrast to the MNIST dataset, images in SVHN are colored. We do not use any preprocessing in this dataset.

Encoder:

$$\begin{aligned} x \in R^{3\times 32\times 32} &\rightarrow Conv_{64} && \rightarrow BN \rightarrow ReLu \\ &\rightarrow Conv_{128} && \rightarrow BN \rightarrow ReLu \\ &\rightarrow Conv_{256} && \rightarrow BN \rightarrow ReLu \\ & && \rightarrow FC_{100} \rightarrow FC_{20} \end{aligned}$$

Decoder:

$$\begin{aligned} y \in R^{20} \rightarrow FC_{100} \rightarrow ConvT_{256} &\rightarrow BN \rightarrow ReLu \\ \rightarrow ConvT_{128} &\rightarrow BN \rightarrow ReLu \\ \rightarrow ConvT_{64} &\rightarrow BN \rightarrow ReLu \\ \rightarrow ConvT_{32} &\rightarrow BN \rightarrow ReLu \\ &\rightarrow FC_1 \end{aligned}$$

Notations are the same as described above.

CelebA

In contrast to the MNIST dataset, images in CelebA are colored. Further, we first took central crops of 140×140 and resize the images to a resolution 64×64.

Encoder:

$$\begin{aligned} x \in R^{3\times 64\times 64} \rightarrow Conv_{64} &\rightarrow BN \rightarrow LeakyReLu \\ \rightarrow Conv_{128} &\rightarrow BN \rightarrow LeakyReLu \\ \rightarrow Conv_{256} &\rightarrow BN \rightarrow LeakyReLu \\ \rightarrow Conv_{512} &\rightarrow BN \rightarrow LeakyReLu \\ &\rightarrow FC_{100} \rightarrow FC_{100} \end{aligned}$$

Decoder:

$$\begin{aligned} y \in R^{100} \rightarrow FC_{100} \rightarrow Conv_{512} &\rightarrow BN \rightarrow ReLu \\ \rightarrow ConvT_{256} &\rightarrow BN \rightarrow ReLu \\ \rightarrow ConvT_{128} &\rightarrow BN \rightarrow ReLu \\ \rightarrow ConvT_{64} &\rightarrow BN \rightarrow ReLu \\ \rightarrow ConvT_{32} &\rightarrow BN \rightarrow ReLu \\ &\rightarrow FC_1 \end{aligned}$$

LeakyReLU uses a negative slope of 0.2, and padding was set to 0 for the last convolutional layer of the encoder and the first of the decoder. All other notations are the same as described above.

C.3 Implementation of Real NVP

In our study, we used a Real NVP (see Dinh et al. 2017) to model the latent space of the autoencoder and serve as a benchmark. For all data sets, we use spatial checkerboard masking, where the mask has a value of 1 if the sum of coordinates is odd, and 0 otherwise. For the MNIST data set, we use 4 coupling layers with 2 hidden layers each and 256 features per hidden layer. Similarly, for the SVHN data set, we also use four coupling layers with two hidden layers each and 256 hidden layer features. Lastly, for the CelebA data set, we use four coupling layers with two hidden layers each and 1024 hidden layer features. For all data sets, we applied a learning rate of 0.0001 and learn for 2000 epochs.

D Image Interpolation of the Autoencoder

We show that our used autoencoder learned a relevant and smooth representation of the data by interpolation in the latent space and, thus, modeling the latent space for generating new images is reasonable. For example, consider two images A and B with latent variables $y_{A,1}, ..., x_{A,100}$ and $y_{B,1}, ..., y_{B,100}$. We now interpolate linearly in each dimension between these two values and feed the resulting interpolation to the decoder to get

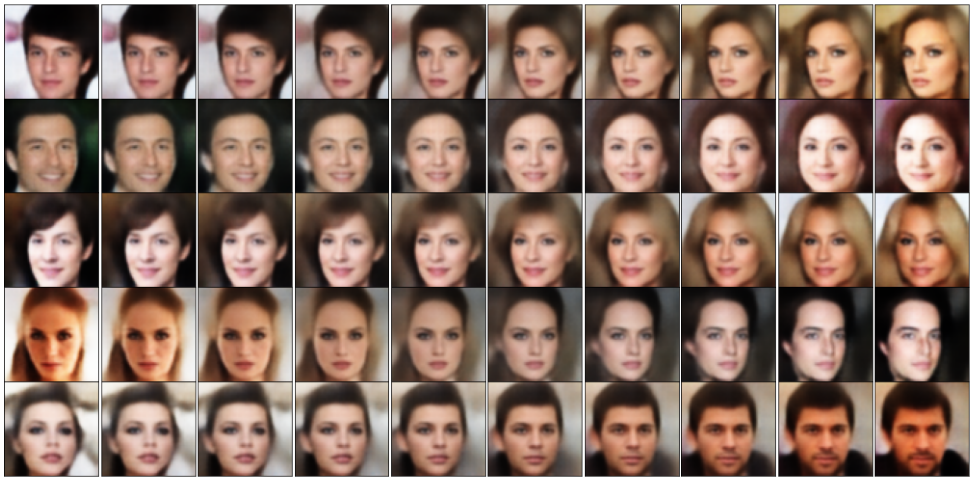

Figure 8: Interpolation in the latent space of samples of the autoencoder.

the interpolated images. Each row in Figure 8 shows a clear linear progression in ten steps from the first face on the left to the final face on the right. For example, in the last row, we see a female with blonde hair slowly transforming into a male with a beard. The transition is smooth, and no sharp changes or random images occur in-between.

E Additional Experiments

E.1 Numerical Assessment of Methods on CelebA

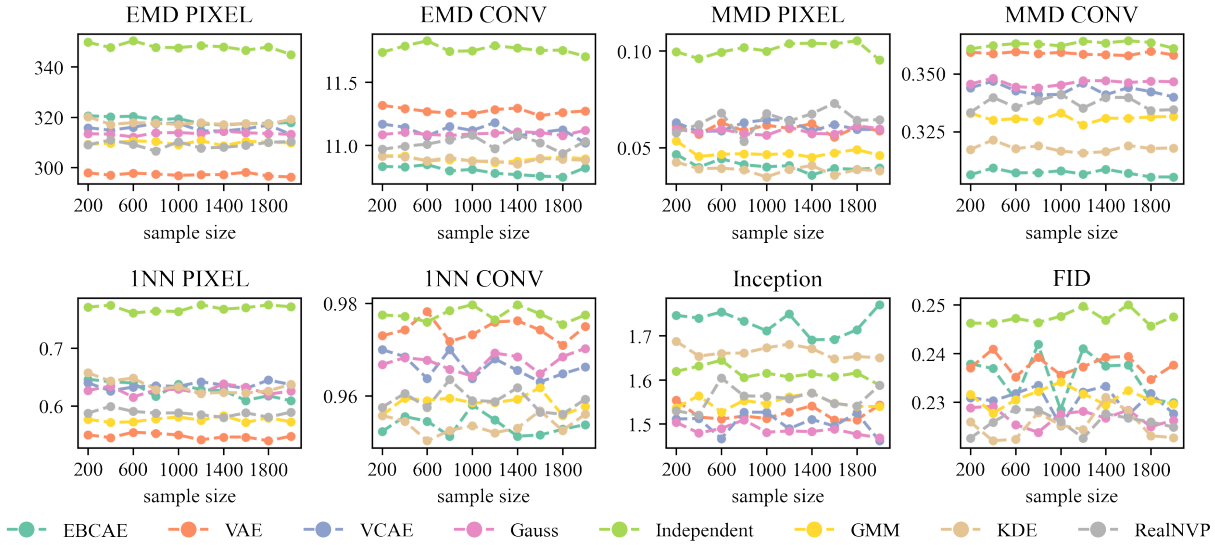

Figure 9: Performance metrics of generative models on **CelebA**, reported over latent space sample size. Note that they only differ in the latent space sampling and share the same autoencoder.

E.2 Numerical Assessment of Methods on MNIST

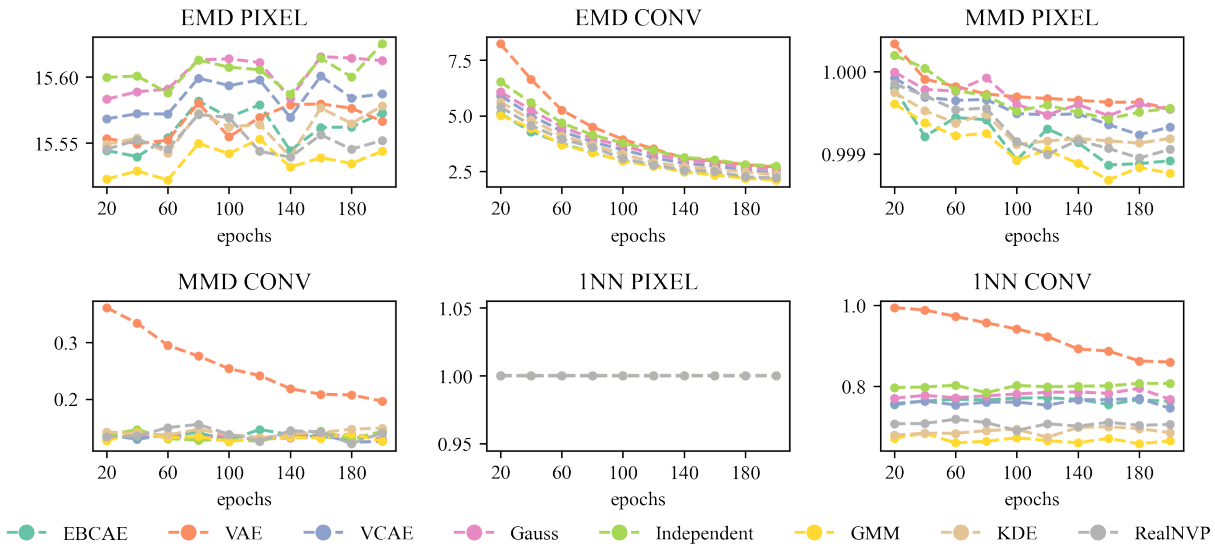

Figure 10: Performance metrics of generative models on **MNIST**, reported over epochs computed from 2000 random samples. Note that they only differ in the latent space sampling and share the same autoencoder.

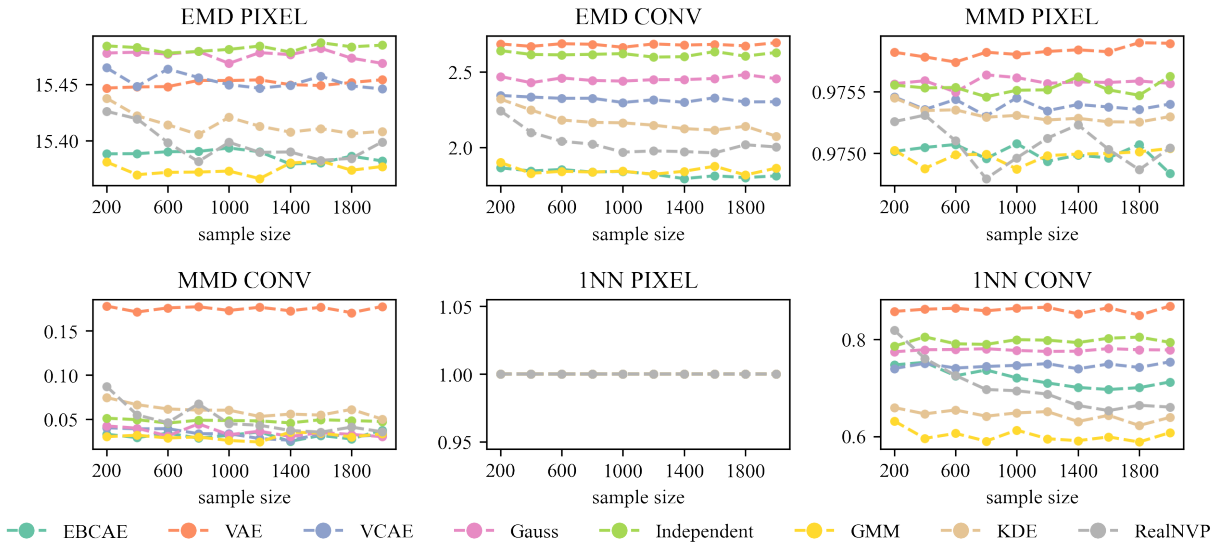

Figure 11: Performance metrics of generative models on **MNIST**, reported over latent space sample size. Note that they only differ in the latent space sampling and share the same autoencoder.

E.3 Numerical Assessment of Methods on SVHN

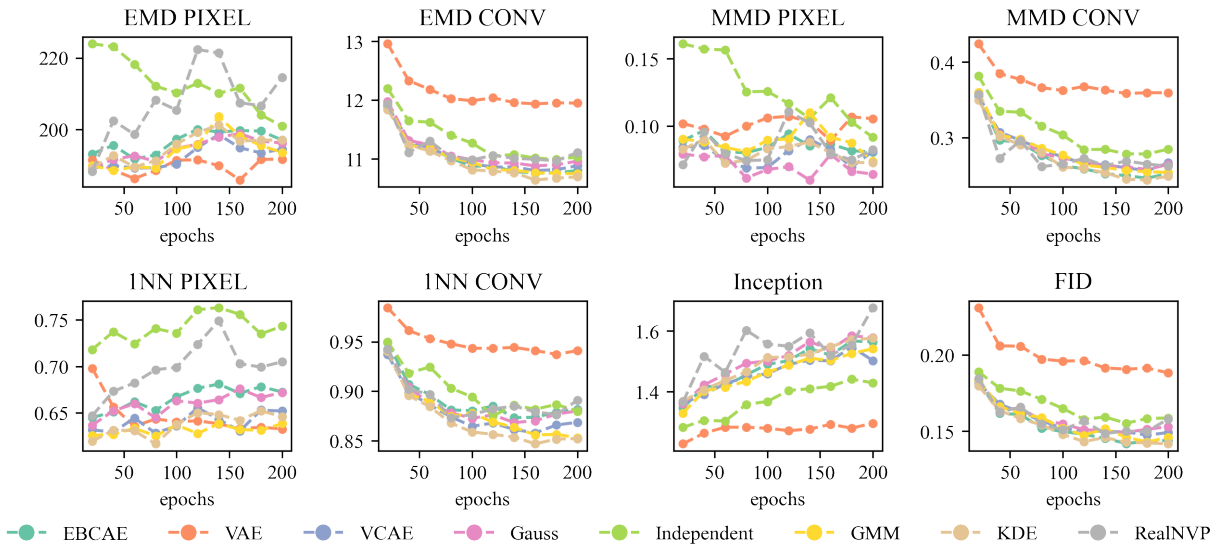

Figure 12: Performance metrics of generative models on **SVHN**, reported over epochs computed from 2000 random samples. Note that they only differ in the latent space sampling and share the same autoencoder.

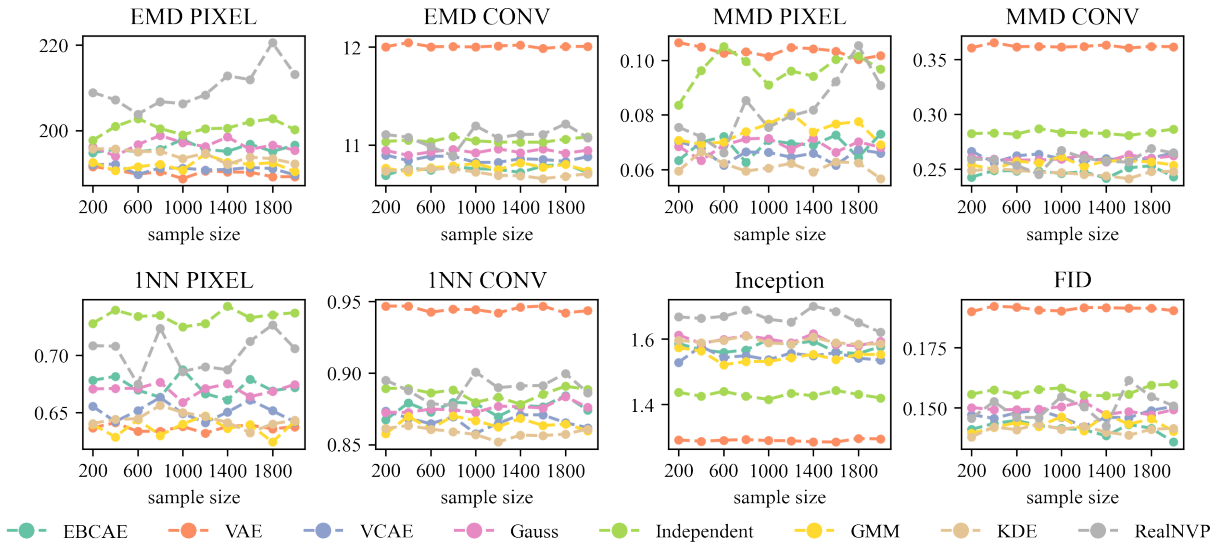

Figure 13: Performance metrics of generative models on **SVHN**, reported over latent space sample size. Note that they only differ in the latent space sampling and share the same autoencoder.

E.4 Generated Images from SVHN

Figure 14: Comparison of synthetic samples of different Autoencoder models. **1**st **row:** Fitted normal distribution, **2**nd **row:** Independent margins, **3**rd **row:** KDE-AE, **4**th **row:** GMM, **5**th **row:** VCAE, **6**th **row:** EBCAE, **7**th **row:** VAE, **8**th **row:** Real NVP, **Last row:** original pictures.

F Code

Code will be provided here. [link to the repository will be inserted]