DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Transport of Discontinuous Densities with Artificial Neural Networks

Michael Plainer

DEPARTMENT OF INFORMATICS

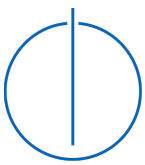
TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Transport of Discontinuous Densities with Artificial Neural Networks

Transport von Unstetigen Dichten mit Künstlichen Neuronalen Netzwerken

Author:	Michael Plainer
Supervisor:	Prof. Dr. Christian Mendl
Advisor:	Dr. Felix Dietrich
Submission Date:	September 15, 2021



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, September 15, 2021 Michael Plainer

Abstract

Nearly all real-world measurements can only record a part of the underlying truth due to technical limitations. In many fields, full comprehension of the system requires an understanding of how the unmeasurable inputs or states map to the measurable outputs. In cases where many individual measurements are performed, the density of the observation can be approximated with histograms. They count the frequency at which measurements fall in a given range. Each observed sample corresponds to exactly one unknown point in the input space that has been mapped by a function to produce exactly this recorded output. When the distribution of these points in the original input space is known (e.g. uniformly distributed), a transport function describing this mapping can be found. Identifying this transport function is the main objective of this thesis. The field of transportation theory is dedicated to finding these transportation maps between two (probability) measures that are optimal according to a metric. Those approaches can fail to identify the *true* underlying transport map, for example if it is not bijective or when the recorded density is discontinuous. Reconstructing this true underlying transport map can be done by employing an observation process that measures consecutive outputs of moving points. This reconstruction procedure is implemented with artificial neural networks and demonstrated by examples. Separately to the transport of measures, another network is implemented that learns the underlying dynamical system based on the observation process, allowing to extrapolate the movement of the points. Apart from fictitious examples, the procedure is also applied to reconstruct the shape of a simulated cell by synthesizing image data (e.g. produced by a microscope) and observing moving bacteria on the cell's surface.

Contents

1. Introduction

Conducting experiments, measuring their results, and adapting theories is the established scientific method. In many cases, the complete state of the underlying system can only be observed partially but not completely. However, access to a predictive system state is often needed to progress understanding and to create new results. Projecting a three-dimensional object to a two-dimensional picture, for example, already loses essential information that might be needed in some scenarios. Similarly, in scientific processes, observing measurements usually involves taking multiple samples, where the exact underlying process that produces the measurements is unknown—otherwise samples could be perfectly simulated. The number of underlying inputs needed to accurately map a point from the input space to the observed space varies. In some cases, an underlying one-dimensional space with a (complicated) map might be sufficient to express the observed values.

Measurements can be processed in multiple different ways, one of which is to create a histogram. With histograms, recorded values are categorized into non-overlapping ranges (i.e. bins) and the number of items in each bin is counted. The higher the frequency of a bin compared to others, the more likely a new sample falls into it. This process estimates the underlying density and can give a lot of insight on its own [49, p. 18]. For example, a satellite image could be used to create a two-dimensional histogram (i.e. a heat map) that shows where most people live. As bins get smaller and the sample size larger, this histogram converges to the true density of the population.

The main focus of this thesis is to reconstruct the true relation that maps between the unknown input space and the observed measurements, and vice versa. It pays particular emphasis to the case where the observed density suggests singularities, as it does in various real-world fields [1, 9, 43]. The optimal transport problem is one way to solve this and was first formulated by Monge [44]. It describes the problem of finding an optimal way to map between two densities [70, 71]. In this context, optimal typically means to minimize the distance individual points must be moved within the space.

Implementations of the optimal transport problem such as the numeric approach presented in [14] often fail to identify the *true* underlying relation, especially when discontinuous densities are involved [45]. Similarly, we will see that other approaches, such as normalizing flows [66, 67], can fail too. However, identifying the true system is needed in fields such as domain adaption [13]. In this thesis, we will follow the

procedure proposed in [45] to reconstruct the true underlying relation. Moosmüller, Dietrich, and Kevrekidis rely in [45] on an observation process that collects multiple measurements from the underlying moving points, instead of only collecting single values. In the previous example of the satellite image producing the heat map, this would mean that multiple pictures are taken to record people's movement. With these additional time delayed measurements, individual points (e.g. people) can be observed and finding the true relation becomes possible (up to isometry). The ideas of Takens' theorem [2, 48, 68] are used to find a similar (i.e. diffeomorphic) solution.

This thesis reproduces most results of the underlying paper [45] but adapts and extends its approach to use artificial neural networks for transportation between the densities. For this, neural normalizing flows are used that can find a series of invertible functions transporting between a simple and a complicated density [38, 50]. The reconstruction procedure is also exercised with a practical application. A microscope takes two-dimensional pictures of a simulated three-dimensional spherical cell with bacteria moving on its surface. Since the picture is only a partial reconstruction of the space, a circle is recorded instead of a sphere. Using our adjusted transportation procedure, the three-dimensional position of the bacteria can be reconstructed. As the bacteria lie on the surface, this reconstructs the cell's shape from two-dimensional measurements alone. The reconstruction often still produces some unexpected artifacts, thus further research is required.

We also illustrate and implement learning of the underlying dynamical system. Understanding the dynamics, allow to predict and extrapolate the movement of the underlying points. To do this, an artificial neural network approximating the vector field that defines the underlying movement is trained based on the ideas presented in [57], but additionally incorporates data from the observation process.

This thesis is composed of two main chapters. In Chapter 2, the current state of the art is presented containing mostly the mathematical foundations and theoretical background needed. Chapter 2 also introduces the origin and use of transportation theory, illustrates how normalizing flows can transport measures, and summarizes and discusses the paper [45] this thesis builds upon. Chapter 3 focuses on the concrete problems and implementation of our adapted procedure. It presents the reconstruction procedure by explaining it with a one-dimensional example and then illustrates how it can be generalized to solve higher-dimensional problems. It continues by showing how neural networks can be used to learn the dynamical system and concludes with a demonstration of the reconstruction of the shape of a simulated cell.

2. State of the Art

Density transportation allows us to find a function that can transform one density to another. This can and has been used over the last years in a variety of fields, especially in economics. For example, it can be used to match machines to resources to maximize throughput, or to determine the price of assets when the price depends on the value of other financial assets [25]. In this part, we will see that there is a lot of research concerning the transport of densities available and will discuss relevant ones.

This chapter is devoted to the current state of the art in density transportation with focus on discontinuous densities. It serves as an introduction to the field, beginning with the mathematical fundamentals in Section 2.1 that are needed for the rest of this thesis. There, we will discuss the underlying definitions for densities, transportation theory, and also give intuition for manifolds. Afterwards, we will take a look at normalizing flows in Section 2.2, a procedure that learns an underlying distribution and allows transportation from and to densities based on a sequence of "simple" functions. Artificial neural networks are described in this section as well. They can facilitate this process to make the model of normalizing flows more expressive. This chapter concludes with an extensive review of [45] in Section 2.3 that is the foundation for this thesis. The definitions for a discontinuous density are presented and the problem with existing approaches is outlined. To conclude, the modified approach is presented that allows finding the "true" transport map, even for discontinuous densities.

2.1. Fundamentals

This section is dedicated to presenting (mathematical) background knowledge. If the reader is already familiar with this knowledge, they can safely continue with Section 2.2. Especially Section 2.1.1 is very technical in nature and serves more as a theoretical underpinning than a necessity for understanding this thesis.

2.1.1. Measure theory

In many different fields, the goal is to measure something, such as the length of an interval, the volume of an object, or the probability of some event happening. While (basic) probability theory does not require measure theory, many aspects become easier

or get a more concise definition with it [55, Ch. 1]. The next paragraphs mostly follow the definitions from [55, Ch. 2–3] and give a brief overview of measure theory with a focus on the probabilistic background.

The triple $(X, \mathcal{A}, \mu)$ is called a *measure space* and is the core underlying object of measure theory. X is a set and describes the general space we operate in. In this thesis $X \subseteq \mathbb{R}^n$ will always be the case. $\mathcal{A}$ is a σ-algebra over X, meaning that it is a set of subsets of X that contains all measurable objects (i.e. subsets). μ is the measure itself: a function that assigns every measurable object a non-negative volume. The following definitions formalize these descriptions.

Definition 1 (σ-Algebra)**.** *Given a set of subsets $\mathcal{A} \subseteq \mathcal{P}(X)$, it is a σ-algebra of X if*

1. $X \in \mathcal{A}$,
2. *if $A \in \mathcal{A}$ so is its complement $(X \setminus A) \in \mathcal{A}$,*
3. *if countable many $A_1, A_2, \ldots \in \mathcal{A}$, so is their union $\cup_i A_i \in \mathcal{A}$.*

Definition 2 (measure)**.** *The function $\mu : \mathcal{A} \to \mathbb{R}^+ \cup \{\infty\}$ is a measure if*

1. $\mu(\emptyset) = 0$,
2. *if all countable many pairwise disjoint $A_1, A_2, \ldots \in \mathcal{A}$ can be measured individually to calculate the combined volume $\mu\left(\cup_i A_i\right) = \sum_i \mu\left(A_i\right)$.*

Additionally, it is called a probability measure *if $\mu\left(X\right) = 1$ also holds.*

Those definitions might seem daunting at first, but when taking a more practical approach they become more intuitive. μ can measure all the elements of our σ-algebra $\mathcal{A}$ and thus it can also measure the complete set X. If μ can measure a set Y, the complement can be trivially measured by subtracting the volume of Y from the volume of the complete space X. Similarly, if multiple disjoint sets can be measured, the complete volume can be measured by summing up the individual parts. Note that μ can only measure objects of the underlying σ-algebra $\mathcal{A}$, and usually there are multiple ways with which the volumes can be assigned by a measure μ.

As this thesis is about probability and densities, we will focus on probability measures specifically. Every probability measure can be thought of as a probability function $\mathbb{P}$ and vice versa. This gives existing concepts new interpretations in measure theory. In classical discrete probability theory, the *density* can be thought of as the probability for a random variable to have a specific value (i.e. $\mathbb{P}\left[X = x\right]$). In the continuous case, this does not hold anymore, since the probability for a single event $\mathbb{P}\left[X = x\right]$ is always

0 and the density at an individual point can be greater than 1. The density of the uniform distribution on $[0, 0.5]$, for example, is 2 in this range. Continuous densities, as in the opposite of discrete densities, allow us to calculate the probability of an area by calculating the integral. Such a continuous density is illustrated in Figure 2.1.

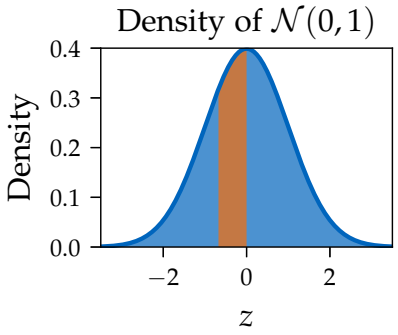


Figure 2.1.: This plot shows the density of the standard normal distribution. While the probability for an individual point is zero, it can be calculated for a range of z. The area highlighted in orange corresponds to the z values that occur 25% of the time.

In measure theory, the integral over a density that describes the probability of a range becomes a way to express one measure with the means of another. Typically, one uses the simplest measure for such an expression, the Lebesgue measure λ. It measures the length of an interval or, in higher dimensions, the volume [55, Ch. 1]. If the measure μ is absolutely continuous with respect to λ (meaning that $\forall A \in \mathcal{A} : \lambda(A) = 0 \implies \mu(A) = 0$), μ can be re-expressed by λ with the density f_μ such that

$$\mu(A) = \int_A f_\mu(x)\, d\lambda(x), \text{ with } f_\mu : X \to \mathbb{R}^+, \tag{2.1}$$

as stated by the Radon-Nikodym theorem. This gives densities a different and, in some cases, more intuitive and flexible interpretation.

2.1.2. Transportation theory

Optimal transportation theory aims to move *mass* from one density to another while minimizing a defined cost [12]. Optimal transport was first formalized by Monge in 1781 [44] describing the problem of finding the best way to move a heap of soil into a hole (e.g. with a shovel) [70, 71]. Usually, the "best" transport is described by the one that minimizes the overall distance that "shovels full of dirt" have to be moved [71]. Later, the work of Kantorovich on couplings [34, 35] led to a relaxation of the original problem and it has since been referred to as the Monge-Kantorovich problem [70,

71]. Contrary to Monge's problem, Kantorovich's formulation allows the mass at an individual point to be split and moved to multiple destinations [53, 70]. This relaxation led to more progress and a rebirth of the field of transportation theory over the past few decades [71]. In the following, we will focus on the formulation of Monge using modern notation.

Going back to the original interpretation, the pile and the hole both have their own distinctive shape and need to have the same volume for the sand to exactly fill up the hole [70]. Thus, the goal is to transform one function to another with equal volume. In this work, the scale of those functions does not matter, so all the volumes are scaled to 1. As for the probability measures, μ and ν with their respective underlying measure spaces X and Y (omitting the σ-algebras) are used. This notation and the problem are visualized in Figure 2.2.

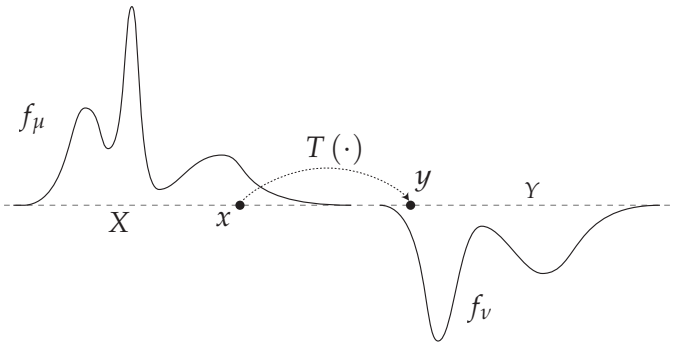


Figure 2.2.: Two density functions f_μ and f_ν can be seen with their underlying spaces X and Y, and their respective measures μ and ν. Continuing the example from earlier, the goal is to transport the heap of sand f_μ into the hole f_ν. Every point of mass $x \in X$ is transported to some $y \in Y$ based on a transport map T.

The aim is to find a function that transforms the original measure μ into ν, the so-called *push-forward* measure. A map $T : X \to Y$ pushes μ forward to ν if T satisfies

$$\nu(A) = \mu(\{T(x) \in A \mid x \in X\}) = \mu\left(T^{-1}(A)\right). \tag{2.2}$$

for all measurable sets $A \subseteq Y$. To say that ν is the push-forward of μ, we write $\nu = T_{\#}\mu$ [42, 45, 53]. Intuitively, Equation 2.2 enforces the constraint that every bit of volume in the measure ν (e.g. sand) has to come from an equally sized volume in μ. This map T can be seen as a function that takes a single point of mass from X and maps it to Y (e.g. move one shovel of sand). The push-forward $T_{\#}\mu$ on the other hand moves an entire probability measure instead of just single points [53].

Since this thesis is about transporting densities, it makes sense to reformulate Equation 2.2 by plugging in the definition for densities from Equation 2.1, yielding to

$$\nu(A) = \int_A f_\nu(y)\, d\lambda(y) = \int_{T^{-1}(A)} f_\mu(x)\, d\lambda(x) = \mu\left(T^{-1}(A)\right). \tag{2.3}$$

While finding any arbitrary push-forward measure might be enough under some circumstances, generally, the goal is to find one that is *optimal* in some regard. For this, a cost function is needed that governs the optimization process and determines the optimal transport map. The cost function $c : X \times Y \to \mathbb{R}^+$ describes the cost of moving one unit of mass, namely $x \in X$, to $y \in Y$ [70]. With this definition, the optimal push-forward map

$$\min_T \left\{ \int_X c(x, T(x)) d\mu(x) \;\middle|\; T_\# \mu = \nu \right\} \tag{2.4}$$

that minimizes the cost globally can be found [42, 53, 70]. While in principle any arbitrary cost function can be used to define the optimum, in practice, the squared Euclidean distance is typically chosen. It is well-researched, many algorithms are optimized for it [5, 23, 53], and there exists a unique, monotonically increasing solution [45].

Optimal transport and the theory behind it have gained much attention over the last decades. Its applications range from brain tractograms to better understand nerve tracts [22] to various economic areas [25].

2.1.3. Manifolds and embeddings

While the basic notion of manifolds is rather intuitive, especially in the lower-dimensional case, a precise definition can be difficult to formulate and to grasp [40, Ch. 1]. For this thesis, no mathematical notation for manifolds is needed and this section only serves as a brief introduction. For a more technical definition and treatment see for example [40].

Manifolds are an extension of curves and surfaces into a higher dimension [69, p. 47] that one can think of as an arrangement of points. By this definition, all objects of the "real" world can be thought of as manifolds. For example, a string can be seen as a curve and thus a manifold, or a sheet of paper that can be viewed as a bounded plane.

A key property of all manifolds is their intrinsic dimension. It can be interpreted as the minimum number of parameters needed to describe a single point on such a manifold [40, Ch. 1]. For a curve for example, a single parameter, such as the cumulative length, is enough to describe every point uniquely. Thus, the dimension of every curve is one, and it is a one-dimensional manifold. Analogously for a surface, a x and a y coordinate can be used to describe every point locally, meaning that it is a two-dimensional manifold.

A hollow sphere on the other hand is generally an object that can only be visualized in three dimensions. But contrary to intuition, it is a two-dimensional manifold. Two coordinates, the longitude and latitude are sufficient to describe a point on the sphere, but they are not continuous functions [40, Ch. 1]. This local two-dimensionality can be experienced every day—locally, the earth can be seen as flat, but this assumption does not hold anymore when viewing the object as a whole.

And this is the second important property of manifolds: an n-dimensional manifold can be described locally around each point as the Euclidean space $\mathbb{R}^n$ [40, Ch. 1]. But, that does not mean that it can be *embedded* into an $\mathbb{R}^n$-dimensional Euclidean space. Here, *embedding* can be taken literally and describes whether one object can "hold" another. Mathematically, it means finding a diffeomorphic function to put one manifold into another space [69, Ch. 3]. In this context, a function is diffeomorphic if it is invertible on the image of the function and both the function and the inverse are smooth (i.e. differentiable) [38, 50].

Although the sphere is a two-dimensional manifold, it cannot be embedded into two dimensions but only in three or more. Whitney's theorems [73] give a limit on how many dimensions are needed to embed a manifold. In this thesis, embeddings are crucial to transport densities, and we will assume that there is access to multiple time delayed measurements. Based on the embedding results of Whitney's theorems, Packard, Crutchfield, Farmer, and Shaw [48], Aeyels [2], and Takens [68] describe a way to use time delays to reconstruct the system attractor [45].

These ideas, referred to as Takens' theorem, state that with enough time delays collected, a diffeomorphic version of the underlying manifold can be reconstructed. Generally, one needs to collect $2d + 1$ time delayed measurements of a dynamical system to find such an embedding, with $d \in \mathbb{N}$ being the dimension of the underlying manifold. The use of this theorem will become clearer in later sections and serves as the theoretical foundation.

2.2. Transport of densities with normalizing flows

In recent machine learning applications, the core goal is often to approximate the underlying distribution given a set of measurements (i.e. samples) [38, 50]. Doing this accurately enough enables two main things: 1) create more data by generating new points based on the underlying distribution, and 2) determine how likely a measurement is, or falls into a certain range (e.g. applicable in outlier detection) [38]. Over the last few years, the first objective has been worked on extensively.

One major contribution in this field that has received significant attention are genera-

tive adversarial networks (GANs) [26] proposed by Goodfellow *et al.* They combine two models: a generator and a discriminator. The generator creates new samples while the discriminator tries to distinguish them from the original dataset [26]. With this setup, the two models can train each other. This simple yet powerful approach leads to many advantages [42], such as its great sampling dynamics [26, 38]. GANs also have their drawbacks such as that this configuration does not allow determining the density of new points (i.e. the second item from the enumeration before) [38].

Similarly to GANs, normalizing flows (NF) [66, 67] are also probabilistic generative models that try to learn the underlying distribution in an unsupervised manner [38]. Due to their construction, they allow for efficient sampling *and* scoring of data points. Apart from outlier detection, this can be useful in many areas, such as in classifying samples, or calculating the expectation of the process [49, p. 12]. It can also be used to validate the model with statistical tests, such as a one-sample goodness-of-fit test [33, 49]. NFs are a crucial part of this thesis and will be discussed in the following.

The general goal of normalizing flows is to find a series of invertible functions that transform an observed complicated distribution to a simple known distribution (e.g. normal distribution). When applied, those functions can then gradually transform samples from one distribution to the space of a simple one, as visualized in Figure 2.3. This setup is also the origin for the name normalizing flow. The data points flow through the function(s) until the "normal" form is reached [38]. Since the functions are computationally invertible by construction, the reverse direction that transforms a normal density into the complicated density, is also feasible.

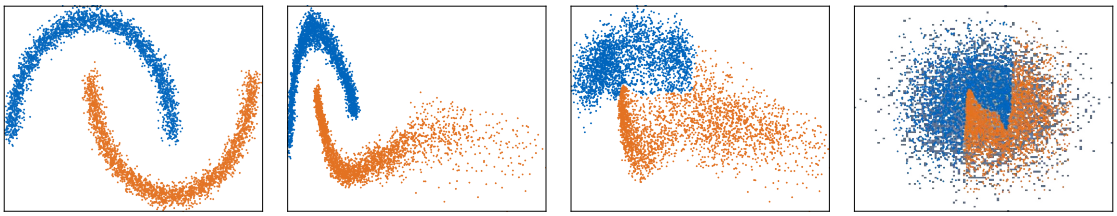


Figure 2.3.: On the left, $5,000$ samples are shown that are drawn from a "complicated" two-dimensional distribution. Through a series of three diffeomorphic functions, those samples are transported to a two-dimensional standard-normal distribution (left to right). In the right-most frame, points that follow a true standard-normal distribution are plotted beneath the transformed samples in gray for comparison.

More formally, assume that the measured samples are real valued D-dimensional vectors $\mathbf{x}_1, \mathbf{x}_2, \cdots \in \mathbb{R}^D$ drawn from a complicated distribution. Flow-based modeling tries to express each of those data points $\mathbf{x}$ by applying a transformation T to a random

variable $\mathbf{u}$ of dimension D [50]. With the correct choice of T, the distribution is then defined by

$$\mathbf{x} = T(\mathbf{u}) \quad \text{with} \quad \mathbf{u} \in \mathbb{R}^D, \mathbf{u} \sim f_u(\mathbf{u}). \tag{2.5}$$

f_u is the probability density function of u and is almost always the density of a normal distribution (typically the standard normal) [38, 49, 50]. The goal—and also the difficult part—is to find an approximation of T based on the observed samples that also has an (easy to find) inverse. More specifically, T needs to be *diffeomorphic*, meaning that the inverse exists, and both the function and its inverse are differentiable [38, 50].

Under those conditions, the change of variables formula [8, pp. 194–197] can be applied and the target density function f_x can be reformulated by the means of the simple known density to

$$\begin{aligned} f_x(\mathbf{x}) &= f_u\left(T^{-1}(\mathbf{x})\right) \left|\det J_T(\mathbf{u})\right|^{-1} \\ &= f_u\left(T^{-1}(\mathbf{x})\right) \left|\det J_{T^{-1}}(\mathbf{x})\right|, \end{aligned} \tag{2.6}$$

as is done in [38, 50]. $J_T(\mathbf{u})$ is the $D \times D$ Jacobian

$$J_T(\mathbf{u}) = \begin{bmatrix} \frac{\partial T}{\partial u_1} & \cdots & \frac{\partial T}{\partial u_D} \end{bmatrix} = \begin{bmatrix} \frac{\partial T_1}{\partial u_1} & \cdots & \frac{\partial T_1}{\partial u_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial T_D}{\partial u_1} & \cdots & \frac{\partial T_D}{\partial u_D} \end{bmatrix}, \tag{2.7}$$

containing all first-order partial derivatives of T. T is then the push-forward of f_u to f_x, namely $f_x = T_\# f_u$ [38]. Note that this is the push-forward of a *density*, not a measure as used before. The change of variable can be viewed as T warping the underlying space to fit the densities and $|\det J_T(\mathbf{u})|$ correcting the volume so f_x still integrates to 1 [50].

While this gives us a way to express the new density f_x based on f_u with a given transformation T, it says nothing about its existence or how to approximate it. It has been formally and constructively proven that, under reasonable assumptions for the densities f_u and f_x, there exists a diffeomorphism T that pushes f_u forward to f_x [7, 50]. Constructing this arbitrarily complex diffeomorphism may be difficult and evaluating this function at a point, inverting it, or calculating the determinant of the Jacobian may be computationally infeasible. To overcome this problem, one typically uses a composition of multiple simple diffeomorphic base functions [38]. These diffeomorphic transforms can then be applied in sequence:

$$T = T_1 \circ \cdots \circ T_N. \tag{2.8}$$

This composition of diffeomorphisms can be seen as a single complex *diffeomorphic* transformation. It can still be inverted easily by inverting each function individually

and reversing the order. $|\det J_T(\mathbf{u})|$ can also be calculated by applying the chain rule from calculus, and multiplying the individual $|\det J_{T_i}(\mathbf{u})|$ [38, 50].

From those multiple individual functions, a complex diffeomorphism can be constructed. The flexibility and complexity of those base functions determine the computational performance and the expressiveness [38, 50]. A trade-off between those two needs to be found. One intuitive approach would be to perform a linear transformation such that

$$T_i(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} \quad \text{with} \quad \mathbf{A} \in \mathbb{R}^{D\times D}, \mathbf{b} \in \mathbb{R}^D. \tag{2.9}$$

Those linear flows are simple and easy to calculate, but they bring some inherent problems. The most crucial disadvantage is that when starting with a normal distribution, a linear transform will only result in another normal distribution with different expectation and variance [38]. Meaning that a unimodal normal distribution could never be transformed into a higher-modal distribution. The same holds for distributions of the exponential family [38]. Another issue lies in the fact that some operations (such as the determinant of the Jacobian) generally take $\mathcal{O}(D^3)$. Especially for higher-dimensional data this might become a problem. To circumvent it, special matrix forms could be used, but while diagonal matrices are easy to work with, they cannot express correlation between dimensions. Triangular matrices are hence often a good middle ground [38].

2.2.1. Artificial neural networks

With the popularity and advantages of neural networks one might be wondering if and how they can be applied in the context of normalizing flows. In practice, neural networks are often used to implement the individual diffeomorphic composite functions $T_1, \ldots, T_N$ (see Equation 2.8) [50]. This section serves as a brief overview of artificial neural networks and can be safely skipped by readers familiar with the foundations. It is loosely based on [3].

A neural network can be viewed as a function Θ with many internal parameters that govern how an input x is mapped to an output y. The goal is to adapt those parameters in a way that function behaves as desired. Typically, this neural network consists of multiple so-called layers, each of which containing neurons. In its simplest form, the neurons of one layer are only connected to the neurons in the subsequent layer. The first layer serves as the input of the function and is referred to as the *input layer*. Analogously, the last layer is the actual output of the function and is the *output layer*. This structure is depicted in Figure 2.4.

Based on the input of the function (i.e. the values of the neurons in the input layer), the values propagate through the network. The value of neurons in layers after the input layer are simply a weighted sum of the values of the preceding neurons, as

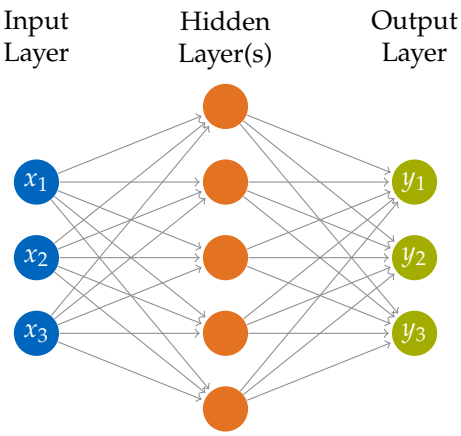


Figure 2.4.: An artificial neural network consisting of three layers. The circles depict the neurons, and the arrows indicate connections between them. A classical neural network can be visualized as a directed acyclic graph.

indicated by the arrows in Figure 2.4. Those weights are the internal parameters of the function and subject to optimization. More formally, the value of a neuron n is $\sum_{m \in N(n)} \Phi\left(W_n^m m\right)$. With $N(n)$ being the neurons that are connected to n, $W_n^m \in \mathbb{R}$ the weight of the connection between neurons n and m, and Φ an in principle arbitrary function. It is the so-called *activation function* and ensures that the network can learn non-linear relations. Often, the sigmoid function $\frac{1}{1+e^{-x}}$ is used.

Another important function is the *loss function* that defines how accurate the output is compared to the ground truth. There are many choices available, but often the mean squared error function is used. With this loss function and samples to compare the output to, an optimization problem can be solved that minimizes the error. The most common technique to solve this optimization problem relies on backpropagation [61], a procedure that determines the gradients of the weights based on the results of the loss function. An optimizer can then adjust the weights based on the gradients and approximate the optimum step-by-step. One simple such optimizer is stochastic gradient descent [58].

The success and ease of this optimization process is greatly governed by the hyper-parameters of the artificial neural network. Such as the number and size of layers, the optimizer used, and the specific choice of functions (e.g. activation, loss). Those hyper-parameters are often searched by an algorithm or determined by trial and error. Once suitable hyper-parameters are found, and the neural network has been trained, it can be used in the desired scenario to make predictions for previously unseen data.

2.2.2. Normalizing flows with artificial neural networks

Artificial neural networks have proven to be a versatile tool in a wide variety of problems and are also used at the core of many normalizing flows algorithms. But two major issues prevent arbitrary complex networks from becoming a drop-in replacement for classical transformations. When the artificial neural network is the function T itself, the inverse T^{-1} must exist. Generally, networks can be constructed in a way that the inverse exists [30] but calculating it may become computationally infeasible with a complex neural network. The second issue is of the same nature: calculating the determinant of the Jacobian [50]. While the problem of invertibility can be overcome by imposing restrictions on the weights and activation function [30], ensuring an easy way to calculate the determinant requires a more elaborate architecture.

There are multiple ways to change the construction of those T_i to overcome these two problems and use arbitrarily complex neural networks. One of the more prominent ways are *coupling flows* that were introduced by Dinh, Krueger, and Y. Bengio in [17].

Coupling flows

Coupling flows are constructed such that the inverse of the transform and the determinant of the Jacobian can be trivially obtained [17]. To achieve this, the authors propose to split the input vector $\mathbf{x} \in \mathbb{R}^D$ into two parts $\mathbf{x}_{1:d-1}, \mathbf{x}_{d:D}$. The first part is used to compute parameters Θ with an arbitrarily complex neural network $\Theta(\mathbf{x}_{1:d-1}) = \mathrm{NN}(\mathbf{x}_{1:d-1})$. The first part of the input vector, $\mathbf{x}_{1:d-1}$, remains unmodified, but the resulting parameters Θ are used as an input for a diffeomorphic function g, the *coupling transform*. g maps $\mathbf{x}_{d:D}$, the second part of the input, based on Θ. More concisely,

$$\begin{aligned} \mathbf{y}_{1:d-1} &= \mathbf{x}_{1:d-1} \\ \mathbf{y}_{d:D} &= g\left(\mathbf{x}_{d:D}; \Theta\left(\mathbf{x}_{1:d-1}\right)\right). \end{aligned} \tag{2.10}$$

This architecture is illustrated in Figure 2.5.

When incorporating the artificial neural network this way, the complete process can now be easily inverted. Since the first part of $\mathbf{x}$ has not been changed, the neural network can be rerun to create the same parameters Θ again. And since g has a computationally efficient inverse by construction, it can be determined based on the parameters Θ. Moreover, as shown in [17], the Jacobian has a triangular form, meaning that the determinant simply equals the determinant of g [38]. So, an arbitrary complex neural network can be used without the need to invert it or to calculate the determinant of the Jacobian of the network itself.

Two questions remain unanswered: how to partition the vector $\mathbf{x}$ into two parts, and how to choose g. Typically, the vector $\mathbf{x}$ is simply split in half [38] but in some

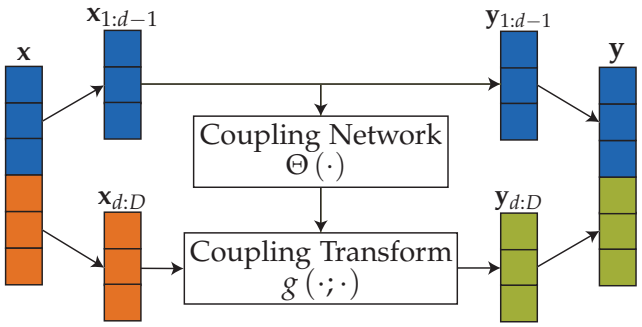


Figure 2.5.: This figure visualizes the forwards pass of coupling flows. The coupling network only operates on part of the data, $\mathbf{x}_{1:d-1}$, and outputs the parameters used to transform the other part, $\mathbf{x}_{d:D}$, with the actual coupling transform g [38, 50].

architectures, more elaborate methods are used [18]. Regarding the second question, usually the coupling transform g is applied to every element individually. The neural network then returns one parameter for each dimension, where those dimensions often have their own function g [38]. More formally,

$$\mathbf{y}_{d:D} = \left(g_d\left(\mathbf{x}_{d:D}; \Theta_d\left(\mathbf{x}_{1:d-1}\right)\right), \ldots, g_D\left(\mathbf{x}_{d:D}; \Theta_D\left(\mathbf{x}_{1:d-1}\right)\right)\right). \tag{2.11}$$

For the specific choice for the coupling transform(s), there are many possibilities, as listed in [38]. In the original paper [17], a simple additive coupling $g(x;\Theta) = x + \Theta$ or affine coupling $g(x;\Theta) = \Theta_1 x + \Theta_2$ is used. But these transforms are not universally expressive and multiple coupling flow layers are needed to describe complex distributions [38, 50]. When stacking multiple coupling layers after another, it is often useful to permute the input vector $\mathbf{x}$ to improve the model's expressiveness [21, 38, 50].

Neural spline flows

While these aforementioned simple coupling transforms are often sufficient, more sophisticated approaches such as those based on splines typically yield better results [21]. Splines [62, 63] are a way to interpolate between multiple points. Typically, polynomials of the same degree are used to connect two points (so-called knots) [38]. As the degree of the polynomials increases, more degrees of freedom for the parameters become available while still connecting neighboring knots. Those degrees of freedoms are usually filled by enforcing constraints on the derivatives. In most applications, splines consist of only low order polynomials that are easy to invert and calculate [50]. The number and position of those knots parameterize the actual function.

There are multiple instances where different forms of splines have been used as a coupling transform, such as in [20, 21, 46]. Contrary to those, Durkan, Bekasov,

Murray, and Papamakarios propose in [21] an approach based on monotonic rational-quadratic splines. This special spline is a piecewise function that divides two quadratic polynomials. The neural network that determines $\Theta(\mathbf{x}_{1:d-1})$ of the coupling flow outputs the parameters for the splines (i.e. knots and derivatives) [21]. Each $\mathbf{x}_i$ is then transformed with its own rational-quadratic spline [21].

This approach is a direct replacement of the affine or additive layers described in [17] for the coupling transform g. The experiments performed in [21] show that neural splines can learn complex distributions more quickly and need fewer parameters compared to other transformations, while still working in higher dimensions.

Library for normalizing flows in Python

Many open-source implementations for normalizing flows in Python are available but since they are relatively new, few implementations provide a mature API. For this thesis, we decided to use neural spline flows with a coupling architecture. They provide enough flexibility for the given problems and learn relatively quickly. As for the specific library, Pyro [6, 54] was used. It builds on top of the machine learning framework PyTorch [51] and adds functionality for working with probabilities.

2.3. Transport of discontinuous densities

Normalizing flows are a great tool to transport densities, and particularly their invertibility allows them to be used in a variety of fields. As we will see later, this invertibility is the reason that normalizing flows often fail to approximate the true underlying transport map. Especially, when dealing with discontinuous densities finding the transport map with normalizing flows directly is usually not possible.

We will now revisit the definition of densities and define what makes a density *discontinuous*. Similarly to [45], only two types of discontinuous functions are of interest in this thesis: those with a jump discontinuity, and those diverging to infinity.

Definition 3 (Discontinuity)**.** *A density function f has a jump discontinuity at $a \in \mathbb{R}$ if*

$$\lim_{x \to a^-} f(x) \neq \lim_{x \to a^+} f(x). \tag{2.12}$$

A density function f diverges to infinity at $a \in \mathbb{R}$ if

$$\lim_{x \to a^-} f(x) = \infty \quad \text{or} \quad \lim_{x \to a^+} f(x) = \infty. \tag{2.13}$$

While it may not be intuitive at first, discontinuous densities can arise easily even with a "nice" underlying definition. For this, we will look at densities defined by

their push-forward from the uniform density. When pushing a continuous density f_μ forward with T to f_ν, discontinuities can arise depending on certain properties of T. For a point x where the map T is flat or folded (i.e. $T'(x) = 0$), the new density f_ν diverges to infinity at this point. A jump discontinuity can either be caused by a non-continuous derivative of the transport map or might arise if T is non-injective [45].

The basic scenario described in [45] is that objects are distributed uniformly on an unknown manifold, move with the same pattern, and measure variables. Often, those measurements only reveal a part of the variables (i.e. only some dimensions of the manifold) and the goal is to reconstruct it fully. This can be seen as reconstructing the original underlying map T. Moosmüller, Dietrich, and Kevrekidis do this by relying on a so-called *observation process* capable of measuring subsequent values. This section summarizes the main ideas of [45] and while it is particularly important as a foundation for the main part, Chapter 3, some parts may be hard to understand at first and will become clearer with the examples presented later.

When the sampling process of some variables gives rise to discontinuous densities, it can become difficult to determine the true underlying relation (i.e. manifold). For a complete understanding of the process and to make new predictions, knowledge of this exact manifold is needed. A typical way to recover such a manifold is to solve the optimal transportation problem to map the observed density to the original space with minimal cost. This results in a manifold that will produce an identical density but might be different from the real manifold. Such a direct approach is tried in Section 3.1.3 with a normalizing flow that yields "incorrect" results. In [45] and in this thesis, the major point of interest is to find the true underlying manifold.

With only a single measurement, reconstruction of this true manifold is not possible. To overcome this problem, the authors of [45] use Takens' theorem (see Section 2.1.3) and collect multiple measurements from each moving object on the underlying manifold. With those additional measurements, a diffeomorphic manifold can be constructed that is then used as a starting point instead of the discontinuous density itself.

While this observation process adds more data, the dimensions typically do not match anymore making it incompatible with transportation. To overcome this issue, the map is parametrized such that the manifold can be embedded in a lower dimension. With this step, transportation is again possible and as this version of the manifold is already diffeomorphic to the underlying true solution, classical transport algorithms can reconstruct the true underlying manifold (up to isometry). Another advantage of this process is that by increasing the dimension with time delays, observed densities with a lower dimension than the original underlying manifold can be transported, for example densities of marginal or conditional distributions. This would not be possible without this procedure, as transportation only works with matching dimensions.

3. Transport of Discontinuous Densities with Artificial Neural Networks

The work presented in [45] is the foundation of this thesis. Many of the ideas of [45] are discussed in this chapter and some of the results recreated. The one-dimensional example from [45] is reproduced in Section 3.2.1 to Section 3.2.3 and the ideas of the two-dimensional case are used in Section 3.4.1 to Section 3.4.5. We extend the ideas so that artificial neural networks (with normalizing flows) are applied and show most intermediate steps that were left out in the original work.

This chapter begins with Section 3.1 where the concrete problem is presented with the help of an example. How normalizing flows can be applied to the problem and why a direct solution can fail is the first topic discussed. Section 3.2 then continues by presenting a way to overcome the problems of the direct approach by including an observation process to recover the manifold. After some additional notes regarding this procedure, a second problem is presented in Section 3.3 that is not discussed in [45]. For the observation process, we assume that objects move on the underlying manifold and time delayed measurements can be taken. This section is dedicated to the underlying dynamical process and how it can be learned with neural networks.

These sections only look at the problem in the one-dimensional case. Section 3.4 is devoted to describing how these procedures can be generalized by applying them to a two-dimensional toy example. It also discusses how a marginal or a conditional density can be transported to a joint density. Section 3.5 shows how this process could be applied in the real world by recovering the manifold of a simulated cell by observing the movement of bacteria on it.

3.1. Overview

The theoretical part already describes the process and requirements for recovering a manifold in detail. This section is dedicated to giving a clear overview and intuition of the problem itself and shows how normalizing flows can be used in practice. With this knowledge, we try to apply normalizing flows directly to solve the presented problem.

3.1.1. Problem description

We will begin with an exemplary one-dimensional problem because it is the easiest to follow. Thus, only one-dimensional values that give rise to a discontinuous density are measured. These discontinuous densities can arise in many scientific areas [1, 9, 43] but imagine a simple use case, where cars drive on a highway. The cars are *uniformly* distributed on the highway, and they all measure a variable y, for example, the temperature. In addition, each car has a unique position x on the highway, but that location is unknown. Only the cars' measurements of the temperature y at their respective positions can be observed. This results in an unordered list of temperatures without any access to the underlying position x on the highway. Our goal is to reconstruct the function $y = T(x)$ that describes the temperature y over the highway. The uniform distribution of the cars on the highway in combination with the map T cause the recorded density which is discontinuous in this case.

Such a concrete example, with a specific choice for T, is shown in Figure 3.1 and will be used throughout this section and in almost all one-dimensional cases. The map used is identical to the one used in [45], and in Section 3.2 we will follow the presented ideas to reconstruct it. Note that we assume $T(x)$ to be unknown, as its reconstruction would not be required otherwise.

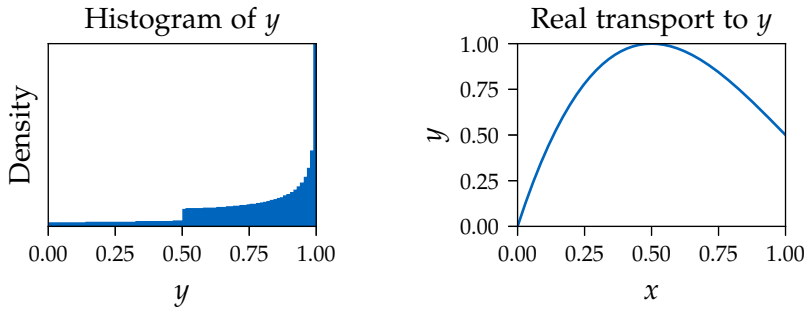


Figure 3.1.: The recorded histogram of y-values is illustrated on the left. It has a jump discontinuity at $y = 0.5$. On the right, the true *unknown* underlying relation $y = T(x) = -2(1-x)^3 + \frac{3}{2}(1-x) + \frac{1}{2}$ is plotted.

While this and all other plots are abstract and do not use practical measures such as the temperature, the car analogy is used throughout this thesis to aid explanation. So, y is the feature that can be measured, the temperature, and x is the unknown underlying space, such as the relative position on the highway.

The discontinuity of the recorded histogram in Figure 3.1 might leads us to believe that the underlying map is very complicated. As illustrated in Figure 3.1 it is a simple,

non-invertible *folded* map that causes the discontinuities. The measurements themselves are uniformly distributed on x (i.e. the highway), but y values smaller than 0.5 can only be produced by the first half of the manifold. Values larger or equal to 0.5 can be observed twice, at the start and the end of the highway. This causes the jump in the observed histogram at $y = 0.5$ because suddenly the same temperature is recorded more often. The fold of the map causes the divergence to infinity at $y = 1$ for infinitely many cars. This is because the gradient of T is zero at $x = 0.5$, and $y = T(0.5) = 1$. As the gradient approaches zero, many cars measure a temperature arbitrarily close to 1, resulting in a high probability for $y \in [1 - \varepsilon, 1]$ and a small positive ε.

The map T is a push-forward of f_x to the density $f_y = T_\# f_x$, and thus, the underlying distribution of the cars is crucial. When changing either T or f_x, the results change and a different density f_y is observed.

3.1.2. Transporting densities with normalizing flows

For now, we will take a step back from this concrete example and investigate how normalizing flows can be applied. In the theoretical part of this thesis, we discussed that NFs can find push-forwards, so they are capable of finding the same type of transformation we are looking for. In Section 3.1.3 we will try to find the push-forward T with a normalizing flow. But first, we will discuss in this section how NFs can be used to transport between arbitrary densities.

Throughout the thesis, normalizing flows are used to express (unknown) complicated probability distributions by the means of a simpler, known distributions. More specifically, they are used to find an invertible transport map between the density of a uniform distribution and the target density. While the theory behind normalizing flows itself has been discussed thoroughly in Section 2.2, the aim of the following is to show their versatility in practice and to get a better feel on how they can be applied.

It is important to keep in mind that normalizing flows operate on *samples* that follow the target distribution and can "flow" through the function(s) to transport them back and forth between the simple source and the complicated target. For that, we use neural spline flows with a coupling architecture in higher dimensions. To fit the transport function T that pushes the normal density forward, the training process maximizes the so-called (log) likelihood function [38]. This likelihood function describes the probability for given samples to happen.

T is then adjusted by an optimizer based on a loss function. To determine the loss, the samples are transported to the simple distribution with T^{-1}, where the (log) likelihood can be calculated. For a known distribution, such as the normal distribution, the probability for a sample can be efficiently determined. Once T has been approximated

well enough, it can be used as a push-forward. A concrete implementation that uses normalizing flows to transport a normal to a uniform density is found in Appendix A.

Normalizing flows use simple functions, which often lack expressiveness, to transport points. One check to ensure that the architecture is flexible enough, is whether it is capable of changing the modality of a distribution. If too simple transformations are used, such as only linear ones, a normal distribution can only be transported to another normal with different variance and mean [38]. To demonstrate versatility, Figure 3.2 illustrates a NF approximating a *bimodal* distribution. While the function T produces some slight deviations and some unexpected bumps or dents, it learns the density accurately. With the push-forward T, transportation in both directions is now possible.

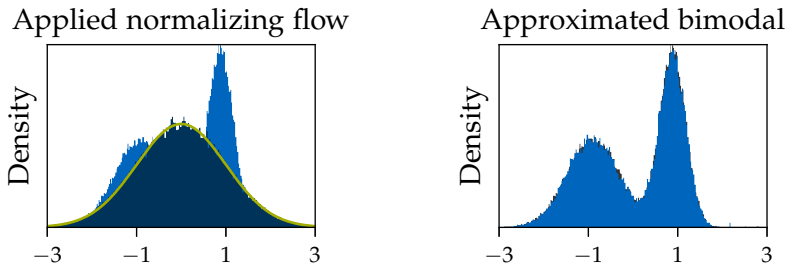


Figure 3.2.: On the left, a histogram of a bimodal normal distribution is visualized in light blue. By applying the normalizing flow T^{-1} to each sample, the dark blue histogram arises. It approximates the density of a normal distribution (green). The right figure shows the inverse direction. Sampling new points from the normal distribution and transporting them with T (blue) to reproduce the original histogram (black).

In many cases, mapping to and from a normal distribution is not enough if one wants to transport a complicated density to another arbitrary density. This could be useful, for example, if the distribution of pictures of faces is learned and one wants to interpolate between two faces, as is done in [37]. This can be achieved with a rather simple trick: both densities are mapped with a normalizing flow to the density of a normal distribution. To go from one distribution to the other, samples are first transported to the known normal distribution and then mapped into the other space. This process has been visualized in Figure 3.3, where the previous bimodal distribution is transported to a trimodal distribution (and vice versa).

For this transport, two normalizing flows, T_1 and T_2, have to be learned. Transporting a sample from the bimodal to the trimodal distribution can then be done with $T_2 \circ T_1^{-1}$. While this is only an approximation, it works well enough in practice. Such a concatenation of the push-forwards T_1, and T_2 is illustrated in Figure 3.4. This intermediate step allows finding a push-forward of an arbitrary source and target.

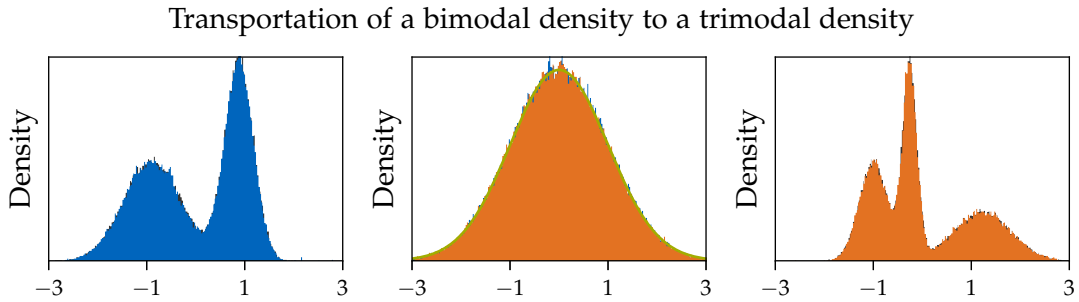


Figure 3.3.: This illustration shows a transport between two unknown normal densities with different modality. A bimodal density (left) can be transported to a normal density (center) with the inverse of a normalizing flow. Transporting the samples with another normalizing flow result in the trimodal density (right). The black histograms in the left and right plot show the original samples, whereas the colored histograms (blue / orange) show points transported from bimodal to trimodal or vice versa. The middle figure shows the density of a normal distribution (green), and the density of the transported bimodal distribution (blue) and trimodal distribution (orange).

It is worth noting that in those aforementioned examples, the densities have always been pushed forward from the density of a normal distribution. In this thesis on the other hand, the source distribution is typically assumed to be uniform. As most other libraries supporting normalizing flows, Pyro allows the use of any known distribution as a base distribution. But in Pyro, some errors with uniform densities were encountered when elements fall out of the uniform bounds. To overcome this issue, a second transformation can be added that maps back and forth from the normal density. This could be done with uniform samples and training another normalizing flow, but the more elegant approach that has been used in this thesis, relies on the Gaussian Φ and Φ^{-1}. These can be used to transport back and forth between the uniform and normal density, as described for example in [24, Appendix A].

Further, it should be mentioned that normalizing the samples before the learning process to have mean 0 and variance 1 greatly aids in the learning process. Libraries such as scikit-learn [52] provide functions to easily (de-)normalize matrices.

3.1.3. Direct transport of discontinuous densities

In the previous sections, we discussed and saw that normalizing flows can find a push-forward function given a simple known density and samples following another complicated distribution. We will now continue with the in Section 3.1.1 proposed problem and with the previous analogy: uniformly distributed cars on a highway

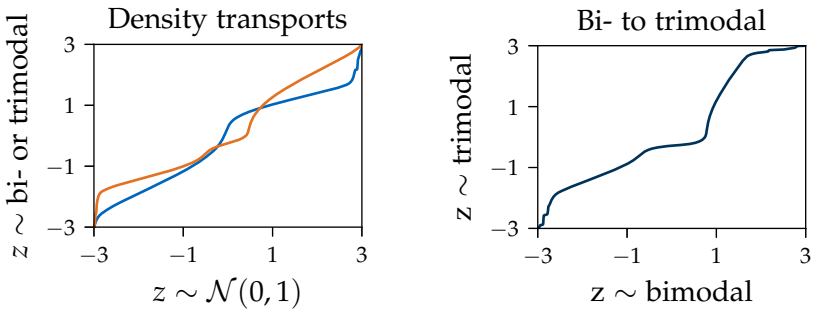


Figure 3.4.: The left figure visualizes the individual push-forward maps showing how mass from a normal distribution can be transported to a bimodal or a trimodal distribution. The blue function transports z to the bimodal density, whereas the orange transports it to the trimodal density. The concatenation describing how to push the bimodal density forward to the trimodal density is visualized in the right plot.

measure the temperature y at their respective position, and the goal is to reconstruct the function T. As mentioned before, the function T is a push-forward from the uniform density. Thus, we can try to use NFs to find this push-forward. So, we continue as in the previous section and learn a NF transporting a uniform density to the observed samples (i.e. the temperatures). Figure 3.5 visualizes the results of this process.

The first observation that becomes apparent when looking at this figure is that while the normalizing flow approximates the underlying density well, it does not find the true transport map T. In some scenarios, finding an arbitrary push-forward might be sufficient, especially when only new samples need to be drawn from a complicated distribution. As for the example where T describes the temperature of the highway, identifying the true underlying system is required to make correct predictions.

Folded maps are surjective and therefore not invertible, but normalizing flows are constructed so that the underlying function must be invertible. Thus, NFs cannot find any maps that are folded [38, 50]. Approaches relying on solving the optimal transport problem to find transport maps can face similar problems. Solving the optimal transport problem proposed by Monge with the usual cost function of $c(a, b) = \|a - b\|^2$ yields a unique monotonically increasing solution (i.e. not folded) [45]. In this example presented, the normalizing flow approximates this optimal solution.

Especially with discontinuous densities, the true underlying map is likely to be folded and thus such an approach will not lead to the correct solution. Even in the continuous case, there are scenarios where multiple different transport maps exist and the true relation cannot be found, as will be discussed in Section 3.2.4.

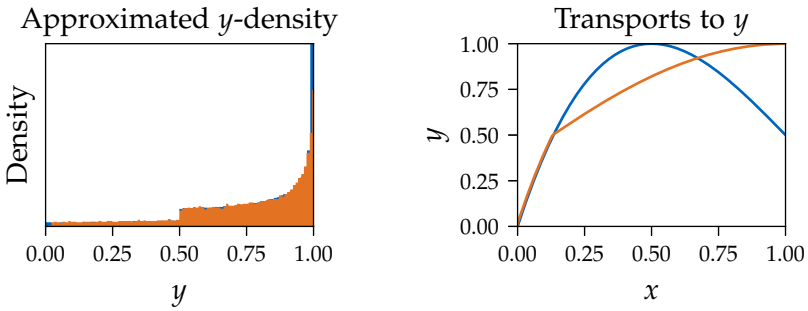


Figure 3.5.: This figure illustrates the push-forward a normalizing flow finds (orange) when directly applied to the observed distribution of y values. The plot on the left shows the true recorded histogram (blue) compared to new samples drawn from the uniform distribution and then transported with the normalizing flow. On the right, the push-forward function of the normalizing flow is compared to the real unknown map T (blue). In the push-forward that has been found, the discontinuity of the derivative at $y = 0.5$ causes the jump discontinuity in the density [45].

3.2. Reconstruction of one-dimensional discontinuous densities

In the previous section, we saw that normalizing flows (and optimal transport approaches) fail to recover the underlying manifold when the map is folded, even in some continuous cases [45]. The goal of this thesis is the identification of the real underlying map regardless of whether it is folded. Therefore, the process must be adapted such that it can approximate the true T. In the following, we will continue the above example with the ideas from [45] to perform reconstruction.

3.2.1. Observational process and time delay embedding

We saw that normalizing flows find unfolded, and for discontinuous densities non-differentiable maps. When the requirement is to reconstruct only a differentiable map from the measurements, one can use the approach presented in [45, Appendix A], but we want to find the correct solution. With the current measuring process of y, one can never be certain whether the approximated map is the true underlying relation. For example, it could be that on some part of the road a different asphalt is used, explaining the abrupt change in the gradient of the temperature, and making the solution found by the normalizing flow the correct one. To overcome this issue, the measurement process has to be changed to be more systematic than only collecting single values. As proposed in [45], an observation process is used that follows an *unknown but consistent dynamical process* where the underlying variable(s) change.

Continuing with the car analogy, this means that all cars move on the highway with the same speed, that is, the objects move on the underlying manifold with a consistent motion. Instead of only measuring the value y (i.e. the temperature) at a specific point in time, access to subsequent values at later positions on the manifold (i.e. highway), $(y_n, y_{n+1}, \dots)$, are required. Since all cars move at the same speed, the difference in x, while unknown, is consistent over all cars. This difference does not need to be consistent over time delayed measurements. For example, the second measurement could be one second after the first, while the third is one minute after the second.

Takens' theorem (see Section 2.1.3) states that with enough time delayed measurements, the time delays can be used to create a diffeomorphic embedding of the manifold we are looking for [45]. For one-dimensional problems, as the one we are currently discussing, the theorem says that up to three time delays are needed. For the example presented in this section, even two time delayed measurements are sufficient.

With this knowledge of consecutive values, and the fact that the embedding is diffeomorphic, reconstruction of the true underlying manifold T becomes feasible. Intuitively, the local movement of the underlying objects can be used to verify a solution. Figure 3.6 shows the diffeomorphic time delay embedding for the measurement process and example discussed before.

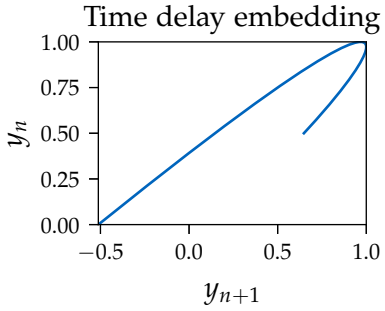


Figure 3.6.: The time delay embedding given two consecutive measurements of y, namely $(y_{n+1}, y_n) = (T(x+\tau), T(x))$ is depicted in this plot. It results in a curve that is diffeomorphic to T. The constant offset τ does not affect the diffeomorphism itself, but changes the "width" of the fold and can be unknown.

While this illustrated curve is not the solution T, it looks "very similar" to it. Takens' theorem says that this curve is diffeomorphic to the true underlying manifold and hence folded the same way. Only an *invertible* (and differentiable) function is required to map it to the correct solution. Thus, no new folds need to be added and normalizing flows can be used again. This diffeomorphic embedding contains more data than the original histogram, and, from now on, will be used instead. The original histogram can be recovered by projecting the data points onto the plotted y-axis.

3.2.2. Unfolding discontinuous densities

While this diffeomorphic curve contains more information, it cannot be used as a starting point by itself. First of all, it is two-dimensional, and it needs to be transported to a one-dimensional space x. Secondly, in the previous sections, and especially when trying the direct transport in Section 3.1.3, we saw that folded transport maps cannot be constructed by most algorithms. The diffeomorphic curve also has a fold that causes the same issues. To overcome those problems, this curve must be cleverly encoded to still allow the added information from the time delay to be used.

Moosmüller, Dietrich, and Kevrekidis propose in [45] to resolve the problem by *unfolding* the underlying density, on which the following section is based on. In this one-dimensional case, the time delay embedding is a curve with respective positions (y_{n+1}, y_n). Assume for a moment that the transport function T is known. With it, the true underlying diffeomorphic curve $c(x) = (T(x+\tau), T(x))$ can be defined.

There is an intuitive approach to unfold a curve: parametrizing it by its arc length. When interpreting a function as a string, the arc length describes the length of this string up to a given point and is monotonically increasing. It is formally defined as

$$\operatorname{arcl}(x) = \int_0^x \|c'(t)\| dt, \tag{3.1}$$

for a curve $c(t)$ and in this case $x \in [0,1]$. The major advantage of this parametrization by the arc length of a curve is that it does not have any folds because it is monotonically increasing. When measuring a string (i.e. a curve) it can only become longer.

The map T is not known (yet), and hence neither the curve c nor the arc length can be described exactly. With a large enough number of samples, both can be approximated numerically. The arc length can be determined by calculating the cumulative Euclidean distance of neighboring points $\sqrt{(p-q)^2}$. The results when numerically parametrizing the diffeomorphic curve by its arc length are illustrated in Figure 3.7.

This unfolding process allows us to overcome the aforementioned problems of the folded time delay embedding while still retaining the additional information. By only looking at the arc length s, the new unfolded version gives rise to a one-dimensional continuous density. Secondly, this process allows us to describe the underlying one-dimensional manifold, the curve, with only a single parameter. The arc length of the curve can be seen as the single intrinsic state that describes the properties of the system. With this, the dimensions now match and a transport to a one-dimensional uniform density is possible. It also gives us information about the system itself, namely that a transport to only one parameter is sufficient (i.e. the highway is one-dimensional). Without those problems, this reparametrization of the time delay embedding can be used as the intermediate space to reconstruct the true underlying transport map T.

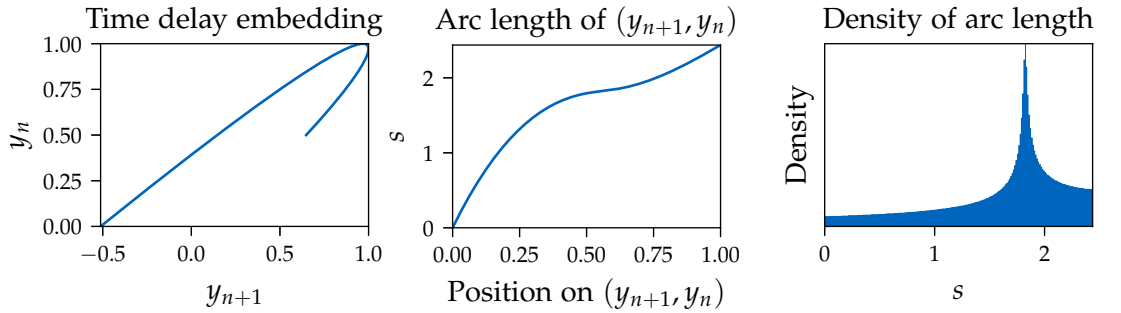


Figure 3.7.: The process of parametrizing the time delay embedding by its arc length is illustrated. On the left, Figure 3.6 is shown again that displays the time delay embedding (y_{n+1}, y_n). The monotonically increasing and hence unfolded parametrization of this time delay embedding is depicted (center) with the corresponding arising continuous density of the arc length s (right).

3.2.3. Reconstructing the underlying manifold

This unfolding process allows us to once again apply a normalizing flow to find a transport map. But contrary to Section 3.1.3, the density is now continuous and based on a diffeomorphic manifold. Hence, the normalizing flow can now approximate an invertible push-forward of the uniform density that will be isometric to the true underlying transport map. Meaning that the normalizing flow can find the true map, with sufficient knowledge about x (e.g. the range).

The steps before can be seen as pre-processing the data such that classical transportation works. The normalizing flow that can now be learned is capable of transporting points $x \sim \mathcal{U}_{[0,1]}$ to the arc length s and vice versa. Figure 3.8 shows the results of a trained normalizing flow and that the underlying density can be approximated well.

With the car analogy, the normalizing flow can map the position of a car on the highway to a position on a diffeomorphic highway. With it, each temperature can be mapped to the position on the highway, allowing to reconstruct previously unknown information. To be able to go in both directions, and to finalize reconstruction, a map from the arc length s to the temperature y_n, $y(s) : s \mapsto y_n$ has to be found. A computationally inefficient method would be to re-measure the arc length of the diffeomorphic curve and stop once it has the desired length s. This yields a position on the curve (y_{n+1}, y_n), allowing reconstruction of the measured value y_n.

As only samples are collected, the complete function of the time delay embedding is unknown, and the diffeomorphic curve and its arc length have to be approximated. In the implementation for this thesis, the first tuple (y_{n+1}, y_n) for which the arc length is greater or equal the desired s is returned. When there are only a few data points

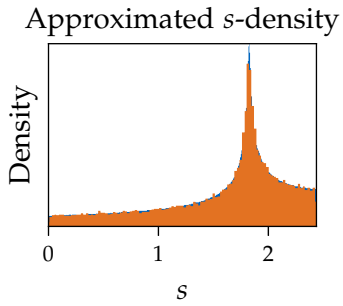


Figure 3.8.: This plot compares the true density created by the arc length of the time delay embedding (blue), and new points sampled from the uniform distribution and transported with the learned normalizing flow (orange).

available, it might be necessary to perform interpolation between the arc lengths.

Based on this implementation, every arc length can be mapped to the original y_n with $y(s)$, as seen in Figure 3.9. When taking a closer look at this function, we can see that it already somehow resembles the map we are looking for. And intuitively this makes sense because we unfolded a diffeomorphic version of curve, and to transform the unfolded version back (i.e. to refold it), the mapping needs the correct folds.

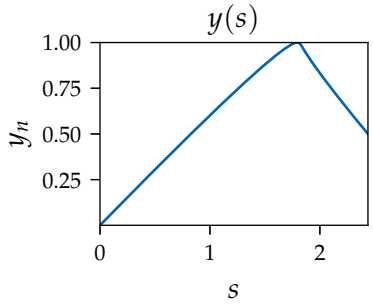


Figure 3.9.: Plot of $y(s)$ that maps each point of the arc length to the respective y_n. A slightly different version of this map can be constructed to recover y_{n+1}.

With all the steps done before, and enough information about x, the true underlying manifold can be reconstructed, and new points sampled. A point x is transported with the normalizing flow $f(x)$ to an arc length s, and then with $y(s)$ mapped into the underlying folded space. This results in the complete function $T(x) = y(f(x))$, and the corresponding density, as visualized in Figure 3.10.

Note that, when the underlying space is unknown, the exact manifold cannot be reconstructed but only a diffeomorphic copy of it. As a simple example, imagine that the true bounds on x are not $[0, 1]$ but $[0, 100]$. The reconstructed manifold would then

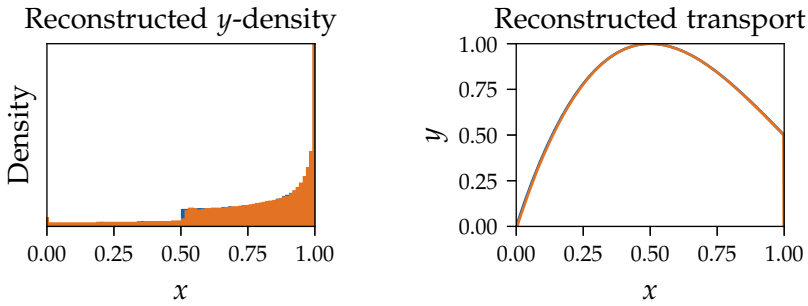


Figure 3.10.: This figure compares the ground truth (blue) with the approximated solution (orange). The true manifold could be approximated (right) which will result in a nearly identical density when sampling new points (left).

be defined on a different range and thus only be diffeomorphic to the true manifold. Similarly in higher dimensions, the variables could be swapped. In this thesis, sufficient knowledge for the complete reconstruction is assumed.

The reconstruction of the original underlying transport map has only minor deviations. With this approximation, the temperature y on a position x on the highway can be determined, without measuring a single x value. In the reconstructed map at $x = 1$, a vertical bar can be seen. Because the NF is only an approximation, when it yields an arc length greater than the length of the curve itself, it will be mapped to zero in this implementation. As there does not exist a point on the time delay embedding for such points out of range, there is no single correct way to resolve this problem.

In this section, we saw that by directly applying a transport algorithm such as normalizing flows, the correct solution cannot always be found, especially when the underlying map is folded. Discontinuous densities are often caused by folded maps and thus require explicit attention. To overcome this problem, multiple time delayed measurements were collected that can be used to find a diffeomorphic version of the underlying manifold. By unfolding the density of the time delay embedding, the dimensionality could be reduced, and we ensured that an invertible map is sufficient. This unfolded version can be used to find a normalizing flow, allowing reconstruction. When *refolding* the values, the map T can be evaluated at arbitrary positions.

3.2.4. Transport of continuous densities

In the previous sections, we discussed that the classical approach can even fail to identify the true manifold for continuous densities. We will now see that time delays are always required when needing to ensure that the identified transport is correct.

We begin by looking at a density without a jump discontinuity, produced by $\bar{y} = T(x) = -4x^2 + 4x$. While this map is again folded, it is not "cut-off" as before, but every y value corresponds to exactly one of two x values. Thus, the arising density of this transport map, Figure 3.11 (left), has no jump discontinuity. A normalizing flow, Figure 3.11 (right), cannot find the true map as it only finds invertible solutions.

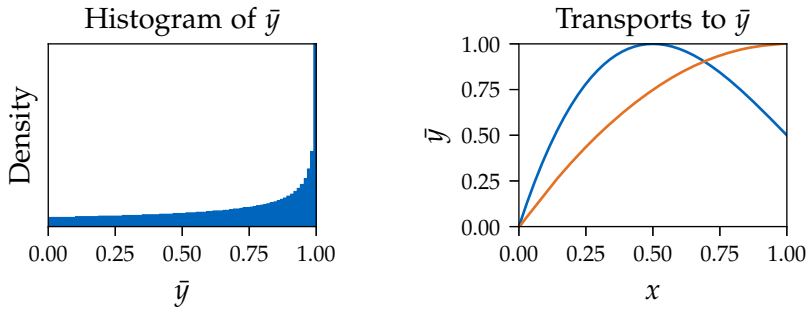


Figure 3.11.: The left plot shows the density that can be observed when one of the two transport functions plotted in the right is used. The blue folded map (right) was used to create the samples, but as this function is non-invertible it cannot be found directly. A normalizing flow finds the orange curve (right).

This again shows that the problem described arises mostly with *folded* transport maps, regardless of whether the density has a jump discontinuity or not. Still, the density presented here is discontinuous, as it diverges to infinity at $\bar{y} = 1$. This time, the solution is also differentiable. As transport maps are not unique, an additional observation process is needed to be certain that the correct solution was found. This also holds for continuous densities, as illustrated by a simple transport map where the isometric version is found by the normalizing flow shown in Figure 3.12.

One can even construct examples, where the normalizing flow (with a specific algorithm) finds a different solution than the isometric one. These examples show the importance of multiple observations and a process that can identify the true system, up to an orthogonal map, described in this thesis, based on [45].

3.2.5. Varying underlying densities

Up until now, we have assumed that the density we are transporting to is uniform. In the car example, this means that the cars are uniformly distributed on the highway. Since normalizing flows are used, any known base density can be used instead. To be more precise, the two-step normalizing flow procedure illustrated in Section 3.1.2, allows transport to any complicated density if enough samples are provided to learn that distribution.

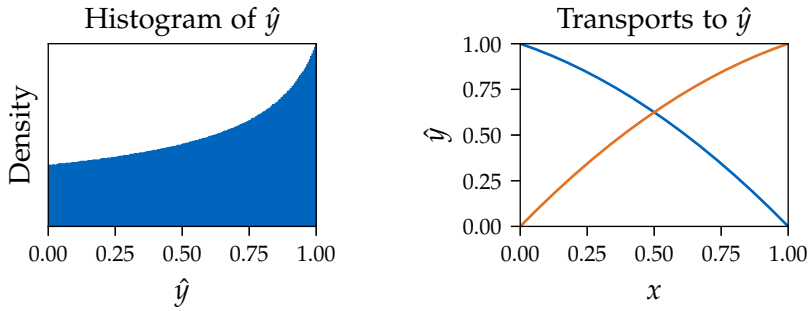


Figure 3.12.: Similarly to Figure 3.11, this plot shows two different transport maps (right) that both produce the density on the left. $\hat{y} = T(x) = -\frac{1}{2}x^2 - \frac{1}{2}x + 1$ (blue) is the actual function used for sampling, but the normalizing flow approximates the isometric version (orange). Without additional data, the correctness cannot be determined.

In Figure 3.13, the same procedure to reconstruct the underlying manifold as before is applied but the assumptions were changed. Instead of transporting the density of the arc length to a uniform density, these push-forwards illustrate the same process with a density from the standard normal distribution, and a bimodal distribution. Thus, the cars on the highway would be distributed accordingly.

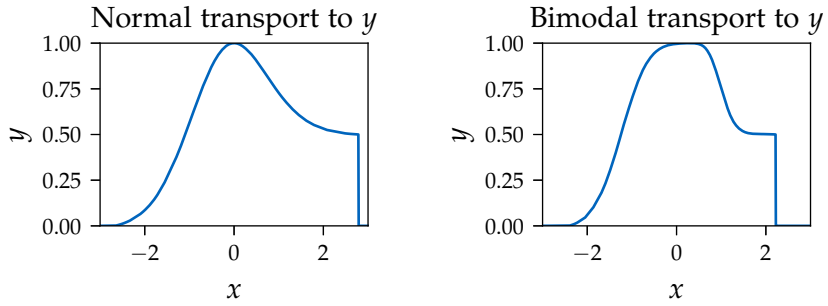


Figure 3.13.: These plots show that different push-forwards are found, when the base density is changed. Only the transport to the intermediate arc length is changed but otherwise the process kept identical. On the left, the push-forward for points drawn from a standard normal distribution are shown. In the plot on the right, the same is shown for the bimodal distribution from Section 3.1.2.

Note that the analogy of a highway might not make sense anymore with an unrestricted base distribution, such as the standard normal. A uniform distribution has clear bounds that one can imagine as the start and end of the road. And while the tails of the density of unbounded densities can become arbitrarily small, they never reach zero, meaning that the highway would need to be infinitely long.

3.2.6. Number of time delays

Takens' theorem gives a clear upper limit on the number of delays needed for a time delay embedding to be diffeomorphic. In practice, it might be desired to collect as few time delays as possible to reduce cost and time. In the previous example (see Section 2.1.3), two time delays from the three necessary were sufficient to find a diffeomorphic embedding. In this section, we will look at the benefit when increasing the number of collected time delays even further than Takens' theorem requires.

With the clear upper limit provided by Takens' theorem, the answer might seem to be that collecting more than $2d + 1$ time delays is unnecessary. Figure 3.14 visualizes how the previous example changes with additional time delays.

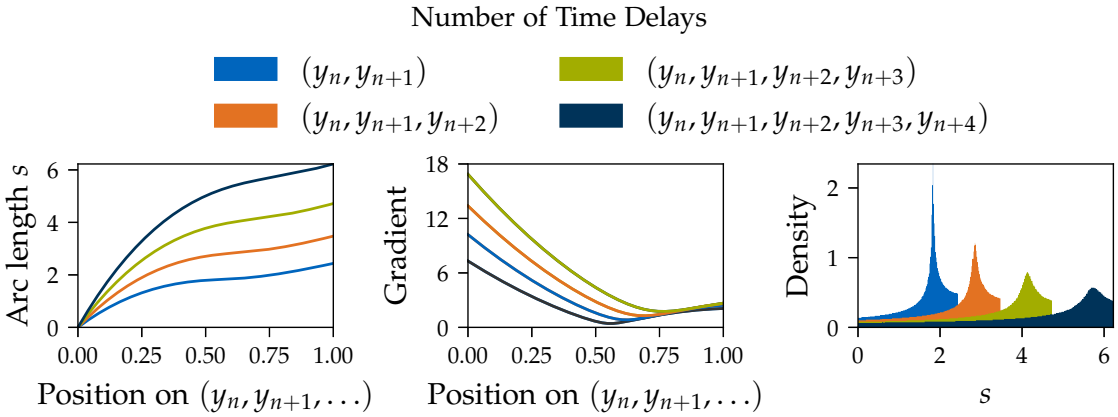


Figure 3.14.: Indicated by different colors, this figure shows the arc length with increasing number of time delays (left), the corresponding gradient (center), and the density (right). In this example, two time delays are sufficient for a diffeomorphic embedding and up to three are generally needed. The gradient of the arc length (center) becomes smoother with more time delays, even when exceeding the maximum number of time delays required.

Those smoother gradients of the arc length make the resulting densities also smoother which can be beneficial when using machine learning. As the relations become more linear, artificial neural networks (e.g. used in normalizing flows) can potentially learn and express the underlying relation more quickly and with fewer neurons. While this might not be a problem in those toy examples presented, when data becomes higher-dimensional, reducing the training time might be required.

3.3. The dynamical system

Reconstruction of the underlying manifold gives much information about the stationary system, which was the highway x and the corresponding temperature y in the previous example. With the introduction of time delays, the underlying objects needed to move on a stationary system (i.e. the manifold). In this section, the goal is to learn the underlying dynamical system—the movement of the objects—with time delayed measurements. More specifically, a dynamical system is a process where a single vector s serves as the intrinsic state and fully determines the future of the system [28]. Continuing with the car analogy, this means that we want to investigate how the cars' measured temperature changes when it drives along the road. Throughout this section, we will continue the previous one-dimensional example and also use the same data.

Even in the simple scenario of only a single measurement y_n it is difficult to predict subsequent y_{n+k} values. Because folded maps are not injective, a single measurement can correspond to multiple positions. Thus, an individual y_n does not uniquely determine the position on the underlying manifold. With this ambiguity, predicting the future is difficult, because it is usually not possible to decide which of the intrinsic states is the correct one.

For example, assume the same folded curve as before (see Figure 3.1) that could describe the temperature on a highway. A measurement from a car could have been taken on the left side of this folded map (i.e. first half of the highway), meaning the temperature will increase in the next timestep, or, it could have been measured on the second half and it will decrease with the movement of the car.

But when describing the system in a different way, prediction of the next state can become easier. For instance, learning the dynamical system of the time-delay embedding is possible, because a point on the curve (y_{n+1}, y_n) uniquely identifies the state and allows prediction how the system will change (i.e. rate of increase / decrease). Using this tuple as the intrinsic state and learning the corresponding dynamical system of the time delay embedding comes with some difficulties. Although the system of the diffeomorphic curve is two-dimensional, prediction of the next state can only be done for points on this one-dimensional curve (y_{n+1}, y_n). While this increase in dimension allows us to overcome the initial problem, it increases complexity without adding higher-dimensional information.

In the previous sections, we were able to use a single dimension to parametrize the underlying curve and hence saw that the intrinsic dimension is one. When using the arc length parametrization as the single state of the system, all information can still be encoded without increasing the dimension. As the parametrized function is monotonically increasing, every arc length has a unique position on this curve making prediction

easier. Since we also saw that there is a mapping $y(s) : s \mapsto y_n$, the dynamical system of y_n (i.e. the temperature) can be learned by using s as the intrinsic state instead. This parametrization allows us to learn the underlying dynamics and predict the observed temperatures at the next timestep, all without increasing the dimension.

Given this apparent advantage, this section is dedicated to learning the nonlinear dynamic of the (unfolded) system. In the one-dimensional case, understanding the dynamics is equivalent to learning the steepness of the curve to estimate the position at the next timestep. Thus, the goal is to learn the gradient. Such dynamical systems are typically described with differential equations [57]. In the higher-dimensional case, this corresponds with learning the underlying vector field [2]. In all dimensions, it describes where to and how quickly a point on this field would "flow" into a direction when imagining the vector field / gradient as water with turbulence [65]. For more information on vector fields and gradients see for example [64], and [65] is a great introduction for dynamical systems themselves. The extension of this concept in higher dimensions is sketched in Section 3.4.6.

3.3.1. Learning the system with a neural network

There are many ways to learn the dynamics of a nonlinear system [28, 39], but in this thesis, the method presented in [57, Sec. III.3] is used. There, Rico-Martínez *et al.* propose to use an artificial neural network that is integrated with a fourth order Runge-Kutta integrator. Runge-Kutta is a family of methods that allow numerical integration and can be used to solve ordinary differential equations. [4] and especially [4, Ch. 4] serves as a great introduction for numerical integration, available methods, and contains much information regarding ordinary differential equations. The method proposed in [57] is more thorough than needed for this simple scenario. Thus, the following only describes the necessary parts to solve the scenario tackled in this thesis with slightly adapted notation. Otherwise, it is very similar to [57, Sec. III.3].

The goal is to learn the underlying vector field of the parametrized version. This possibly higher-dimensional gradient $\mathbf{f}$ we are looking for, serves as the right-hand side of the ODE

$$\vec{s}_n{}' = \mathbf{f}(\vec{s}_n), \text{ with } \vec{s} \in \mathbb{R}^d, \ \mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d, \tag{3.2}$$

where $\vec{s}_n$ are the coordinates of the d-dimensional space. Continuing the one-dimensional example from before, this means that $\vec{s}_n$ is the arc length with a dimension of $d = 1$. This unknown function $\mathbf{f}$ is the vector field and describes the underlying dynamical system. To approximate it with the observed data, a neural network is used as the function $\mathbf{f}$ itself.

The weights of the network $\mathbf{f}$ have to be adjusted such that it approximates the gradient correctly. When training a neural network with supervised learning, some inputs and outputs of the function need to be pre-determined for the neural network to learn, and the weights to be adjusted accordingly. In this case, there are only measurements for the arc length itself, but not for the gradient. To be able to compare the output of the network $\mathbf{f}$ (i.e. the gradient) with the measured arc length, the neural network is integrated numerically with Runge-Kutta [57]. Given the current state of the system $\vec{s}_n$, this numerical integration method can approximate the subsequent state $\vec{s}_{n+1}$ with $\hat{\vec{s}}_{n+1}$ by

$$\hat{\vec{s}}_{n+1} = \vec{s}_n + \frac{1}{6}\left(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4\right), \tag{3.3}$$

with

$$\begin{aligned}
\vec{k}_1 &= h\mathbf{f}\left(\vec{s}_n\right) \\
\vec{k}_2 &= h\mathbf{f}\left(\vec{s}_n + \frac{\vec{k}_1}{2}\right) \\
\vec{k}_3 &= h\mathbf{f}\left(\vec{s}_n + \frac{\vec{k}_2}{2}\right) \\
\vec{k}_4 &= h\mathbf{f}\left(\vec{s}_n + \vec{k}_3\right),
\end{aligned} \tag{3.4}$$

as formulated in [57]. This can be used for learning the correct $\mathbf{f}$, up to a global fourth-order error of $\mathcal{O}\left(h^4\right)$ that depends on the step size h [4].

With Runge-Kutta, the right-hand side of the ODE (i.e. the neural network $\mathbf{f}$) can be used to continue the "flow" of the dynamical system. In this one-dimensional example, this means that with a single position on the arc length s_n, and the gradient $\mathbf{f}$, the arc length at the next time step s_{n+1} can be approximated with $\hat{s}_{n+1}$.

3.3.2. Architecture and training

We will now apply this method proposed by Rico-Martínez *et al.* to learn the dynamical system of the arc length that in turn allows us to understand the dynamical system of the temperature. As data, we use the previously collected points on the arc length. As only samples are available, the arc length was approximated numerically and thus a monotonically increasing list of s_n values is available. Based on those points, tuples of (s_n, s_{n+1}) can be created: s_n would be an arc length and s_{n+1} the subsequent arc length in the list. This gives inputs and outputs that can be used to train $\mathbf{f}$.

In the car analogy, the goal is to predict the temperature observed by a moving car at subsequent timesteps. In this construction, we assume that at one timestep, each car drives to the exact position of the following car. If this is not desired, additional data

has to be collected. For example, collecting $(y_n, y_{n+1}, y_{n+2}, y_{n+3})$ can also be used to create $s_n = (y_n, y_{n+1})$, and $s_{n+1} = (y_{n+2}, y_{n+3})$ with possibly a different step size.

Now that the training data has been acquired, the following procedure is executed for random (s_n, s_{n+1}) tuples to train the neural network:

1. Apply the neural network $\mathbf{f}$ four times with input as described in Equation 3.4.
2. Calculate $\hat{s}_{n+1}$, the approximation of s_{n+1}, with Runge-Kutta from Equation 3.3.
3. Adjust the weights of $\mathbf{f}$ based on the difference between $\hat{s}_{n+1}$ and s_{n+1}.
4. Repeat with 1. until the global error is sufficiently small.

In the following, we will first look at the design decisions of the neural network and then discuss the results in the next section. For the timestep h, $dt = \frac{1}{\text{sample size}}$ is used with a sample size of $50,000$ (i.e. number of cars on the highway). This small h ensures that the underlying space is in $[0, 1]$ and it also normalizes the values to sane boundaries so that training the neural network is more efficient and accurate.

As for the internal architecture used, this constructed example is very small and hence a very basic neural network is sufficient. It contains two fully connected hidden layers with 20 neurons each. For the activation function, the sigmoid function has been used to ensure non-linear expressiveness. To train the neural network, the data set is split into random batches of 300 samples. Every 300 processed samples of s_n, s_{n+1}, the sum of their losses is calculated, and the weights updated accordingly. To calculate the loss function, the mean absolute error $\frac{\sum_{i=1}^{300} \|s_{i+1} - \hat{s}_{i+1}\|}{300}$ is used [51]. Based on this loss, the weights are adjusted by the AMSGrad optimizer, which is a variant of Adam [36, 56].

PyTorch [51] was used for implementing this approach. For the neural network following the schematics proposed in [57, Sec. III.3] itself, the implementation by [15] was used only with slight alterations.

3.3.3. Results

With this internal architecture and choice of hyper-parameters, the neural network can be trained for a few hundred epochs (i.e. times the training data is processed fully) to correctly approximate the true underlying $\mathbf{f}$. With further fine-tuning of hyper-parameters such as the learning rate, the training could be achieved faster. Figure 3.15 shows the approximation of this neural network $\mathbf{f}$, the right-hand side of the ODE. As is apparent, the overall approximations are very accurate.

Learning the dip of the gradient to nearly zero is difficult for the neural network to approximate and needs many epochs. To ease this problem, more neurons might be

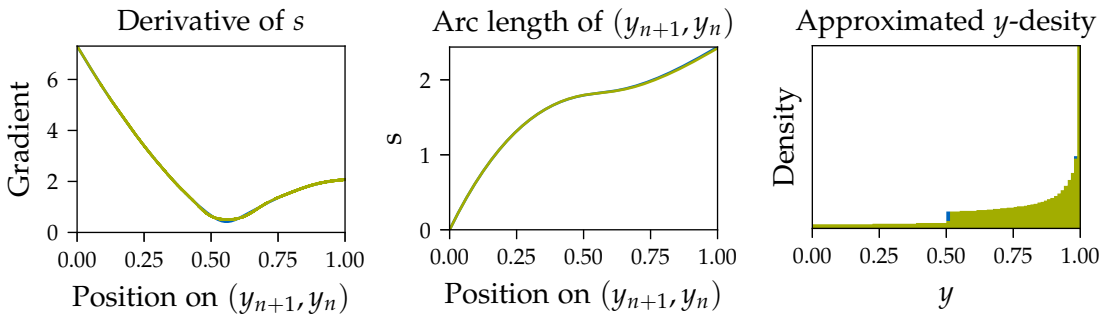


Figure 3.15.: Those plots compare the performance of the neural network (green) with the true values (blue). The gradient (left) can be reconstructed by taking the measured arc length values s_n and calculating $\mathbf{f}(s_n)$. By integrating the gradient $\mathbf{f}$ numerically, the arc length can be reconstructed (center). The right plot shows the histogram of y-values created by the integrated, neurally approximated arc length.

beneficial. Collecting more time delays, as discussed in Section 3.2.6, makes this bump flatter and the resulting function smoother, and facilitates the learning process.

As the function $\mathbf{f}$ is the gradient, it can be used to solve the initial value problem for $s_0 = 0$. When imagining the gradient as the turbulence in water, this would mean that a leaf on this water is placed at the start and then observed as it moves according to the flow. Solving for the positions of the leaf at specific points in time can be done with Runge-Kutta by applying Equation 3.3 repeatedly. The more sophisticated initial value problem solver provided in SciPy [72] was used for Figure 3.15 (center).

One has to keep in mind that while we were able to reconstruct this arc length completely by solving the initial value problem in this one-dimensional case, it has a different meaning. Intuitively, this solution means that a car starts at the beginning of the highway and records the arc length while it drives along. Before, there were multiple cars that recorded their own respective temperature. This difference becomes apparent, when looking at the problem in a higher-dimensional case, as the manifold cannot be reconstructed anymore. This will be discussed in Section 3.4.6.

With this procedure, the underlying dynamical process could be learned. Meaning that the neural network allows us to continue the movement of an object on the underlying manifold and observe it at arbitrary timesteps. Together with the reconstructed map, this can give much insight about the system and can be used for an even better understanding of the underlying system.

3.4. Approach for higher-dimensional discontinuous densities

In the previous sections, the one-dimensional case has been extensively discussed. When extending the problem to higher dimensions, some parts, especially those regarding the parametrization and unfolding, cannot be trivially transferred. This is mostly because the arc length, the function used for the unfolding procedure, cannot be used for higher-dimensional manifolds. This section is dedicated to extending the one-dimensional approach to work in higher dimensions. This is done by investigating a two-dimensional problem that again follows [45]. With this increase in dimensionality, the procedure will not only be generalized but also allows for more applications. One such application is presented in Section 3.4.5, where we will see that in higher dimensions the underlying procedure does not only allow us to find the underlying manifold but even enables us to transport densities when the dimensions do not match.

3.4.1. Two-dimensional discontinuous densities

The analogy of cars moving on a highway and measuring their respective temperature can be slightly adjusted such that it makes sense in the two-dimensional case. Instead of cars, we could imagine ships on an ocean, as suggested by [16]. For this analogy to work properly, the curvature of the earth needs to be ignored. For the third dimension, this could be easily changed to spaceships moving in the universe, but the two-dimensional scenario will be sufficient to explain the higher-dimensional concepts.

Imagine ships uniformly distributed on a two-dimensional ocean (x, β_1) sailing with a consistent motion in β_1-direction, where each of those ships measures the temperature β_2. Figure 3.16 is based on [45] and illustrates measurements that could be observed in such a scenario. While in the previous example the underlying manifold that produced this discontinuous density was one-dimensional (i.e. a curve), this time, the unknown manifold is two-dimensional (i.e. a surface), as seen in the bottom center of Figure 3.16. A two-dimensional discontinuous density is recorded (top left) because the underlying map is again folded. Apart from the difference in dimensions, this problem is very similar to the one-dimensional case.

In this example, each ship can measure the temperature β_2 and also one coordinate of the ocean, β_1. We can imagine this with people living on an island on the edge of the ocean that have a telescope. For each ship, the inhabitants can tell how far "left" or "right" it is (i.e. measure one coordinate). Measuring how far away each ship is (i.e. recording the x value) is not possible. Thus, two of the three variables can be measured, namely β_1, and β_2. Of course, this is just an example to make the concept more feasible and the concrete measurements can be arbitrary.

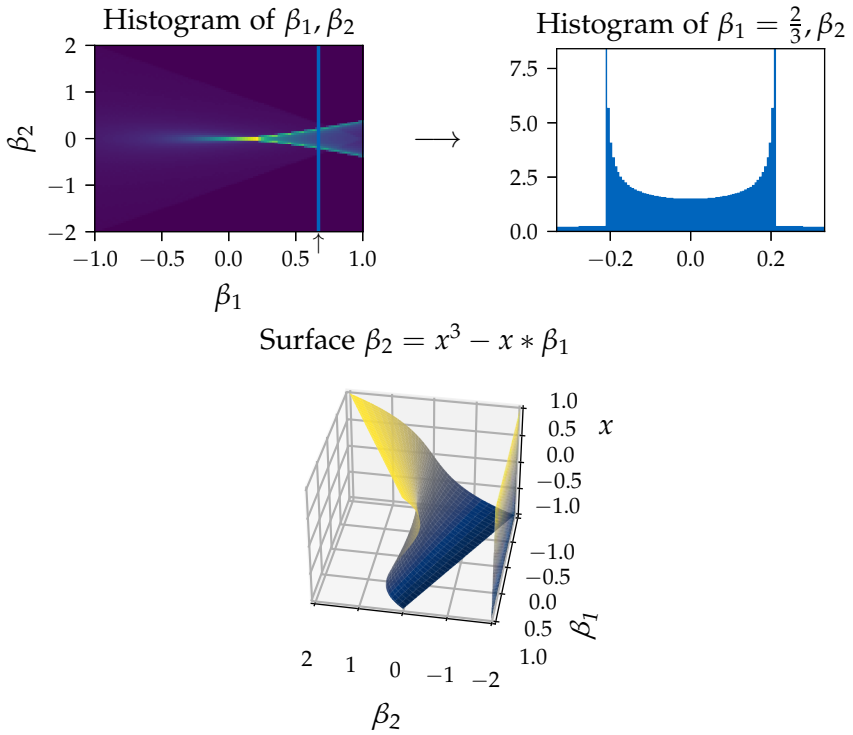


Figure 3.16.: In this example, a two-dimensional histogram (top left) with the values β_1, β_2 can be observed. As it becomes difficult to look at a two-dimensional density, the plot at the top right depicts a discontinuous slice of this density at $\beta_1 = \frac{2}{3}$. Instead of uniformly sampling on a single axis as before, the density was created by sampling uniformly from the plane $x, \beta_1 \in [-1, 1]$ (center right) and determining the value on the cusp surface $\beta_2\,(x, \beta_1) = x^3 - \beta_1 x$ (center).

3.4.2. Constructing a diffeomorphic surface with time delays

In this example, the recorded two-dimensional density of (β_1, β_2) and the space originally uniformly sampled from (x, β_1) are both two-dimensional. This means that similarly to Section 3.1.3, a normalizing flow can be trained that directly maps between those two measures. Such a direct transport with normalizing flows is visualized in Figure 3.17. As the underlying map is once again folded, the same problem as before is faced and it cannot be reconstructed immediately.

To overcome this issue, the procedure proposed by [45] is repeated. Similar to the one-dimensional case, the first step is to look at multiple time delayed measurements of the moving underlying objects. In this case, all objects (i.e. ships) move on the manifold in β_1-direction with consistent speed (i.e. the wind is the same for the complete ocean).

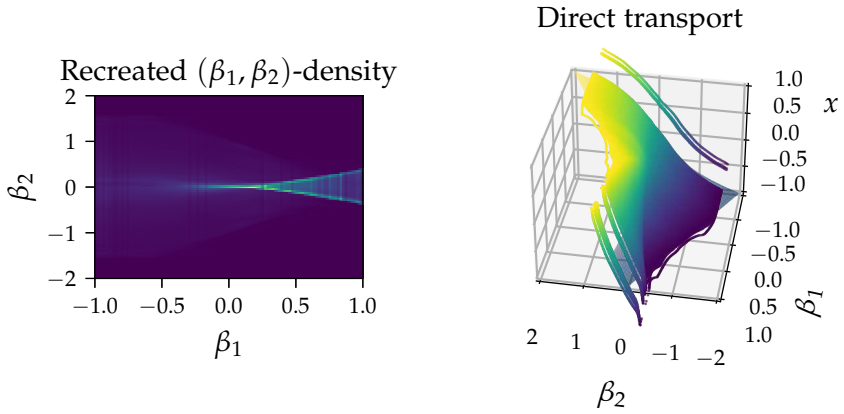


Figure 3.17.: The density (left) can be reconstructed when performing a direct transport with an underlying folded map. The reconstructed surface (right) can never be the real solution (right, translucent), as normalizing flows find invertible maps.

By Takens' theorem, up to five dimensions are needed to find a diffeomorphic time delay embedding. In Figure 3.18 (left), a plot of the time delays $\left([\beta_1]_{n+1}, [\beta_2]_n, [\beta_2]_{n+1}\right)$ with three dimensions is illustrated. In this case, it becomes difficult to see if three dimensions are sufficient to embed the manifold. To make it easier to visualize, and to facilitate training under some circumstances, PCA is applied.

Principal component analysis (PCA) [29] can be seen as a way of rotating the underlying space such that the axes are ordered by the variance they express. Those axes are referred to as principal components (PCs) and the first principal component (PC_1) explains most of the change in the data. In plots, axes are typically scaled automatically, so data is often not only rotated in PCA space but the axes also scaled.

The PCA version of the time delays in Figure 3.18 (right) is a better visualization of the surface, shows that those observations are sufficient for a diffeomorphic embedding, and contains all the folds needed. In the following, and also in all subsequent examples, the PCA version will be used as the diffeomorphic version instead of the time delays $\left([\beta_1]_{n+1}, [\beta_2]_n, [\beta_2]_{n+1}\right)$ themselves.

3.4.3. Parametrization with manifold learning

While the arc length allows the parametrization of every one-dimensional manifold in a way that it becomes unfolded, it cannot be used to parametrize higher-dimensional manifolds. For this, a more elaborate process is required. In the following, one way of parametrization is described with the two-dimensional manifold presented before.

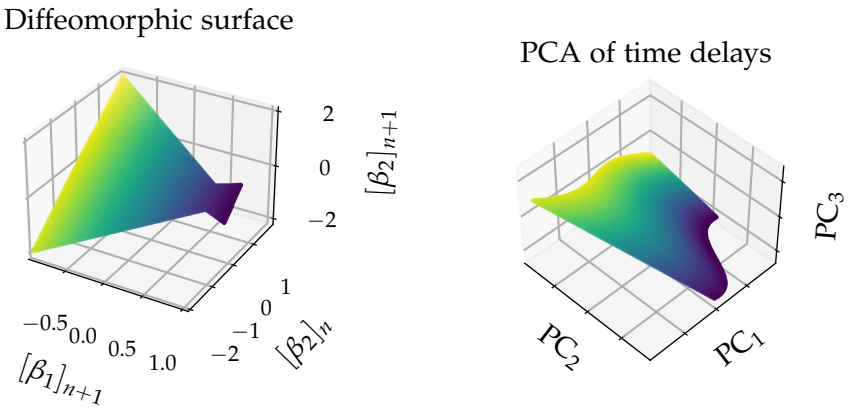


Figure 3.18.: In the left plot the time delays $\left([\beta_1]_{n+1}, [\beta_2]_n, [\beta_2]_{n+1}\right)$ are visualized. It is not clear whether this version is diffeomorphic. The right plot shows the same data but transformed by PCA. While it does not reconstruct the underlying manifold, it is a diffeomorphic version. The coloring changes along the second principal component.

When looking at the diffeomorphic surface in Figure 3.18 (right), it can be seen that it is a two-dimensional manifold (i.e. a surface) embedded into three dimensions. By construction, and because this is a diffeomorphism, it is known that the intrinsic dimension of this manifold is two. To transport to a two-dimensional uniform density, the dimension has to be reduced by reparametrizing the surface.

This parametrization (i.e. the unfolding) can be done with nonlinear *manifold learning*. Intuitively, this surface we are looking for can be imagined as a crumbled sheet of paper in three dimensions. While three coordinates are required to address a position on this surface, it is still a piece of paper with only two dimensions. The goal of manifold learning is to find a way to describe it as a two-dimensional object again. This is done by finding a (non-)linear way of mapping every point in three dimensions such that it lies on a flat plane again. There are many algorithms available that solve this problem.

Moosmüller, Dietrich, and Kevrekidis use in [45] so-called *diffusion maps* (DMAPs) [10] with a Mahalanobis-like metric [19] to reduce the dimension. In this thesis, DMAPs are used for the core process without the Mahalanobis distance. Diffusion maps are a nonlinear manifold learning technique that are based on a diffusion process. In those, a Markov matrix describing the probability to move from one data point to another is constructed. The probability is higher, the closer two points are in the feature space according to some measure (i.e. a kernel function). The eigenfunctions of this matrix (i.e. eigenvectors of a function) are then used to calculate the position embedded in a lower dimension [10]. Apart from the original paper for DMAPs [10], a great concise

introduction to the topic of manifold learning can be found in [31].

One major aspect of manifold learning techniques is that they typically need to know the intrinsic dimension beforehand. As all examples in this thesis are constructed, the intrinsic dimension is always known or easy to find out. With a sufficiently small dimension, the manifold can be visualized and hence the intrinsic dimension inferred by looking at a plot. Once the dimension becomes higher than three, this cannot be done anymore. A dimension estimation algorithm such as the one presented in [60] can be used in higher dimensions.

Continuing with the original problem, we now know that algorithms exist to find a (non-)linear embedding of a higher-dimensional manifold in a lower dimension. This process is the parametrization needed that can unfold the density and is used as a substitute for the arc length in higher dimensions. For this thesis, DMAPs are used provided by the library datafold [41].

In Figure 3.19 DMAP and another manifold learning algorithm are applied to the PCA version of the time delay embedding. In this case, different manifolding algorithms can find an embedding relatively easily. Of course, every algorithm has its own parameters that possibly require fine-tuning. Only the embedding found by DMAP shown in Figure 3.19 (left) will be used in the next sections.

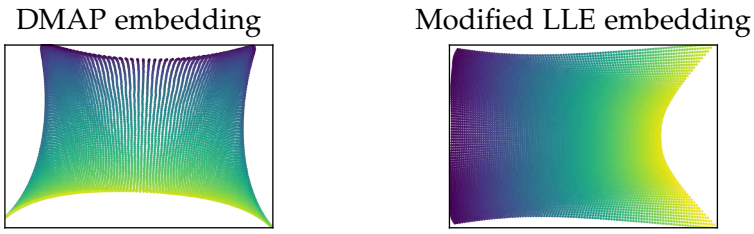


Figure 3.19.: Lower-dimensional embeddings of the diffeomorphic surface from Figure 3.18 (right) are visualized. Ideally, a perfect rectangle (i.e. a plane) is found. On the left, the DMAP algorithm has been applied with a continuous nearest neighbor kernel function. The plot on the right shows the embedding performed with a modified version of locally linear embedding (LLE) [59, 74]. Modified LLE is another manifold learning technique available in Python with [52]. Both algorithms can embed the surface into two-dimensional space. The coloring changes with the second principal component PC_2 from the diffeomorphic surface in Figure 3.18 (right). From the color gradient, we see that points are mapped differently by those algorithms.

3.4.4. Recovering the underlying manifold

With the parametrization of the surface in two dimensions, we can continue analogously to the one-dimensional scenario, as in Section 3.2.3. Namely, we train a normalizing flow that learns to map points from a two-dimensional uniform distribution $\mathcal{U}_{[-1,1]^2}$ to the embedding space displayed in Figure 3.19 (left).

The normalizing flow allows transportation from the space of time delays (3D) to a point on the uniform plane (2D). With this, plotting the reconstructed manifold is already possible and this might be sufficient for some scenarios. But the goal discussed in this thesis is to be able to transport in both directions. Most importantly, the ability to go from a uniformly distributed point to the time delay space allows sampling new points on the underlying surface and thus can be used to reconstruct the density.

Both, the normalizing flow and the application of PCA are invertible. The only step where invertibility has not yet been discussed is the DMAP. And while there is no easy way to invert DMAP itself, *geometric harmonics* [11] can be used to learn a mapping that goes from the embedded space Ψ to the constructed diffeomorphic space X.

Geometric harmonics refer to a special class of wave functions that are solutions of an eigenproblem of a kernel matrix. Based on the Nyström method [47], geometric harmonics can interpolate data on arbitrary manifolds. The manifold learning library used, datafold [41], also implements those, allowing us to solve the pre-image problem $f^{-1} : \Psi \to X$, which is the inverse direction of the DMAP process. For the following, it is sufficient to know that geometric harmonics allow us to find a function from the embedding to the true underlying higher-dimensional manifold. The results of learning the pre-image $f^{-1} : \Psi \to X$ with geometric harmonics are visualized in Figure 3.20.

With this pre-image mapping f^{-1}, both directions become feasible and the underlying manfiold can be reconstructed by sampling. This is done similar to the one-dimensional case by transporting uniform samples (2D) with a normalizing flow to the parametrization (2D embedding). This surface is then refolded by mapping the points with the geometric harmonics to the PCA of the diffeomorphic surface (3D). By applying the inverse of the PCA, the position on $\left([\beta_1]_{n+1}, [\beta_2]_n, [\beta_2]_{n+1}\right)$ recovers the original β_2. The results of this complete process are depicted in Figure 3.21.

With increasing dimensions, approximating the correct solution requires some additional fine-tuning. The results presented here show that the essential parts, and especially the fold, are well captured. Future work is needed for further improvement, specifically at the tails of the map. The part of the approximation where improvement seems most promising is the function f^{-1}. This can either be done by using a different process or by improving hyper-parameters and the learning process.

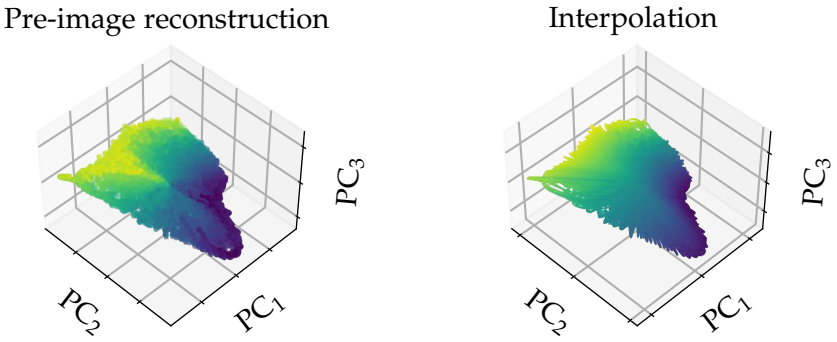


Figure 3.20.: In the left plot, the points from the two-dimensional DMAP embedding are transported back into three dimensions with f^{-1}. The color scheme in the left figure is according to PC_2 of the previously constructed diffeomorphic surface. Points are shifted to the center and thus not mapped back perfectly to their original position. The right figure shows points sampled from $\mathcal{U}_{[-1,1]^2}$ that are transported with a normalizing flow to the DMAP space and then mapped with f^{-1}. The color gradient in this plot is *not* based on PC_2 of the diffeomorphic surface but changes with PC_2 of this plot.

3.4.5. Transporting marginal to joint densities

In all previous examples, direct transportation from the observed space to the original space was possible because the known intrinsic dimension matched the one of the observed space. In the following, we will look at how a measurement of merely a marginal density $p(\beta_2)$ can be used to recover the underlying manifold, and with it, the joint density $p(\beta_1, \beta_2)$. This was proposed by [45], and the same cusp surface from Figure 3.16 (center) is used to explain the concept.

Namely, the function of the surface is $\beta_2(x, \beta_1) = x^3 - \beta_1 x$ and it produces a discontinuous joint density, as seen before. This time, we assume that only β_2 can be observed. Previously, we had the example of ships on the ocean, where those ships could measure the temperature β_2, and additionally there is a person on an island capable of measuring the coordinate β_1 with a telescope. Now, there is no island and only the temperature can be recorded. The density that arises with this example is illustrated in Figure 3.22.

Unlike in previous examples, this one-dimensional density cannot be directly transported to two dimensions because of the mismatching dimensions. With an observation process, the dimensionality can be artificially increased with time delays. In this section, the objects (i.e. the ships) are still uniformly distributed on the manifold (i.e. the sea), but this time, they all move in x-direction. Similarly to the one-dimensional example from Section 3.2.1, we begin by collecting one additional time delay $([\beta_2]_n, [\beta_2]_{n+1})$

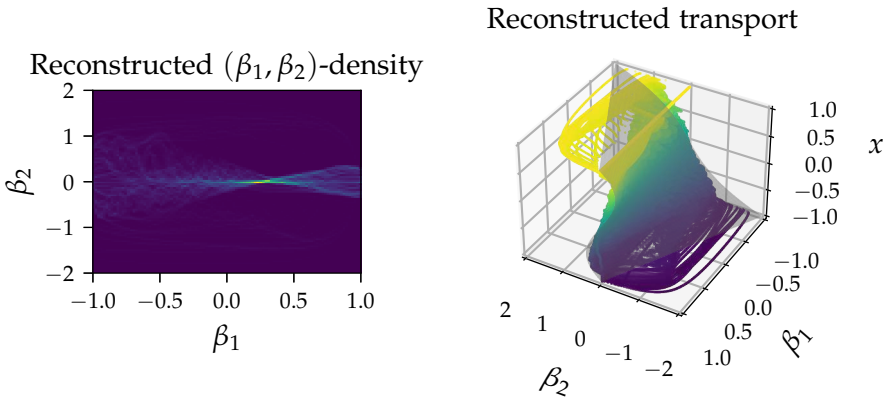


Figure 3.21.: The reconstructed density of (β_1, β_2) is shown in the left, and the right visualizes the approximated manifold with the real solution plotted translucently. Compared to the direct approach presented in Figure 3.17, this solution contains a fold.

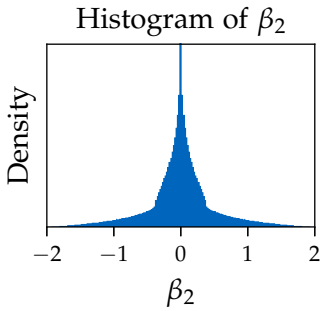


Figure 3.22.: The marginal density of β_2 can be seen, where the distribution is determined by the cusp surface. It is again discontinuous, but the jumps are more subtle.

to create the diffeomorphic manifold. As seen in Figure 3.23 (left), contrary to the one-dimensional example, this plot has overlaps and does not look like a curve. Thus, more time delays need to be collected for a diffeomorphic version. In Figure 3.23 (right), when collecting $\left([\beta_2]_n, [\beta_2]_{n+1}, [\beta_2]_{n+2}\right)$ instead, a non-overlapping surface can be found. This visual process is not possible in higher dimensions, but one can use dimension estimation algorithms [60] to find the intrinsic dimension of the manifold [45].

With this procedure, the dimension was increased, and a diffeomorphic version of the manifold can be reconstructed. From that point onwards, this scenario is identical to the one before. By learning how to unfold and refold the density with DMAPs and geometric harmonics, and transporting this parametrized surface with a normalizing flow to a two-dimensional uniform density, the underlying manifold can be reconstructed. The results of this reconstruction process are depicted in Figure 3.24.

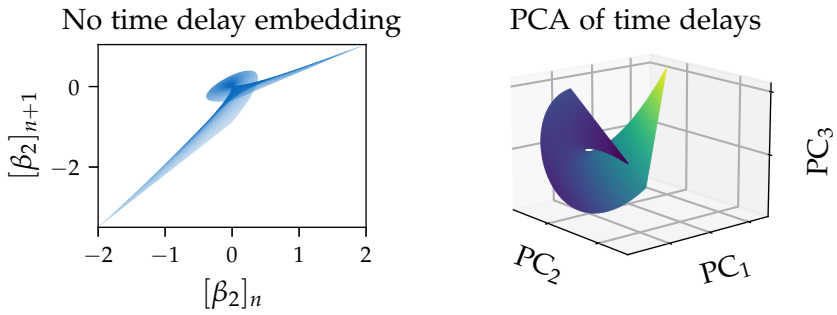


Figure 3.23.: The left image shows that an embedding with one additional time delay overlaps and is thus not diffeomorphic. In the right image, the PCA of three time delays does not overlap and illustrates a diffeomorphic time delay embedding. The coloring scheme for the three-dimensional plots is according to the first principal component.

The results presented face the same issues as reconstruction did in Section 3.4.4 and need further work to better approximate the real manifold. In this case, it is even more impressive that a transportation can be found, because previously it was not possible at all. And not only any arbitrary transport map could be found, but the real underlying surface could be approximated up to numerical errors. This shows a major benefit when using this procedure instead of classical transportation theory.

3.4.6. Learning the dynamic in higher dimensions

As discussed in Section 3.3, when the underlying map is folded, the observed state is often insufficient to predict the future behavior of the dynamical system. In the one-dimensional case, this problem was overcome by using the parametrization instead of the observed measurement y directly. As the arc length of the curve is monotonically increasing, and is thus invertible, the future could be predicted deterministically. This section is dedicated to how this concept can be used for higher-dimensional manifolds, where the parametrization by arc length is not possibly anymore.

Before, the parametrization by arc length has been substituted by DMAPs and geometric harmonics. These extensions can also be made here by using the position on the DMAP embedding as the intrinsic state. With subsequent measurements from the moving objects, the dynamical system of higher-dimensional manifolds can be learned analogously to the one-dimensional case described in Section 3.3.1. When the mapping to and from the embedding is accurate enough, learning it on this parametrization can be advantageous, as the lower dimension reduces unnecessary computational overhead. Most algorithms, especially machine learning ones, can benefit from this pre-processing step, as they do not need the complexity to express a higher-dimensional manifold.

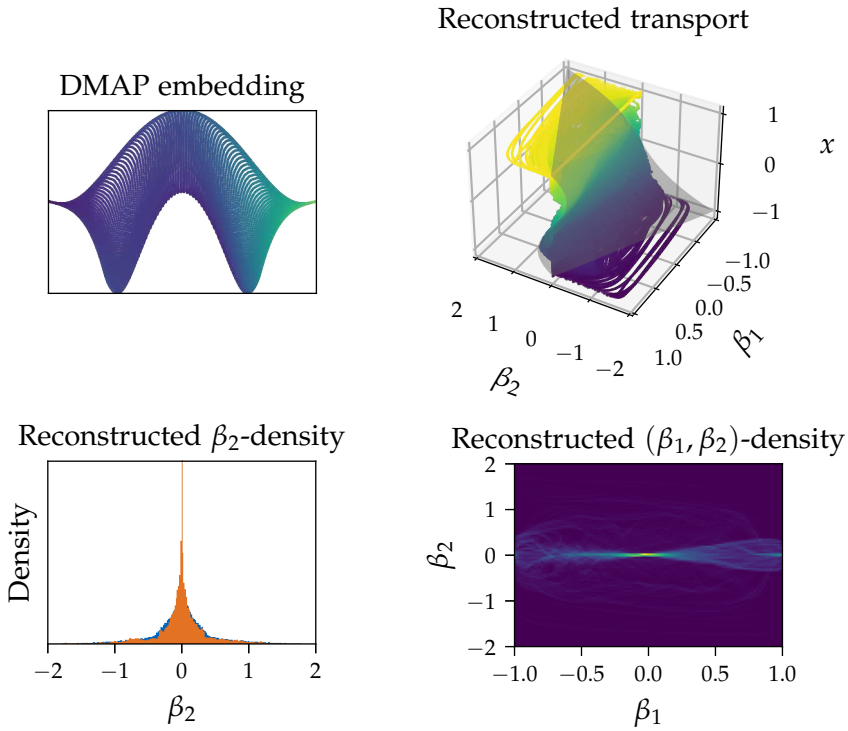


Figure 3.24.: The top left image shows the two-dimensional embedding that can be found for the PCA of the diffeomorphic surface. The top right illustration shows the reconstructed underlying manifold and the original one printed translucently. With this manifold, the marginal density of β_2 (bottom left) and the joint density of (β_1, β_2) (bottom right) can be reconstructed.

But, as we saw in the previous section, it is not always trivial to use manifold learning to approximate the truth well enough, especially when the transformation has to be performed in both directions. This parametrization step is not strictly necessary because the time delay embedding itself gives every point a unique state—the position on the manifold in higher-dimensional space. Thus, when learning the dynamical system, the position on the time delay embedding, or the PCA of it, can be used as the intrinsic state.

For the actual training process, every measurement $\vec{s}_n$ requires a successor state $\vec{s}_{n+1}$. In the one-dimensional example, we simply used the arc length tuples as states and interpreted them as a car that moves to the position of the next car in one timestep. This did not require any change in the observation process, but in higher dimensions a "next" point does not trivially exist anymore.

Depending on the actual physical measurement process underlying the observations, one might need to follow an object for even more time delays to be able to infer the subsequent states on the time delay embedding. In some cases, it is sufficient to keep the current measurement process unaltered. With the ship analogy, it might be possible to interpret the measurements as a row of ships. After each timestep, every ship in that row sails to the next position, and so on. This results in multiple lists (i.e. columns of ships' measurements) where the successor states are known. In other cases, the measurements are unordered, and the objects need to be followed for a longer time.

By integrating the neural network $\mathbf{f}$ that approximates the vector field, the complete arc length was reconstructed in the one-dimensional example before. In higher dimensions, the manifold cannot be fully reconstructed anymore by solving the initial value problem. Solving it will only result in the trajectory of a single object with its corresponding measurements. When again viewing this vector field as turbulence in water (e.g. a river), this makes more sense. A leaf is placed on the start and its positions observed as it floats on the water. The river is higher-dimensional, but the leaf (i.e. a single underlying object) is only a point moving through that space. This will always result in a curve that does not reconstruct the movement of the complete river.

3.5. Learning the shape of a cell from image data

All previous examples were very abstract and only the analogies made them somehow directly applicable to the real world. While the data in this section is still artificially generated, it aims to illustrate a more practical use case for the reconstruction of the underlying manifold. In this scenario, movement of bacteria on a single cell is simulated and recorded by a series of pictures. These two-dimensional pictures only record partial information about the three-dimensional position of bacteria on the cell's surface. The goal is to learn the underlying "rules" for the movement of bacteria on this cell, when the microscope records the film. This movement is guided by the shape of the cell, and thus we want to reconstruct the true underlying manifold.

Single-cell biology, see for example [32], has become an interesting topic for many biologists over the last few years, due to the increase in computational power and better microscopes becoming available. Understanding the behavior of objects on the cell can be useful in a variety of biological applications. When observing viruses, or bacteria that can penetrate cells, finding out where they move to could be useful to determine the most probable points of entry. It could also be interesting to observe how the cell's surface changes when infected with bacteria or to simply determine the difference between various cell types and compare them.

Contrary to before, the goal in this section is only to recover the shape of the cell, instead of also approximating the underlying density. Figure 3.25 shows the simulated image data that will be used for reconstruction. The bacteria are uniformly distributed on the cell's surface [27] and move according to a sine wave up and down. A time series of the (x, y)-coordinates of each bacterium relative to the cell can be observed. Since cells are translucent, bacteria can even be observed when they are "behind" the cell. Further, with this construction, the paths do not intersect, otherwise the procedure would find a higher-than-three-dimensional manifold that describes the movement, not the shape itself. The resulting density when only collecting two coordinates is again discontinuous, because the map is folded (multiple times). Singularities arise where the gradient is zero: at the equator and the poles. Due to this, and because the dimensions do not match, classical transport to a three-dimensional manifold is not possible.

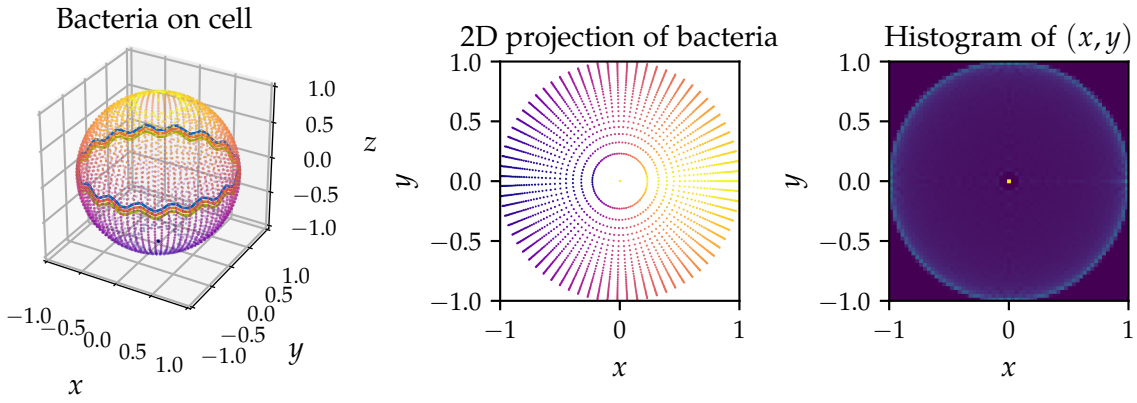


Figure 3.25.: The left illustration shows the generated image data with the movement of three bacteria. As the microscope can only take two-dimensional images, the center plot shows what the microscope might observe when directly looking at the cell from the top. The arising discontinuous density is seen in the right figure.

The simulation is implemented in such a way that a single "column" of bacteria rotates around the sphere. They move according to a sine wave along the great circle they lie on. The recorded film gives access to (x, y)-coordinates at specific timesteps. With this data, reconstruction of the sphere is nearly identical to the previous examples. Takens' theorem limits the number of time delays to 5, hence the measurements $(x_n, y_n, x_{n+1}, y_{n+1}, x_{n+2}, y_{n+2})$ are sufficient for a diffeomorphic version of the cell. Visually, this cannot be verified anymore because the dimension is too high. As we did in Section 3.4, PCA is applied to this time delay embedding of which the first three principal components are visualized in Figure 3.26 (left).

Analogously to before, a three dimensional embedding is constructed with DMAP, and a normalizing flow trained to transport this embedding to the density of the uniform distribution $\mathcal{U}_{[-1,1]^3}$. In Section 3.4, the manifold was two-dimensional (as is in this case) but it could be embedded into two dimensions. While a sphere can be parameterized by the longitude and latitude, it needs three dimensions for an embedding. Thus, transportation is performed to a three-dimensional uniform density.

In previous examples, and figures, the manifold was reconstructed by sampling new points that were transported with geometric harmonics. This can also be done here, but as we are only interested in the shape of the cell, the inverse direction will be omitted. Figure 3.26 (right) shows the approximated reconstructed shape. The results need some further fine-tuning to perfectly approximate the cell, but this section demonstrates real-world applicability of this thesis and serve as a proof of concept. Learning the dynamical system of the bacteria itself could also be interesting, when the movement of the objects on the cells needs to be investigated.

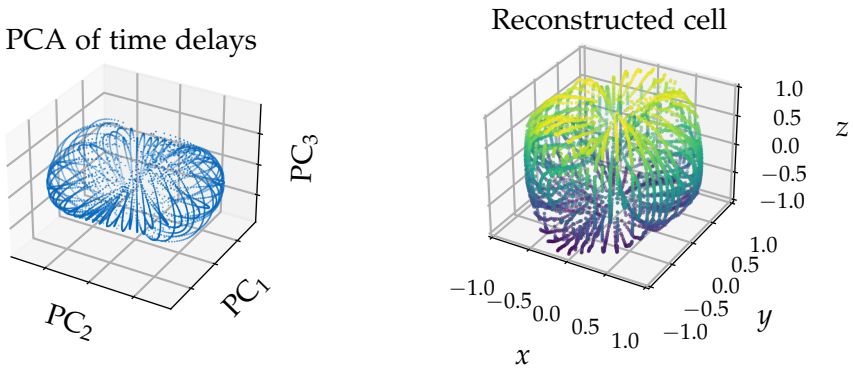


Figure 3.26.: The first three principal components of the six-dimensional time delay embedding are shown in the left figure. The right plot shows the reconstruction of the cell by transporting (x, y) with the presented procedure to $\mathcal{U}_{[-1,1]^3}$ and adding z.

4. Conclusion

In this thesis, we investigated the dynamical process of objects moving on (higher-dimensional) unknown manifolds. When the measurement process only gives partial information regarding the position of those objects on the manifold, the observed behavior can seem complex, even for simple manifolds. The case where the observed density of those partial measurements is discontinuous was particularly important in this thesis because those singularities suggest that the underlying map might be folded. Most transportation algorithms—including normalizing flows—fail to reconstruct the true structure of the underlying manifold when the map is folded.

To recover the true underlying manifold, a simple measurement process is already insufficient to decide whether the found solution is the correct one because the functions only differ in the pointwise mapping. As a solution, an observation process that follows the underlying objects and collects multiple time delayed measurements was employed. With enough time delays, Takens' theorem states that a diffeomorphic version of the true manifold can be constructed. As the number of time delays is typically larger than the intrinsic dimension, the lower-dimensional manifold is embedded into a higher dimension. By using manifold learning—DMAPs in this thesis—the lower-dimensional parametrized manifold can be learned and transported back to the original (unknown) space with normalizing flows. Geometric harmonics complete reconstruction by allowing transformation back into the higher dimension, so the transport can map in both directions.

This observation process enables reconstruction of the true underlying manifold (up to diffeomorphism) as well as the learning of the underlying dynamical system. To illustrate this, a neural network was trained that acts as the underlying vector field (i.e. the right-hand side of a corresponding ODE) that can be used to predict subsequent states. To train this neural network, the vector field is numerically integrated one step with Runge-Kutta and the output is then compared to the ground truth.

Learning the dynamical process and identifying the true underlying system (or a diffeomorphic version of it), can be applied in many fields and used in various contexts where measurements are taken. In the last section, we illustrated this versatility with an example that shows how the shape of a simulated cell can be reconstructed by observing the movement of bacteria on it. This example demonstrates real world-applicability of the thesis, even when conventional measurement processes (e.g. photography) are

used. We also saw that the area of normalizing flows is a promising new research field that is worth exploring further.

Most approximations, especially in the one-dimensional case, can already be useful and be applied in practice. Higher-dimensional examples may require more fine-tuning, as highlighted in the illustrative-examples and when reconstructing the manifold of the cell (i.e. sphere). During testing, normalizing flows approximated the target density well. Hence, we suggest improving the application of manifold learning with DMAPs, and the pre-image mapping with geometric harmonics. Once better reconstruction in the examples presented in this thesis is feasible, the procedure can be applied to real-world examples with potentially inaccurate data.

The latest version of this thesis, and the accompanying source code are available at https://github.com/plainerman/density-transport. Note that the provided implementation is not ready for production but intended as a proof of concept.

A. Normalizing flows with Pyro and PyTorch

In this thesis, the library Pyro [6, 54] was used for normalizing flows. To illustrate how a NF can be trained, Listing A.1 shows an exemplary implementation to find the transport between a uniform and a normal density. `spline_transform` is the approximated transport function T, capable of mapping from the normal density to the uniform density. The inverse function T^{-1} can be calculated with `spline_transform.inv`. This code excerpt along with the complete implementation to produce the figures presented in this thesis, and the code for this document itself can be found at https://github.com/plainerman/density-transport.

```
1  import numpy, torch, pyro.distributions as dist
2  from tqdm import tqdm
3
4  samples = numpy.random.random((10000, 2)) # draw uniform samples
5  dataset = torch.tensor(samples, dtype=torch.float)
6
7  base_dist = dist.Normal(torch.zeros(2), torch.ones(2)) # standard normal
8
9  spline_transform = dist.transforms.spline_coupling(2, count_bins=16) # T
10 flow_dist = dist.TransformedDistribution(base_dist, [spline_transform])
11
12 optimizer = torch.optim.Adam(spline_transform.parameters(), lr=1e-2)
13
14 steps = 1000
15 iterator = tqdm(range(steps), desc="Training NF")
16 for step in iterator:
17     optimizer.zero_grad()
18
19     # optimize based on maximum likelihood of dataset
20     loss = -flow_dist.log_prob(dataset).mean()
21     loss.backward()
22     optimizer.step()
23
24     flow_dist.clear_cache()
25     iterator.set_postfix(loss=loss.item()) # update progress bar
26
27 # draw new samples
28 new_samples = flow_dist.sample(torch.Size([1000,])).detach().numpy()
29 # transport observed uniform samples with inverse of T to standard normal
30 normal_samples = spline_transform.inv(dataset).detach().numpy()
```

Listing A.1: This code example implements a transport from the uniform to a standard normal density. With 10,000 uniform random samples, and a neural spline flow with 16 bins, new samples can be drawn or the original uniform points transported to the normal space. When a different two-dimensional distribution should be learned, only line 4 needs to be adjusted.

Bibliography

[1] J. Adler, I.-M. Sintorn, R. Strand, and I. Parmryd, "Conventional analysis of movement on non-flat surfaces like the plasma membrane makes brownian motion appear anomalous," *Communications biology*, vol. 2, p. 12, 2019. DOI: 10.1038/s42003-018-0240-2.

[2] D. Aeyels, "Generic observability of differentiable systems," *SIAM Journal on Control and Optimization*, vol. 19, no. 5, pp. 595–603, 1981, ISSN: 0363-0129. DOI: 10.1137/0319037.

[3] C. C. Aggarwal, "An introduction to neural networks," in *Neural Networks and Deep Learning*, C. C. Aggarwal, Ed., Cham: Springer International Publishing, 2018, pp. 1–52, ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463-0_1.

[4] U. M. Ascher and L. R. Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations*. Philadelphia: Society for Industrial and Applied Mathematics, 1998, ISBN: 978-0-89871-412-8.

[5] J.-D. Benamou, B. D. Froese, and A. M. Oberman, "Numerical solution of the optimal transportation problem using the monge–ampère equation," *Journal of Computational Physics*, vol. 260, pp. 107–126, 2014, ISSN: 0021-9991. DOI: 10.1016/j.jcp.2013.12.015.

[6] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep Universal Probabilistic Programming," *Journal of Machine Learning Research*, 2018.

[7] V. I. Bogachev, A. V. Kolesnikov, and K. V. Medvedev, "Triangular transformations of measures," *Sbornik: Mathematics*, vol. 196, no. 3, pp. 309–335, 2005, ISSN: 1064-5616. DOI: 10.1070/SM2005v196n03ABEH000882.

[8] V. I. Bogachev, *Measure theory*. Berlin: Springer, 2007, ISBN: 978-3-540-34513-8.

[9] L. Chen and R. K. Mehra, "A study of nonlinear filters with particle flow induced by log-homotopy," in *Signal Processing, Sensor Fusion, and Target Recognition XIX*, I. Kadar, Ed., ser. SPIE Proceedings, SPIE, 2010, p. 769 706. DOI: 10.1117/12.853001.

[10] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 5–30, 2006, ISSN: 1063-5203. DOI: 10.1016/j.acha.2006.04.006.

[11] R. R. Coifman and S. Lafon, "Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 31–52, 2006, ISSN: 1063-5203. DOI: 10.1016/j.acha.2005.07.005.

[12] C. Cotar, G. Friesecke, and C. Klüppelberg, "Density functional theory and optimal transportation with coulomb cost," *Communications on Pure and Applied Mathematics*, vol. 66, no. 4, pp. 548–599, 2013, ISSN: 0010-3640. DOI: 10.1002/cpa.21437.

[13] N. Courty, R. Flamary, and D. Tuia, "Domain adaptation with regularized optimal transport," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, T. Calders, F. Esposito, E. Hüllermeier, and R. Meo, Eds., vol. 8724, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 274–289, ISBN: 978-3-662-44847-2. DOI: 10.1007/978-3-662-44848-9_18.

[14] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26, Curran Associates, Inc., 2013.

[15] L. Dickmanns and M. Sellami, *Masters practicum mlcms*, https://github.com/ldickmanns/masters-practicum-mlcms, commit: 13b99fb2b37d1c2d4bd2, 2020.

[16] F. Dietrich, Chair of Scientific Computing in Computer Science, Department of Informatics, Technical University of Munich, personal communication, https://fd-research.com, Jul. 28, 2021.

[17] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[18] L. Dinh, J. Sohl-Dickstein, and S. Bengio, *Density estimation using real nvp*, 2016. arXiv: 1605.08803v3.

[19] C. J. Dsilva, R. Talmon, C. W. Gear, R. R. Coifman, and I. G. Kevrekidis, "Data-driven reduction for a class of multiscale fast-slow stochastic dynamical systems," *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 3, pp. 1327–1351, 2016. DOI: 10.1137/151004896.

[20] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, *Cubic-spline flows*, 2019. arXiv: 1906.02145v1.

[21] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, *Neural spline flows*, 2019. arXiv: 1906.04032v2.

[22] J. Feydy, P. Roussillon, A. Trouvé, and P. Gori, "Fast and scalable optimal transport for brain tractograms," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, ser. Lecture Notes in Computer Science, D. Shen, T. Liu, T. M. Peters, L. H. Staib, C. Essert, S. Zhou, P.-T. Yap, and A. Khan, Eds., vol. 11766, Cham: Springer International Publishing, 2019, pp. 636–644, ISBN: 978-3-030-32247-2. DOI: 10.1007/978-3-030-32248-9_71.

[23] B. D. Froese and A. M. Oberman, "Convergent finite difference solvers for viscosity solutions of the elliptic monge–ampère equation in dimensions two and higher," *SIAM Journal on Numerical Analysis*, vol. 49, no. 4, pp. 1692–1714, 2011, ISSN: 0036-1429. DOI: 10.1137/100803092.

[24] C. Gaetan and X. Guyon, *Spatial Statistics and Modeling*. New York, NY: Springer New York, 2010, ISBN: 978-0-387-92256-0. DOI: 10.1007/978-0-387-92257-7.

[25] A. Galichon, *Optimal transport methods in economics*. Princeton: Princeton University Press, 2018, ISBN: 978-0-691-18346-6.

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

[27] D. C. Hall, "Sampling random directions within an elliptical cone," *Computer physics communications*, vol. 219, pp. 87–90, 2017, ISSN: 0010-4655. DOI: 10.1016/j.cpc.2017.05.010.

[28] A. Hefny, C. Downey, and G. J. Gordon, "Supervised learning for dynamical system learning," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015.

[29] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417–441, 1933, ISSN: 0022-0663. DOI: 10.1037/h0071325.

[30] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville, "Neural autoregressive flows," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 2078–2087.

[31] A. J. Izenman, "Introduction to manifold learning," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 5, pp. 439–446, 2012, ISSN: 1939-5108. DOI: 10.1002/wics.1222.

[32] J. P. Junker and A. van Oudenaarden, "Every cell is special: Genome-wide studies add a new dimension to single-cell biology," *Cell*, vol. 157, no. 1, pp. 8–11, 2014. DOI: 10.1016/j.cell.2014.02.010.

[33] C. Kacper, S. Heiko, and G. Arthur, "A kernel test of goodness of fit," *International Conference on Machine Learning*, pp. 2606–2615, 2016, ISSN: 1938-7228.

[34] L. V. Kantorovich, "On the translation of masses," *Dokl. Acad. Nauk SSSR*, vol. 37, no. 7-8, pp. 227–229, 1942.

[35] L. V. Kantorovich, "On a problem of monge," *Uspekhi Mat. Nauk*, vol. 3, no. 2, pp. 225–226, 1948.

[36] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. arXiv: 1412.6980v9.

[37] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018.

[38] I. Kobyzev, S. Prince, and M. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. PP, 2020. DOI: 10.1109/TPAMI.2020.2992934.

[39] J. Langford, R. Salakhutdinov, and T. Zhang, "Learning nonlinear dynamic models," in *Proceedings, twenty-sixth international conference on machine learning*, L. Bottou and M. Littman, Eds., United States: ICML, 2009, pp. 1–8, ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553451.

[40] J. M. Lee, *Introduction to topological manifolds*, ser. Graduate texts in mathematics. New York and London: Springer, 2000, vol. 202, ISBN: 0-387-98759-2.

[41] D. Lehmberg, F. Dietrich, G. Köster, and H.-J. Bungartz, "Datafold: Data-driven models for point clouds and time series on manifolds," *Journal of Open Source Software*, vol. 5, no. 51, p. 2283, 2020. DOI: 10.21105/joss.02283.

[42] N. Lei, K. Su, L. Cui, S.-T. Yau, and D. X. Gu, *A geometric view of optimal transportation and generative model*, 2017. arXiv: 1710.05488v2.

[43] D. Marchenko, D. V. Evtushinsky, E. Golias, A. Varykhalov, T. Seyller, and O. Rader, "Extremely flat band in bilayer graphene," *Science advances*, vol. 4, no. 11, eaau0059, 2018. DOI: 10.1126/sciadv.aau0059.

[44] G. Monge, "Mémoire sur la théorie des déblais et des remblais," *Histoire de l'Académie Royale des Sciences de Paris*, pp. 666–704, 1781.

[45] C. Moosmüller, F. Dietrich, and I. G. Kevrekidis, "A geometric approach to the transport of discontinuous densities," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 8, no. 3, pp. 1012–1035, 2020. DOI: 10.1137/19M1275760.

[46] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák, *Neural importance sampling*, 2018. arXiv: 1808.03856v5.

[47] E. J. Nyström, "Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben," *Acta Mathematica*, vol. 54, no. 0, pp. 185–204, 1930, ISSN: 0001-5962. DOI: 10.1007/BF02547521.

[48] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a time series," *Physical Review Letters*, vol. 45, no. 9, pp. 712–716, 1980, ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.45.712.

[49] G. Papamakarios, *Neural density estimation and likelihood-free inference*, 2019. arXiv: 1910.13233v1.

[50] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[52] 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*, vol. 12, pp. 2825–2830, 2011.

[53] G. Peyré and M. Cuturi, "Computational optimal transport," *Foundations and Trends in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[54] D. Phan, N. Pradhan, and M. Jankowiak, *Composable effects for flexible and accelerated probabilistic programming in numpyro*, 2019. arXiv: 1912.11554v1.

[55] D. Pollard, *A user's guide to measure theoretic probability*, ser. Cambridge series in statistical and probabilistic mathematics. Cambridge: Cambridge University Press, 2002, ISBN: 978-0-521-80242-0.

[56] S. J. Reddi, S. Kale, and S. Kumar, *On the convergence of adam and beyond*, 2019. arXiv: 1904.09237v1.

[57] R. Rico-Martínez, K. Krischer, I. G. Kevrekidis, M. Kube, and J. L. Hudson, "Discrete- vs. continuous-time nonlinear signal processing of cu electrodissolution data," *Chemical Engineering Communications*, vol. 118, no. 1, pp. 25–48, 1992, ISSN: 0098-6445. DOI: 10.1080/00986449208936084.

[58] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951, ISSN: 0003-4851. DOI: 10.1214/aoms/1177729586.

[59] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science (New York, N.Y.)*, vol. 290, no. 5500, pp. 2323–2326, 2000, ISSN: 0036-8075. DOI: 10.1126/science.290.5500.2323.

[60] A. Rozza, G. Lombardi, M. Rosa, E. Casiraghi, and P. Campadelli, "Idea: Intrinsic dimension estimation algorithm," in *Image analysis and processing - ICIAP 2011*, ser. Lecture notes in computer science, 0302-9743, G. Maino and G. L. Foresti, Eds., vol. 6978, Heidelberg: Springer, 2011, pp. 433–442, ISBN: 978-3-642-24084-3. DOI: 10.1007/978-3-642-24085-0_45.

[61] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, ISSN: 0028-0836. DOI: 10.1038/323533a0.

[62] I. J. Schönberg, "Contributions to the problem of approximation of equidistant data by analytic functions. part a. on the problem of smoothing or graduation. a first class of analytic approximation formulae," *Quarterly of Applied Mathematics*, vol. 4, no. 1, pp. 45–99, 1946, ISSN: 0033-569X. DOI: 10.1090/qam/15914.

[63] I. J. Schönberg, "Contributions to the problem of approximation of equidistant data by analytic functions. part b. on the problem of osculatory interpolation. a second class of analytic approximation formulae," *Quarterly of Applied Mathematics*, vol. 4, no. 2, pp. 112–141, 1946, ISSN: 0033-569X. DOI: 10.1090/qam/16705.

[64] G. Strang, *Calculus*. Wellesley, Mass.: Wellesley-Cambridge Press, 1991, ISBN: 978-0-9614088-2-4.

[65] S. H. Strogatz, *Nonlinear dynamics and chaos: With applications to physics, biology, chemistry and engineering*, Second Edition. Philadelphia, PA: Westview Press, 2015, ISBN: 978-0-8133-4910-7.

[66] E. G. Tabak and C. V. Turner, "A family of nonparametric density estimation algorithms," *Communications on Pure and Applied Mathematics*, vol. 66, no. 2, pp. 145–164, 2013, ISSN: 0010-3640. DOI: 10.1002/cpa.21423.

[67] E. G. Tabak and E. Vanden-Eijnden, "Density estimation by dual ascent of the log-likelihood," *Communications in Mathematical Sciences*, vol. 8, no. 1, pp. 217–233, 2010, ISSN: 1539-6746. DOI: 10.4310/CMS.2010.v8.n1.a11.

[68] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*, D. A. Rand and L.-S. Young, Eds., ser. Lecture Notes in Mathematics, vol. 898, Berlin: Springer, 1981, pp. 366–381, ISBN: 978-3-540-38945-3. DOI: 10.1007/bfb0091924.

[69] L. W. Tu, *An Introduction to Manifolds*. New York, NY: Springer New York, 2011, ISBN: 978-1-4419-7400-6.

[70] C. Villani, *Topics in Optimal Transportation*, ser. Graduate studies in mathematics, 1065-7339. Providence, R.I. and Great Britain: American Mathematical Society, 2003, vol. v. 58, ISBN: 0-8218-3312-X.

[71] C. Villani, *Optimal transport: Old and new*, ser. Grundlehren der mathematischen Wissenschaften, 0072-7830. Berlin and London: Springer, 2009, vol. 338, ISBN: 978-3-540-71050-9.

[72] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[73] H. Whitney, "Differentiable manifolds," *The Annals of Mathematics*, vol. 37, no. 3, p. 645, 1936, ISSN: 0003-486X. DOI: 10.2307/1968482.

[74] Z. Zhang and J. Wang, "MLLE: Modified locally linear embedding using multiple weights," vol. 19, Jan. 2006, pp. 1593–1600.