Mathematics in Data Science

Technische Universität München

Master's Thesis

Isometric Embedding of Manifolds with Auto-Encoder Networks

Sarah El Beji

Mathematics in Data Science

Technische Universität München

Master's Thesis

Isometric Embedding of Manifolds with Auto-Encoder Networks

Author:	Sarah El Beji
Supervisor:	Prof. Dr. rer. nat. habil. Hans-Joachim Bungartz
Advisor:	Dr. rer. nat. Felix Dietrich
Submission Date:	September 22nd, 2022

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

September 22nd, 2022

Sarah El Beji

[illegible]

Abstract

We use Auto-Encoders in the context of Manifold Learning to learn lower dimensional embeddings of higher dimensional manifolds. Using different loss functions expresses diverse regularization approaches. In this thesis, we compare two auto-encoders that create an approximately isometric embedding in the latent space. These methods are I-AE and LOCA. We compare their mathematical background and results and establish use cases for which we compare them. Our results show that LOCA has an advantage when burst sampling is available and can find an isometric embedding that I-AE cannot. Furthermore, when burst sampling is unavailable, LOCA gives close results to I-AE at the cost of two additional hyper-parameters.

Kurzfassung

Wir verwenden Auto-Encoder im Zusammenhang mit Manifold Learning, um niedrigdimensionale Einbettungen von höherdimensionalen Mannigfaltigkeiten zu lernen.Unterschiedliche Verlustfunktionen erlauben uns, unterschiedliche Regularisierungsansätze zu encodieren. In dieser Arbeit vergleichen wir zwei auto-encoder, die Eingangsdaten annähernd isometrisch einbetten, um einen. Diese Verfahren sind I-AE und LOCA. Wir vergleichen ihre mathematischen Hintergründe und Ergebnisse und ermitteln Anwendungsfälle, für die wir sie vergleichen. Unsere Ergebnisse zeigen, dass LOCA einen Vorteil hat, wenn Burst-Sampling verfügbar ist, und eine isometri-sche Einbettung finden kann, die I-AE nicht finden kann. Darüber hinaus gibt LOCA, wenn Burst-Sampling nicht verfügbar ist, Ergebnisse ähnlich zu denen von I-AE auf Kosten von zu I-AE auf Kosten von zwei zusätzli-chen Hyperparametern.

Contents

Part I.

Introduction and Background Theory

1. Introduction

Many real-life problems ranging from human motion analysis to fraud detection or recommender systems, represent complex phenomena expressed in high dimensional data that boil down to a lower dimensional space. For instance, in computer vision, a picture of a face in high-resolution pixels is at the end, only a combination of the particular positioning of our facial features, which lies in a much lower dimension than the picture itself. In scientific computing, we can use different measures gathered from an object to uncover a particular set of information: high-dimensional astrophysical data of stars can reveal their composition. It is thus crucial to develop methods that can compress this type of information since it would mean smaller amounts of data with, hopefully, the same knowledge.

Manifold learning encompasses many such models that cross various fields and have become increasingly sophisticated throughout the last two decades. In this thesis, we study two state-of-the-art models developed in different contexts.

On the one hand, Peterfreund et al. [26] introduce, in the context of scientific computing, the Local Conformal Autoencoder for Standardized Data Coordinates (LOCA). This model finds normalized coordinates in a lower space of high dimensional measurements. It uses repeated sampling on the high dimensional manifold to find an isometric embedding in a lower dimension using Auto-Encoder Neural Networks.

On the other hand, Gropp et al. [13] introduce Isometric Auto-Encoders (I-AE), which like LOCA, uses auto-encoders to impose an isometry when finding a lower dimensional embedding. Nevertheless, I-AE does not use burst sampling and defines the isometry differently. It is thus interesting to compare these seemingly similar approaches and uncover if we can use one model in the context of the other, which is our contribution. To this end, we divide our thesis into, a first section 2 and a main part (Part II) that encompasses three sections 3, 4 and 5.

In section 2, we detail the mathematical background used in this thesis and the different historical approaches for manifold learning. In particular, we cover isometric manifold learning methods. In section 3, we lay down the methodology to approach our comparison of LOCA and I-AE. In section 4, we explain both I-AE and LOCA separately. First, we detail their proofs for isometrical embeddings and explain how their mathematical approaches differ. Secondly, we introduce their contextual use cases and present a measure of the isometry quality. In section 5, we show the results of our implementations of the empirical comparison framework we purpose in section 4. Finally, in part III, we discuss our results and propose further continuations of our work.

2. Background Knowledge and Related Work

We divide this first section into three subsections. First, we present the mathematical definition of manifolds. One algorithm discussed in the papers studied in this thesis establishes proof that the proposed methods learn isometric embeddings using geometrical properties of the manifold; it is essential to establish the mathematical notions behind Manifold Learning which stems from Differential Geometry. In the second section, we discuss the first classical manifold learning methods. Methods like PCA that do not infer an isometric embedding will not be discussed in depth. However, the methods we find crucial to illustrate how the problem of uncovering isometric embedding was approached in literature are discussed in more detail. It is essential to point out that the study of these different models paved the way for the framework to establish a comparison between the two studied models. We were inspired by how texts in literature compare former methods to the methods they introduce, and we picked their line of thought when establishing our comparison scheme. The third section introduces the concept of Deep Learning, Neural Networks, and Auto- Encoders, which are the backbone model used to uncover isometric embeddings in this thesis.

Manifold learning encloses a variety of techniques that seek to represent high-dimensional data through a transformation from one higher-dimensional space $\mathcal{M}$ to another lower dimensional space $\mathcal{N}$. In the following section, we explain the notion of manifolds and how we can attempt to isometrically map them to Euclidean space in the context of manifold learning. We start by defining the abstract notion of a manifold which is any collection of points with a specific topology. We assume that this manifold represented in high-dimensional space actually lies on a lower dimensional space.

A recurrent intuitive example of a manifold is how the sphere $\mathbb{S}^2$, a 2-dimensional manifold, is embedded in a higher 3-dimensional space $\mathbb{R}^3$. Thus the 3-d data actually lies in a 2-d manifold. Differential Geometry studies the geometry of smooth (infinitely differentiable) manifolds like $\mathcal{S}^2$ and gives important results on how to parametrize such topological spaces.

In the following, we establish rigorously the mathematical background [15][28][17] for manifold learning and the terms we used in the example of the sphere also illustrated in figure 2.1.

2.1. Mathematical Prerequisites

We have followed the outline of [15] [28] and [17] to present the mathematical prerequisites.

2.1.1. Manifolds in Differential Geometry

Definition 2.1 (Topological Space) *A topological space $\mathcal{T} = (X, \Omega)$ where X is a set and Ω a collection of open subsets such that :*

- $\emptyset \in \Omega$
- $X \in \Omega$
- *a finite intersection of sets from Ω is in Ω.*
- *a finite or infinite union of sets in Ω is in Ω*

Definition 2.2 (Homeomorphism) *A function ϕ between two topological spaces is a homeomorphism if it is a continuous invertible function and admits a continuous inverse.*

Definition 2.3 (Charts and local coordinates) *Let $\mathcal{M}$ be a topological space and $p \in \mathcal{M}$. A chart defined on $\mathcal{M}$ in p is the pair (ϕ, U) where :*

$$\phi\colon U(p) \mapsto \tilde{U}$$

is a homeomorphism between $U(p)$ an open set in $\mathcal{M}$ around pand $\tilde{U}$ an open set of $\mathbb{R}^d$. A chart $(\phi, U(p))$ represents a local coordinate map of the point p and we can write : $\phi(p) = (p_1, p_2, .., p_d)$.

Definition 2.4 (Smooth charts) *Let $(\phi_1, U(p_2))$ and $(\phi_2, U(p_1))$ be two charts. If $\phi_1(U(p_1) \cap U(p_2))$ and $phi_2(U(p_1) \cap U(p_2))$ are open and the map $\phi_21 = phi_2 \circ \phi_1^{-1}$ is a diffeomorphism of order r (the map phi_21 and its inverse are r-differentiable, which differentiable r times) then equivalently r-differentiable compatible.*

Two charts are smoothly compatible if they are they are infinitely-differentiable compatible.

Definition 2.5 (Atlas and Smooth Atlas) *An atlas A defined on $\mathcal{M}$ is a collection of charts defined $\mathcal{M}$ that cover $\mathcal{M}$ with countably many open subsets $U(p)$:*

$$A = \{(\phi, U(p) \text{ such that} \bigcup U(p) = \mathcal{M}\}$$

The atlas is maximal smooth when every of its two charts are smoothly compatible.

The above definitions introduce the definition of a smooth manifold.

Definition 2.6 (Smooth Manifold) *A smooth manifold $\mathcal{M}$is the pair $(\mathcal{M}, \mathcal{A})$ where $\mathcal{M}$ is a set and $\mathcal{A}$ is a maximal smooth atlas on M. When $\mathcal{M}$ has dimension d then $\mathcal{M}$ is an d-dimensional smooth manifold.*

Now, that the structure of a manifold is defined, we introduce the notion of transformations on manifolds. In fact, since in manifold learning we are looking for a transformation of the manifold, it is important to understand how we can map a manifold to another and how to characterize such transformations.

Definition 2.7 (Maps between Manifolds) *A map ϕ between two smooth manifolds $\mathcal{M}$ and $\mathcal{N}$ is defined as $\phi\colon \mathcal{M} \mapsto \mathcal{N}$.*

Definition 2.8 (Smooth real-valued function on a manifold) *Let $\psi\colon \mathcal{M} \mapsto \mathbb{R}$. If for any point p in $\mathcal{M}$, the transition map $\psi \circ \phi^{-1} : \phi(U) \mapsto \mathbb{R}$ is smooth with $(\phi, U(p))$ the smooth chart that contains p.*

We note $C^{\infty}(\mathbb{R})$ the subspace of all such functions.

Definition 2.9 (Derivation) *A derivation at a point $p \in \mathcal{M}$ is any linear map $D\colon C^{\infty}(\mathbb{R}) \mapsto \mathbb{R}$ that satisfies the product rule.*

A derivation at a point p can be seen as a tangent vector at p to the manifold $\mathcal{M}$ with the idea of tangent as we know it in Euclidean space.

Definition 2.10 (Tangent Space) *The tangent space $\mathcal{T}_p\mathcal{M}$ is the set of all derivations in point p.*

The tangent space can be seen as the tangent of a curve in 1-dimension. Similarly to the line going through a 1-d curve, and tangent to it at a point p, is a linear approximation of the curve at that point, the tangent space $\mathcal{T}_p\mathcal{M}$ is a linear approximation of the manifold $\mathcal{M}$ at that point p. It is an affine translation of a Euclidean space to p tangent to $\mathcal{M}$.

We have introduced the idea of locally approximating linearly a manifolds. This notion is important as it gives a natural local coordinate system for the tangent spaces that they inherit from $\mathbb{R}^d$ when the manifold is d-dimensional.

Thus to characterize a mapping between two manifolds, we can characterize it on the tangent spaces, as it paramterizes the transformation on the basis. Therefore, we define next, the notion of a differential of a map on a manifold.

Definition 2.11 (Differential of a smooth Map) *The differential $d\phi_p$ of a smooth map $\phi\colon \mathcal{M} \mapsto \mathcal{N}$ is defined on Euclidean spaces to represent the linear map between the tangent spaces of the manifolds.*

$$d\phi_p\colon \mathcal{T}_p\mathcal{M} \to \mathcal{T}_{f(p)}\mathcal{N}$$

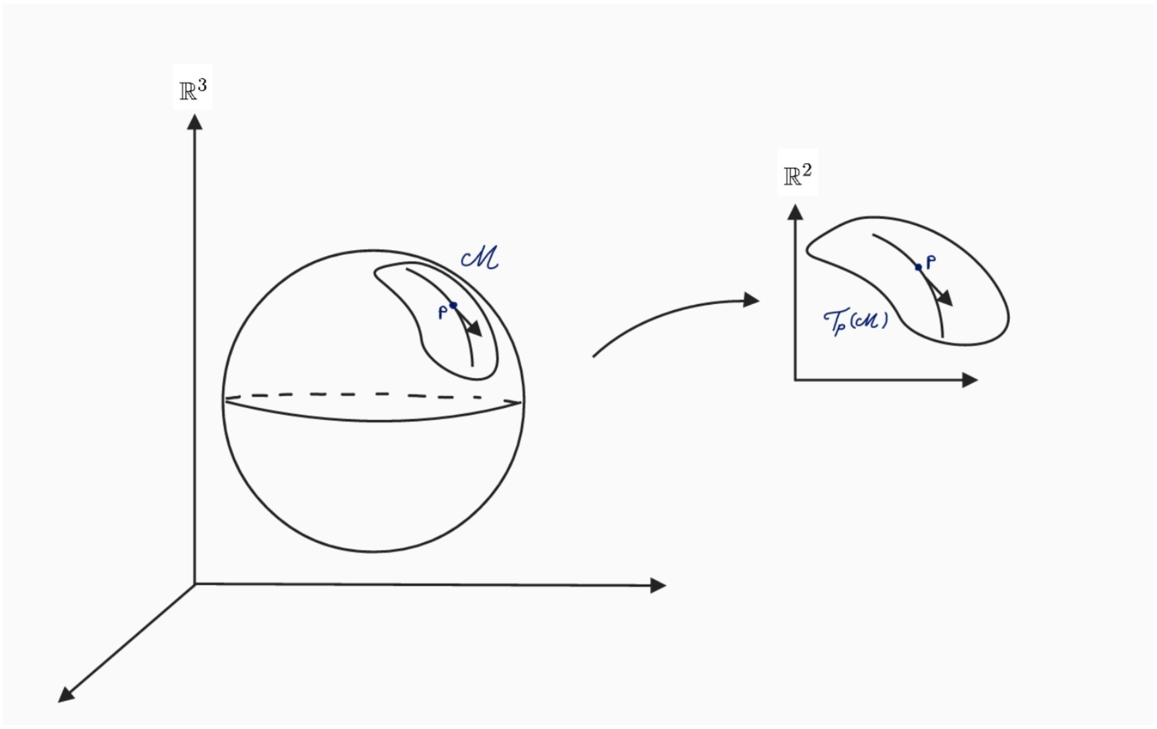


Figure 2.1.: Illustration of how we can map locally the manifold using the tangent space and the tangent to a 1-d curve is in the tangent space to the manifold around the curve. This illustration is adapted from figure 1.3 in [28]

Definition 2.12 (Jacobian at a point on the manifold) *The matrix notation of the differential $d\phi_p$ is $J_\phi(p)$ the jacobian of ϕ at point p expressed on the inherited local Euclidean coordinates.*

$$J_\phi(p) \in \mathbb{R}^{s,d},$$

where $\mathcal{M} \subset \mathbb{R}^d$ and $\mathcal{N} \subset \mathbb{R}^s$.

Definition 2.13 (Pullback and Pushforward) *Let ϕ be a smooth map $\phi\colon \mathcal{M} \mapsto \mathcal{N}$. Then its differential $d\phi$ is called pushforward. If ϕ is an isomorphism then $d\phi^{-1}$ is called pullback.*

Definition 2.14 (Diffeomorphism between manifolds) *If ϕ is an isomorphism then its differential $d\phi_p$ at point p is an isomorphism.*
We can link its differential to the differential of its inverse by

$$d\phi^{-1}_{\phi(p)} = d\phi_p^{-1}.$$

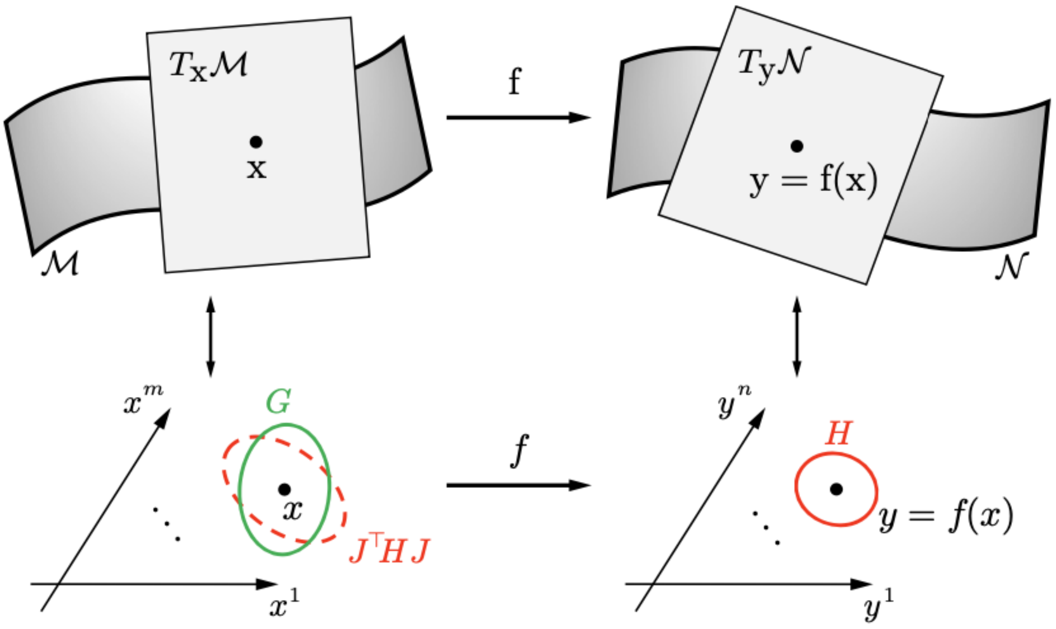


Figure 2.2.: Illustration of tangent spaces and Riemannian metrics for a mapping $f : \mathcal{M} \to \mathcal{N}$, reprinted from [14]. The matrices G and H correspond respectively to the expression of the metric on $\mathcal{M}$ at x in the local coordinate system of the tangent space $\mathcal{T}_x\mathcal{M}$ and the expression of the metric on $\mathcal{N}$ at $f(x)$ in the local coordinate system of the tangent space $\mathcal{T}_{f(x)}\mathcal{N}$. J is the jacobian of f in x. This illustration is essential in picturing the local transformation of the metrics and how accessing such deformations can help find f.

2.1.2. Isometry on Manifolds

The purpose of our thesis is to characterize a transformation between manifolds that preserves distances on the manifolds which would impose that the transformation is an isometry.
It is thus important to first introduce the concept of distance on a manifold, the definition of an isometry in general and how we would approach it on manifolds. In fact, on Euclidean spaces the distances are the Euclidean distances that stem from the Euclidean scalar product. A distance between two points in a Euclidean space is the length (norm that stems from the usual scalar product) of the line that connects both points.

However, manifolds can be curved and we would need first the equivalent of a path and the equivalent of a measure of the length of such a distance.

In the following, we define terms used throughout this thesis to call such curved shortest paths and their lengths on the manifold.

Definition 2.15 (Geodesic) *A geodesic is the shortest path between two points on the manifold.*

A manifold can have many geodesics.

We introduce a general definition of a metric that can vary from point to point and can generalize the notion of metric to non-Euclidean spaces.

Definition 2.16 (Riemannian Metric) *Let $\mathcal{M}$ be a smooth manifold. We equip $\mathcal{M}$ at each point p, with an intrinsic positive-definite scalar product $<,>_p$ defined as:*

$$g_p =<,>_p \colon \mathcal{T}_p\mathcal{M} \times \mathcal{T}_p\mathcal{M} \mapsto \mathbb{R}.$$

This defines at each point a metric that locally coincides with the Euclidean scalar product as we know it in the Euclidean space. Thus locally :

$$\|x_1 - x_2\| = g_p(x_1 - x_2, x_1 - x_2) \; \forall x_1, x_2 \in \mathcal{T}_p\mathcal{M}.$$

Since g_p is defined as a linear map on the tangent space at p, it can be written as a matrix G_p such that : $< x_1, x_2 >_p = x_1^T G x_2$, where x_1 and x_2 are expressed in the local coordinates of $\mathcal{T}_p\mathcal{M}$such that $G_{i,j} =< g_p^{(i)}, g_p^{(j)} >$ where i, j index the basis of local coordinates of $\mathcal{T}_p\mathcal{M}$ inherited from the local Euclidean space. Since G_p represents a scalar product, it is positive definite.

Figure 2.2 illustrates the notions of tangent spaces as well as Riemannian metrics on two different Riemannian manifolds and how we can relate both by a map f.

Definition 2.17 (Riemannian Manifold) *A Riemannian Manifold is the pair $(\mathcal{M}, g)$ where $\mathcal{M}$ is a smooth manifold and g is defined as the Riemannian metric on $\mathcal{M}$ at each point $p \in \mathcal{M}$.*

Equipped with a metric that will allow the measure of distances, we specify the notion of isometry. First, we define isometry as a general term. The notion of "isometrically" embedding a manifold into another space is different than the notion of an isometry.

Definition 2.18 (Isometry in general) *Let $d_{\mathcal{M}}$ an arbitrary distance measure on a smooth manifold $\mathcal{M}$. A map $\phi \colon \mathcal{M} \to \mathcal{N}$ both of dimension d, is an isometry iff it is a bijective map such that*

$$d_{\mathcal{M}}(p_1, p_2) = d_{\mathcal{N}}(\phi(p_1), \phi(p_2)) \qquad \forall p_1, p_2 \in \mathcal{M}.$$

The definition of differentials defines equivalently the notion of isometry when ϕ is differentiable. Then if ϕ is a diffeomorphism then its differential $d\phi_p$ is an orthogonal isomorphism for all $p \in \mathcal{M}$.

When the smooth map ϕ is not a bijection, we still can extend the notion of isometry to local isometry by defining it directly on the tangent space.

Definition 2.19 (Local Isometry) *A map $\phi \colon \mathcal{M} \mapsto \mathcal{N}$ is a local isometry iff its differential in p is an orthogonal linear isomorphism in p.*

Let $J_\phi(p)$ be the jacobian of ϕ_p at p, this is equivalent to saying that J_ϕ is invertible and :

$$J_\phi(p)J_\phi(p)^T = I_d \qquad \forall p \in \mathcal{M}.$$

Definition 2.20 (Riemannian Isometry) *Let $(\mathcal{M}, g)$ and $(\mathcal{N}, h)$ be Riemannian manifolds, let $\phi \colon \mathcal{M} \mapsto \mathcal{N}$ a smooth map. ϕ is an isometry iff ϕ is a diffeomorphism that preserves the intrinsic Riemannian metrics on each manifold.*

$$g(x_1, x_2)_p = h(df_p(x_1), df_p(x_2))_{f(p)} \forall x_1, x_2 \in \mathcal{T}_p\mathcal{M},\ \forall p \in \mathcal{M}. \tag{2.1}$$

Equivalenty (ϕ is an isomorphism), ϕ is an isometry iff ϕ^{-1} is a diffeomorphism such that :

$$h(y_1, y_2)_f(p) = g(df^{-1}_{f(p)}(y_1), df^{-1}_{f(p)}(y_2))_{f(p)} \forall y_1, y_2 \in \mathcal{T}_f(p)\mathcal{N},\ \forall p \in \mathcal{M}. \tag{2.2}$$

Definition 2.21 (Local Riemannian Isometry) *Let $(\mathcal{M}, g)$ and $(\mathcal{N}, h)$ manifolds, $f \colon \mathcal{M} \mapsto \mathcal{N}$ a smooth map. f is an isometry iff $f_{|U_p} \colon U_p \mapsto U_{f(p)}$ is a diffeomorphism that preserves the intrinsic Riemannian metrics on a neighborhood U_p around p to a neighborhood on $\mathcal{M}$.*

The pure definition of isometry on a manifold as stated above can be restrictive. Even for a local isometry, we would need an isomorphism. Such a definition is impractical when we do not have all the information about the transformations and we want to infer them. In fact, isometrically embedding a d-dimensional manifold $\mathcal{M}$ is restricted to actually find an injective mappping that preserves the Riemannian metrics between the tangent spaces of the manifold $\mathcal{M}$ and $\mathbb{R}^s$ with $d \leq s$.
Nash's theorem [21][22] enunciates the existence of such an injection. We enunciate it without giving the details of the range in which s lies as it is not of interest to us in this thesis.

Theorem 2.22 (Nash's theorem) *Let N be a d-dimensional Riemannian manifold r-differentiable, such that $3 \leq r \leq \infty$. Then there exists $s = O(d^3)$ such that there exists $\phi \colon \mathcal{N} \to \mathbb{R}^s$*

$$< x_1, x_2 >=< d\phi_p(x_1), d\phi_p(x_2) > \ \forall x_1, x_2 \in \mathcal{T}_p\mathcal{N} \forall p \in \mathcal{N}.$$

In the following, we suppose the manifold hypothesis. That is, given a set of data points

$$Y = y_i : i \in [1, N]\ for N \in \mathbb{N}, y_i \in \mathbb{R}^m,$$

the points in Y lie in a manifold $\mathcal{M}$ of lower dimension d though embedded in $\mathbb{R}^D$ with $d \leq D$.
In general, trying to learn $\mathcal{M}$ boils down to modeling the problem as follows : We suppose the existence of a diffeomorphism

$$\phi \colon \mathcal{M} \mapsto \mathbb{R}^d.$$

The main idea of manifold learning, though approached differently throughout literature involves finding an approximate to ϕ^{-1}. The objective is then to uncover ϕ while preserving a certain structure of the embedded manifold.

In the following section, we present a no-exhaustive history of manifold learning techniques.

2.2. Classic Manifold Learning methods

We define terms that will be used throughout this section.

Definition 2.23 (Graph) *A graph G is defined as a pair (V, E) where $V = v_{i i \in C}$ where C is finite subset of . $E = (v_i, v_j)_{ij \in C, ij}$ is a set of paired points in V. Elements of E are edges and elements of V are vertices. A graph is connected if there exists a path between any two points in the graph.*

Definition 2.24 (Adjacency and Degree Matrix of a graph)

Adjacency matrix A is defined on graph $G = (V, E)$ as $A = (a_{ij})_{ij \in C}$ with $a_{ij} = 1$ if $e_{ij}$is an edge in E and 0 otherwise.

The degree matrix D is the diagnoal matrix defined on graph $G = (V, E)$ as $D = (a_{ij})_{ij \in C}$ with d_{ii} the number of nodes that are connected to i.

Definition 2.25 (Graph Laplacian) *Let matrix L be the laplacian matrix of the graph G.*

$$L = D - A$$

where A is the adjacency matrix of the graph and D is the degree matrix of the graph.

Definition 2.26 (Hessian Matrix) *Let $f\colon \mathbb{R}^n \mapsto \mathbb{R}$ a real-valued function. The Hessian matrix of f is defined as :*

$$H_{ij} = (\frac{\partial^2 f}{\partial x_i \partial x_j})$$

Definition 2.27 (Hessian on a manifold) *Let $f\colon \mathcal{M} \mapsto \mathbb{R}$. The Hessian of f in a point p of $\mathcal{M}$ is defined on $T_p\mathcal{M}$ as :*

$$h_f\colon T_p\mathcal{M} \mapsto \mathbb{R}^d$$

is written can be written as a matrix and identified as the usual Hessian matrix on the Tangent Space at point 0 which is image of p on the tangent space.

2.2.1. Principal Component Analysis (PCA)

PCA [24] is introduced as a solution for finding the most "be-fitting" line to regress N data points in $\mathbb{R}$. A principal component is the principal axis of the residual ellipsoïd which centroïd is the points' mean. For higher dimensional points, PCA projects them in the direction of higher variances to find their lower dimensional representation. Those directions are calculated by an eigen-decomposition of the covariance matrix of the centered data points and each direction is a principal component. [19] introduces a generalization of PCA using kernels.

2.2.2. Kernel and eigenmap methods

Laplacian Eigenmaps (LE) [4] The Laplace Beltrami-operator is defined as the generalization of the Laplacian on manifolds. Recall that the Laplacian of a function on a point translates how much the evaluation of the function at that points differs from the neighboring points. Intuitively, if we perceive the points lying on the manifold as a graph, we can apply the Graph Laplacian to learn the underlying manifold. In fact, if we can map the manifold in a lower space such that the graph Laplacian is preserved, then we have found a lower d-dimensional embedding that preserve local neighborhood structure. This entails that we must first define neighborhood and construct a graph of the manifold.

Local Linear Embedding (LLE) [27] The idea of LLE is reflected simply in its name : model the manifold as an aggregation of small linear affine spaces at each of its points. The affine spaces are not perfectly tangent to the manifold. The intuition is that at each point, the affine space is a best befitting linear space that describes the neighborhood of that point. The manifold is, as introduced in LLE, is thus perceived as a collection of "patches" defined at each point for the best local approximation.

The neighborhood of a point is defined as a ball of fixed radius r.

The affine spaces are then found by calculating $\tilde{w}_{ij}$ as the solution of a least square constrained optimization problem. We thus find in higher dimensions, the weights that represent locally the high dimensional points.

Once the weights are analytically calculated they are plugged into the loss function

$$loss = \Sigma_i \left\| y_i - f(y_i) \right\|^2 ,$$

where

$$f(y) = \Sigma_{ij} \tilde{w}_{ij} * (y_i - y_j).$$

The minimization of the loss has an analytical closed form and yields a lower representation of the data while preserving the local high dimensional neighborhood linearly weighted structure. This solves the manifold learning problem as stated above.

Hessian Eigenmaps (HE) HE learns the inverse of the transformation ϕ by supposing that the manifold is smooth and that ϕ in a neighborhood of the manifold is smooth isometric, thus smooth and locally isometric. It is based on characterizing the local Hessians at the points on the manifold of a real-value function f that describes the local neighborhoods. This function corresponds to the graph adjacency matrix. HE proceeds in a manner close to LE, while replacing the Laplacian with the Hessian matrix and is based on the important following corollary defined in [10]

Corollary 2.28 (Local isometric function on smooth manifolds and hessian Matrix) *Hypothesis: ϕ, as defined in the manifold problem, is locally isometric. We can find the transformed coordinates of $\tilde{y}$ in the d-lower dimensional embedding by finding a suitable basis for the null space of the Hessian Matrix.*

The algorithm for estimating the Hessian matrix at each point and characterizing the null space of H has the same steps as LE.

Isotropic Diffusion Maps (DM) For DM [8] also perceives the manifold as a graph. A random walk is performed iteratively through the manifold. The probability transition matrix P is initialized as the adjacency matrix on the graph G using a diffusion kernel on the neighborhood as a similarity measure. The ensuing transition matrix P provides a better characterization of the manifold structure at each new step t when the random walk is iterated again. The transformation ϕ is then found by an eigendecomposition of P.

Anisotropic Diffusion Maps (ADM) We remark that both LE and HE builds a similarity distance on the manifold that approximates the Euclidean distance in the higher dimensional space. For example, in LE, this "low"-dimensional measure is the heat kernel K

$$K(y_1, y_2) = \exp\left(-\frac{\|y_1 - y_2\|^2}{2\sigma^2}\right),$$

when y_1 and y_2 are two high dimensional points. The transformation that LE and HE try to uncover is rooted in the high dimensional distance even though they approximate, as mentioned, a good representation of the local neighborhood around the points. This can still be considered a weak approximation of the embedding's real, local distance measure. AMD uses the same idea as DM but with a kernel approximating the Euclidean distance between the inaccessible embedded x_1 and x_2 in a lower dimension. The kernel is defined as

$$K(y_1, y_2) = \exp\left(-\frac{(\left\|d\phi_{y_1}^{-1}(y_1 - y_2)\right\|^2 + \left\|d\phi_{y_2}^{-1}(y_1 - y_2)\right\|^2)}{4\epsilon}\right).$$

$d\phi^{-1}$ at a point y is not calculated directly but approximated using an eigendecomposition of the covariance matrix of neighbors sampled around y.

2.2.3. Multi-Dimensional Scaling (MDS)

MDS [12], maps the N data points

$$y_i : i \in [1, N]\ for\ N \in \mathbb{N}, y_i \in \mathbb{R}^D,$$

for which pairwise distances $d_ij = d_{\mathcal{M}}(y_i, y_j)$ are provided $(D = (d_{ij}))$, to the wanted lower d-dimensional space while keeping the distances between the projections as close as possible to the original points. The map provides a solution to the manifold problem, as we stated above.

MDS envelopes various techniques with the same stemming idea but differ in the type of transformation functions, the distances, and the scaling factor used.

The general idea is to minimize the following function loss (referred to as stress or strain depending on the nature of the mapping)

$$loss = \frac{\Sigma_{i,j}(f(d\mathcal{M}_{ij}) - d_{ij})^2}{scaling}.$$

Intuitively, the idea of stress is introduced to model the problem as a graph structure with connected nodes weighted by distances, and we want to transform the nodes so that we impose minimal stress on the structure. Classical MDS uses linear transformations and euclidian distance. Other variants use isotropic (constant) or anisotropic scaling (adapted to the data). A strength of MDS is that it can be used as an intermediate step when a dissimilarity or adjacency matrix is provided. MDS was in [7] used MDS for data representation in hypermedia.

Closer to our focus in this thesis, we present an overview of isometric embedding techniques in the literature.

2.3. Isometric Manifold Learning methods

The methods in subsections 2.2.2 extract the embedding to preserve a local or global structure without explicitly imposing geometrical property conservation. The interactions between neighborhoods in the presented non-linear methods were so far presented as graphs. They minimize a particular loss to keep an inherent characteristic of the graph (eigenvalues of functions defined on the graph or the stress on the structure) not the geometry on the manifold as it would be mathematically defined (uncover d-dimensional embeddings that preserve the distance measures on the real d-dimensional manifold).

A more constraining way to uncover the manifold transformation is to suppose that ϕ is isometric or locally isometric. This explicit constraint is fruitful in better characterizing the manifold, and learned low-dimensional points close to the manifold are retrieved as close as that on the actual low-dimensional manifold.

In isometric methods, the isometry condition is explicitly formulated between the distances on the manifold and the distances of the retrieved manifold

$$d_{\mathcal{M}}(x_1, x_2) = d_{\tilde{\mathcal{M}}}(\tilde{x}_1, \tilde{x}_2),$$

where $d_{\mathcal{M}}(x_1, x_2)$ is a particular distance measure on $\mathcal{M}$ between x_1 and x_2.

2.3.1. Generalized MDS

Introduced by Bronstein in 2006, Generalized MDS [6] generalizes MDS to use non-Euclidean distances on manifolds and offers an optimized framework that finds the loss by numerically approximating (no analytical expression used) the distorted metric on the embedding. It was introduced as a framework for matching isometry-invariant shapes, which matches shapes that were transformed in a manner that did not change intrinsic distances on the manifold. It does not produce a perfect isometry since, in general, it is too rigid to impose a perfect isometry. However, it does produce a ”decent” similarity measure that identifies shapes ”probes” coming from the same isometrically distorted ”model” shape.

2.3.2. Isomap

Isomap assumes, additionally to the isometrical embedding, that the manifold is also a convex region, which means that for every two points on the manifold, there is a straight path that joins intermediate points in a convex way. In Isomap, the manifold is viewed as a weighted graph, and the local distances calculated between points are the length of geodesics on the graph. For a graph, a geodesic is the shortest path between two vertices.

Isomap finds a new adjacency matrix based on the geodesic distance as a similarity measure. Recall that MDS can be used as a step to approximate the Euclidean distance in a d-dimensional space based on a similarity matrix between objects. Isomap uses this property to find the new embeddings that guarantee that the geodesic distances on the manifold are the same as the distances calculated d-dimensional Euclidean space.

2.3.3. Preserving the metric on the Manifold

In [25], the authors introduce a geometric framework that approximates the actual geometric distance locally on the manifold. The manifold is considered Riemannian.

The authors give a mathematical guarantee that the proposed method extracts the Riemannian metric on the embedding. This framework can be used to correct any embedding that was not retrieved isometrically with a mathematical guarantee. We use the results of this paper in our thesis to evaluate the quality of the isometry retrieved in both cases we study.

Megaman The authors propose a package Megaman available in Python that codes the following results that will be used in this thesis:

1. Computing adjacency matrices using FLANN (Fast Library for Approximate Nearest Neighbors)[20]
2. Computing the Riemannian metric and showing the distorted distances between the real embedding and the learned embedding.

In [18], the same authors of [25] construct the algorithm that uses the geometric framework to construct the isometric embeddings. The manifold is mapped to a graph with weights are calculated using a kernel function K, builds Laplacian of the graph and upon it builds the Riemannian metric matrix H. It imposes the isometry by calculating a loss function that makes H converge to I, the identity matrix.

[14] builds a framework fro isometric embeddings close to [25] but that generalizes also to non-Euclidean data.

2.4. Auto-Encoders for manifold Learning

2.4.1. Concept of Machine Learning

Machine Learning [16] represents a family of methods that solves a set of problems without having the rules of the solution coded explicitly. The rules are instead inferred by minimizing a specific loss. It encompasses two large sets of problems: Supervised and Unsupervised.

On the one hand, the task in Supervised-Learning is to find a function that, given N points, maps them to their labels. The idea is to use the inherent data structure to learn the patterns that differentiate or describe best the correct labeling of the data points.

On the other hand, the task in Unsupervised-Learning is to find, within the data structure, the similarity between data points. No labels are given to orient the learning towards a specific solution. We can approach Manifold Learning as a machine learning problem since we want to find an embedding that captures the structure of the data. In this thesis, Auto-Encoders, a specific type of Neural-Networks, were used to solve the problem of learning an isometric embedding.

Neural Networks (NN) Neural Networks [5] can be represented as a global function f that we want to learn to solve a particular problem.

They are stacked layers of connected nodes, which is more of a representation than the definition of an actual graph as illustrated in figure 2.3.

Each node represents an activation function, and the connections between the nodes carry the weights of the inputs that will fire (activate) or not the node. Activation functions that introduce non-linearity in the NN's overall learned function, allowing more flexibility and complexity for the decision boundaries. A vanilla loss function determines the deviation of the neural network's outputs $f(x)$ to the wanted outputs y. In general, it can be any function of the network's parameters and the input data.

To solve a specific task, one needs finding the correct architecture of NN and the right parametrization of the connections between the nodes which optimizes the loss minimization.

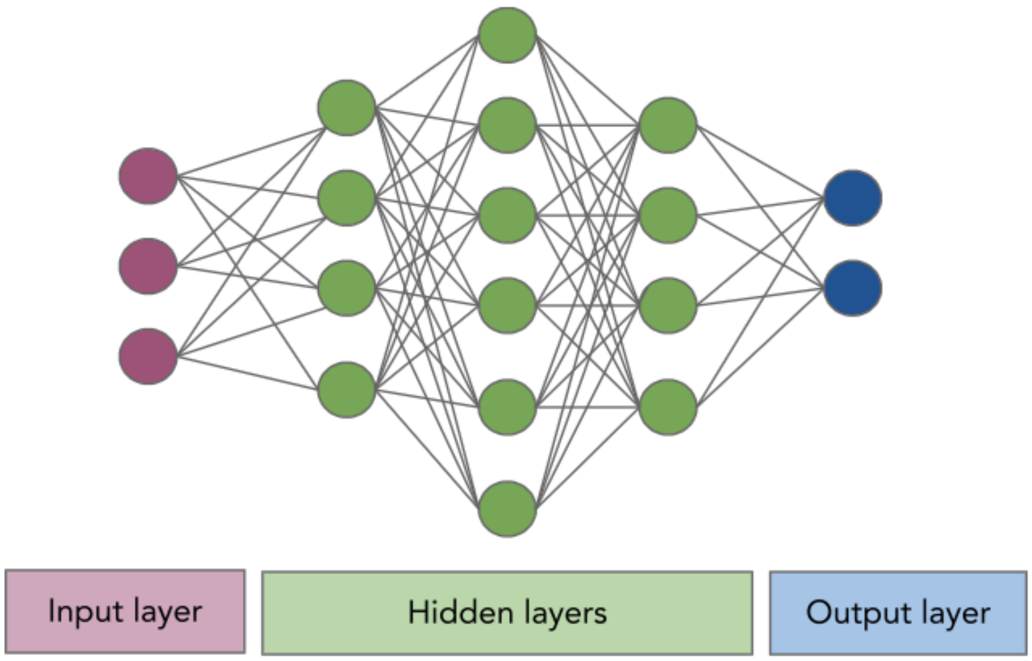


Figure 2.3.: Example of a neural network with an input in 3-d, an output in 2-d and 3 hidden layers with 4,6 and 4 neurons orderly.

2.4.2. Vanilla Auto-Encoders

An Auto-encoder [11] is a neural network that learns the input data fed to it. The global function of the Auto-Encoder would be the identity since the purpose is to learn the inputs and would not give further information on the data points. However, when a bottleneck is introduced in the architecture of the NN as well as other constraints to the architecture, it learns an expressive function that maps the inputs to a lower dimensional space and then to a higher dimensional space as pictured in Figure 2.4. This idea is the main purpose of an Auto-Encoder.

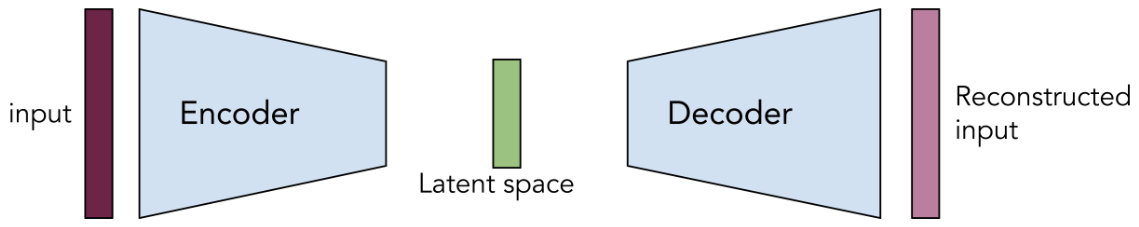


Figure 2.4.: Illustration of the auto-encoder structure. The latent space represents a compression of the input data. The auto-encoder learns a reconstruction of the input.

Let

$$Y = y_i : i \in [1, N]\ for N \in \mathbb{N}, y_i \in \mathbb{R}^D,$$

the N input points to the auto-encoder.

We note the functions of the auto-encoder as follows

- ψ is the general function the auto-encoder tries to learn

$$\begin{aligned} \psi \colon \mathbb{R}^D &\to \mathbb{R}^D \\ y &\mapsto \tilde{y}. \end{aligned}$$

- ψ_e is the function of the encoder that maps the input to the lower dimensional Euclidean space $d < D$

$$\begin{aligned} \psi_e \colon \mathbb{R}^D &\to \mathbb{R}^d \\ y &\mapsto \tilde{x}. \end{aligned}$$

- ψ_d is the function of the decoder that remaps the lower dimensional points to the higher dimensional input space

$$\begin{aligned} \psi_d \colon \mathbb{R}^d &\to \mathbb{R}^D \\ \tilde{x} &\mapsto \tilde{y}. \end{aligned}$$

Thus, the auto-encoder is the composition of the encoding and decoding functions

$$\psi = \psi_d \circ \psi_e.$$

Using $d \leq D$ introduces the network bottleneck that permits uncovering a latent representation of the input data. The embedding is learned by minimizing a loss function

$$\mathcal{L}_{rec}(\psi) = \|\psi(y) - y)\|^2 .$$

2.4.3. Regularized auto-encoders

Regularization methods are used to impose the extraction of a certain data structure which enforces a certain characteristic of the learned embedding function ψ such as sparsity [23] or robustness to noise [29].

In this thesis, isometry can be perceived as a regularization technique [13]. The loss function is modified to account for the regularization term R as follows:

$$\mathcal{L}(\psi) = \mathcal{L}_{rec}(\psi) + R(\psi)$$

Many authors during the last two years (LOCA and I-AE were published in 2020), have used the papers we are interested in, for other applications. For instance, [3] applies the strategy of isometric auto-encoders in I-AE to GANs where the input of the Generator is the learned isometric embedding from the I-AE encoder.

Part II.

Isometric Embedding of Manifolds with Auto-Encoder Networks

3. Methodology

As stated in section 1, this thesis aims to compare two approaches to manifold learning using auto-encoders. We frame our approach to this problem by referring to the main challenging points the algorithms introduced in section 2 focused on when introducing new approaches. We retain that the following points are the recurrent points we look for when studying manifold learning method :

1. The capability of learning the embedding recovered by the algorithm
2. The power to generalize over more than the observed points and extend the embedding in the non-observed domain
3. The robustness to noise on the manifold
4. The robustness to small data samples
5. The computation comlexity of the model

Based on the following points, we establish a line of comparison between both approaches. Furthermore, in comparing two objects, one must generally establish a common framework where both objects can be comparable and establish the points of alignment and the divergence between them. Thus, we establish experiments that compare the methods to state their similarities and differences for sure of the abovementioned points. Therefore, we formulate the following framework for our comparison

1. **Idea and Mathematical approaches** We explain separately the ideas and methodologies developed in both papers. We state the different hypotheses of the models and establish a comparing conclusion on the mathematical guarantees both methods offer.
2. **Use Cases** Based on the mathematical guarantees, both papers tackle the problem differently. Therefore, each method must be used with its hypotheses for a fair comparison. First, we state the different use cases for which each paper was introduced. Then, we implement them for each method in its spectrum of hypotheses. We also adapt them so we can compare them when the guarantees of one are unavailable for the other.
3. **Implementation** We briefly discuss the implementation of both methods. Although LOCA is an open source project [1], I-AE is not. The source code was not available,

and we proceeded to implement I-AE. We will discuss the implementation procedure's details briefly. Furthermore, since use cases differ, the input is modified to suit how [26] implements LOCA.

4. **Isometry comparison** The methods introduce further losses that approximate different entities. For a fair comparison, we use the Euclidean distances on the lower dimensional manifold when the distribution is available. Otherwise, we use a Riemannian metric retrieval method. We, thus, compare both methods congruently on this common ground that [25] introduces within a solid mathematical background.

It is worth noting that the comparison in this context is a difficult task since I-AE and LOCA define completely different isometries. In the next section, we detail each of the steps mentioned in our methodology points.

4. Comparison Framework

4.1. Mathematical problem setting

The purpose of LOCA and I-AE is to find the inverse of supposedly isometric transformation from an observed embedding to a latent embedding. As stated in section 2, by isometric, we mean "isometrically" embedding the lower dimensional manifold in a lower dimensional space.

Let $(\mathcal{N}, h)$ be a d-dimensional Riemannian manifold that we refer to as embedding, $(\mathcal{M}, g)$ a D-dimensional Riemannian manifold that we refer to as the observed manifold, such that $d \leq D$. ($h$and g are the Riemannian metrics defined on respectively $\mathcal{N}$ and $\mathcal{M}$) Let $y_i \in \mathbb{R}^D$, be N observed samples near the observed manifold $\mathcal{M}$.

We suppose there is a map ρ that transformed the embedding $\mathcal{N}$ to the observed embedding $\mathcal{M}$:

To find perfectly ρ^{-1} when the transformation is not apparent, we would still need information about ρ or the actual embedding itself, which we do not have. Furthermore, imposing that ρ is an isometry is mathematically incorrect. The definition of isometry needs to be an isomorphism between spaces with the same dimension, and we do not always have $d = D$.

Thus, we find a strategy to approach ρ^{-1} when ρ is invertible and $\rho^{-1}_{|\mathcal{M}}$ when ρ is supposed to be smooth and injective.

For a unified framework between LOCA and I-AE, we decompose the transformation ρ as :

$$\rho = \psi_d \circ \phi$$

Where :

$$\phi \colon \mathcal{N} \to \mathbb{R}^s \tag{4.1}$$
$$x \mapsto \tilde{x} \tag{4.2}$$

$$\psi_d \colon \mathbb{R}^s \to \mathbb{R}^D \tag{4.3}$$
$$\tilde{x} \mapsto y \tag{4.4}$$

For LOCA, ϕ is an isometric embedding from the d-dimensional $\mathcal{N}$ into $\mathbb{R}^s$ with $d \leq s$. For I-AE, ψ_d is the isometric embedding from $\mathcal{N}$ and $\mathbb{R}^D$ and ϕ could be seen as an isomorphism that is not necessarily defined between $\mathcal{N}$ and $\mathbb{R}^d$ ($d = s$); for no further confusion, for I-AE, we write $\rho = \psi_d$.
$\rho_{|\mathcal{M}}$ is supposed to be invertible. Thus, ψ_d must be invertible. For both LOCA and I-AE, ψ_e is chosen to be the pseudo-inverse of ψ_d.
For both LOCA and I-AE, although the functions that are imposed to be isometric embeddings are different, they both impose local Riemannian isometries on the tangent space of Riemannian manifolds.

After establishing a common ground for LOCA and I-AE, we explain the strategies taken to justify that the different isometries they propose are isometries and how the pseudo-inverse is imposed.

4.1.1. LOCA's approach

LOCA imposes ϕ such that for

$$d\phi_x \colon \mathcal{T}_x\mathcal{N} \to \mathbb{R}^s,$$

we have

$$g_x(x_1, x_2) =< d\phi_x(x_1), d\phi(x_2) >,$$

which can be written in matrix form as

$$J_\phi(x)^T H_{\phi(x)} J_\phi(x) = G_x \; \forall x \in \mathcal{N},$$

where $H_{\phi(x)}$ is the inherited metric on $\mathbb{R}^s$. Furthermore, the Riemannian metric is supposed to be constant over the manifold $\mathcal{N}$, thus

$$G_x = I_d \; \forall x \in \mathcal{N},$$

$$H_{\phi(x)} = I_s.$$

Thus, imposing the Riemannian isometry boils down to imposing

$$J_\phi(x)^T J_\phi(x) = I_d. \tag{4.5}$$

LOCA imposes this using a lemma that relates the covariance of samples on the transformed tangent space to the Jacobian on the original lower dimensional manifold.
LOCA has access to these samples using a burst sampling strategy which consists in repeated sampling on the manifold on a small time window.
In fact, each y_i is not observed once on the observed manifold, but each y_i is sampled M times. This translates into direct access to a distribution on the manifold.
LOCA characterizes the deformation of each burst as follows:

Before the deformation, on the unknown manifold $\mathcal{N}$ lie N non-deformed samples which we noted x_i such that :

$$x_{i,m} \sim \mathcal{N}(x_i, \sigma^2 I_d).$$

We do not ave access to this information however we can relate to it using the learned bursts. Then the samples that form a burst i we can approach are

$$\tilde{x}_{i,m} = \phi(x_{i,m}).$$

Figures 4.1 and 4.2 show a burst respectively before and after the transformation.

The intuition behind using bursts comes from the fact that we can parametrize the tangent space $\mathcal{T}_x\mathcal{N}$using the local $\mathbb{R}^s$ basis.
Recall the idea of PCA, where we can have an idea about the occurred transformation by exploiting the residual ellipsoid, which gives the axis on which we can project the data according to the directions of highest variance.
Here, the idea is the same, on the local euclidean space $\mathbb{R}^s$ passing through $\phi(x) = \tilde{x}$, the transformed residual ellipsoid by ϕ gives us a parametrization of the transformed manifold.
We can intuitively relate the variances on each transformed dimension of the samples to the differential in the transformed space. Thus, at each point $\tilde{x_i}$, the burst (which, as we said, is equivalent to a residual ellipsoid) can parametrize locally the transformation ϕ.

The authors of LOCA introduce this reasoning as a lemma under certain hypotheses

Lemma 4.1 (covariance of samples and differential on the tangent space) *Let ϕ be a 3-diffrentiable and injective map from $\mathbb{R}^d$ to $\mathbb{R}^s$ such that $d \leq s$. Let $\sigma \in \mathbb{R}_+^*$, and let X be a random variable such that $X \sim N(x, \sigma^2 I_d)$ with σ small enough that $J_\phi(x)$ is constant on the neighborhood $\mathcal{B}_(x, \sigma)$ and $\tilde{X} = \phi(X)$.Then*

$$J_\phi(x) J_\phi(x)^T = \frac{1}{\sigma^2} Cov(\tilde{X}) + \mathcal{O}(\sigma^2).$$

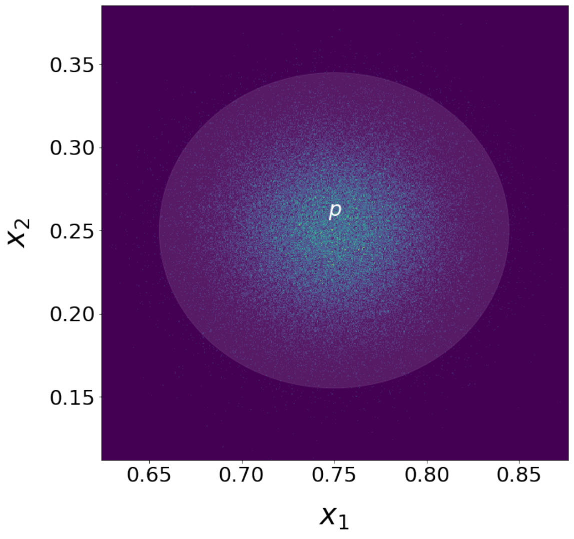


Figure 4.1.: Example of a burst B in R^2 that reflects the distribution on the embedding $\mathcal{M}$ before deformation. The points are sampled using $N((0.25, 0.75), 0.001 * I_2)$

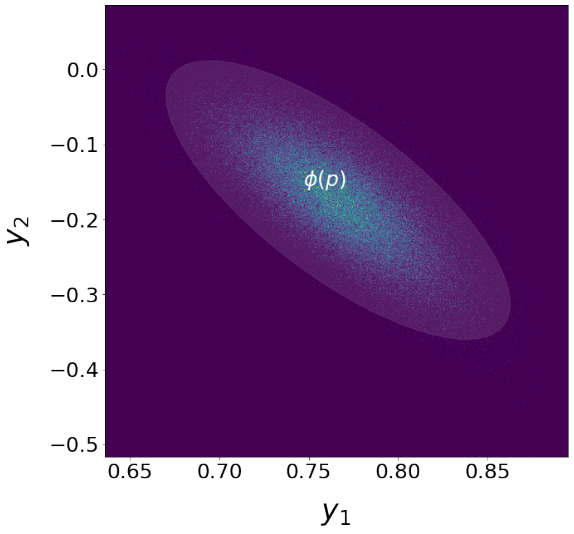


Figure 4.2.: The burst B transformed using a non-linear function ϕ. This shows the deformation of the burst when observed on the manifold.

As defined in equation 4.5, if ϕ is supposed to be an isomorphism then $s = d$. Then

$$J_\phi(x)^T J_\phi(x) = J_\phi(x) J_\phi(x)^T.$$

LOCA thus rewrites the local isometry condition as

$$\frac{1}{\sigma^2} Cov(\tilde{X}) = I_d \, \forall \tilde{x} \in \mathbb{R}^d. \tag{4.6}$$

Now, the objective is to find the mapping ψ_d defined by equation 4.3 that injects the low dimensional bursts $\tilde{x}_i, m$ to $\mathbb{R}^D$ as close to y_i, m as possible for every $\tilde{x}_i \in \mathbb{R}^d$.
This condition is written as :

$$\begin{aligned} \rho(x_{i,m}) = \psi_d(\tilde{x}_{i,m}) = \tilde{y}_{i,m}, \\ \rho^\dagger(y_{i,m}) = \psi_e(y_{i,m}), \end{aligned}$$

which means to impose

$$\forall y_{i,m} \in \mathcal{M}, \quad \psi_d \circ \psi_e(y_{i,m}) - y_{i,m} = 0. \tag{4.7}$$

This condition is easily enforced using auto-encoders by setting the reconstruction loss to

$$\mathcal{L}_{rec}(\theta_{\psi_e}, \theta_{\psi_d} | Y) = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \left\| \psi_d \circ \psi_e(y_{n,m}) - y_{n,m} \right\|^2. \tag{4.8}$$

To enforce the condition of isometry in equation4.6, LOCA sets a new loss that we note $\mathcal{L}_{white}$

$$\mathcal{L}_{white}(\theta_{\psi_e}, \theta_{\psi_d} | Y) = \frac{1}{N} \sum_{n=1}^{N} \left\| \frac{1}{\sigma^2} Cov(\psi_d(Y_i)) \right\|^2. \tag{4.9}$$

LOCA optimizes θ_{ψ_e} and θ_{ψ_d}, the parameters respectively of the encoder and decoder using the algorithm in [26].

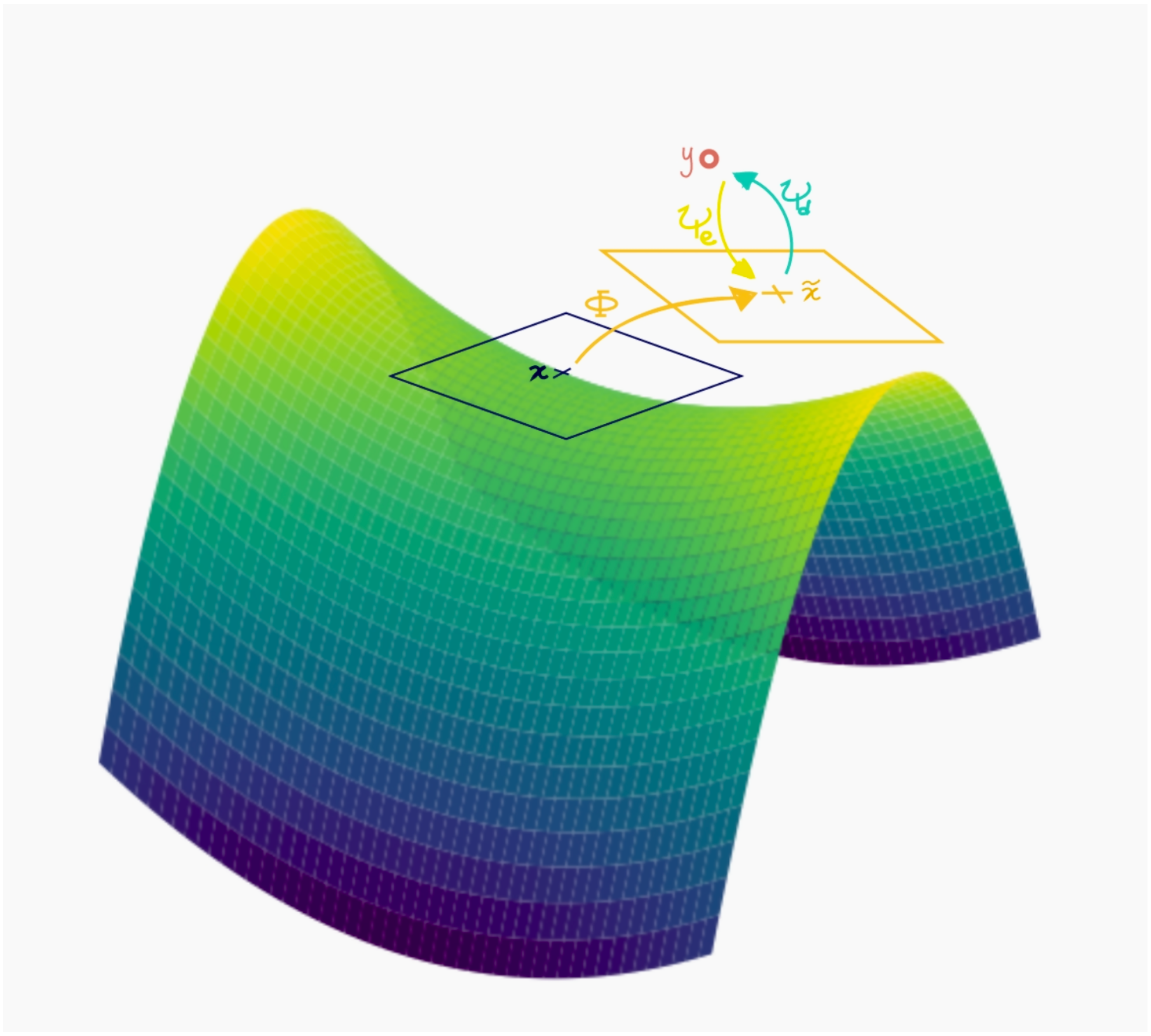


Figure 4.3.: Illustration of how LOCA enforces the isometry and learns the embedding. Using burst information, it links the actual embedding points x to the learned points $\tilde{x}$. It then, finds a map ψ_d and its pseudo-inverse ψ_e that maps y to $\tilde{x}$. We optimize the auto-encoder to uncover such transformations. ψ_e and ψ_d are respectively the encoder's and decoder's functions while ϕ represents the isometry as defined by LOCA.

4.1.2. I-AE's approach

Recall the definition of ρ in subsection 4.1. For I-AE, the isometric embedding is ψ_d. For notation purpose we denote ψ_d also as $\psi^{(d)}$.

The isometry condition for $\psi^{(d)}$ is written as

$$d\psi_x^{(d)} : \mathcal{T}_x\mathcal{N} \to \mathcal{T}_x\mathcal{M},$$

where $d\psi_x^{(d)}$ is the differential of ψ_d in x. We have

$$\forall x_1, x_2 \in \mathcal{T}_x\mathcal{N}, \quad g_x(x_1, x_2) =< d\psi_x^{(d)}(x_1), d\psi_x^{(d)}(x_2) >,$$

which can be written in matrix form as

$$J_{\psi^{(d)}}(x)^T H_{\psi^{(d)}(x)} J_{\psi^{(d)}}(x) = G_x \; \forall x \in \mathcal{N},$$

where $H_{\psi^{(d)}(x)}$ is the inherited metric on $\mathbb{R}^D$. Furthermore, the Riemannian metric is supposed to be constant over the manifold $\mathcal{N}$, thus

$$G_x = I_d \; \forall x \in \mathcal{N},$$

$$H_{\psi^{(d)}} = I_D.$$

This results in

$$J_{\psi^{(d)}}^T J_{\psi^{(d)}}(x) = I_d. \tag{4.10}$$

For LOCA, we supposed that $s = d$ and the injective map ϕ are locally an isomorphism. In this case, the isometry is between a d-dimensional and a D-dimensional space. Thus equation 4.10 would have been enough if we imposed also $d = D$. However, limiting the cases to only $d = D$ is not desirable as we are looking for a lower dimensional embedding where we can have $d < D$.

In other words, for LOCA when $s = d$, it was equivalent to define the isometry as preserving locally using the push-forward $d\phi$ or the pullback $d\phi^{-1}$, since

$$J_\phi(x)^T J_\phi(x) = J_{\phi^{-1}}(\phi(x))^T J_{\phi^{-1}}(\phi(x)) = I_d.$$

For I-AE, since ψ_d is not invertible, we also need to define the isometry in terms of the pullback.

However, the injective map ψ_d does not have a well-defined ψ_d^{-1}; thus, we define the pullback as the pseudo-inverse of ψ_d

$$\chi = \psi_d^\dagger. \tag{4.11}$$

Then ψ is an isometry if the equation 4.10 is verified, and the pseudo-inverse χ verifies the isometry condition defined by the pullback 2.2 as well

$$\forall x \in \mathcal{N},\ \forall y_1, y_2 \in \mathcal{T}_{\psi(x)}\mathcal{M}, \quad h_{\psi(x)}(y_1, y_2) = g_{\chi\circ\psi(x)}(d\chi_x(y_1), d\chi_x(y_2)),$$

which can be written in matrix form as :

$$\forall y \in \mathbb{R}^D,\ J_\chi(y)^T G_{\chi(y)} J_\chi(y) = H_y,$$

where H_y is the inherited metric on $\mathbb{R}^D$.

Furthermore, the Riemannian metric is supposed to be constant over the manifold $\mathcal{N}$, thus :

$$\forall x \in \mathcal{N},\ \ G_{\chi\circ\psi_d(x))} = I_d,$$

$$\forall x \in \mathcal{N},\ \ H_{\psi_d(x)} = I_D,$$

This results in

$$\forall y \in \mathbb{R}^D,\ J_\chi(y) J_\chi(y)^T = I_d. \tag{4.12}$$

To summarize, I-AE imposes an isometric embedding of $\mathcal{N}$ into $\mathbb{R}^D$ by imposing the three conditions 4.10, 4.12 and 4.11.

I-AE uses an auto-encoder like LOCA to find ψ_d and χ which is ψ_e. To impose 4.10 and 4.12, I-AE introduces respectively loss $\mathcal{L}(\theta_{\psi_d})$ and $\mathcal{L}(\theta_\chi)$).
To impose the condition 4.11, I-AE uses the reconstruction loss as defined in 2.

4.1.3. Comments on the mathematical approaches

We do not detail the expression of the losses as the idea of this mathematical framework was to explain more the way I-AE and LOCA imposed the isometry.
It is clear that LOCA has the advantage of introducing a local isomorphism that is isometric to the inaccessible manifold. This translates into a more powerful definition of isometry and a better preserving of the local geometry as it accesses information about the latent space that is not available in I-AE.

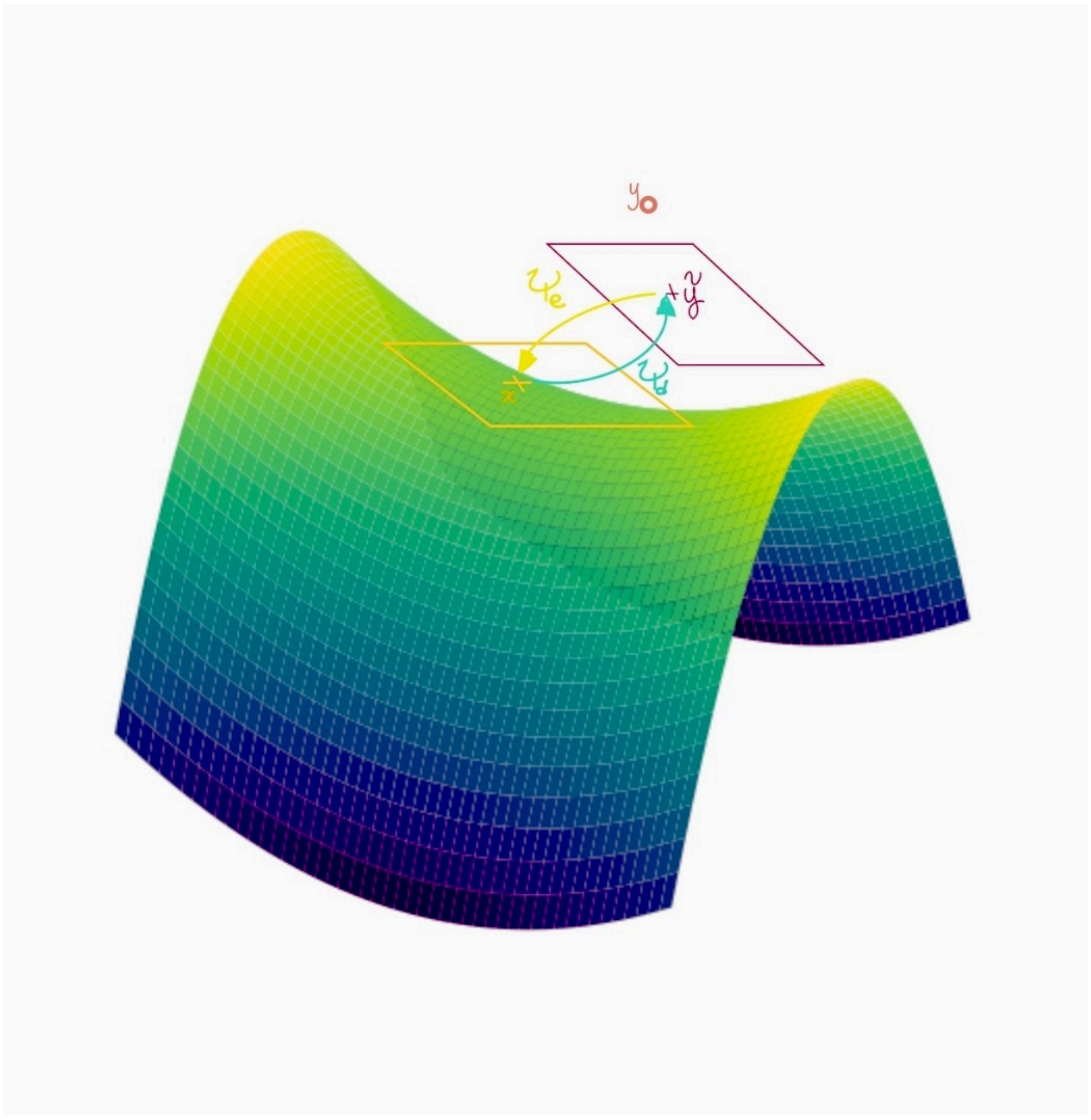


Figure 4.4.: Illustration of how I-AE enforces the isometry and learns the embedding. Minimizing the losses has the effect of projecting y on the tangent manifold of the learned space where $\tilde{y}$ and learning an isometry (injective isometry and its inverse) that creates a mapping between $\tilde{y}$ and $\tilde{x}$. The composition ψ of ψ_e and ψ_d creates a mapping between $\tilde{y}$ and y. ψ_e and ψ_d are respectively the encoder's and decoder's functions.

4.2. Use Cases

LOCA and I-AE have different approaches for isometric auto-encoders. While LOCA uses bursts, I-AE does not. We can use LOCA for uncovering measurements from different measurement devices. We can use I-AE in general manifold learning tasks. Thus, for a fair comparison, we are interested in use cases developed for LOCA and applying them for I-AE. Similarly, we are interested in applying a general use case where bursts are not available to be fair to I-AE since it is not made for cases where burst sampling is available.

4.2.1. Manifold Learning for Dimensionality Reduction

Many datasets are made available for testing manifold learning in dimensionality reduction. We test both LOCA and I-AE on one dataset: Swiss Roll.

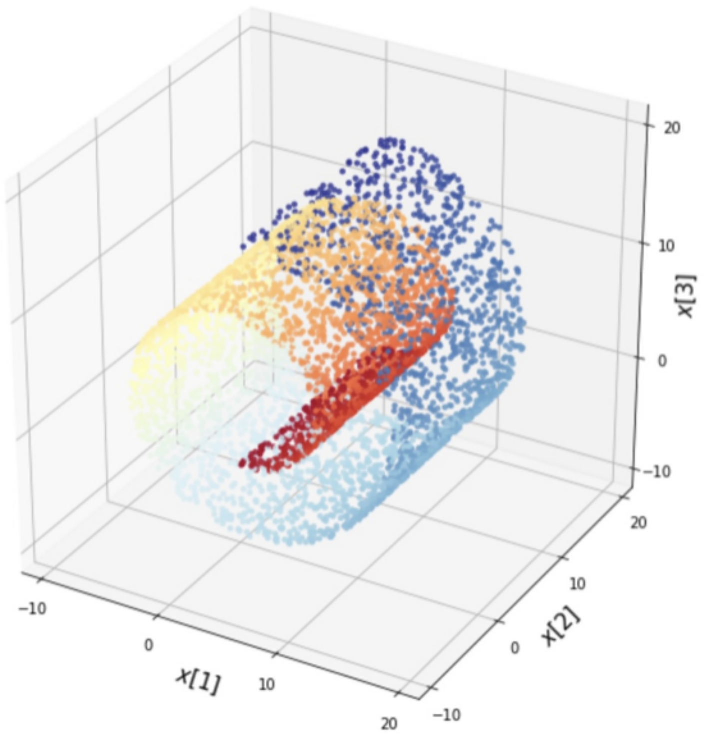


Figure 4.5.: Swiss Roll dataset represents the manifold $\mathcal{M}$. Five thousand points y_i is sampled on the swiss roll manifold.

We choose this data set as it represents a smooth manifold and does not need the strategy of burst sampling for a fair comparison between LOCA and I-AE, as mentioned in section

3.
The manifold $\mathcal{M}$ on which lie the sampled points y_i is a 2-dimensional manifold embedded in $\mathbb{R}^3$. Thus the manifold learning task is to learn an isometric 2-d embedding of $\mathcal{M}$.

4.2.2. Manifold Learning using burst sampling

In this case, the experiment uses the burst sampling method. N points $x = (x_1, x_2)$ are sampled uniformly on a 2-dimensional manifold to correspond to the unknown manifold $\mathcal{N}$ such that

$$x_i \sim U_{[0,1]^2}.$$

To construct a burst around each ith point x_i, M points $x_{i,m}$ are sampled such that

$$x_{i,m} \sim N(x_i, \sigma^2 I_2).$$

We transform the manifold $\mathcal{N}$ using the following non-linear equation to give the points $y = \rho(x)$ is given by

$$\rho(x) = \begin{pmatrix} x_1 + x_2^3 \\ -x_1^3 + x_2 \end{pmatrix}.$$

Since the authors in [13] do not define I-AE for burst sampling, we need to modify a modification the input. We discuss how we modify the burst input to suit the burst sampling for I-AE in . We discuss the results in section 5. We do not compare the quality of the learned isometry, which we also explain in section 5.
Thus we obtain bursts that are supposed to be sampled on the deformed manifold and observed in $\mathbb{R}^2$.

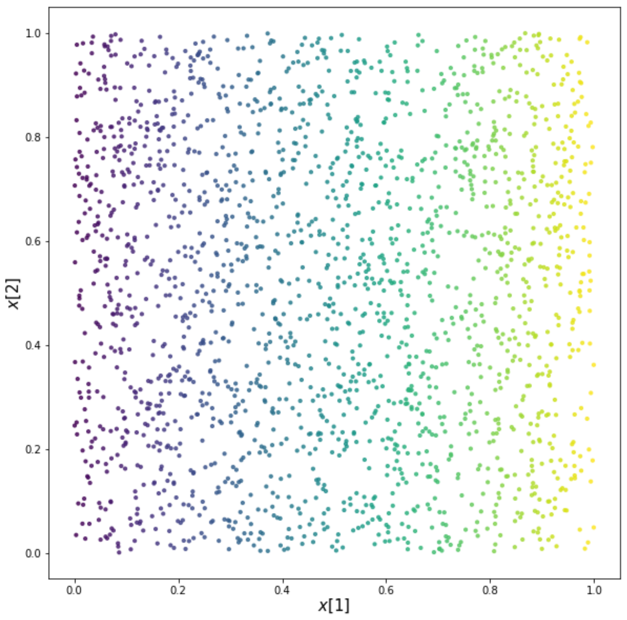


Figure 4.6.: Illustration of the real latent space N in which the points x_i lie. Here $N = 200$ and $M = 200$. Each point represents a burst that we do not draw for simplicity.

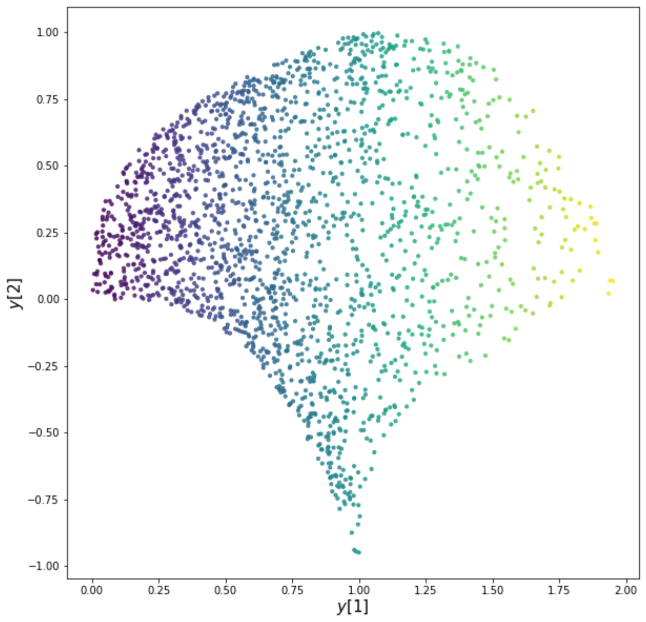


Figure 4.7.: Illustration of the transformed points y_i that lie on the observed manifold $\mathcal{M}$.

The figures 4.1 and 4.2 represent a burst around $y_i = (0.75, 0.25)$ respectively before and after the transformation ρ was applied on the manifolds represented by figure 4.6. The transformation of figure 4.6 is illustrated in figure 4.7.

4.3. Implementation

4.3.1. I-AE

We implement I-AE based on the authors' paper [13]. We use TensorFlow which was also used for LOCA.

Hyper-parameter tuning For each of the experiments where I-AE was used, the model was hyper-tuned using the $KerasHypertuner$ [2]. The parameters are optimized to find the best overall loss the fastest.

- **number of hidden layers** : It references the number of hidden layers in the encoder and we mirror the architecture for the encoder. This value ranges in the set

$\{1, 2, 3, 4, 5\}$

- **number of units per layer** range in the set $\{32, 64, 128, 256\}$
- **type of architecture** : ”same”: we use the same number of units for each hidden layer.
- **activations for the hidden layers** : softmax, relu, tanh or softplus. Each has its limitations.

Adapting I-AE For the mushroom experiment that uses burst sampling, the input to the encoder is a matrix

$$I \in \mathbb{R}^{N \times M},$$

with N the number of bursts and M the number of sampled points in each burst. M is the same for each burst.
I-AE is fed this matrix just flattened. This makes sure we use the same information as in LOCA. Nevertheless, since the algorithm of I-AE does not use the observed correlation matrices, we cannot adopt this input further.
Thus I-AE is just fed I

$$I \in \mathbb{R}^{NM}.$$

4.3.2. LOCA

We use the implementation of LOCA that is available in the open-source project [1].

Nevertheless, LOCA needs the presence of bursts as input. Let I be the input example for LOCA. In order to use the Swiss Roll without knowledge of bursts (for a fair comparison with I-AE), we need to add a heading function on top of LOCA that re-adapts the input I as sampled bursts.
To do so, we follow the reasoning of the methods in section 2. We define a burst around a point y_i near the observed manifold $\mathcal{M}$ as a neighborhood of y_i and use k-NN to choose such neighborhoods.
If theoretically, for LOCA, a burst represents M points uniformly sampled on the lower manifold, here we do not have access to this lower manifold. We content ourselves in defining the bursts on the higher dimensional manifold. Therefore, we select $k = M$ neighbors of the point y_i and define the resulting neighborhoods as our observed bursts. We do not have further information about the manifold. We do not want to learn a distribution to select the right neighborhoods (using geodesics on the observed manifold or a kernel method that relates to the covariance on the observed manifold), which would defeat the purpose and incorporate the input to LOCA. As a result, we take the simplest approach of k-NN for the neighborhood selection.

Thus in LOCA, we optimize additionally

- **k** Number of neighbors in the k-NN algorithm
- σ Standard deviation of the bursts, since the variance of the bursts is not known as the points are not sampled using the bursts sampling strategy.

4.4. Isometry comparison

To compare the quality of the isometry, we choose two methods.

4.4.1. Euclidean local distances

When we use burst sampling, the distances on the latent manifold are accessible. Thus, we can use the euclidean distances on the manifold to compare the quality of the isometry imposed between LOCA and I-AE using the local euclidean distances.

4.4.2. Testing for finding the real metric

We use the results of the paper [25] and the package Megaman, introduced in section 2, to uncover the distortions between the real embedding and the learned metric.

5. Computational Experiments

5.1. Experiment A : Mushroom Experiment

This section discusses the results of implementing the mushroom experiment discussed in section **??**. For implementing this experiment, we use the same type of architecture as in LOCA (fully connected layers with non-linear activations). However, we hyper-tune the number of units per layer and the number of layers to give more freedom to I-AE to find the right transformation but still be comparable to the results given by LOCA.

5.1.1. I-AE

I-AE is hyper-tuned to find the best structure that minimizes the loss and converges the fastest. We report the results of the experiment using the parameters in table 5.1.

Table 5.1.: Best parameters retained after hyper-tuning

number of layers	1
number of units per layer	256
learning rate	0.00024698
λ_{reg}	100

We inspect each of the losses introduced by I-AE that individually reinforce one condition for imposing the isometry. We report their convergence in training. Furthermore, to understand which transformation the auto-encoder learns, we visualize the learned embeddeding in $\mathbb{R}^2$ and the reconstructed manifold in $\mathbb{R}^2$ of the mushroom experiment. Using I-AE yields the results reported in Figures 5.1, 5.2 and 5.3.

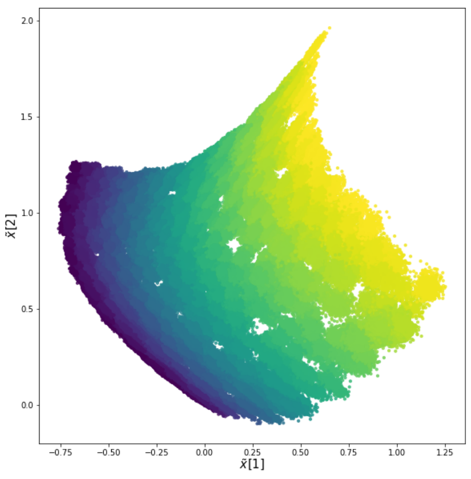


Figure 5.1.: The learned isometric embedding of the unknown latent space $\mathcal{N}$ into $\mathbb{R}^2$. It represents the predicted points $\tilde{x}_i$, which are the codes learned by the auto-encoder I-AE

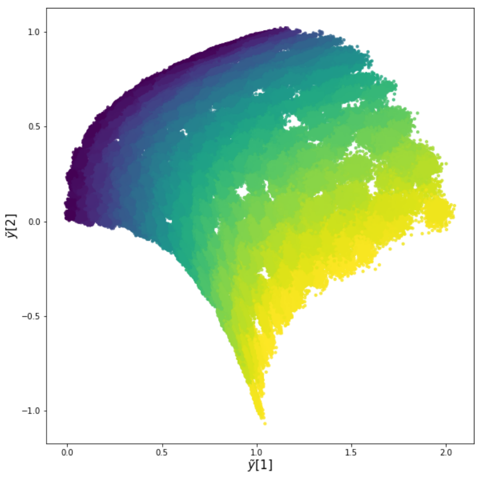


Figure 5.2.: The reconstructed mushroom cloud that was input in figure 4.4.2. The points correspond to the predicted points $\tilde{y}_i$ decoded from the learned embedding in figure 5.1.

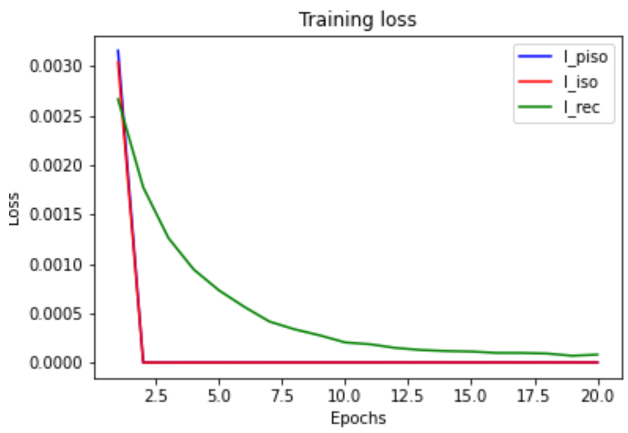


Figure 5.3.: Illustration of the convergence of the losses (reported on linear scale) included in the optimization of I-AE for the mushroom experiment. l_{rec}, in green, refers to the reconstruction loss l_{rec}. l_{iso}, in red, refers to the reconstruction loss l_{iso}, which tracks how ”far” the auto-encoder is to finding an isometric embedding. l_{piso}, in blue, refers to the isometry loss l_{piso}, which reinforces that the encoder is the pseudo-inverse of the decoder.

The losses converge empirically to 0 as the number of epochs grows. The l_{piso} and l_{iso} drop fast to 0 as the auto-encoder overfits on the output. We are empirically sure that the losses enforce the isometry conditions as introduced in section 4.
For auto-encoders to learn something, there must be a bottleneck in encoding the input space. However, along with an imposed inverse function for the decoder that does not add information on encoding the unknown manifold isometrically, the auto-encoder learns a rigid transformation of the observed manifold. The learned latent space in figure 5.1 is a rotation of the observed manifold in figure 4.7. We expected such result, as I-AE imposes an isometry based on the observed distances and does not link it to the distances on the unknown manifold using the extra information given by the distributions of the bursts. Thus, despite the regularization in the form of isometry, the auto-encoder still cannot uncover the real latent "code".

5.1.2. LOCA

This experiment's results are available in [1]. LOCA links the variance on the latent space to the transformation made on the unknown manifold, which permits LOCA to be an auto-encoder that can capture more than just a rigid transformation of the ambient space.
The authors of LOCA suggest calculating the deformation of local Euclidean distances to compare the quality of the isometry. Since I-AE failed to uncover an isometry that is informative, the distances would be the same as with an isometry, but it is irrelevant. Thus, comparing the quality of the isometry uncovered with LOCA and I-AE would not be correct.

5.2. Experiment B : Swiss Roll dataset

We compare LOCA and IAE's performance on the swiss roll set. On the one hand, we use the burst sampling method to generate the dataset, and on the other hand, we do not use burst information but keep the center of the bursts only.

5.2.1. Using burst sampling

As described in the mushroom experiment, we sample N points $x = (x_1, x_2)$ uniformly on a 2-dimensional manifold to correspond to the unknown manifold $\mathcal{N}$ such that

$$x_i \sim U_{[0,1]^2}.$$

To construct a burst around each ith point x_i, M points $x_{i,m}$ are sampled such that

$$x_{i,m} \sim N(x_i, \sigma^2 I_2).$$

We transform the manifold $\mathcal{N}$ using the following non-linear equation to give the points $y = \rho(x)$ is given by

$$\rho(x) = \begin{pmatrix} t_1 cos(t_1) \\ 21 t_2 \\ t_1 sin(t_1) \end{pmatrix},$$

with

$$t = (1.5\pi(1 + 2x_1), x_2).$$

Here we use the same transformation as given in available data sets ($sklearn$). The data set is re-generated instead of being used directly to access the actual embedding and assess the quality of the isometry.

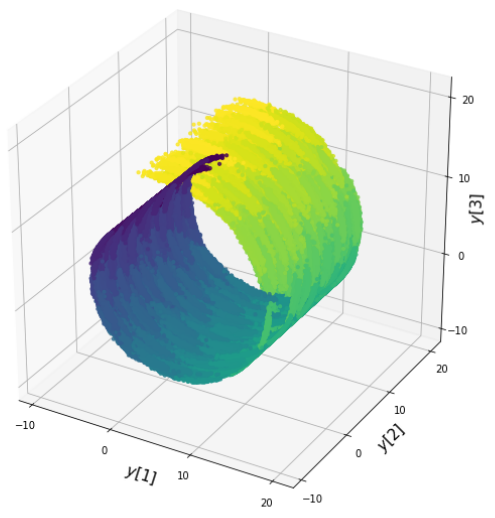
We only use a cut part of the manifold to accelerate our experiments. Thus we cut the point of the Swill Roll to not engage more computational force into unfolding the swiss-roll entirely.

Thus only a part of the manifold shaped similarly to an open cylinder is kept. It still embodies the problem of short circuits at the close boundaries of the dataset and as well as the difficulty of unfolding a 3-d manifold into a lower 2-d space which still corresponds to our problem.

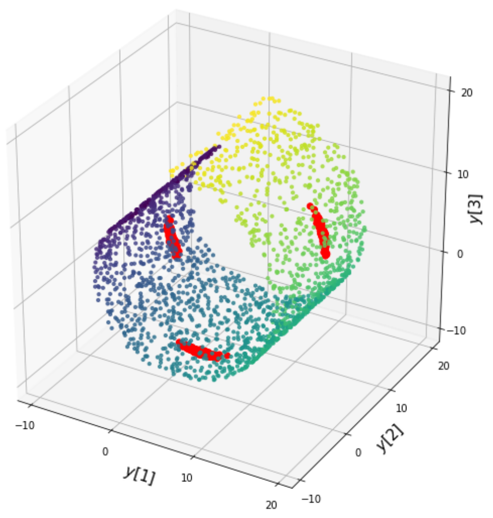
I-AE We train I-AE on the above data set, using 80% for training and 20% for testing. In training, we use the parameters in table 5.2. Figures 5.4 and 5.5 show the results of our experiments.

Table 5.2.: Best parameters retained after hyper-tuning

number of layers	3
number of units per layer	256
learning rate	0.0003
λ_{reg}	100
batch size	64



(a)



(b)

Figure 5.4.: The bursts generated on the real 2-d embedding $\mathcal{N}$ with $\sigma_{burst} = 0.01$. The manifold is folded and cut to keep only a part that is faster to unfold but still exhibits the same difficulty as the manifold learning problem. Here 5000 bursts are generated, each having 200 points, and only keep 2000: a total of 400000 data points. In figure 5.4a, we show the manifold with all the bursts. In figure 5.4b, we show the centers of the deformed bursts and the three full bursts in red. The transformation makes the bursts much more elongated along the first axis.

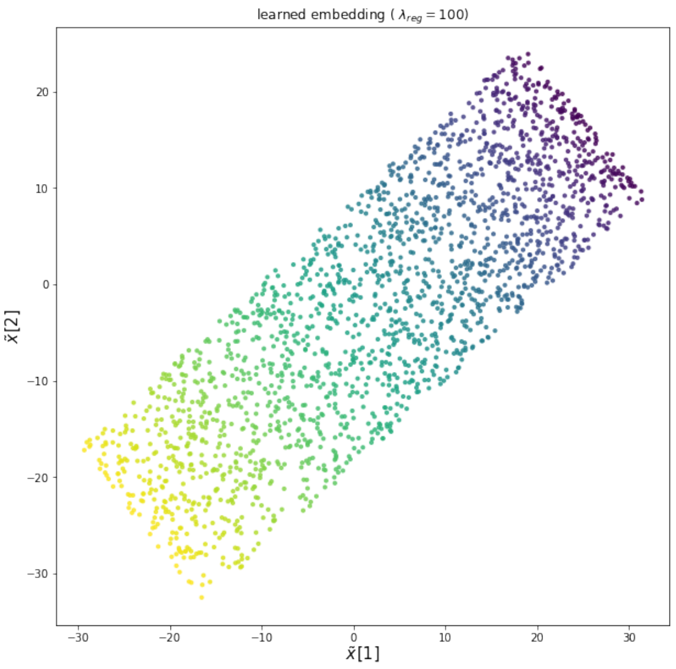


Figure 5.5.: I-AE learns a rigid transformation of the 2-d embedding using a flattened version of the burst information which amounts to 400000 data points. We show only the centers of the bursts.

I-AE reconstructs as well the 3-d manifold in the right way. The losses l_{piso}, l_{iso} and l_{rec} empirically converge after 500 epochs.

LOCA We retain the parameters used for the architecture when training I-AE in order to train LOCA. An additional needed parameter is σ_{burst} which is the standard deviation of the bursts on the 2-d manifold. Figure 5.6 reports the results of hyper-tuning σ_{burst}.

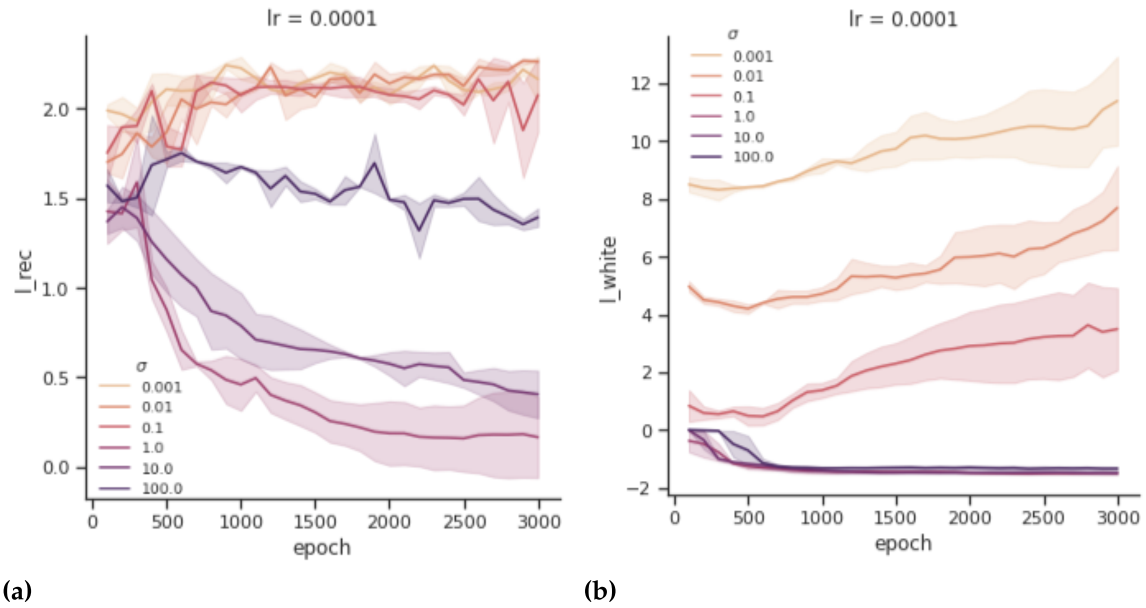


Figure 5.6.: Hyper-tuning σ is done by running LOCA for over 3000 epochs until the losses (reported on logarithmic scale) empirically do not exhibit very promising change. This logic stems from the same logic used when manually hyper-tuning, a model which is also the logic behind $KerasHypertuner$. We explore for 3000 epochs initially the most promising hyper-parameters, which will be used for training for more epochs. We report losses on a logarithmic scale. Figure 5.6a shows the evolution of l_{rec} and 5.6b the evolution of l_{white}, both by the value of σ along the number of epochs. The error in hue color represents the variation in the losses' values' due to the use of different batches. Although for generating the bursts, we used $\sigma_{burst} = 0.01$, we notice that the training is not possible with such value. In fact, the losses diverge for low values of σ as illustrated in figure 5.6b. For the swiss roll dataset, the smaller the value of σ $(0.01, 0.001)$, the more inconsistent the results are. Actually, the l_{rec} losses do not converge to low values and the l_{white} diverge. Acceptable values of losses are obtained when σ is 1 or 10.

A possible explanation is a difference in the architecture of the neural network. Since the parameters of networks are not optimized in the same manner, this could affect the capacity needed for learning the underlying manifold. However, to keep a comparative framework. We use the same structure and optimize the used σ_{bursts}. Furthermore, in more general use cases, σ is unavailable, and we need to find it. Therefore, it is interesting to see the influence of finding the embedding while hyper-tuning σ's value as well.

Table 5.3.: Best parameters retained (same architecture used as in I-AE)

number of layers	3
number of units per layer	256
learning rate	0.0001
batch size	64
σ	1

Using the parameters in table 5.3, we train LOCA using the same training and testing datasets used in training I-AE. Figures 5.7, 5.8 and 5.9 show the results of our experiments.

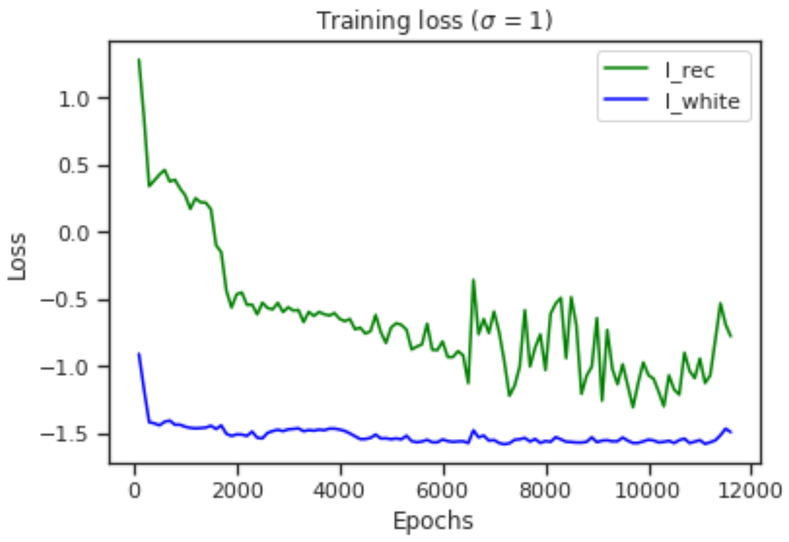
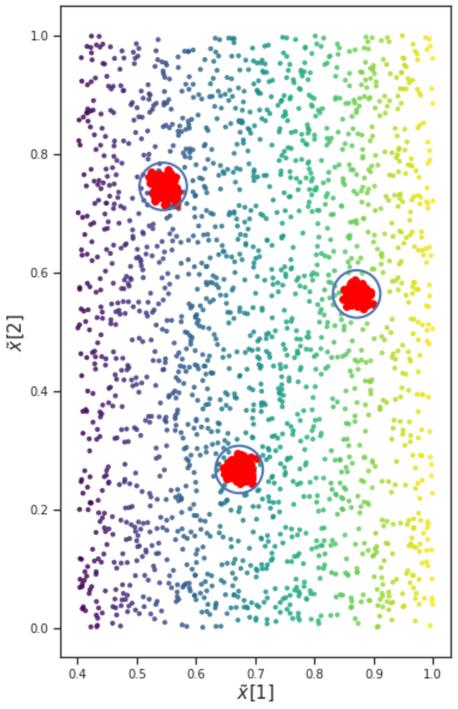
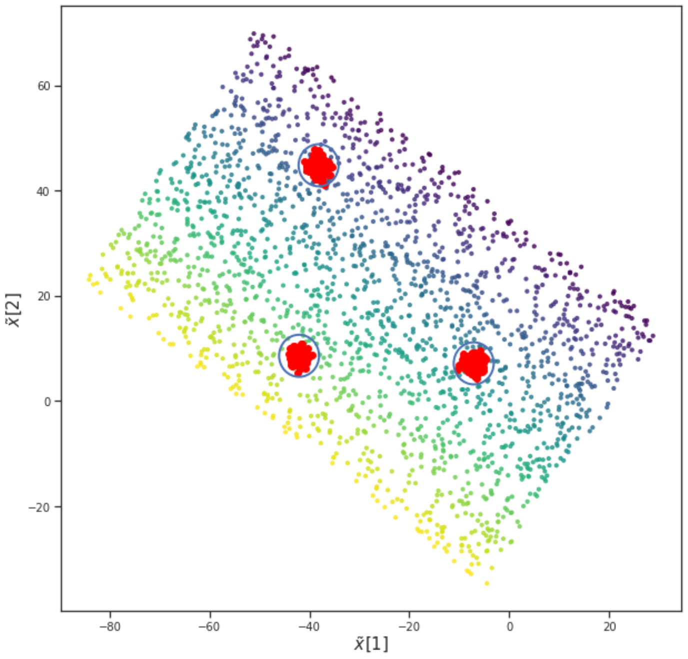


Figure 5.7.: We train the model for 12000 epochs (until having stable empirically converging losses (reported on logarithmic scale)). Unlike I-AE, for this experiment, the losses do not drop very close to 0. However, the results are acceptable as the losses converge to values close to 0 (order of 0.01).





(a) **(b)**

Figure 5.8.: LOCA is the 2-d manifold. Figure 5.8a shows the true manifold from which stems the training dataset. Only the centers of the bursts are shown, with three bursts fully drawn. Figure 5.8b shows the learned 2-d manifold which is a rigid transformation of the real 2-d manifold. LOCA retrieves the right distributions of the bursts.

Comparison of results In contrast to I-AE, LOCA learns the distribution of the real manifold. In fact, as portrayed in 5.5, I-AE learns an unfolding of the prolonged distributions of the real embedding 5.8a. In addition to learning a lower representation of the data, LOCA retrieves the original distributions by taking into account the variances of the bursts as illustrated in figure 5.8b.

Since we have access to the distances between points on the actual embedding, we can assess the quality of the learned isometry by assessing how the distances are kept empirically. The results are reported in figure 5.9.

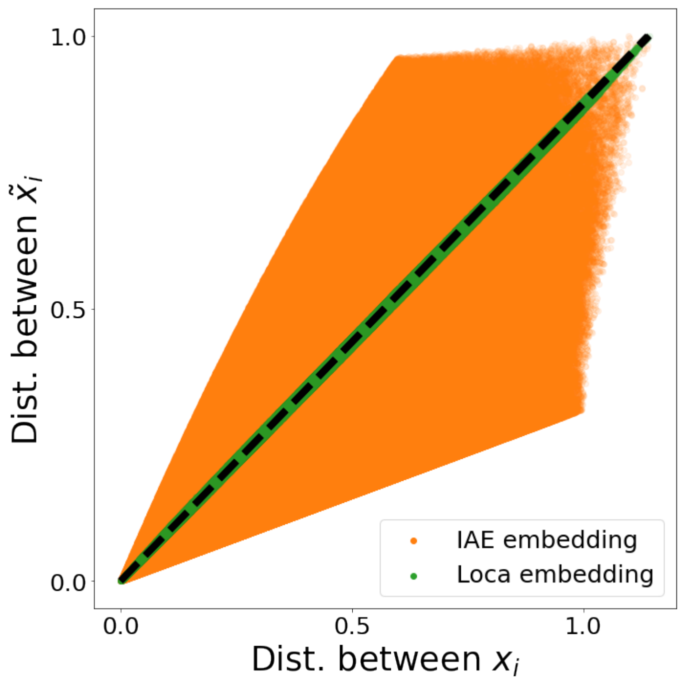


Figure 5.9.: On the x-axis, we report the distances between points on the original 2-d manifold (standardized by the maximum). On the y-axis, we report the distances between the points on the learned embedding. A model that imposes an isometry between the learned and real manifold would have an identity correspondence between both distances. LOCA's results fall well on the identity line. However, I-AE has a high variance around the identity line. It struggles even more, to impose the isometry as the standardized distances are close to 1. The higher the distances, the more the distribution is close to a random distribution over the distances.

5.2.2. Without using burst sampling

In this experiment, we do not use the burst sampling to uncover a 2-d embedding of the manifold illustrated in figure 5.15.

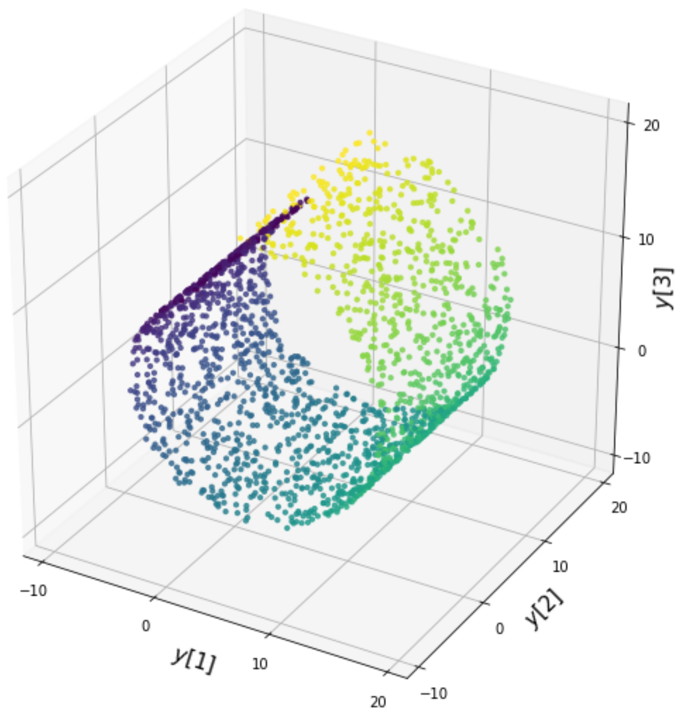


Figure 5.10.: represents the manifold $\mathcal{M}$ for which the models I-AE and LOCA try to uncover an embedding in 2-d without using the bursts sampling method.

I-AE This experiment represents more of the natural setting of I-AE. We try to explore more the effect of finding the proper regularization coefficient λ_{reg}.
If hyper-tuned, optimizing the losses will cause the hyper-tuner to automatically choose the smallest λ_{reg} since it directly minimizes the total loss as they are all positive entities as reported in Figure 5.11.

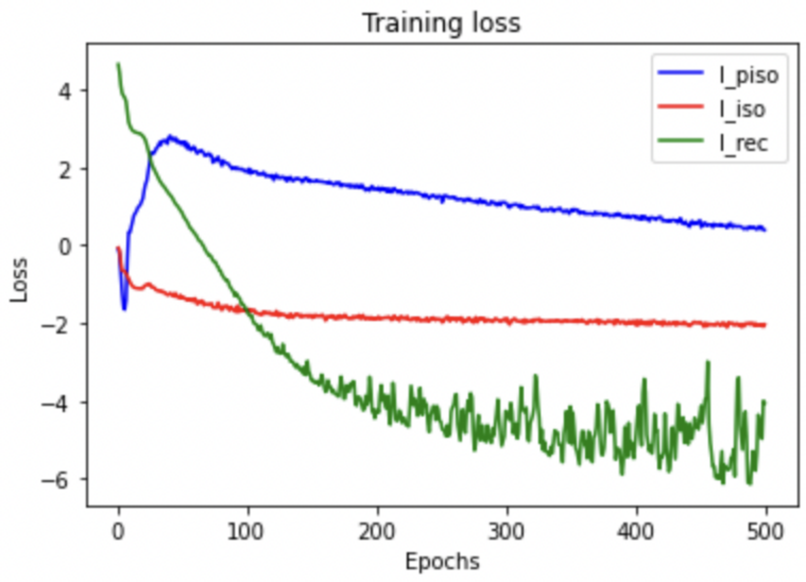
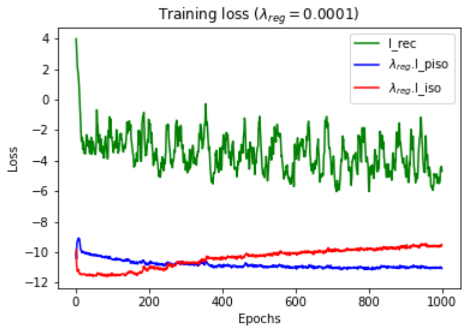


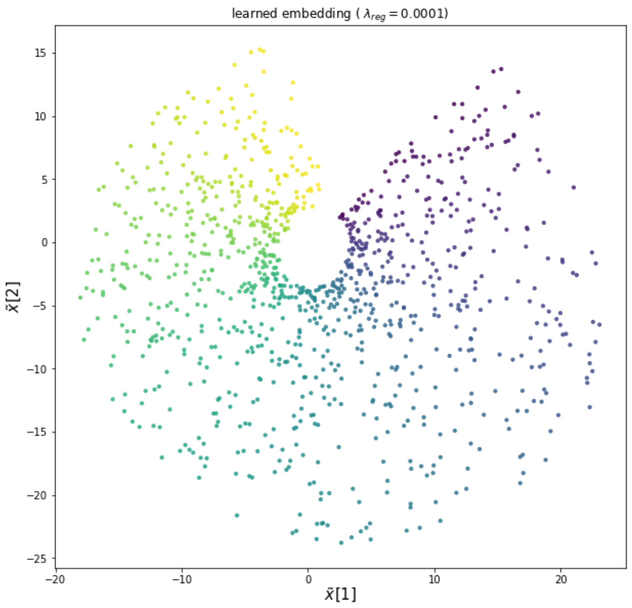
Figure 5.11.: Losses' convergence (reported on logarithmic scale) when λ_{reg} is also hyper-tuned show that the hyper-tuner fixes λ_{reg} as low as possible. This makes reinforcing the isometry impossible as $\lambda_{reg} << 0$, since l_{piso} and l_{iso} are practically disregarded.

We fix λ_{reg} which we choose to range in $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}\}$. For each of the values of λ_{reg}, we investigate whether the isometry is reinforced as defined in section 4 by visualizing the losses' convergence and the learned embeddings.

Figures 5.12,5.13 and 5.14 show the evolution of l_{piso}, l_{iso} and l_{rec}. The l_{piso} and l_{iso} are multiplied by λ_{reg} and all losses are reported in logarithmic scale.

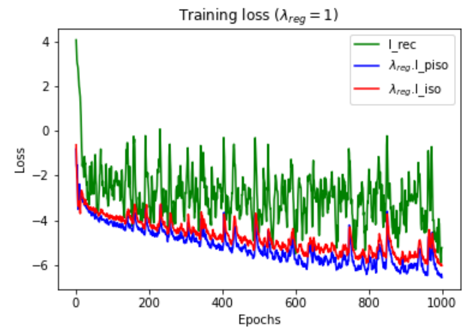


(a)

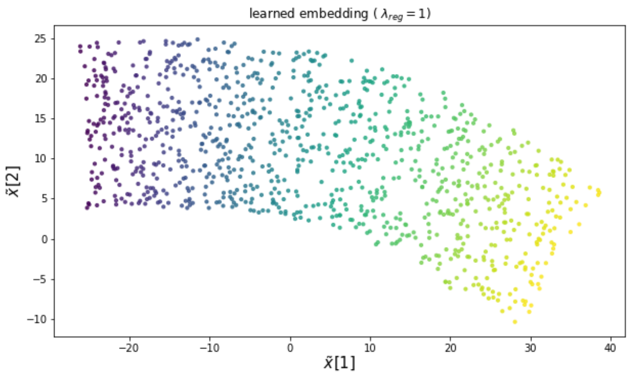


(b)

Figure 5.12.: For small values of λ_{reg}, the I-AE behaves as a vanilla auto-encoder that cannot unfold the swiss roll.

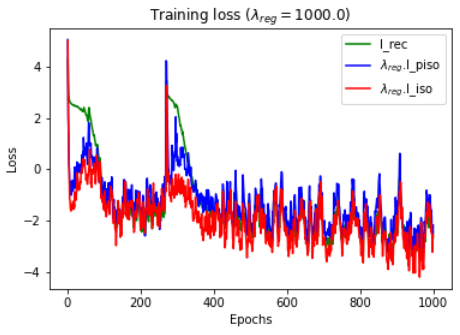


(a)

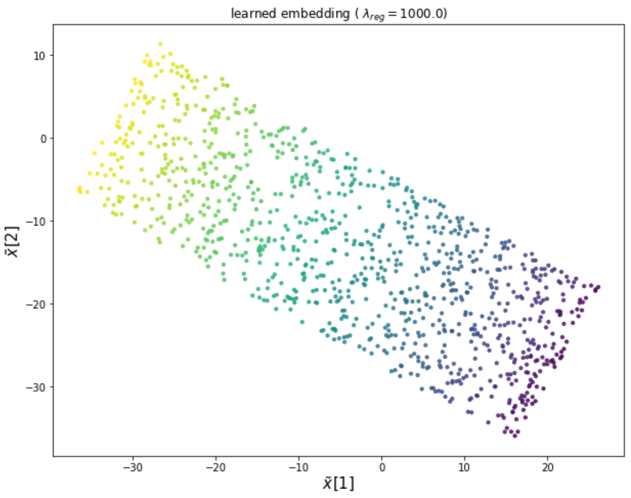


(b)

Figure 5.13.: The higher the value of λ_{reg}, the more I-AE can unfold the swiss roll. l_{red} is not smooth, oscillates overall with a variance of 0.1 and slowly decreases to $\epsilon = 0.0001$.



(a)

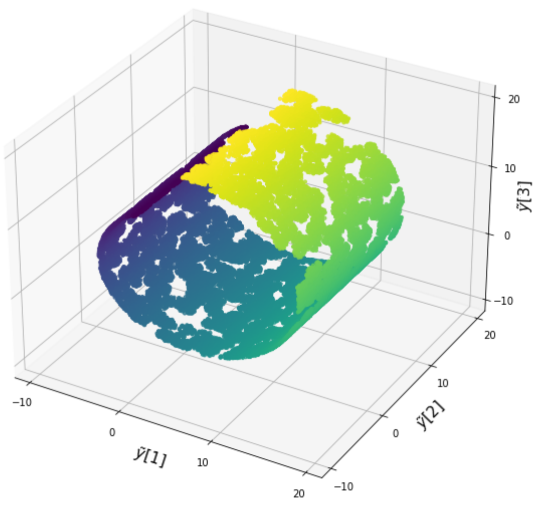


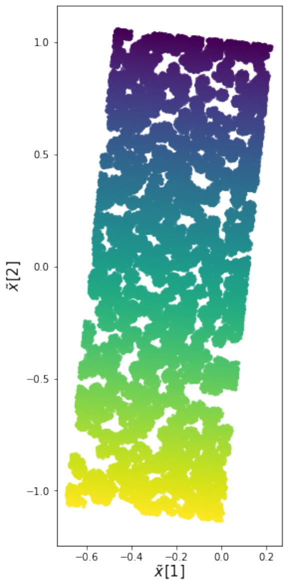
(b)

Figure 5.14.: I-AE unrolls perfectly the 3-d manifold when $\lambda_{reg} \in \{10^2, 10^3\}$. The l_{rec}, l_{iso} and l_{piso} have the same trend. When multiplied by λ_{reg}, the losses have the same order of value to impose the isometry.

I-AE can unfold the swiss roll when the isometry can be enforced by a strong enough λ_{reg}. The isometry brings regularization to the architecture of the auto-encoder. Therefore, contrary to a vanilla auto-encoder, it can unfold the swiss roll dataset.

The losses in I-AE are not very stable and can have sudden peaks to re-converge to 0, and only a small number of epochs (order of (10^2) is necessary for empirical convergence. The figures 5.12,5.13 and 5.14 result on using a dataset with 1000 points only. To compare later to LOCA, which uses k-NN bursts, we also train I-AE on flattened k-NN bursts. Figure 5.15 shows the results of I-AE trained on such a dataset while keeping the same parameters used for a dataset without bursts. The manifold generated using N k-NN bursts with $k = 64$ and $N = 1000$, which mounts to 64000 data points.
The results of these experiments will be used later to compare the isometry with LOCA using the Megaman package.





(a) **(b)**

Figure 5.15.: While using k-NN bursts, I-AE can learn a 2-d embedding.

LOCA

As described in 4, we use k-NN bursts to unfold the swiss roll. In figures 5.16 and 5.17, we report the optimization process for LOCA.

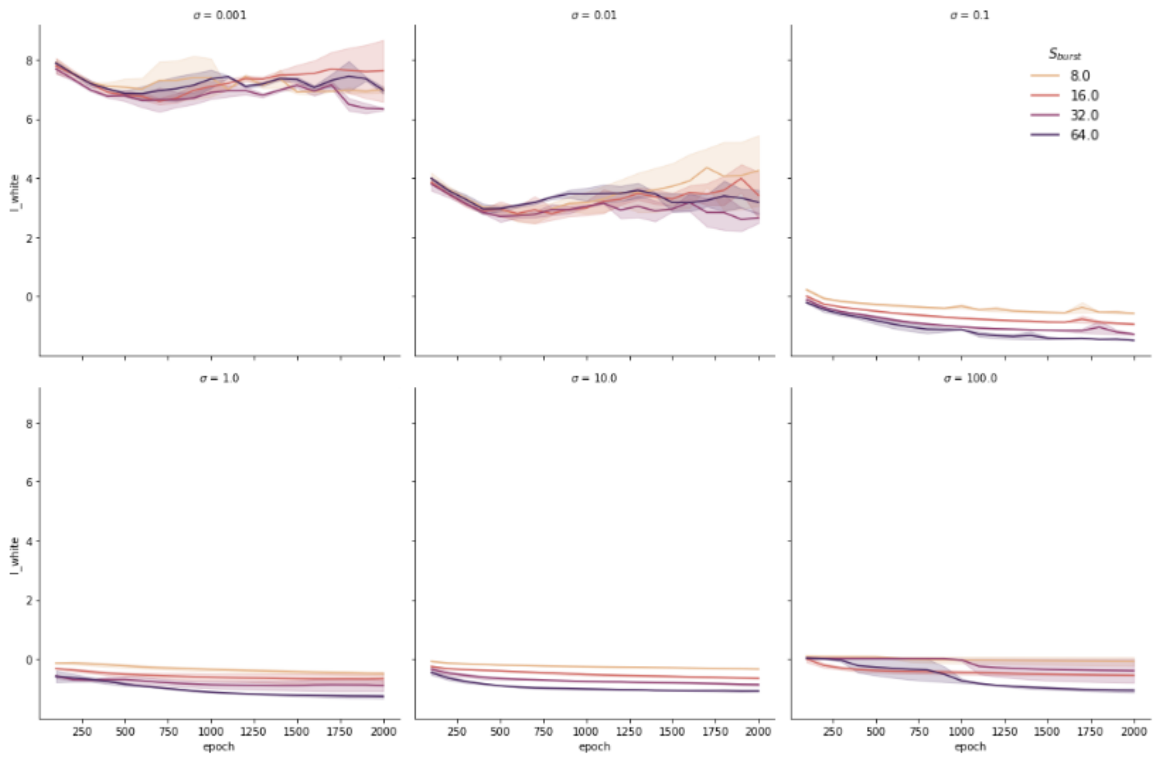


Figure 5.16.: Losses are reported on logarithmic scale. l_{white} diverges for values of σ higher than 10^{-1}. No matter what value of k, the converge nature for l_{white} is the same. The hue used in the figure represents different learning rates $\{10^{-3}, 10^{-4}\}$. The values of l_{white} when there is convergence are only $O(10^{-1})$. Higher values of k have lower values of l_{white} and smoother losses, which means the more points in the neighborhood, the more stable the calculations. Nevertheless, the neighborhood should not be too large. Otherwise, the assumption that the manifold can be locally approximated on the burst as a linear space fails. Although the assumptions for LOCA impose a value of σ that is small, the algorithm is still robust and calculates an empirically converging loss.

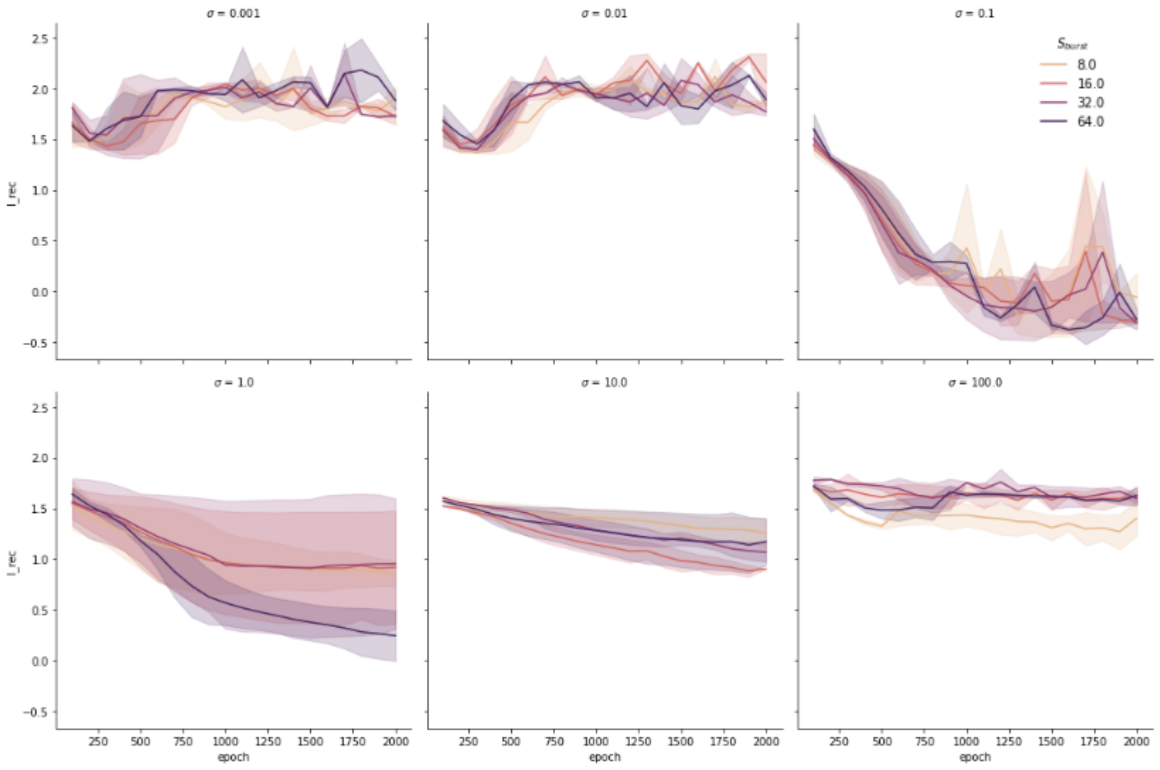


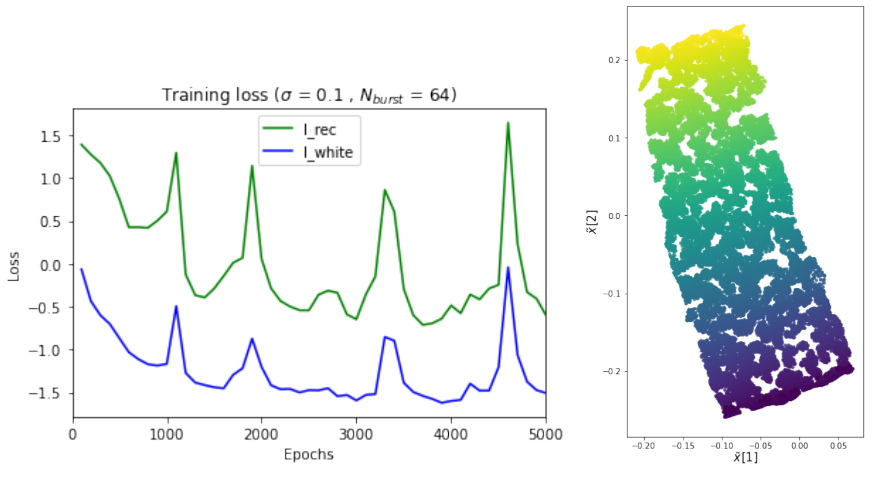
Figure 5.17.: l_{rec} and l_{white} have the same tendencies. l_{rec} diverges for low values of σ. The hue represents a different learning rate (same as for l_{white}). l_{rec} converges empirically to 0 for $\sigma = 0.1$. Values of high neighborhoods converge to lower values than lower k values, although all oscillate relatively to the learning rate or k or both. Losses are reported on logarithmic scale

We retain the best parameters mentioned in the table 5.4.

Table 5.4.: Best parameters retained (same architecture used as in I-AE)

number of layers	3
number of units per layer	256
learning rate	0.0001
batch size	64
σ	0.1
k	64

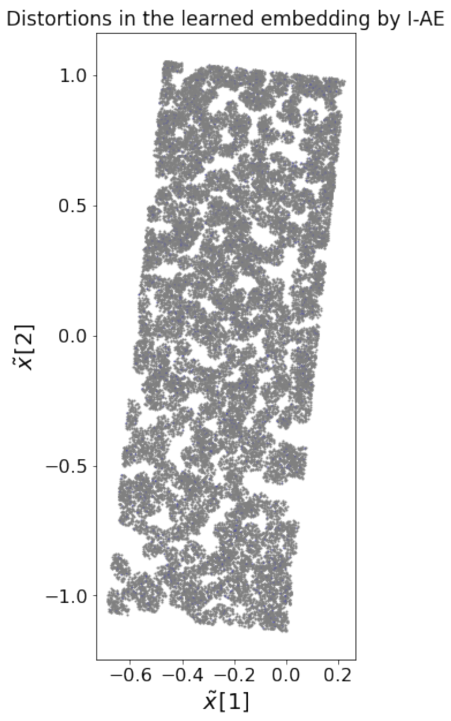
Comparing results We run LOCA on the swiss roll obtained with N bursts with $k = 64$ bursts and $N = 1000$ and obtain the results analyzed in Figure 5.18.



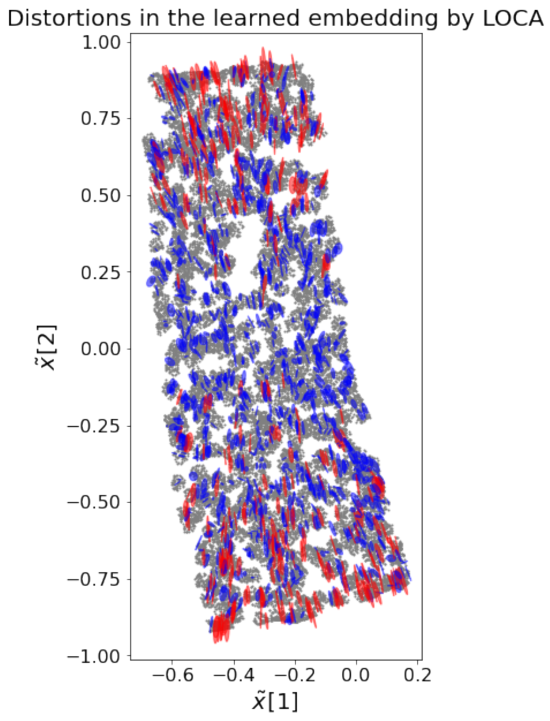
(a) **(b)**

Figure 5.18.: While using k-NN bursts, LOCA can learn a 2-d embedding. Nevertheless, it distorts the manifold where the boundaries were close on the higher dimensional manifold. Losses (reported on logarithmic scale) are unstable during training.

To compare the quality of the isometry that is induced respectively by LOCA and I-AE, we use the package Megaman introduced in section 2. We use the resulting embedding from both I-AE and LOCA and extract the deformation on the learned manifold. The results are reported in form of deformation variance ellipses. The large the ellipse in on axis the more the deformation is important in that particular direction. These results are reported in 5.19. Furthermore, for a better inspection of these variances, we report the distributions of the ellipses' heights and widths that relate directly to the variance as their are a linear transformation of the spectral decomposition of the covariance matrices extracted by Megaman. We report these distributions in 5.20.



(a)



(b)

Figure 5.19.: Ellipses in red represent variance in one axis higher than 0.00015; otherwise, the ellipses are blue.

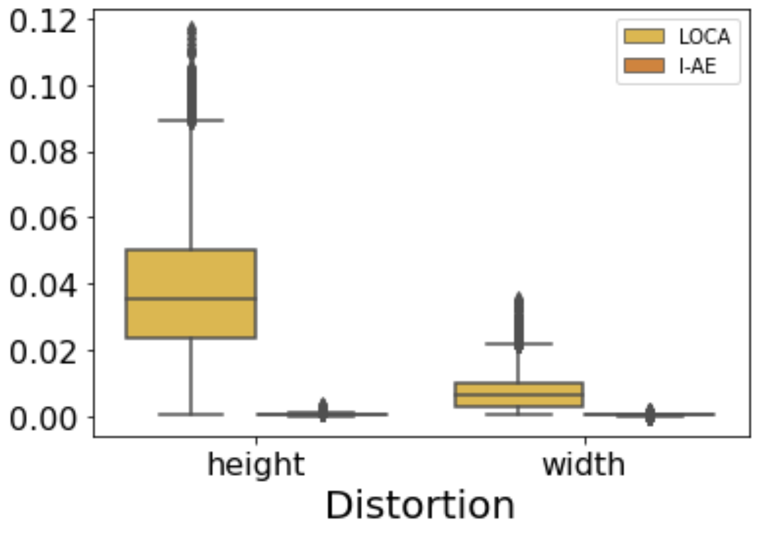


Figure 5.20.: Descriptive statistics of the distribution of the heights and widths of the distortion ellipses. As reported on Figure 5.19, the deformations for I-AE are negligible in comparison to LOCA. The deformations for LOCA are more important on the borders and even more on the lower half than I-AE. The embedding is slightly larger than the upper half.

Comments on results Although Megaman helps quantify the deformation in learning the manifold, it is worth noting that the mathematical theory behind it is close to that of I-AE. Both have the same approach when defining isometry. The approach for I-AE to find the suitable embedding uses auto-encoders, while Megaman calculates the Laplacian of the manifold when seen as a connected graph.

Part III.

Conclusion and Future Work

In this chapter, we will review our approach to comparing two isometric embedding models, I-AE and LOCA, and our results. We will further present future possibilities for continuing this work.

In section 3, we presented the framework we used to compare both models. In section 4, we first presented a common mathematical framework to establish the theories behind both LOCA and I-AE. This part explained how isometry is defined theoretically. We proceeded to find a common practical way to compare the results of implemented experiments. We focused on use cases that are the natural settings for LOCA or I-AE. We divided our experiments into two main experiments. In the first experiment (mushroom experiment), we compare the power of both models to extract useful information on manifolds in $\mathbb{R}^2$ having embedded information in $\mathbb{R}^2$ as well. The second experiment (swiss roll dataset) involves finding an embedding of a manifold in $\mathbb{R}^3$ with a latent space in $\mathbb{R}^2$. We use a more generic dataset that is famous for testing Manifold Learning algorithms. Moreover, we used first the burst sampling method and second a standard setting where only data points are provided with no further information about the distributions over the neighborhoods of points.

In section 5, we presented the results of the experiments as presented above. We have found that for the mushroom experiment, LOCA has an advantage in the case where we dispose of the burst sampling strategy. Furthermore, unlike I-AE and vanilla autoencoders, LOCA can extract useful information from the 2-d manifold and embed it again in 2-d. In the first part of the swiss roll experiment, we found that LOCA, as expected and when burst sampling is available, can find a mapping in $\mathbb{R}^2$ of a manifold embedded in $\mathbb{R}^3$. I-AE can also do the same when the burst information is flattened as input. However, I-AE does not incorporate information about bursts' deformations. Thus, while for LOCA, we retrieved the embedding as defined in $\mathbb{R}^2$ before the deformation, I-AE could only recuperate an extended version of it as it cannot shrink back the elongated manifold. Furthermore, we compared the quality of isometry on the manifold. We remark that LOCA, although using a value of σ that is different from the theoretical one, the model converges and keeps the exact distances between the learned and embedded manifold.

I-AE, on the other hand, deforms points on close boundaries. This has two explanations: on the one hand, the observed points are on an elongated 3-d manifold, and thus the distances are more distorted as we move away from the axis around which we folded the data. On the other hand, I-AE imposes the isometry between the observed manifold and the latent variables; therefore, the distances are more distorted with I-AE.

In the second part of the swiss roll experiment, we did not use the burst information. We uncovered, using I-AE, the 2-d elongated embedding. LOCA gave similar results with the cost of hyper tuning additionally k, the number of neighbors in k-NN bursts, and σ the used variance of the burst when optimizing LOCA. We used Megaman to compare the isometry quality and discovered that LOCA deforms the embedding slightly on closer boundaries in 3-d.

We have only tackled two comparison points. As stated in section 3, further comparison angles such as the robustness to noise on the manifold and the robustness to small datasets can be explored. This needs first to define the concept of noise on a manifold and then explore whether we can prove theoretically or empirically which of LOCA and I-AE is indeed robust to noise. Furthermore, specific use cases, such as the MNIST [9] data set that uses other layers in the auto-encoder, would be interesting to explore to see if LOCA can still produce interesting results when using convolutional layers [11]rather than linear ones.

Bibliography

[1] Code supplement for ”local conformal autoencoderfor standardized data coordinates” by peterfreund, e., lindenbaum, o., dietrich, f., bertalanc, t., gavish, m., kevrekidisc, i. g., and coifman, r.r. (2020). `https://purl.stanford.edu/zt044bg9296`. Accessed: 15-08-2022.

[2] KerasTuner description. `https://keras.io/keras_tuner/`. Accessed: 2022-09-21.

[3] Ranya Almohsen, Matthew R Keaton, Donald A Adjeroh, and Gianfranco Doretto. Generative probabilistic novelty detection with isometric adversarial autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2003–2013, 2022.

[4] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001.

[5] Chris M Bishop. Neural networks and their applications. *Review of scientific instruments*, 65(6):1803–1832, 1994.

[6] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences*, 103(5):1168–1172, 2006.

[7] Chaomei Chen. Generalised similarity analysis and pathfinder network scaling. *Interacting with computers*, 10(2):107–128, 1998.

[8] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[9] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[10] David L Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[12] Patrick Groenen and Michel van de Velden. Multidimensional scaling. Technical report, 2004.

[13] Amos Gropp, Matan Atzmon, and Yaron Lipman. Isometric autoencoders. *arXiv preprint arXiv:2006.09289*, 2020.

[14] Cheongjae Jang, Yung-Kyun Noh, and Frank Chongwoo Park. A riemannian geometric framework for manifold learning of non-euclidean data. *Advances in Data Analysis and Classification*, 15(3):673–699, 2021.

[15] Dietmar A. Salamon Joel W. Robbin. *Introduction to differential geometry*. Springer Studium Mathematik, 2021.

[16] Miroslav Kubat and Kubat. *An introduction to machine learning*, volume 2. Springer, 2017.

[17] Jeffrey M. Lee. *Manifolds and differential geometry*. 1956.

[18] James McQueen, Marina Meila, and Dominique Joncas. Nearly isometric embedding by relaxation. *Advances in Neural Information Processing Systems*, 29, 2016.

[19] Sebastian Mika, Bernhard Schölkopf, Alex Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. *Advances in neural information processing systems*, 11, 1998.

[20] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[21] John Nash. C1 isometric imbeddings. *Annals of mathematics*, pages 383–396, 1954.

[22] John Nash. The imbedding problem for riemannian manifolds. *Annals of mathematics*, pages 20–63, 1956.

[23] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[24] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[25] Dominique Perrault-Joncas and Marina Meila. Metric learning and manifolds: Preserving the intrinsic geometry. *Preprint Department of Statistics, University of Washington*, 2012.

[26] Erez Peterfreund, Ofir Lindenbaum, Felix Dietrich, Tom Bertalan, Matan Gavish, Ioannis G Kevrekidis, and Ronald R Coifman. Local conformal autoencoder for standardized data coordinates. *Proceedings of the National Academy of Sciences*, 117(49):30918–30927, 2020.

[27] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[28] Stefan Sommer, Tom Fletcher, and Xavier Pennec. *Introduction to differential and Riemannian geometry*. Elsevier, 2020.

[29] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.