Radboud University Nijmegen


Faculty of Social Sciences

# Marginalizing Flows

Thesis BSc Artificial Intelligence

*Author:*
Stijn de Boer
s1003731

*Supervisor:*
dr. Luca Ambrogioni

*Second reader:*
dr. Umut Güçlü

**Abstract**

We introduce marginalizing flows, an extension to normalizing flows which allow for better density estimation by marginaliziation of auxiliary random variables. We give an outline of the shortcomings of normalizing flows and motivate our approach. We trained models with an architecture based on Real NVP by Dinh et al.[5] on several datasets. For low-dimensional data we see that marginalizing flows consistently predict higher likelihood than normalizing flows, but our results do not generalize to higher-dimensional data like images.

July 30, 2020

# Contents



# 1 Introduction

A main objective in statistics is to model the distribution of some continuous random variables given observed data. Ideally, a model is able to represent the target distribution well and makes it easy to perform inference, but it is not always easy or possible to find a simple parametric model that fits both these criteria. Much of present-day research in statistical modeling is devoted to finding methods for approximate density estimation, and with today's computational possibilities, the problem of density estimation is often rephrased as an optimization problem. Using modern methods like Normalizing Flows, it is possible to train a model to approximate a target distribution by performing gradient descent on the negative log-likelihood. NFs are flexible enough to come arbitrarily close to a target distribution, and offer a mapping to a domain in which inference is easy, such as a Gaussian distribution. These two properties make them outstanding for approximating arbitrary probability distributions. In the next section, we will discuss the properties of Normalizing Flows in more detail.

## 1.1 Normalizing Flows

A normalizing flow is a bijective map with the key property that the probability density in the mapped space corresponds to the probability density in the space that is mapped to, given a correction for the compression that happens during the transformation. In this section we will lay out the mathematical foundation of NFs and we will give an overview of the history and current practice.

### 1.1.1 Definition

Let $\boldsymbol{x}$ and $\boldsymbol{u}$ be D-dimensional vectors with $\boldsymbol{x} \sim p_x(\boldsymbol{x})$ and $\boldsymbol{u} \sim p_u(\boldsymbol{u})$. A normalizing flow $T$ is a transformation such that

$$\boldsymbol{x} = T(\boldsymbol{u}) \tag{1}$$

A visualization of the transformation $T$ can be seen in figure 1a. What characterizes a normalizing flow is that it connects the densities $p_u$ and $p_x$ in a way that allows evaluation of $\boldsymbol{x}$ in terms of $p_u$, notably, without evaluation of $p_x(\boldsymbol{x})$:

$$p_x(\boldsymbol{x}) = p_u(\boldsymbol{u}) |\det J_T(\mathbf{u})|^{-1} \qquad \text{where } \boldsymbol{u} = T^{-1}(\boldsymbol{x}) \tag{2}$$

Where $J_T(\boldsymbol{u})$ denotes the Jacobian of $T$ with respect to $\boldsymbol{u}$. The Jacobian is a matrix $J_T$ of partial derivatives, such that $J_T(\boldsymbol{u})_{i,j} = \frac{\partial T_i}{\partial \boldsymbol{u}_j}$.

As an intuition, one can think of $T$ as moving, warping, and compressing the $u$-space until the density of $\boldsymbol{u}$ in $u$-space matches the density of $T(\boldsymbol{u})$ in $x$-space. However, as a result of stretching and squeezing, the volume around $\boldsymbol{u}$ is compressed with a certain factor. The Jacobian determinant expresses exactly how much the compression has been, so to eliminate the effect of the compression, the Jacobian determinant has to be divided out. Two requirements for $T$ follow from 2:

1. $T$ is differentiable
2. $T$ is invertible

The Jacobian of $T$ is a $D$-dimensional partial derivative, making the first requirement obvious. The invertibility requirement comes from the dependency on $T^{-1}(\boldsymbol{x})$. Dividing by the Jacobian determinant as in equation 2 is only possible if the Jacobian determinant is non-zero, so that has to be forced. Invertibility follows from the inverse function theorem which states that a continuously differentiable transformation is invertible if it has a non-zero Jacobian determinant. Methods for constructing flows that have a tractable inverse and an easy Jacobian determinant are a topic of main interest in the literature.

Corrolary to 2 since $T$ is invertible, we also have:

$$p_u(\boldsymbol{u}) = p_x(\boldsymbol{x}) |\det J_{T^{-1}}(\boldsymbol{x})| \qquad \text{where } \boldsymbol{x} = T(\boldsymbol{u})$$

It is not generally feasible to find a map that can transform $x$ to $u$ directly, which is why we seek to increase the expressiveness of the flows without having to define exotic types of parametric maps. Like a composition of simple maps can overcome the limitations of parametric maps, flows gain expressive power when they are joined in a composition. Suppose $T$ is a composition of $N$ flows: $T = T_N \circ T_{N-1} \circ ... \circ T_0$, we can use equation 2 in the same way where we have the Jacobian determinant of $T$ be the

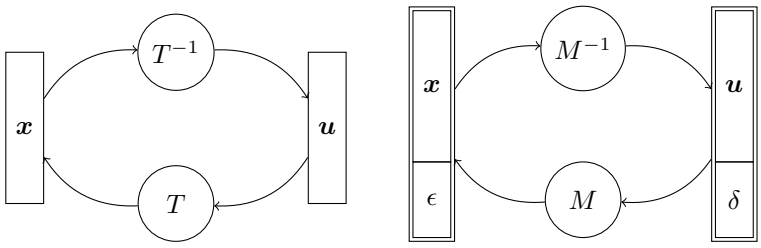



**Figure 1:** 1a represents a forward and inverse transformation on $x$ and $u$ respectively. Similarly, 1b represents the operations on $x$ when it is concatenated with some $\epsilon$. Note that the representation of $u$ and $\delta$ is not entirely truthful, since it suggests that the two are separable by a discrete cut. In practice, the information of $u$ and $\delta$ can be shared accross dimensions.

product of the Jacobian determinants of its component flows, i.e.:

$$\begin{aligned} |\det J_T(\boldsymbol{u})| &= \prod_{n=0}^{N} |\det J_{T_n}(\boldsymbol{u}_n)| && \text{where } \boldsymbol{u}_n = (T_n \circ \ldots \circ T_0)(\boldsymbol{u}) \\ |\det J_T^{-1}(\boldsymbol{x})| &= \prod_{n=N}^{0} |\det J_{T_n^{-1}}(\boldsymbol{x}_n)| && \text{where } \boldsymbol{x}_n = (T_n^{-1} \circ \ldots \circ T_N^{-1})(\boldsymbol{x}) \end{aligned} \tag{3}$$

Where we applied the lemma that the Jacobian of a composite function is the product of the Jacobians of the components[1]. We can force all $\det J_{T_n}$ to be non-zero, and since $\det(AB) = \det(A)\det(B)$, we can invoke the inverse function theorem again and find that the composite $T$ meets the requirements of differentiability and invertibility. From here on, we will no longer distinguish between flows and composite flows.

### 1.1.2 Stochastic gradient descent on the negative log-likelihood

A machine learning model like an NF is trained by minimizing the loss function by performing gradient descent on the parameters. For many variational inference problems, a natural candidate for the loss function is the KL-divergence, since it expresses exactly the quantity that we want to minimize: the deviation from one distribution to another distribution. The KL divergence between $D$-dimensional continuous distributions $p$ and $q$ is:

$$D_{KL}(p||q) = \int_{\mathbb{R}^D} p(\boldsymbol{x}) \log\Big(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\Big) d\boldsymbol{x}$$

Suppose $T_\phi$ is a flow with parameters $\phi$. To compute the KL divergence between the target distribution $p_x$ and our flow model $p^{T_\phi}$, we substitute and find:

$$\begin{aligned} D_{KL}(p_x(.)||p_x^T(.;\phi)) &= \int_{\mathbb{R}^D} p_x(\boldsymbol{x}) \log\Big(\frac{p_x(\boldsymbol{x})}{p_x^T(\boldsymbol{x};\phi)}\Big) d\boldsymbol{x} \\ &= -\int_{\mathbb{R}^D} p_x(\boldsymbol{x}) \log(p_x^T(\boldsymbol{x};\phi)) d\boldsymbol{x} + C \qquad (4) \\ &= -\mathbb{E}_{p_x} \log(p_x^T(\boldsymbol{x};\phi)) + C \\ &= -\mathbb{E}_{p_x}\left[\log\Big(p_u(T^{-1}(\boldsymbol{x};\phi))\Big) + \log\Big(|\det J_{T^{-1}}(\boldsymbol{x};\phi)|\Big)\right] + C \qquad (5) \end{aligned}$$

[1] The chain rule

Where 4 is allowed because $\int_{R^{\mathbb{D}}} p_x(\mathbf{x}) \log(p(\mathbf{x}))d\mathbf{x}$ does not depend on $\phi$. The constant term introduced at 4 does not depend on $\phi$, thus we can ignore it during optimization. Equation 5 is obtained by substituting 2 and by observing that

$$|\det J_T(\boldsymbol{u})|^{-1} = |\det J_{T^{-1}}(\boldsymbol{x})| \qquad \text{whenever } \boldsymbol{x} = T(\boldsymbol{u})$$

Intuitively, what follows from 5 is that minimizing the KL-divergence is equivalent to fitting the model parameters to the data by maximum likelihood estimation. Since the KL-divergence expresses a goodness-of-fit, we can use this as a loss function, and optimize parameters $\phi$ by stochastic gradient descent. Suppose we have observed $N$ data points $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$, we can get an estimate of the KL-divergence (and negative log-likelihood) by:

$$\mathcal{L}(\phi) = -\frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{p_x}\left[\log\Big(p_u(T^{-1}(\boldsymbol{x}_n;\phi))\Big) + \log\Big(|\det J_{T^{-1}}(\boldsymbol{x}_n;\phi)|\Big)\right] \tag{6}$$

A consequence of equation 3 in the context of the loss equation 6, is that the log-product rule allows us to sum the logs of the Jacobian determinants of all the components.

### 1.1.3 Historical overview

Before normalizing flows appeared, a number of popular frameworks for density estimation existed. The approach of kernel smoothing techniques [15] (often mentioned in concurrence with MCMC) is to convolute the data - or a histogram - with a parameterized kernel, for example a Gaussian with variance $W$. The problem of this approach is picking a good $W$, which is often non-trivial. Another approach to density estimation is Gaussian mixture models [11](with the EM-algorithm on the frontline). This method is similar to kernel smoothing in the sense that it superimposes (Gaussian) kernels over the data, but opposed to kernel smoothing, the number of kernels is fixed, and in practice[12] an iterative method is used to converge the kernels to the optimal configuration in terms of means and covariances.

The inspiration for NFs came from Tabak et al., who introduced the idea of finding an approximate map from an observed distribution to an isotropic Gaussian by gradient ascent on the log-likelihood in[18], and coined the term normalizing flow in[17]. Notably, Tabak provides proofs for the convergence of compositions of flows. Rezende et al.[16] first considered normalizing flows as an inference model in the context of variational inference[1, 2]. Since then, a lot of novel implementations have been investigated in the literature [4, 5, 6, 8, 14]. An excellent overview of the current methods and techniques is provided in [13].

## 1.2 Marginalizing flows

We have seen that NFs are in theory capable to express any mapping between distributions, but in practice it is not always possible to find a suiting parametric transformation. It has been shown that stacking multiple flows in a pipeline fashion improves the performance of the system as a whole[16], which makes more complex transformations possible. However, there is no real investigation into the effect of widening an NF as opposed to lengthening it. The motivation for widening is twofold.

Firstly, not all transformations are bijections, some information can be lost during the transformation. For instance, if $u$ is sampled from a normal distribution, i.e. $u \sim \mathcal{N}(0, 1)$, we can transform $u$ by squaring it: $x = f(u) = u^2$, such that $x \sim \mathcal{N}(0, 1)^2$. This transformation is non-invertible, and thus it is not possible to perfectly model this

transformation with an NF, since this would require the model to be non-invertible as well, which is not allowed. The best we can do is an approximation. Suppose now we can capture the lost information in an auxiliary variable, $\epsilon$, which models the sign of $u$: $\epsilon = \text{sgn}(u)$, such that $f(u) = \langle x, \epsilon \rangle$. Now, an inversion is possible: $f{-}1(u) = \epsilon\sqrt{x}$. Aside from learning the distribution of $x$, we can now also learn the distribution of $\epsilon$. After learning the distribution of $\epsilon$, we can marginalize $\epsilon$ out as follows:

$$p(x) = \int p(x, \epsilon) d\epsilon = \int p(x|\epsilon) p(\epsilon) d\epsilon = \mathbb{E}_\epsilon p(x|\epsilon)$$

It is our hypothesis that adding an auxiliary dimension to the transformation will improve the performance in terms of the likelihood of $x$.

The second motivation for widening flows is the Universal Approximation Theorem. Leshno et al.[9] showed that an artificial neural network with one hidden layer containing a finite number of neurons is capable of expressing any function, and since NFs are commonly modeled by artificial neural networks, this result begs the question what the effect would be of increasing the width of an NF.

### 1.2.1 Definition

Given a $D$-dimensional vector $\mathbf{x} \sim p_x(\mathbf{x})$, a marginalizing flow (MF) model is a model $p_x^M$ such that $p_x^M(\mathbf{x}) = p_x(\mathbf{x})$. an MF has a number of auxiliary dimensions $B$, and an underlying normalizing flow model $p_y^T$, where $y \in \mathbb{R}^Q, Q = D + B$. See figure 1b for an illustration.

Essential to MFs is the idea of marginalization. Suppose a transformation $M$ exists from a $Q$-dimensional isotropic Gaussian to a space of which the lowest $D$ dimensions model some observed data, and the highest $B$ dimensions capture the distribution of information that was not captured in the lowest $D$ dimensions. For example, we could map samples from a 2-dimensional Gaussian to a space $\psi$ where $\psi_0 \sim \mathcal{N}(0_{1,B}, \mathbf{I}_B^2)^2$ and $\psi_1 \sim \mathcal{N}(0_{1,B}, \mathbf{I}_B^2)$.

We can then get a density estimate of some new observation $\mathbf{x}$ by

$$\begin{aligned} p_x^M(\mathbf{x}) &= p_y^T(\mathbf{x}, \epsilon) \\ \epsilon &\sim \mathcal{N}(\boldsymbol{\mu}, \Sigma^2) \\ \boldsymbol{\mu} &= 0_{1,B} \\ \Sigma &= \mathbf{I}_B \end{aligned} \tag{7}$$

And we can acquire a Marginalized density estimate by importance sampling as follows

$$p_x^M(\mathbf{x}) = \frac{1}{S} \sum_{s=1}^{S} \frac{p_y^T(\mathbf{x}, \epsilon_s)}{p(\epsilon_s)} \tag{8}$$

$$\epsilon_s \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2) \qquad s \in \{1, \ldots, S\} \tag{9}$$

$$\boldsymbol{\mu} = 0_{1,B} \tag{10}$$

$$\sigma = \mathbf{I}_B \tag{11}$$

Assuming that some information $\epsilon$ is lost during the transformation in equation 1, we can fit the distribution of $\epsilon$ to the Gaussian in 8, enabling us to marginalize over $\epsilon$. Note that the choice for a Gaussian is arbitrary, we may choose to model $\epsilon$ by any other distribution equivalently. The choice for the Gaussian distribution is made here because it supports efficient sampling.

Fitting a marginalizing flow to data is not much different from fitting a Normalizing flow [2]. Given some observed data $x \in \mathbb{R}^D$, we concatenate it with some $\epsilon \in \mathbb{R}^B$ and train a mapping to a $D + B$-dimensional known distribution. For fitting the marginalizing flow it suffices to optimize with respect to equation 7, but it is important that every observation $x$ is concatenated with many different $\epsilon$ during training. By concatenating every observation with many different $\epsilon$ and maximizing equation 7, maximization of equation 8 is implicitly achieved.

### 1.2.2 Related work

Similar to marginalizing flows, Noisy Injective Flows by maaloe et al [10] introduce marginalization over auxiliary variables to arrive at a density estimate. In contrast to our approach, their approach is supervised in the sense that they use class labels as auxiliary variables, wheres here we employ random auxiliary variables.

Gemici et al. [7] address the problem of density estimation on Riemannian manifolds with non-euclidean geometry. Mappings from these manifolds to a euclidean space would allow the use of methods like Normalizing flows, but these mappings are not generally invertible. Gemici et al. introduce the *Inverse Stereographic Transform* which provide an inverse operation from for instance a unit sphere $S^2$ (embedded in $R^3$) to a euclidean space $R^2$, allowing to apply normalizing flows on spheres by first mapping $S^2$ to $R^2$, transforming the space $R^2$ by a normalizing flow, and then mapping back to $S^2$.

In an approach that bears some similarities to that of Gemici et al., Cunningham et al. [3] (currently under review) provide auxiliary Deep Generative Models (ADGM), establishing their point using a Normalizing Flow that works across dimensions. Under the assumption that a low-dimensional manifold underlies higher-dimensional data, their method first uses a bijective map from the low dimensional manifold to the high-dimensional space, and then adds noise to the generated point. The inverse transformation that NFs require is provided by some means of marginalization, but in a different sense than that of marginalizing flows. In MFs, the auxiliary dimensions are marginalized out, whereas in ADGMs, the marginalization is done to eliminate noise as to project an observed point on the embedded manifold.

## 1.3 Main questions

The main question that will be addressed here is whether MFs perform better than normalizing flows. Secondly, it will be investigated whether there exists a relation between the number of auxiliary dimensions $B$ and the performance of the model.

# 2 Methods

We used the NVP architecture described by Dinh et al.[5] to model the NF underlying the MF. We trained our models on synthetic datasets with varying characteristics, as well as on MNIST and the half-moons dataset.

## 2.1 Model architecture

The NVP architecture is a repeating structure of three layers; a special type of batch normalization called Actnorm, a coupling layer, and a permutation layer.

[2] This should come as no surprise, since an MF with $B = 0$ is an NF

### 2.1.1 Actnorm layer

In order to improve numerical stability to allow deeper models and smaller minibatches, Dinh used a special type of batch normalization called actnorm. The actnorm layer takes an input $\boldsymbol{x}$ and outputs $\boldsymbol{u}$ where $u_i = \frac{(x_i - \beta_i)}{\alpha_i + \gamma}$, with small $\gamma > 0$ to avoid division by zero. The log Jacobian determinant is given by $-\sum_i \log(\alpha_i + \gamma)$, and the inverse operation is simple, $x_i = (u_i \cdot (\alpha_i + \gamma)) + \beta_i$. It is important that every $\alpha_i$ is positive, and in practice we achieved this by modeling every $\alpha_i$ by an exponential function: $\alpha_i = \exp(\alpha_i')$. In the first call to the actnorm transformation, the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are initialized to the dimension-wise standard deviation $\sigma_1$ and mean $\mu_1$ of the first minibatch respectively. This results in the aimed normalization, and causes the input to the following layer to be numerically stable. Dinh et al. [5] also discuss improving the transformation by using a moving average with decay $\eta$ instead, where $\boldsymbol{\alpha_{t+1}} = \eta * \boldsymbol{\alpha_t} + (1 - \eta) * \boldsymbol{\sigma_t}$, and similarly $\boldsymbol{\beta_{t+1}} = \eta * \boldsymbol{\beta_t} + (1 - \eta) * \boldsymbol{\mu_t}$. This is not implemented here, but it is worth to consider as an extension to the current method.

### 2.1.2 Coupling layer

The coupling layer is the most important transformation in the NVP architecture, since it supports complex non-linear interactions between variables. The coupling layer is an autoregressive layer, which means to say that the transformation of $x_i$ to $u_i$ strictly depends on $x_i$ and $u_{<i}$. Autoregressive flows uses two types of functions, transformers and conditioners. For every $x_i$, the output $u_i$ is given by $c_i(x_i, \tau_i(u_{<i}))$, where $\tau_i$ is the $i$'th transformer, and $c_i$ is the $i$'th conditioner. This transformation is invertible if the conditioner is invertible, and the inverse can be computed in an incremental fashion:

$$
\begin{aligned}
x_1 &= c_1^{-1}(u_1) \\
x_i &= c_i^{-1}(u_i, \tau_i(x_{<i})) \qquad\qquad i > 1
\end{aligned}
$$

In other words, the transformation that is applied to the higher dimensions of $x$ depends on the result of the transformations on the lower dimensions of $x$. This simple restriction makes the transformation invertible, and results in a diagonal Jacobian. MAF as described by Papamakarios et al.[14] and NVP described by Dinh et al. represent the two outer extremes of the spectrum of autoregressive flows. In MAF, every $u_i$ strictly depends on $x_i$ and $u_{<i}$, whereas in NVP, the lower half of the dimensions is not transformed at all, and the transformations of the higher half of the dimensions depend on all the lower dimensions, see figure 2. This means that if we transform $N$ dimensions, the first $N/2$ dimensions of $\boldsymbol{u}$ are given by the first $N/2$ dimensions of $\boldsymbol{x}$, and the last $N$ dimensions of $\boldsymbol{u}$ are given by $\tau_i(x_i, c_i(u_{<N/2}))$ This results in a quick forward and inverse operation, since both require a single application of the conditioner and a single evaluation of the transformer(or its inverse). We use the NVP architecture, and we model the transformer by a neural network, see figure 2. We define $\tau_i(x_i, c_i(u_{<N/2}))$ by $x_i * e^{s_i} + m_i$ where $c_i(u_{<N/2}) = \langle s_i, m_i \rangle$. The Jacobian of this transformation is lower triangular with only ones in the upper left diagonal, and all the $e^{s_i}$ on the lower right diagonal. The log Jacobian determinant is thus simply the sum of all the $s_i$'s.

**Neural Network** As mentioned above, the transformer in the coupling layer is modeled by a neural network. The structure of a coupling layer in an $N$-dimensional NF is depicted in figure 2.

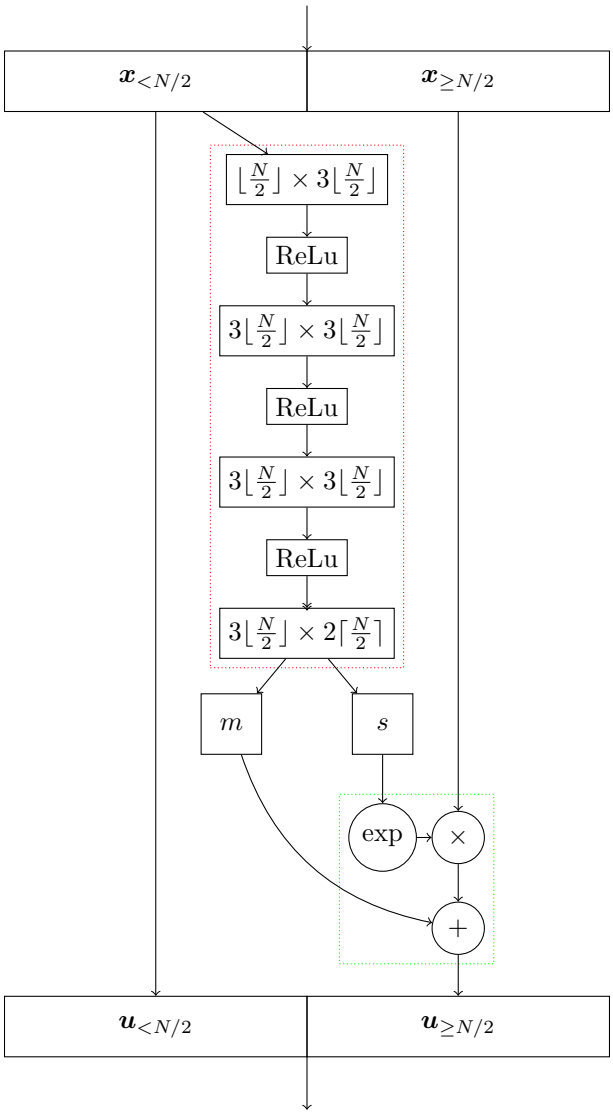



**Figure 2:** An NVP coupling layer with a simple neural network (within the red dotted box) as the transformer. Loosely speaking, the operations within the green dotted box together make up the conditioner. As should be clear from this diagram, the whole layer is invertible if the conditioner is invertible.

### 2.1.3 Permutation layer

The coupling layer only transforms the first half of the dimensions, and the actnorm layer does not allow interaction between dimensions. This implies that using only those two layers, some dimensions would only be linearly transformed (by the actnorm layer). In order to flexibly transform all the dimensions we need to permute the dimensions between the coupling and actnorm layers. We implemented two types of permutation, one where the dimensions are simply inverted, and one where the dimensions are randomly shuffled. The specific random permutation is generated upon initialization, and is considered a non-learned parameter. We alternated between the two types of permutation, starting with the inverting type. This ensured that all the dimensions are transformed by one coupling layer after two coupling layers, since all the dimensions that receive an identity mapping in the first coupling layer are transformed by the conditioner in the second coupling layer, and vice versa. The log determinant of every permutation layer is 0, which should come as no surprise as permuting dimensions does not compress them.

### 2.1.4 R-block

In our models, the first layer is always an actnorm layer, followed by $L - 1$ composed blocks of a fixed structure:

1. Coupling layer
2. Permutation layer (inversion)
3. Coupling layer
4. Permutation layer (random shuffle)
5. Actnorm layer

We will refer to this composition of five layers as an *R-block*. The last layer in our models is also an R-block, but where the last two components have been omitted. We can omit the last two layers because the actnorm and permutation layers are only used to, respectively, ensure numerical stability in later coupling layers and to ensure that dimensions which were not coupled in preceding coupling layers can be coupled in later coupling layers. There are no more coupling layers after the last coupling layer in the last R-block, so these properties are not needed.

An R-block guarantees that all dimensions are transformed by a coupling layer; a coupling layer always non-linearly transforms the higher half of the dimensions, and the two coupling layers are separated by a permutation layer of the kind that inverts the order of the dimensions.

## 2.2 Data

To address the main question - whether MFs are able to improve density estimates by marginalizing over auxiliary random variables - we used two sources of synthetic data. One type of synthetic data was generated by taking samples from a Gaussian and raising them to a specified power. This allowed us to control exactly the distribution and invertibility of the target transformation, and to address our questions with regard to those properties. For illustration, the question whether MFs do not impair the performance of NFs can only be conclusively addressed in the case where the transformation that is modeled does not cause information to be lost, e.g. when the transformation is a bijection. In addition to the Gaussian synthetic case, we also tested on the slightly more exotic half-moons[3]. This data is generated by taking a uniform partition of two 1-D line segments embedded in 2-D space and adding a specified amount of Gaussian

[3]See https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html

noise. To assess our model under a 'real-world' example, we also trained and tested on MNIST and CIFAR-10. These datasets are well known in the scientific community and are commonly used to assess the performance of generative and probabilistic models. They can both be freely accessed online.

In the next sections, we provide details on the datasets, the pre-processing, and how we constructed the models in each case.

### 2.2.1 Gaussian synthetic data

The Gaussian synthetic data was generated by taking $n_s$ samples from a $D$-dimensional isotropic unit Gaussian with mean zero and raising every dimension to the power $\psi$. The resulting distributions were used as training data for MFs with a number of auxiliary dimensions: $B$. We trained models with every combination of parameters from these ranges:

$$\begin{aligned} n_s &= 2048 \\ D &= \{2, 3, 4\} \\ \psi &= \{1, 2, 3\} \\ B &= \{0, 1, 2, 3, 4\} \\ r &= 5 \end{aligned}$$

A datapoint $x_{D,\psi}$ is distributed by $x_{D,\psi} \sim \mathcal{N}(0^D, I^D)^\psi$.

We trained five separate models for each combination of parameters, leading us to train $3 \times 3 \times 5 \times 5 = 225$ models, with a total of 75 unique configurations of data generation parameters and model hyperparameters.

Here we go into more detail on our choice of parameters.

**Dimensionality ($D$)** Firstly, the NVP architecture requires our data to be at least 2-dimensional, which is why we only generate data with dimensionality greater than or equal to 2. Secondly, it would be interesting to see if the number of lost bits of information, the number of auxiliary dimensions, and the performance of the model are correlated. Squaring more dimensions means more bits of information are lost, so we expect to see that a larger number of auxiliary dimensions is needed to account for the bits that are lost by squaring a larger number of gaussian dimensions.

**Exponent ($\psi$)** If the transformation between the base and the target distribution is invertible, we expect an NF to be capable of modeling it quite well. We expect that in that case, using an MF, and increasing the number of auxiliary dimensions of that MF does not have a significant effect on the performance of the model - especially in the case where the transformation is an identity mapping. In order to test this, we transformed samples from a Gaussian with three different transformations. One of which the identity mapping $\psi = 1$, one a non-invertible transformation $\psi = 2$, and one a non-linear invertible transformation $\psi = 3$.

**Number of auxiliary dimensions ($B$)** In case MFs provides an improvement on NFs, it is a reasonable question to ask what is the best choice for $B$. We do not expect to give a conclusive answer here, but as mentioned before, we might see a correlation between the number of lost bits of information and the optimal choice for $B$.

| Data | \|Train set\| | \|Minibatch\| | Lr | # Epochs | Architecture | \|Grad\| |
|---|---|---|---|---|---|---|
| Synth. Gaussian | 2048 | 256 | $1E-3$ | 1024 | 1 coupling, 4 R-blocks | None |
| Halfmoons | 2048 | 64 | $1E-3$ | 1024 | 1 coupling, 4 R-blocks | None |
| MNIST | 60000 | 32 | $5E-4$ | 10 | 1 coupling, 4 R-blocks | 0.6 |
| Fashion-MNIST | 60000 | 32 | $5E-4$ | 10 | 1 coupling, 4 R-blocks | 0.6 |
| K-MNIST | 60000 | 32 | $5E-4$ | 10 | 1 coupling, 4 R-blocks | 0.6 |
| CIFAR-10 | 50000 | 32 | $5E-4$ | 16 | 1 coupling, 4 R-blocks | 0.6 |

**Table 1:** Training parameters. The |Grad| parameter indicates the gradient clip norm that was applied. A gradient $\theta$ is clipped to a norm $\eta$ by $\theta' = \eta \frac{\theta}{|\theta|}$.

#### 2.2.2 Half-moons data

The procedure by which the half-moons data is generated is simple and can be found here. We used the same range for $B$ as for the Gaussian synthetic data, and we trained ten models for each configuration of data-generation parameters.

#### 2.2.3 Image data

We trained and tested on MNIST, Fashion-MNIST, and KMNIST where we first applied a normalization on the data and then added a uniform noise sampled from $[-\frac{3}{10}, \frac{3}{10}]$. For CIFAR10 we only normalized the data, and did not add noise.

### 2.3 Training parameters

We have summarized all our training parameters in table 1. We found that a smaller batch size was required for multimodal distributions such as the image data. For simple monomodal distributions - i.e. the synthetic Gaussians - a smaller batch size provided no benefit so a larger batch size was used to speed up training. For all models a $Q$-dimensional isotropic Gaussian was used as base distribution, and a $B$-dimensional isotropic Gaussian was used for sampling $\epsilon$.

## 3 Results

### 3.1 Synthetic Gaussian data

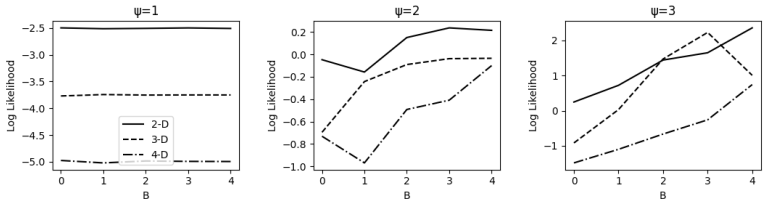
The results on the synthetic datasets are reported in table 2 and a visualization can be seen in figure 3. We selected the best of ten models, and we set the marginalization parameter $S$ from equation 8 to 200. Some important observations can be made from the data. Firstly, we see that for all $D$, if $\psi = 1$, adding auxiliary dimensions does not improve nor deteriorate the performance of the model. This result is reflected by the flat lines in figure 3a. According to this result we have to draw the conclusion that MFs do not hamper the performance of NFs in the case where any information may be lost[4]. Secondly, an arguably more interesting result can be observed for the cases where $\psi > 1$. In those cases the likelihood increases significantly with higher $B$, which can be regarded as being in line with the speculation that the auxiliary dimensions can account for lost information. The effect is quite drastic; the greatest numerical improvement at

[4]Recall that an MF with $B = 0$ is equivalent to an NF

$\psi = 3, D = 3$, with the difference between $B = 0$ and $B = 3$ there being close to $\pi$. Interestingly though, with all other parameters equal, increasing $B$ to 4 results in a drop of 1.21.

| $\psi$ | $D$ | $B = 0$ | $B = 1$ | $B = 2$ | $B = 3$ | $B = 4$ |
|---|---|---|---|---|---|---|
| 1 | 2 | -2.5 | -2.51 | -2.51 | -2.5 | -2.51 |
| 1 | 3 | -3.77 | -3.74 | -3.75 | -3.75 | -3.75 |
| 1 | 4 | -4.98 | -5.02 | -4.98 | -4.99 | -5.0 |
| 2 | 2 | -0.05 | -0.16 | 0.15 | 0.24 | 0.21 |
| 2 | 3 | -0.7 | -0.24 | -0.09 | -0.04 | -0.03 |
| 2 | 4 | -0.73 | -0.97 | -0.49 | -0.41 | -0.1 |
| 3 | 2 | 0.25 | 0.72 | 1.44 | 1.65 | 2.35 |
| 3 | 3 | -0.92 | 0.03 | 1.47 | 2.22 | 1.01 |
| 3 | 4 | -1.48 | -1.1 | -0.66 | -0.26 | 0.74 |

**Table 2:** The log-likelihood of Gaussian synthetic data as predicted by marginalizing flows. We display the prediction of the best of 10 models for each configuration of parameters.
The positi




**(a)** **(b)** **(c)**

**Figure 3:** The data in table 2 represented as a line. The positive correlation between $B$ and the log-likelihood is apparent.

Generated samples are plotted in figure 4. Here too, it seems like a larger $B$ results in a better approximation of the target distribution.

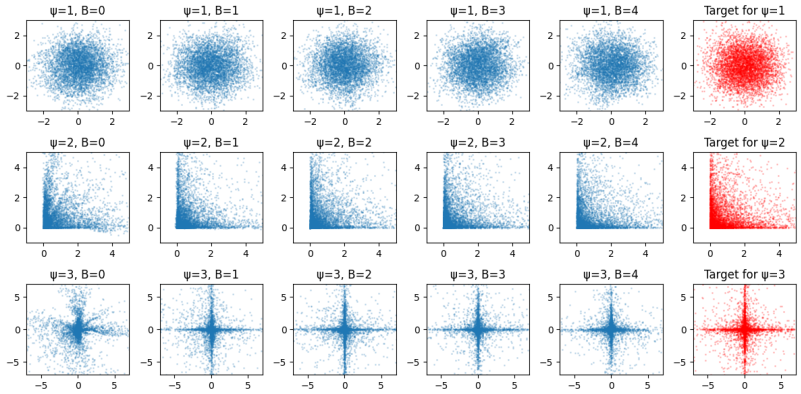
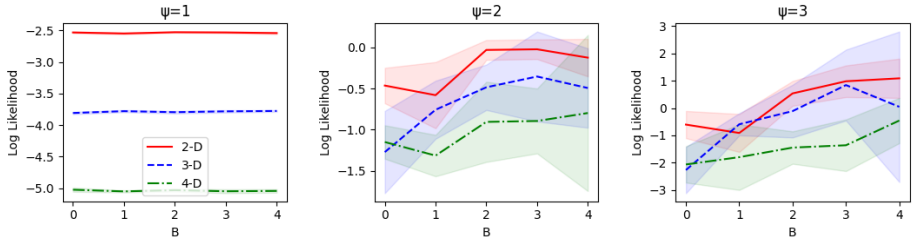



**Figure 4:** Samples generated from random noise by the best of 10 models for each combination of parameters. We see that for $\psi > 1$, the transformation visually becomes better with increasing $B$.




**(a)** **(b)** **(c)**

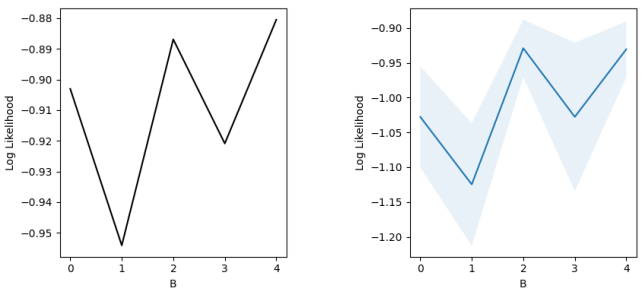
**Figure 5:** The average log-likelihood of the synthetic Gaussian data as predicted by 10 models plotted with a shaded 95% confidence interval. Figure 5a shows that MFs do not improve on the easy case where the target distribution is identical to the base distribution. Figures 5b and 5c show that for the more complex cases where $\psi > 1$, a higher $B$ corresponds to better performance in terms of log-likelihood.

## 3.2 Half moons data

We computed the negative log-likelihood of the half-moons dataset, see table 3 and figure 6a. Out of ten models, we took the best performing one for each $B$, and we set the marginalization parameter $S$ from equation 8 to 200. We found that there were some cases where a model trained with a number of auxiliary dimensions yielded no predictably better results than models trained without auxiliary dimensions. As becomes clear from the confidence intervals, the performance of the models at any $B$ are quite unpredictable. It is not clear that the predicted log-likelihood improves with higher $B$, which is in contrast to the results on the synthetic data where $\psi = 3$. Both the half-moons data, as well as that particular synthetic data could be perfectly generated from an isotropic gaussian through a bijective map, keeping in mind that the mapping required to produce the half-moons data is less trivial than the mapping required to produce the synthetic data where $\psi = 3$, reason being the bimodality of the half-moons data.

| B | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| Log-likelihood | -0.9 | -0.95 | -0.89 | -0.92 | -0.88 |

**Table 3:** The log-likelihood of half-moons data as predicted by marginalizing flows. We display the prediction of the best of 10 models for each $B$.




**(a)** The data in table 2 represented as a line.

**(b)** The mean and 95% confidence interval of 10 models.

However, there is one advantage that the models with higher $B$ provide. Looking at the three left panels of image 6, we can distinguish a line of increased density connecting the two semicircles. In the cases where $B = 3$ and $B = 4$, we see that the two modes of the distribution are disconnected, apart from some very faint noise. Figure 14 in the appendices sheds more light on this. The images depict generations of the models in order of their predicted log-likelihood, with the first row giving the highest prediction, and the last row the lowest. Not only can we see that the models where $B = 0$ have failed to converge to an accurate representation of the target distribution in many cases, it is also apparent that the model with $B = 4$ shows much less connectivity between the modes than for example models $B = 1$ and $B = 3$. Strikingly, models $B = 2$ show a much more pleasing reconstruction than models $B = 3$. Because the underlying architecture has some even splits, this might be due to symmetric properties of the model, but one wonders if the effect disappears with higher $B$, or if the effect would also occur if a different architecture is used. This all together leads us to believe that the convergence properties of MFs are subtle.

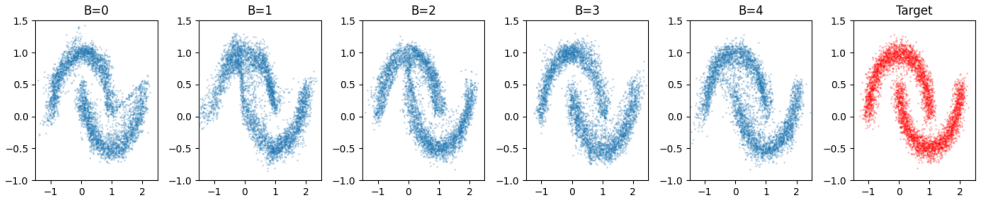


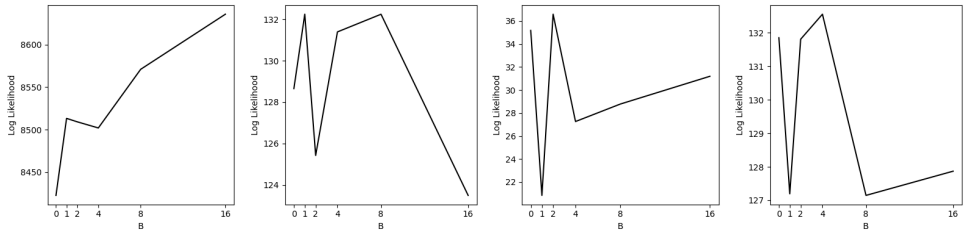

**Figure 6:** Samples generated from random noise by the best of 10 models for each $B$. We see that all models have converged an approximation that is visually similar to the target distribution. However, in the generations by models with $B \leq 2$ we can distinguish a line of increased density connecting the two semicircles. This effect is not seen for models with $B = 3$ and $B = 4$. More generations can be seen in figure 14

## 3.3 Image Data

The results on the image were not without ambiguity. See table 4 and figures 7 and 8. We selected the best of 10 models for each $B$ to arrive at the results in table 4 and figure 7, and we set the marginalization parameter $S$ from equation 8 to 100.

| B | 0 | 1 | 2 | 4 | 8 | 16 |
|---|---|---|---|---|---|---|
| CIFAR-10 | 8422.63 | 8513.6 | 8505.94 | 8502.32 | 8571.83 | 8635.47 |
| MNIST | 126.27 | 145.32 | 137.66 | 119.85 | 129.4 | 128.28 |
| K-MNIST | 33.26 | 33.63 | 28.07 | 27.93 | 28.82 | 35.09 |
| F-MNIST | 132.82 | 141.58 | 132.22 | 132.62 | 131.13 | 138.14 |

**Table 4:** Results in log-likelihood on image data.

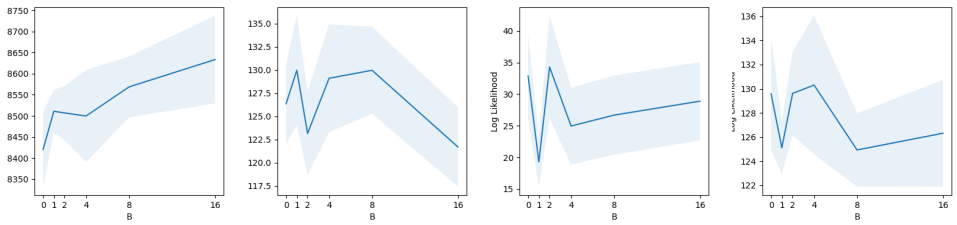



**(a)** CIFAR10 **(b)** MNIST **(c)** KMNIST **(d)** Fashion-MNIST

**Figure 7:** With the exception being CIFAR10, no predictable improvement in the negative log-likelihood is gained by increasing $B$ on image data.

On the CIFAR10 data, the log-likelihood appears to increase predictably as $B$ gets larger. The average log-likelihood as well as the confidence intervals in 8a increase steadily with $B$.

On KMNIST and Fashion-MNIST, the log-likelihood at $B = 1$ was significantly less than the log-likelihood at $B = 0$, but at $B = 2$, both models recovered to a performance that was on par with models with $B = 0$. For regular MNIST, almost the opposite effect can be seen; at $B = 1$, the log-likelihood is higher than at $B = 1$, and it drops again at $B = 2$. Furthermore, the wide confidence intervals, and their large lateral overlaps in figures 8b,8c and 8c suggest that the predicted log-likelihood is subject to fluctuations of such degree that these models cannot be analyzed in such a straightforward manner.




**(a)** CIFAR10 **(b)** MNIST **(c)** KMNIST **(d)** Fashion-MNIST

**Figure 8:** The average log-likelihood of the image data as predicted by 10 models plotted with a shaded 95% confidence interval.

We hypothesized that this might be due to the fact that the manifold underlying MNIST is of a too low dimensionality for adding auxiliary dimensions to have any effect; the 784 dimensions that MNIST were believed to five the NF enough flexibility to model the target transformation. In figure 9 it can be seen that the first 300 principal components of MNIST capture more than 95% of the variance. However, this speculation was proved incorrect when the PCA of the CIFAR10 models was superimposed. We see that the distribution of variance in the principal components of CIFAR10 is even more skewed towards the larger PCs than that of the MNIST data. Consequently to conclude that the skewness of the distribution of variance is not a good indicator for the efficacy of marginalizing Flows. As an aside, PCA is a linear transformation, and the MNIST manifold is not likely to lay in a linear space, but it is enough to show that a lower dimensionality than the actual dimensionality of MNIST-space is enough to capture most of the variance in it. Perhaps a nonlinear principal component analysis could provide more insight.

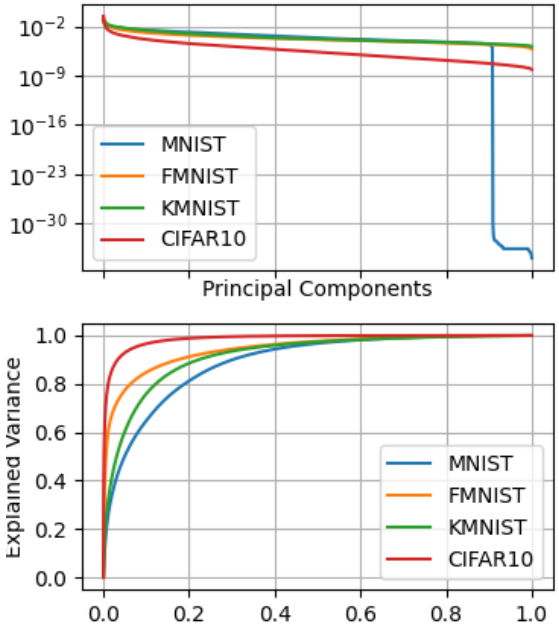



**Figure 9:** The variance explained by principal components of MNIST. The first 331 PCs explain 95% of the variance, and the first 681 PCs explain 99.9% of the variance.

To show the degree of convergence our models achieved we have included a series of images that we generated by transforming samples from an isotropic Gaussian in figures 10, 11, 12 and 13. The image samples were generated from random noise by MFs with $B = (0, 1, 2, 4, 8, 16)$ for the rows respectively. The last row shows samples from the training set.

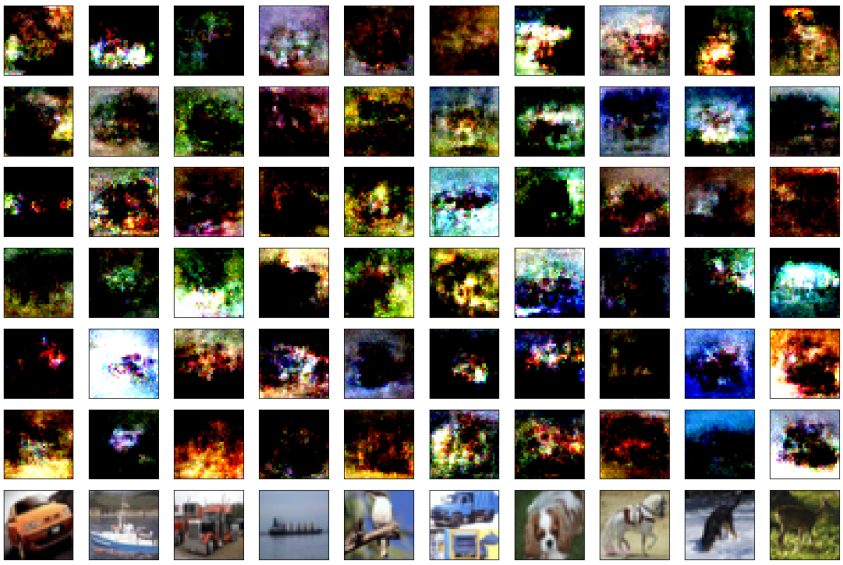


**Figure 10:** CIFAR10

The CIFAR models provide generations in which objects can be discerned - the coloring and the shading seems coherent and in continuity - although it is hard to tell to which class the images belong. Although the predicted likelihood indicates auxiliary dimensions are beneficial, it is not clear from the generated samples whether the MF's with a higher $B$ produce better images than the models with low $B$ or even with $B = 0$.

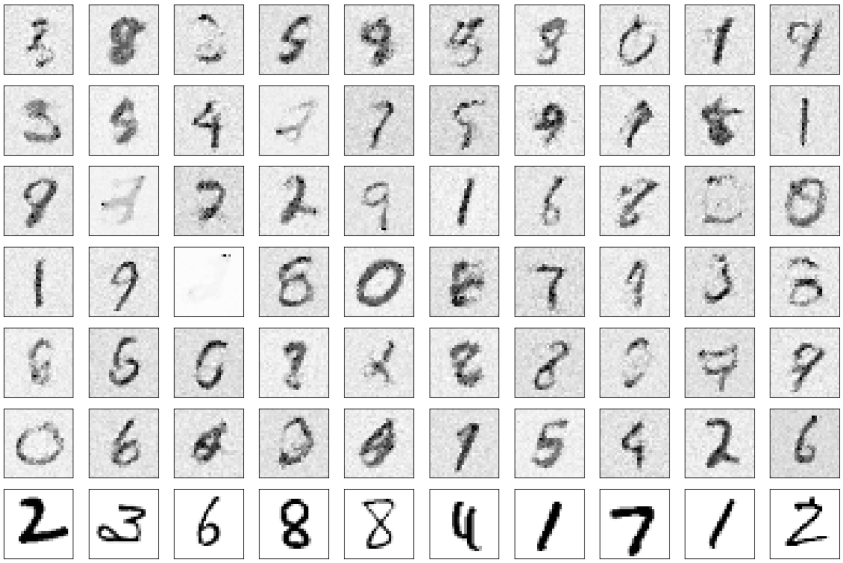


**Figure 11:** MNIST

The generated MNIST digits are generally good, and the latent classes can easily be distinguished. There is also some variability within the classes which suggests the models have picked up the general structure of the manifold well. As with the CIFAR models, the auxiliary dimensions do not seem to improve on the visual quality of the images.

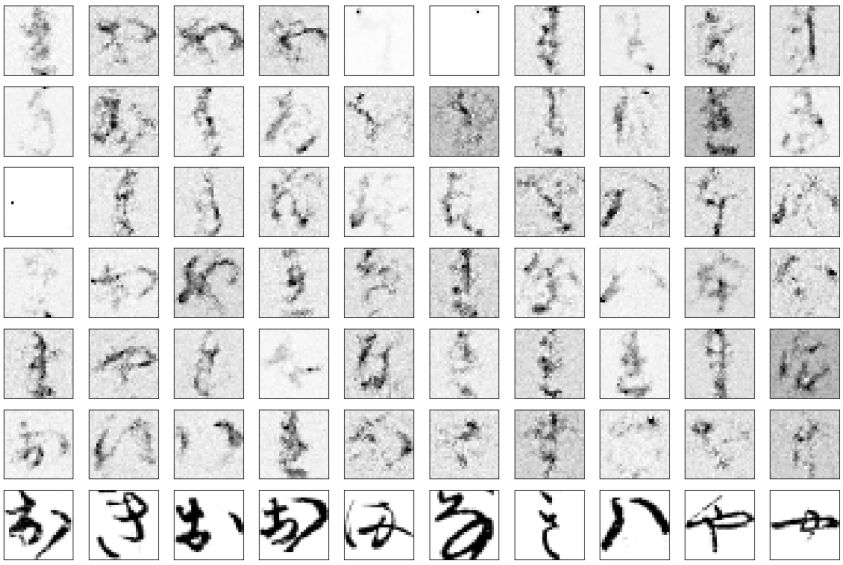


**Figure 12:** KMNIST

The Kuzushiji-MNIST is composed of ten classes of cursive Japanese characters, but those classes seem to contain a lot more intravariability then MNIST (see figure 15) making the dataset more challenging than MNIST. The models have picked up on the general shape of characters, but the generated samples are unclear and ambiguous. For someone with a very limited knowledge of the Japanese language it is already non-trivial to classify the images in figure 15, but it remains unknown whether even a native reader would be able to make sense of many of the generated samples in 12. On this dataset as well, there is no striking qualitative difference between the images on the first row, and the images on the second to last row.

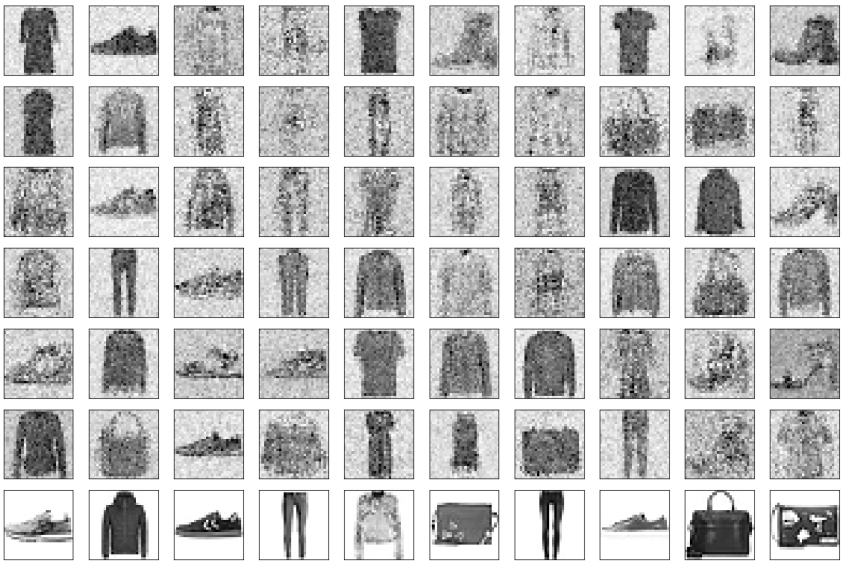


**Figure 13:** Fashion-MNIST

Models trained on Fashion-MNIST are capable of producing images which would

entice those with a predilection for spending lavishly on attire. Classes can easily be distinguished in many samples, although some images are ambiguous. Qualitatively, it appears as though the images on the sixth row are more detailed, with the dress (fifth from the left) having clear shading, and the shoe (third from the left) seemingly showing a line detail on the sole. These kinds of details can not be seen on the images produced by models with lower $B$.

Generally, there seems to be not much visual improvement in the samples generated by the different models. The models with higher $B$ seem to produce images of similar quality than models with a lower $B$.

# 4 Conclusion

For the Gaussian synthetic datasets, in the cases where $\psi > 1$, we saw that a higher $B$ consistently yielded a higher predicted likelihood, as well as a visually better approximation of the target density. This indicates that MFs may provide improvements, even in cases where the target transformation is a bijection; i.e. for $\phi = 3$. We also observed that when the target transformation is trivial (i.e. the case where $\psi = 1$), the likelihood predicted by marginalizing flows and normalizing flows lay very close to one another, indicating that marginalizing flows do not necessarily perform worse than normalizing flows.

On the half-moons data, a higher $B$ did not predictably increase the likelihood, but it did visually improve the approximation of the target density.

MFs do not predict a higher likelihood than normalizing flows on various versions of MNIST data, which we believe might be due to the essential simplicity of MNIST. More interesting results were found on data with a more complex structure; CIFAR10. There we see that as the number of auxiliary dimensions is increased, the predicted likelihood of CIFAR10 increases in a stable manner. Our results must all be seen in the light of the choice of architecture; we have only used the NVP architecture here, but MFs with a different architecture may yield different results.

All in all, our results seem to indicate that MFs outperform NFs on low-dimensional data. However, the exact conditions under which MFs converge to a higher optimum than NFs are subtle, and need more investigation.

# 5 Future Work

In our models we did not exploit the spatial properties of the image data as is done in [5]. The convolutions and squeezing operations used there could be used as an extension to the current method.

# References


[1] David M. Blei and Michael I. Jordan. Variational inference for Dirichlet process mixtures. *International Society for Bayesian Analysis*, (1):121—-143, 2006.

[2] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[3] Edmond Cunningham, Renos Zabounidis, Abhinav Agrawal, Ina Fiterau, and Daniel Sheldon. Normalizing Flows Across Dimensions. *arXiv preprint arXiv:2006.13070*, 2020.

[4] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 1(2):1–13, 2015.

[5] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019.

[6] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural Spline Flows. *Advances in Neural Information Processing Systems*, pages 7511–7522, 2019.

[7] Mevlana C. Gemici, Danilo Rezende, and Shakir Mohamed. Normalizing Flows on Riemannian Manifolds. *arXiv preprint arXiv:1611.02304*, pages 17–19, 2016.

[8] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[9] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

[10] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary Deep Generative Models. *arXiv preprint arXiv:1602.05473*, 2016.

[11] Jean Michel Marin, Kerrie Mengersen, and Christian P. Robert. Bayesian Modelling and Inference on Mixtures of Distributions. *Handbook of Statistics*, 25:459–507, 2005.

[12] Todd K. Moon. The Expectation Maximization Algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[13] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *arXiv preprint arXiv:1912.02762*, pages 1–60, 2019.

[14] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems*, pages 2339–2348, 2017.

[15] E Parzen. On the Estimation of Probability Density Functions and Mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[16] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *32nd International Conference on Machine Learning, ICML 2015*, 2:1530–1538, 2015.

[17] E. G. Tabak and Cristina V. Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[18] Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

# 6 Appendices

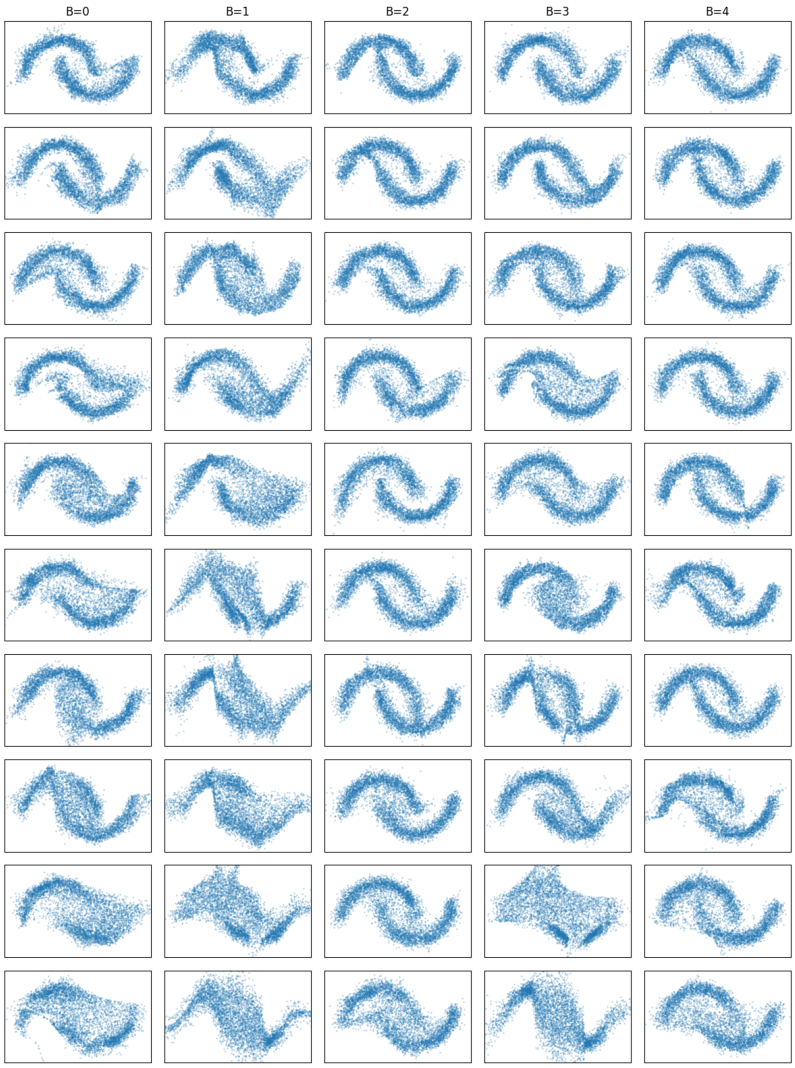



**Figure 14:** Generations produced by models trained on the halfmoons dataset, sorted from best to worst performing from top to bottom.

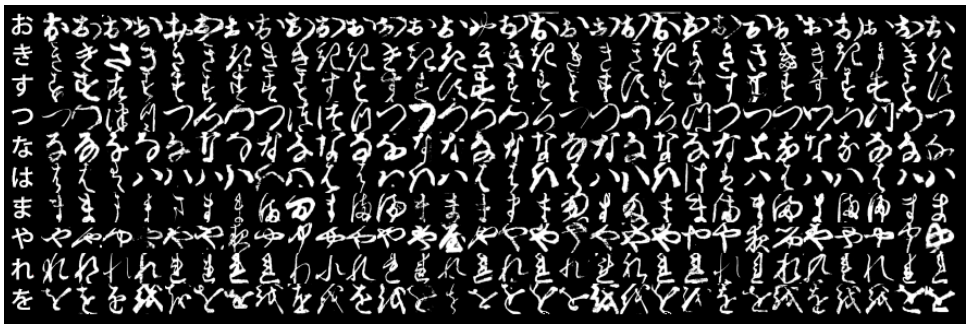


**Figure 15:** The ten classes in the KMNIST dataset. The rows correspond to the classes, with the leftmost character being the label.