

# Representation Learning via Manifold Flattening and Reconstruction

**Michael Psenka**

*Department of Electrical Engineering and Computer Science  
University of California, Berkeley  
Berkeley, CA 94720-1776, USA*

PSENKA@EECS.BERKELEY.EDU

**Druv Pai**

*Department of Electrical Engineering and Computer Science  
University of California, Berkeley  
Berkeley, CA 94720-1776, USA*

DRUVPAI@BERKELEY.EDU

**Vishal Raman**

*Department of Electrical Engineering and Computer Science  
University of California, Berkeley  
Berkeley, CA 94720-1776, USA*

VRAMAN@BERKELEY.EDU

**Shankar Sastry**

*Department of Electrical Engineering and Computer Science  
University of California, Berkeley  
Berkeley, CA 94720-1776, USA*

SASTRY@COE.BERKELEY.EDU

**Yi Ma**

*Department of Electrical Engineering and Computer Science  
University of California, Berkeley  
Berkeley, CA 94720-1776, USA*

YIMA@EECS.BERKELEY.EDU

**Editor:** Sayan Mukherjee

## Abstract

A common assumption for real-world, learnable data is its possession of some low-dimensional structure, and one way to formalize this structure is through the *manifold hypothesis*: that learnable data lies near some low-dimensional manifold. Deep learning architectures often have a compressive autoencoder component, where data is mapped to a lower-dimensional latent space, but often many architecture design choices are done by hand, since such models do not inherently exploit mathematical structure of the data. To utilize this geometric data structure, we propose an iterative process in the style of a geometric flow for explicitly constructing a pair of neural networks layer-wise that linearize and reconstruct an embedded submanifold, from finite samples of this manifold. Our such-generated neural networks, called *Flattening Networks* (FlatNet), are theoretically interpretable, computationally feasible at scale, and generalize well to test data, a balance not typically found in manifold-based learning methods. We present empirical results and comparisons to other models on synthetic high-dimensional manifold data and 2D image data. Our code is publicly available.

**Keywords:** Unsupervised learning, autoencoders, manifold learning, geometry, deep learning

## 1. Introduction

*Autoencoding* (Kramer, 1991) remains an important framework in deep learning and representation learning (Bengio et al., 2013), where one aims to both encode data into a more compact, lower-dimensional representation and then decode it back into the original data format. This framework

is often used on top of deep learning applications to ease downstream learning and processing (Rombach et al., 2022; Turian et al., 2010; He et al., 2022; Hafner et al., 2019) as well as a design principle for the network itself (Ronneberger et al., 2015; Devlin et al., 2018).

However, these models are typically so-called “black-box models”: many important parameters of the model cannot be directly optimized or chosen from mathematical principle, and instead need to be chosen from either extensive trial-and-error or from previous knowledge likewise obtained from trial-and-error. When one is building autoencoder models in a novel domain, this issue can hinder the speed and ease of development.

To resolve this issue, we design a framework for automatically building up an autoencoder model given a *geometric* model of the dataset; namely, we assume that the “manifold hypothesis” (Vidal et al., 2005; Fefferman et al., 2016; Brown et al., 2022; Yu et al., 2020) holds for the dataset we wish to encode/decode. Given this assumption, along with some needed regularity and sampling conditions, we can greedily construct the layers of an encoder/decoder pair in a forward fashion, similarly in spirit to recent works (Chan et al., 2022; Hinton, 2022), such that the learned encoding is as low-dimensional as possible, in as few layers as possible. In this manner, the architecture is automatically constructed to be optimal. While there are a few scalar parameters that need pre-specification by the network designer, these are all mathematically interpretable and in practice not sensitive for training results.

Our approach to this problem is through *manifold flattening*, or *manifold linearization*: if we can globally deform the “data manifold” into a linear structure, then we can apply more interpretable linear models on top of these constructed features, such as PCA for compression. Our motivation for focusing on manifold linearization comes from common practice: any successfully trained deep learning model whose last layer is linear (e.g. multi-layer perceptrons) has “linearized” their problem, since the deep model has implicitly learned a map (all but the last layer) such that the target problem is solvable by a linear map (the last layer).

In particular, our approach to this linearization process and its respective reconstruction method focuses on the *extrinsic curvature* of the data manifold. By iteratively constructing maps that model local curvature, the methodology works to flatten out and reconstruct the manifold, and these local flattenings are constructed in such a way that the aggregate eventually converges to a global flattening, akin to classically-studied *geometric flows*. This approach enables a deeper understanding and manipulation of the data’s second order geometric properties, facilitating a more accurate and efficient representation learning process.

Many deep learning models, such as multi-layer perceptrons with ReLU activations, are restricted to piecewise linear models, and thus primarily engage with first-order geometric properties of the data. Recent theoretical developments have shed light on the nature and capabilities of such piecewise linear, or first-order, models (Yu et al., 2024), but it still remains an open question to study the higher-order geometric effects of deep learning models. While our work does not study existing deep learning techniques directly, its natural development of an MLP-like model sheds insight on how one could develop a deep learning model to explicitly model extrinsic curvature.

The core of this paper will be to construct a manifold flattening algorithm that constructs a dense manifold flattening and reconstruction given a finite set of points (the data set). Through this flattening process arises a MLP-like deep learning architecture, which we call *flattening networks* (FlatNet), that is able to be constructed layer-by-layer rather than trained end-to-end. We begin by establishing the mathematical setting and desiderata of our work, and establish the tight connection between manifold flattening and minimal representation learning. We then establish a proposed discrete geometric flow, where we use local curvature models to iteratively and injectively push the manifold into its convex hull, eventually resulting in a global flattening, and show its constructibility from finite samples and scalability to high dimensions. We then provide some theoretical analysis

on the convergence of these “convexification flows,” and finally provide experimental results on both synthetic, randomly generated manifold data as well as low-dimensional 2D image data.

### 1.1 Problem formulation

We now give a mathematical formulation for the autoencoding problem that we want to solve, and define notation used for the rest of the paper. We assume foundational knowledge of differential geometry; for an introduction and review, see Appendix B.

**Fundamental problem.** Consider a *dataset*  $\mathcal{X} = \{x_1, \dots, x_N\} \subseteq \mathbb{R}^D$ . We make the *geometric* assumption that our data  $x_i$  are drawn near a low-dimensional embedded sub-manifold  $\mathcal{M} \subset \mathbb{R}^D$  which has intrinsic dimension  $d \leq D$ ; we say that  $\mathcal{M}$  is the *data manifold*. We aim to construct an *encoder*  $f: \mathbb{R}^D \rightarrow \mathbb{R}^p$ , and a *decoder*  $g: \mathbb{R}^p \rightarrow \mathbb{R}^D$ , which are a *minimal autoencoding pair* in the following sense.

**Definition 1.** *Given an embedded submanifold  $\mathcal{M} \subset \mathbb{R}^D$  and a scalar parameter  $\varepsilon > 0$ , we say that a pair of continuous functions  $f: \mathbb{R}^D \rightarrow \mathbb{R}^p$  and  $g: \mathbb{R}^p \rightarrow \mathbb{R}^D$  are an  $\varepsilon$ -minimal autoencoding pair for  $\mathcal{M}$  if both of the following conditions hold:*

- (a) *(Autoencoding.)  $f$  is injective on  $\mathcal{M}$ , and for all  $x \in \mathcal{M}$  we have  $\|g(f(x)) - x\|_2 \leq \varepsilon$ , and*
- (b) *(Minimality.)  $p$  is the smallest integer such that there exist continuous  $f, g$  such that (a) holds.*

**Computational considerations.** We mainly concern ourselves with computational time, and the scalability requirements often needed for modern data requirements. Ideally, our autoencoder algorithm follows the following scaling laws:

1. *Linear time in the number of samples  $N$ ,  $O(N)$ .* Datasets have become incredibly large, easily reaching the millions for many modern tasks, so even quadratic time in  $N$  quickly becomes uncomputable in practice.
2. *Linear time in the ambient dimension  $D$ ,  $O(D)$ .* Similarly with the number of samples, the ambient dimension in many cases can easily reach the thousands or millions, so anything above linear time becomes practically uncomputable.
3. *Polynomial time in the intrinsic dimension  $d$ ,  $O(d^k)$  for some  $k \in \mathbb{N}$ .* In contrast to the sample size and ambient dimension, we expect the intrinsic dimension of data manifolds to be relatively small (Wright and Ma, 2022); for example, MNIST (Deng, 2012) has ambient dimension  $D = 784$ , but the intrinsic dimension of a single class of digits is estimated to be around  $12 \leq d \leq 14$  (Hein and Audibert, 2005; Costa and Hero, 2006; Facco et al., 2017). Still, manifold learning in generality requires exponential time and samples in the intrinsic dimension,  $O(c^d)$ , which is infeasible for even relatively low  $d$ .

**Assumptions on the data manifold  $\mathcal{M}$ .** An important area of interest is to determine the correct assumptions for the kinds of manifolds often found in real world data. For this paper, our assumptions are relatively minimal, but it is important to be clear about them. For more information about the motivation of the following assumptions, see Theorem 6 and its proof in Appendix D.

1. We assume  $\mathcal{M}$  is smooth. This is a standard regularity assumption.

2. We assume  $\mathcal{M}$  is compact. This assumption is reasonable in that it is fulfilled for closed and bounded sub-manifolds of Euclidean space, which we encounter in many application contexts. For example, the manifold of  $28 \times 28$  greyscale images as embedded in  $\mathbb{R}^{784}$  (where  $784 = 28 \cdot 28$ ) is bounded in  $[0, 1]^{784}$ . Adding closedness simply makes analysis easier.
3. We assume  $\mathcal{M}$  is connected. In spirit, we focus on a single class of data in this work. While it is an important future direction to extend to multi-class settings, we focus on the single-class setting in this work to emphasize the role of compression.
4. We assume  $\mathcal{M}$  is flattenable. Due to requiring differential geometry concepts to define, flattenability is more rigorously defined later in the work, in Theorem 5. For now, this is the most restrictive assumption of the four, but there is still reason to believe this assumption is commonly satisfied for commonly encountered data manifolds in neural network-based applications; see the end of Section 3 for more discussion about this point.

**Assumptions on the dataset  $\mathcal{X}$ .** The main assumption we make is a *geometric* property which quantifies how much the data and the manifold are representative of each other.

**Definition 2.** *Given an embedded submanifold  $\mathcal{M} \subset \mathbb{R}^D$ , a finite dataset  $\mathcal{X} = \{x_1, \dots, x_N\} \subseteq \mathbb{R}^D$ , and scalar parameters  $\varepsilon, \delta > 0$ , we say that is  $(\varepsilon, \delta)$ -faithful to  $\mathcal{M}$  if both of the following hold:*

1. ( $\mathcal{M}$  represents  $\mathcal{X}$ .)  $\inf_{x \in \mathcal{M}} \|x - x_i\|_2 \leq \varepsilon$  for all  $i \in [N]$ .
2. ( $\mathcal{X}$  represents  $\mathcal{M}$ .)  $\min_{i \in [N]} \|x - x_i\|_2 \leq \delta$  for all  $x \in \mathcal{M}$ .

*One may re-phrase these conditions as saying that  $\mathcal{M}$  is an  $\varepsilon$ -cover for  $\mathcal{X}$  and  $\mathcal{X}$  is a  $\delta$ -cover for  $\mathcal{M}$ .*

Because  $\mathcal{M}$  is compact, for every  $\varepsilon, \delta > 0$ , there exists a finite  $(\varepsilon, \delta)$ -faithful dataset for  $\mathcal{M}$ .

We do not make explicit statistical assumptions of our data, such as the assumption that the data  $x_i$  are drawn i.i.d. from a distribution on  $\mathbb{R}^D$ . This distinguishes our work from other autoencoder analysis and, more generally, much of statistical learning theory.

**Putting it all together.** Finally, we lay down the defining properties of our desired algorithm for efficient computation of a minimal autoencoding pair. Note that we drop parameters for brevity.

**Definition 3.** *Given an embedded submanifold  $\mathcal{M} \subset \mathbb{R}^D$  of intrinsic dimension  $d$  and an faithful dataset  $\mathcal{X} \subset \mathbb{R}^D$ , an algorithm is said to efficiently compute a minimal autoencoding pair for  $\mathcal{M}$  if it computes a minimal autoencoding pair in  $O(DNd^k)$  flops for some  $k \in \mathbb{N}$ , with access only to  $\mathcal{X}$  (and not  $\mathcal{M}$  or  $d$ ).*

## 2. Related work

We now provide a brief overview of works in this domain. The first two, VAEs and manifold learning, are the most direct alternative algorithms to solve the problem given in Theorem 3. However, each of these have fundamental limitations, which we address with our work. We also include methods that match our problem and algorithm in spirit, but in reality address fundamentally different problems.

## 2.1 Variational autoencoders

Arguably the most well-known class of autoencoders at the time of writing is the class of *variational autoencoders* (VAEs) (Kingma and Welling, 2013), which have seen much empirical success, both traditionally in computer vision problems (Higgins et al., 2017; Van Den Oord et al., 2017; Kim and Mnih, 2018), and recently as a black-box method for nonlinear dimensionality reduction within the framework of within so-called *diffusion* models (Vahdat et al., 2021; Rombach et al., 2022). Despite their empirical success, VAEs have a few endemic shortcomings, including posterior collapse (Lucas et al., 2019). However, the most common issue, obeyed by all flavors of VAE thus far to our knowledge, is that the encoder and decoder networks are *black-box* neural networks, about which comparatively little is understood. Important design choices, including hyperparameter selection, thus are either made through extensive trial-and-error or using a history of trial-and-error for the problem (e.g., 2D image autoencoding), rendering application of VAEs in a novel problem a potentially expensive challenge (Rezende and Viola, 2018). In this work, we aim to provide an alternative *white-box* model which ameliorates and linearizes the data geometry, essentially performing nonlinear dimensionality reduction via transforming the data structure to a lower-dimensional affine subspace, all with fewer hyperparameters and greater robustness to hyperparameter changes than the alternatives.

## 2.2 Manifold learning

Manifold learning methods (Izenman; Zheng et al.) and intrinsic manifold learning methods (Schwartz and Talmon; Lin and Zha; Qiao et al.; Zhang et al.) are a well-known class of algorithms that seek to find low dimensional representations of originally high dimensional data while preserving geometric structures. This includes methods that find subspaces by preserving local structure, such as local linear embedding (LLE) (Roweis and Saul, 2000) and its variants, local tangent space alignment (LTSA) (Zhang and Zha, 2003), t-distributed stochastic neighbor embedding (t-SNE) (Van der Maaten and Hinton, 2008), Laplacian eigenmaps (LE) (Belkin and Niyogi, 2003) etc. and methods that try to preserve global structures such as isometric mapping (ISOMAP) (Tenenbaum et al., 2000) and uniform manifold approximation and projection (UMAP) (McInnes et al., 2018). Although these have offered useful results on a broad class of manifolds, there are limitations when applying these to real-world data (Li et al., 2019), which we discuss now and aim to mitigate through our algorithm.

The first limitation is that for many the methods presented above, there is no explicit projection map between the original data and the corresponding low dimensional representation. This means that in order to find a projection for a point in the original space outside of the training data, the entire method needs to be repeated on the original dataset with the new point appended. This additional computational cost is especially problematic in the case of large-scale datasets. One possible extension maps the original data into a reproducing kernel Hilbert space (RKHS) using kernel functions (Xu et al., 2008), avoiding this “out-of-sample” problem (Li et al., 2008). Although this is robust, it requires good kernel function selection and parameter selection, which can be challenging. Another approach which has been explored is parameterizing the features through a neural network architecture (Jansen et al., 2017; Schmidt et al., 1992; Chen, 1996). This presents similar training difficulties to VAEs due to the black-box structure of the corresponding architectures.

Another potential limitation is that for all of the aforementioned methods, it is required to specify the intrinsic dimension of the data as an input. In practice, this is hardly ever known a priori, though there are a large class of intrinsic dimension estimators (Campadelli et al., 2015;

Levina and Bickel, 2004b; Carter et al., 2009) that can be used at an additional computational cost.

### 2.3 Distribution learners

Three forms of models for distribution learning in recent years have been GANs (Goodfellow et al., 2014), normalizing flows (Kobyzev et al., 2019), and diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song and Ermon, 2019). While such models are effective at learning the distribution of the data (Arora et al., 2018) and are able to sample from it (Chen et al., 2022), they do not learn *representations* or *features* for individual data points. Thus, they solve different problems than the one we choose to solve in this work, as per our discussion in Section 1.1.

## 3. Manifold flattening for data manifolds

We now introduce our high-level approach for finding an efficient algorithm which generates a minimal autoencoding pair: *manifold flattening*.

As a warmup, let us suppose that  $\mathcal{M}$  were actually an affine subspace, with no nonlinearity or curvature. Then principal component analysis (PCA) would give an efficient autoencoding; we could estimate the dimension using the subspace structure in any of a variety of ways, then the encoder  $f: \mathbb{R}^D \rightarrow \mathbb{R}^d$  would be the orthogonal projection onto the first  $d$  principal components of the data, and the decoder  $g: \mathbb{R}^d \rightarrow \mathbb{R}^D$  would be the adjoint map of  $f$ .

Motivated by the ease of the problem when the data lies on an affine subspace, our high-level approach is to first remove the nonlinearities in the data manifold  $\mathcal{M}$ , and then use the previously-discussed PCA-type algorithms once we have “flattened”  $\mathcal{M}$ .

In this section, we formalize the notion of flattening a manifold and the equivalence between manifold flattening and minimal autoencoding as described in Theorem 1. For all definitions we use for geometric objects, such as the tangent space, please see Appendix B.

### 3.1 Manifold flattening creates minimal representations

We first provide a classical definition of flatness for a manifold  $\mathcal{M} \subset \mathbb{R}^D$  using the second fundamental form (see Theorem 11 in Appendix B).

**Definition 4.** Let  $\mathcal{M} \subset \mathbb{R}^D$  be a smooth embedded submanifold of dimension  $d$  and second fundamental form  $\mathbb{I}$ . We say that  $\mathcal{M}$  is flat if and only if  $\mathbb{I}_{x_0}(u, v) = 0$  for all  $x_0 \in \mathcal{M}$  and  $u, v \in T_{x_0}\mathcal{M}$ .

This definition says that a manifold is flat if it has no extrinsic curvature. We are not restricted to only studying the extrinsic curvature of the original data manifold  $\mathcal{M}$ , but also of its image through a smooth map:  $\phi(\mathcal{M}) \doteq \{\phi(x) \mid x \in \mathcal{M}\}$ , where  $\phi: \mathbb{R}^D \rightarrow \mathbb{R}^D$ . Ideally, then, the extrinsic curvature gives a measure of when a manipulation  $\phi$  has successfully removed the nonlinearity of  $\mathcal{M}$ , and one can thus safely use PCA on top of  $\phi$  to generate a minimal autoencoding. We formalize this idea in the following definition and theorem.

**Definition 5.** Let  $\mathcal{M} \subset \mathbb{R}^D$  be a compact, connected, smooth embedded submanifold of dimension  $d$ . We say that  $\mathcal{M}$  is flattenable if there exist smooth maps  $\phi, \rho: \mathbb{R}^D \rightarrow \mathbb{R}^D$  such that:

1.  $\rho(\phi(\mathcal{M})) = \mathcal{M}$ , and
2.  $\phi(\mathcal{M})$  is flat.

In this case, we also say that  $(\phi, \rho)$  flatten and reconstruct  $\mathcal{M}$ .

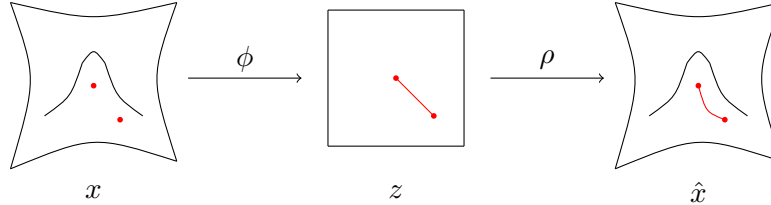


Figure 1: A depiction of interpolation through manifold flattening on a manifold in  $\mathbb{R}^3$  of dimension  $d = 2$ . To interpolate two points on the data manifold, map them through the flattening map  $\phi$  to the flattened space, take their convex interpolants, and then map them back to the data manifold through the reconstruction map  $\rho$ .

**Theorem 6.** *Let  $\mathcal{M} \subset \mathbb{R}^D$  be a flattenable submanifold of dimension  $d$  which is flattened and reconstructed by the maps  $\phi, \rho: \mathbb{R}^D \rightarrow \mathbb{R}^D$ . Then we have:*

1.  $\phi(\mathcal{M})$  is a convex set.
2.  $\phi(\mathcal{M})$  is contained within a unique affine subspace of dimension  $d$ .
3. Let  $U \in \mathbb{R}^{D \times d}$  be an orthonormal basis for the aforementioned subspace, stacked into a matrix, and  $z_0 \in \phi(\mathcal{M})$ . The encoder  $f$  given by  $f(x) = U^\top(\phi(x) - z_0)$  and decoder  $g$  given by  $g(z) = \rho(Uz + z_0)$  form a minimal autoencoding pair satisfying Theorem 1 for  $\varepsilon = 0$ . If  $\phi, \rho$  only reconstruct approximately, in the sense that  $\max_{x \in \mathcal{M}} \|\rho(\phi(x)) - x\|_2 \leq \epsilon$ , then  $f, g$  form a minimal autoencoding pair for the same error  $\epsilon$ .

Furthermore, if  $\phi, \rho$  admit some error  $\epsilon$  (e.g.  $\max_{x \in \mathcal{M}} \|\rho(\phi(x)) - x\|_2^2 \leq \epsilon$ ), then the claims of Theorem 6 still hold for the same  $\epsilon$ . As the proof relies on some outside differential geometry, we leave the proof for Appendix D. The main message for this theorem is a formalization of our previous intuition: *finding flattening and reconstruction maps is equivalent to finding a minimal autoencoding*. Thus, in the rest of the work, we focus on constructing flattening and reconstruction maps  $\phi$  and  $\rho$ .

A first visualization of manifold flattening and reconstruction is depicted in Figure 1, along with the primary benefit outside of Theorem 3 for manifold flattening: nonlinear interpolation.

### 3.2 A note on the flattenable assumption for $\mathcal{M}$

Under the assumption that  $\mathcal{M}$  is flattenable, the converse of Theorem 6 also holds: any lossless, smooth autoencoding pair generates a flattening. We argue that this is a reasonable assumption for data manifolds, as for any continuous autoencoding pair  $f, g$  (e.g. neural network encoder/decoder pair), exactly one of the following must occur:

1.  $f(\mathcal{M})$  is a convex set, in which case  $\mathcal{M}$  is flattenable; or
2.  $f(\mathcal{M})$  is not convex, in which case samples generated via  $g(z)$  for any normally distributed  $z$  will fall outside  $\mathcal{M}$  with high probability.

This sampling method is a common technique within the VAE framework, so manifolds which are amenable to autoencoding via VAEs should generally be flattenable.

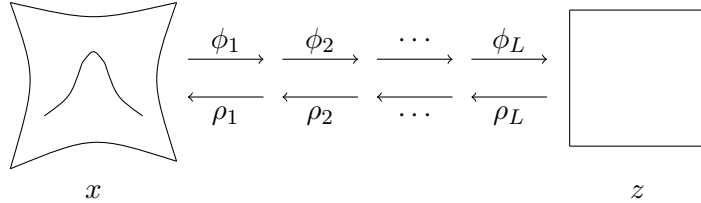


Figure 2: A depiction of the construction process of the flattening and reconstruction pair  $(\phi_{\text{net}}, \rho_{\text{net}})$ , where the encoder  $\phi_{\text{net}} = \phi_L \circ \phi_{L-1} \circ \dots \circ \phi_1$  is constructed from composing flattening layers, and the decoder  $\rho_{\text{net}} = \rho_1 \circ \rho_2 \circ \dots \circ \rho_L$  is composed of inversions of each  $\phi_\ell$ .

#### 4. Flattening Networks: an algorithmic approach to manifold flattening

In this section we discuss a computational method for constructing globally-defined flattening and reconstruction maps. In our method, such maps will be a composition of multiple simpler functions, akin to layers in neural networks, as depicted in Figure 2. Accordingly, we call our method the *Flattening Networks*, or *FlatNet*.

However, a large difference between Flattening Networks and traditional neural networks is that our “layers” are constructed iteratively in a white-box process with no back-propagation. This design is qualitatively similar to other white-box networks such as described in Chan et al. (2022), but is very different in the details.

We summarize the overall construction of our network now, and then give a more detailed description and motivation in the following sections.

1. Initialize composite flattening and reconstruction maps  $\phi_{\text{net}} = \rho_{\text{net}} = \text{id}_{\mathbb{R}^D} : \mathbb{R}^D \rightarrow \mathbb{R}^D$ .
2. For each layer  $\ell \in [L]$ :
  - (a) Sample a point  $x_0$  uniformly at random from  $\mathcal{X}$ .
  - (b) Fit a local quadratic model to a neighborhood  $N_0$  of  $x_0$ , and compute a local flattening map  $\phi_{\text{loc}} : \mathbb{R}^D \rightarrow \mathbb{R}^D$ .
  - (c) Use a so-called *partition of unity* and the local flattening map  $\phi_{\text{loc}}$  to form a global flattening map  $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^D$ .
  - (d) Compute an analytic global approximate inverse, i.e., a global reconstruction map  $\rho : \mathbb{R}^D \rightarrow \mathbb{R}^D$  such that  $\rho \circ \phi \approx \text{id}_{\mathbb{R}^D}$ .
  - (e) Compose the constructed layer with existing layers: set  $\phi_{\text{net}} \leftarrow \phi \circ \phi_{\text{net}}$  and  $\rho_{\text{net}} \leftarrow \rho_{\text{net}} \circ \rho$ .
3. Return the composite maps  $\phi_{\text{net}}$  and  $\rho_{\text{net}}$ .

In the following sections, we will discuss the motivation and implementation of each of these steps.

##### 4.1 Local quadratic models for data manifolds

Since we aim to manipulate and reconstruct a dense manifold  $\mathcal{M}$  from finite samples  $\mathcal{X}$ , we need some way to model manifolds to fit to the dataset  $\mathcal{X}$ . Commonly, the structures of manifolds



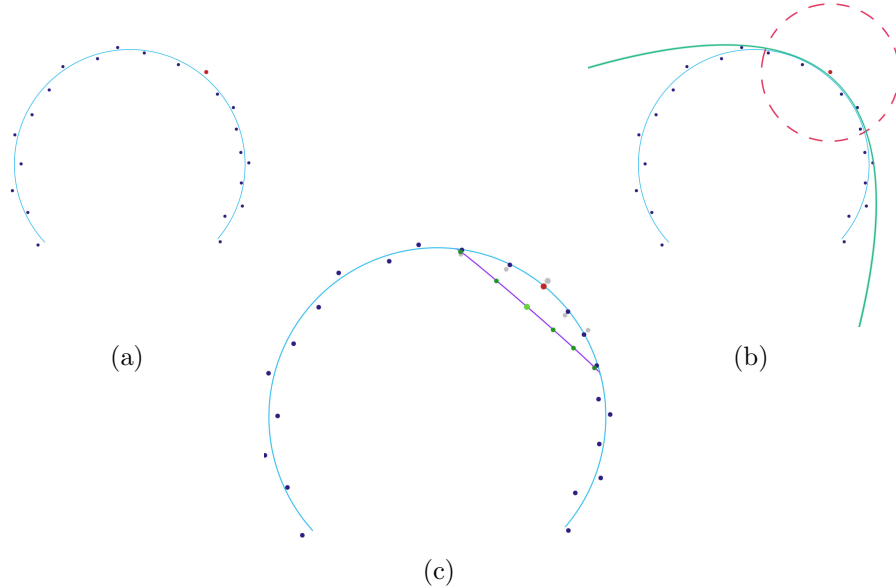


Figure 3: A graphical depiction of one step for our manifold flattening and reconstruction algorithm. **(a)** From a dataset  $\mathcal{X}$  (dots) drawn near a manifold, randomly choose one data point  $x_0$  (red). **(b)** We then find the largest radius around the selected point where our simplified model (green) fits the data well, and **(c)** We use this model to both flatten this patch of the data (purple) and reconstruct onto our local model. This process is repeated on the resulting manifold until the manifold is flattened.

exploited for manifold learning are local. Even beyond the classical “locally Euclidean” definition for manifolds, many manifold processing methods use the fact that, near any fixed point  $x_0 \in \mathcal{M}$ , the manifold  $\mathcal{M}$  will be approximately linear for a small enough ball around  $x_0$ .

While linear models will not be sufficient to solve our problem, as we need to reconstruct the nonlinearities that our method flattens out, we still build our method from the above core principle: *while a data manifold  $\mathcal{M}$  may be complicated and hard to model globally, restricting our attention to one neighborhood at a time allows us to fit a much simpler model accurately to that patch of the manifold.* Such a local model not only tells us how to reconstruct the original manifold after flattening, but also the “best” way to flatten the manifold. Concretely, this allows us to construct a map  $\phi_{\text{loc}}: \mathbb{R}^D \rightarrow \mathbb{R}^D$  which flattens points in a chosen neighborhood  $N_0 \subset \mathbb{R}^D$  around  $x_0$ , but whose behavior is necessarily ill-defined far from  $N_0$ . We now discuss the construction of such a map.

The *exponential map* at  $x_0$ , denoted  $\exp_{x_0}: T_{x_0}\mathcal{M} \rightarrow \mathcal{M}$ , is a local bijection from the  $d$ -dimensional subspace  $T_{x_0}\mathcal{M}$  to the full manifold  $\mathcal{M}$ . Hence,  $\exp_{x_0}$  perfectly models the local structure of the manifold  $\mathcal{M}$ . However, not only is  $\exp_{x_0}$  hard to estimate from finite data, the exponential map is typically only analytically known for very structured manifolds. However, we can estimate approximations to  $\exp_{x_0}$ , and restrict our attention to neighborhoods small enough such that this approximation holds. A well-studied approximation scheme are Taylor approximations. A linear approximation is too restrictive, as this removes all local nonlinear information about  $\mathcal{M}$ , thus making it impossible to reconstruct after flattening. Thus, we use the next simplest Taylor

series model, the second-order Taylor approximation (Monera et al., 2014):

$$\exp_{x_0}(v) \approx x_0 + v + \frac{1}{2}\mathbb{I}_{x_0}(v, v), \quad (1)$$

where  $\mathbb{I}_{x_0}: \mathbb{T}_{x_0}\mathcal{M} \times \mathbb{T}_{x_0}\mathcal{M} \rightarrow \mathbb{N}_{x_0}\mathcal{M}$  is the second fundamental form of  $\mathcal{M}$  at  $x_0$  (see Theorem 11). Based on this expansion we will construct principled local flattening and reconstruction maps  $\phi_{\text{loc}}$  and  $\rho_{\text{loc}}$ .

## 4.2 Proposed local flattening and reconstruction maps

We propose a local flattening map  $\phi_{\text{loc}}: \mathbb{R}^D \rightarrow \mathbb{R}^D$  which is an affine projection operator:

$$\phi_{\text{loc}}(x) = \mathcal{P}_{S+x_c}\{x\}, \quad (2)$$

where  $S$  is a linear subspace and  $x_c \in \mathbb{R}^D$  is a fixed base point (both to be defined next), and the sum is taken in the usual sense, i.e.,  $S + x_c = \{s + x_c \mid s \in S\}$ . There are then two variables we need to decide: the subspace  $S$  and the offset  $x_c$ . The choices we make here are not claimed to be optimal in any sense, but their design serves two important roles:  $S$  is chosen to maintain local invertibility of  $\phi_{\text{loc}}$ , and  $x_c$  is chosen to make the overall flattening process more likely to converge.

First, we choose  $S = \mathbb{T}_{x_0}\mathcal{M}$ . If the approximation in (1) holds, then for any  $v \in \mathbb{T}_{x_0}\mathcal{M}$  we have

$$\begin{aligned} \phi_{\text{loc}}(\exp_{x_0}(v)) &= \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}+x_c}\{\exp_{x_0}(v)\}, \\ &\approx \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}+x_c}\left\{x_0 + v + \frac{1}{2}\mathbb{I}_{x_0}(v, v)\right\}, \\ &= \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\left\{x_0 + v + \frac{1}{2}\mathbb{I}_{x_0}(v, v) - x_c\right\} + x_c, \\ &= \underbrace{\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{v\}}_{=v} + \frac{1}{2}\underbrace{\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{\mathbb{I}_{x_0}(v, v)\}}_{=0} + \underbrace{\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x_0 - x_c\}}_{=\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}+x_c}\{x_0\}} + x_c, \\ &= v + \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}+x_c}\{x_0\}, \\ &= v + \phi_{\text{loc}}(x_0). \end{aligned} \quad (3)$$

Here  $\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{v\} = v$  because  $v \in \mathbb{T}_{x_0}\mathcal{M}$  and  $\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{\mathbb{I}_{x_0}(v, v)\} = 0$  because  $\mathbb{I}_{x_0}(v, v) \in \mathbb{N}_{x_0}\mathcal{M}$ . Thus the local flattening map

$$\phi_{\text{loc}}(x) = \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}+x_c}\{x\} \quad (4)$$

is (approximately) invertible in the neighborhood  $N_0$  of  $x_0$  for which the quadratic approximation in (1) holds, say by the local reconstruction map

$$\rho_{\text{loc}}(z) = x_0 + z - \phi_{\text{loc}}(x_0) + \frac{1}{2}\mathbb{I}_{x_0}(z - \phi_{\text{loc}}(x_0), z - \phi_{\text{loc}}(x_0)). \quad (5)$$

This local invertibility property follows from the below calculation:

$$\begin{aligned} \rho_{\text{loc}}(\phi_{\text{loc}}(\exp_{x_0}(v))) &\approx \rho_{\text{loc}}(v + \phi_{\text{loc}}(x_0)), \\ &= x_0 + v + \frac{1}{2}\mathbb{I}_{x_0}(v, v), \\ &\approx \exp_{x_0}(v). \end{aligned} \quad (6)$$

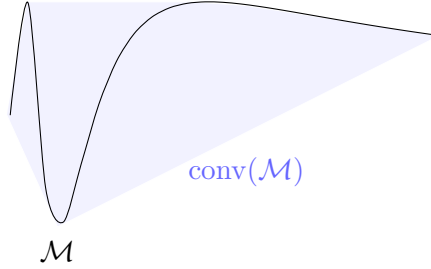


Figure 4: A depiction of the convex hull of a manifold  $\mathcal{M}$  versus the manifold itself. If there is no difference between  $\text{conv}(\mathcal{M})$  and  $\mathcal{M}$  itself, the resulting manifold is convex and thus flattened. We thus design local flattenings to push  $\mathcal{M}$  into its convex hull.

This choosing of the subspace  $S = T_{x_0}\mathcal{M}$  ensures that the local flattening map is (approximately) invertible in the local neighborhood  $N_0$  of  $x_0$  for which the quadratic approximation in (1) holds. While the choice of  $S$  as the tangent space is clearly optimal in preserving the metric at  $x_0$ , the intrinsic volume within the neighborhood can compress at arbitrary ratios depending on the curvature at  $x_0$ . Nonetheless, flattening to the tangent space gives an easy verification that the flattening is invertible.

#### 4.2.1 CHOOSING $x_c$

The offset  $x_c$  induced in (3) is an important design choice of the flattening process, as it directs the global direction of the process induced from repeated local flattenings, and thus controls the convergence of the process. In order to make sure the process converges, we need to design an  $x_c$  that is both (a) pushing the output manifold closer to a converged, globally flattened state, but (b) close enough to our dataset  $\mathcal{X}$  to make the eventual global flattening map  $h$  reasonably smooth.

Our certificate of convergence is the *convex hull* of the manifold. We use the connection between the flatness and convexity of  $\mathcal{M}$  furnished by Theorem 6. Indeed, such an embedded submanifold of Euclidean space is fully flat if and only if it is convex. Thus, any difference between the convex hull of  $\mathcal{M}$ , denoted  $\text{conv}(\mathcal{M})$ , and itself, gives an indication of where to flatten the manifold into. This idea is depicted in Figure 4.

Note that for any probability measure  $P$  supported on  $\mathcal{M}$ , we have that the extrinsic average

$$\bar{x} \doteq \int_{\mathcal{M}} x \, dP(x) \quad (7)$$

lies within the convex hull. Thus, if we chose our distribution  $P$  to have density w.r.t. the Haar measure on  $\mathcal{M}$  be proportional to a smooth function  $\pi: \mathbb{R}^D \rightarrow [0, 1]$  such that  $\pi(x) \approx 1$  for all  $x \in N_0$  and  $\pi$  decays rapidly outside  $N_0$ , to be determined in the next section, then the weighted local average given by the following:

$$x_c = \bar{x} \doteq \frac{\int_{\mathcal{M}} x \pi(x) \, dx}{\int_{\mathcal{M}} \pi(x) \, dx}, \quad (8)$$

lies in the convex hull,  $x_c \in \text{conv}(\mathcal{M})$ , while  $x_c$  is still relatively close to  $x$  due to the imposed structure on  $\pi$ .

Thus, to meet the desired criteria for the local flattening, we choose  $S$  to be the tangent space  $T_{x_0}\mathcal{M}$  and the offset  $x_c$  to be the local average  $x_c = \frac{\int_{\mathcal{M}} x \pi(x) \, dx}{\int_{\mathcal{M}} \pi(x) \, dx}$ .

### 4.3 From local to global

Once we find a local flattening map  $\phi_{\text{loc}}: \mathbb{R}^D \rightarrow \mathbb{R}^D$ , the next step to make it globally well-defined is to “glue” it with a well-behaved globally-defined map outside of the chosen neighborhood  $N_0$  of  $x_0$ . Classically in differential geometry, this is done through a *partition of unity* (Dieudonné, 1937; Lee, 2012), a smooth scalar function  $\psi: \mathbb{R}^D \rightarrow [0, 1]$  such that  $\psi(x) \approx 1$  for all  $x \in A$  for some closed and arbitrarily close  $A \subsetneq N_0$  and  $\psi(x) \approx 0$  for all  $x \notin N_0$ . Given the local flattening map  $\phi_{\text{loc}}: \mathbb{R}^D \rightarrow \mathbb{R}^D$  computed with respect to the neighborhood  $N_0$ , we can define a globally-defined flattening map  $\phi: \mathbb{R}^D \rightarrow \mathbb{R}^D$  by:

$$\phi(x) \doteq \psi(x)\phi_{\text{loc}}(x) + (1 - \psi(x))x, \quad (9)$$

so that  $\phi(x) \approx \phi_{\text{loc}}(x)$  if  $x \in N_0$  and  $\phi(x) \approx x$  otherwise.

Of course, there are many choices for  $\psi$ . However, as we design a practical algorithm satisfying Theorem 3, our choices become more limited. The following are our two main concerns that will limit the design choice for  $\psi$ :

1.  $\psi$  should lead to relatively smooth maps constructed as in (9). Once we flatten a portion of the manifold, we will need to both further flatten and reconstruct the manifold, and if we create cusps as in Figure 3c, the downstream flattening and reconstruction tasks will be unnecessarily difficult.
2.  $\psi$  should allow for computable inverses of maps constructed as in (9). Even if we are able to invert the local flattening function  $\phi_{\text{loc}}$  locally around  $x_0$  using the local reconstruction function  $\rho_{\text{loc}}$ , it’s not always trivial (or even possible) to invert the globally well-defined function  $\phi$ . Nonetheless, this task is crucial for constructing a true autoencoder.

To motivate our choice of  $\psi$ , we attempt to compute an approximate inverse of  $\phi$ , i.e., a function  $\rho$  such that  $\rho \circ \phi \approx \text{id}_{\mathbb{R}^D}$ . To do this we first simplify the expression for  $\phi$ :

$$\begin{aligned} \phi(x) &= \psi(x)\phi_{\text{loc}}(x) + (1 - \psi(x))x, \\ &= \psi(x)(\mathcal{P}_{T_{x_0}\mathcal{M}+x_c}\{x\}) + (1 - \psi(x))x, \\ &= \psi(x)(\mathcal{P}_{T_{x_0}\mathcal{M}}\{x - x_c\} + x_c) + (1 - \psi(x))x, \\ &= \psi(x)\mathcal{P}_{T_{x_0}\mathcal{M}}\{x\} - \psi(x)\mathcal{P}_{T_{x_0}\mathcal{M}}\{x_c\} + \psi(x)x_c + x - \psi(x)x, \\ &= \psi(x)\mathcal{P}_{T_{x_0}\mathcal{M}}\{x\} - \psi(x)\mathcal{P}_{T_{x_0}\mathcal{M}}\{x_c\} + \psi(x)\mathcal{P}_{T_{x_0}\mathcal{M}}\{x_c\} + \psi(x)\mathcal{P}_{N_{x_0}\mathcal{M}}\{x_c\} \\ &\quad + \mathcal{P}_{T_{x_0}\mathcal{M}}\{x\} + \mathcal{P}_{N_{x_0}\mathcal{M}}\{x\} - \psi(x)\mathcal{P}_{T_{x_0}\mathcal{M}}\{x\} - \psi(x)\mathcal{P}_{N_{x_0}\mathcal{M}}\{x\}, \\ &= \mathcal{P}_{T_{x_0}\mathcal{M}}\{\psi(x)x - \psi(x)x_c + \psi(x)x_c + x - \psi(x)x\} + \mathcal{P}_{N_{x_0}\mathcal{M}}\{\psi(x)x_c + x - \psi(x)x\}, \\ &= \mathcal{P}_{T_{x_0}\mathcal{M}}\{x\} + \mathcal{P}_{N_{x_0}\mathcal{M}}\{\psi(x)x_c + (1 - \psi(x))x\}. \end{aligned}$$

This simplification reveals a geometric interpretation of the flattening map  $\phi$ ; the component of  $x$  towards the tangent space  $T_{x_0}\mathcal{M}$  is preserved, while the component of  $x$  towards the normal space  $N_{x_0}\mathcal{M}$  is replaced by a convex combination of  $x_c$  and  $x$  whose weights are given by the partition

of unity  $\psi$ . The following computation now motivates an approximate inverse of  $\phi$ :

$$\begin{aligned}
 & \phi(x) + \psi(x) \left( x_0 - \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}+x_c}\{x_0\} + \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{\phi(x) - x_0\}, \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{\phi(x) - x_0\}) \right), \\
 &= \phi(x) + \psi(x) \left( x_0 - \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}+x_c}\{x_0\} + \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}, \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}) \right), \\
 &= \phi(x) + \psi(x) \left( x_0 - \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x_0\} - \mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{x_c\} + \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}, \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}) \right), \\
 &= \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x\} + \mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{\psi(x)x_c + (1 - \psi(x))x\} \\
 &\quad + \psi(x) \left( x_0 - \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x_0\} - \mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{x_c\} + \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}, \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}) \right), \\
 &= \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x\} + \psi(x)\mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{x_c\} + \mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{x\} - \psi(x)\mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{x\} \\
 &\quad + \psi(x) \left( \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x_0\} + \mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{x_0\} - \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x_0\} - \mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{x_c\} + \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}, \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}) \right), \\
 &= \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x\} + \mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{x\} + \psi(x) \left( -\mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{x - x_0\} + \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}, \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}) \right), \\
 &= x + \psi(x) \underbrace{\left( -\mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{x - x_0\} + \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}, \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}) \right)}_{\approx 0}, \tag{10}
 \end{aligned}$$

$$\approx x. \tag{11}$$

Here we make the approximation  $\mathcal{P}_{\mathbb{N}_{x_0}\mathcal{M}}\{x - x_0\} \approx \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\}, \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{x - x_0\})$  which follows from the definition of the second fundamental form.

Thus, we are motivated to set  $z = \phi(x)$  and obtain the reconstruction map

$$\rho(z) \doteq z + \psi(x) \left( x_0 - \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}+x_c}\{x_0\} + \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{z - x_0\}, \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{z - x_0\}) \right). \tag{12}$$

However, this is not a well-defined global map because the coefficient  $\psi(x)$  depends on  $x \approx \rho(z)$ . To solve this problem we use the following lemma:

**Lemma 7.** *Let  $\lambda > 0$ , let  $\psi(x) = e^{-\lambda\|x-x_0\|_2^2}$ , and  $\phi$  be defined as in (9). There exists a unique smooth function  $\xi: \mathbb{R}^D \rightarrow \mathbb{R}$  such that  $\xi(\phi(x)) = \psi(x)$  for all  $x \in \mathbb{R}^D$ .*

The proof of this lemma is deferred to Appendix D. It is not apparent from the theorem statement, but clear from the proof, is that  $\xi(z)$  is efficiently computable, requiring only the inversion of a globally invertible smooth scalar function, which we use the secant method to compute.

This lemma motivates our choice of  $\psi(x) = e^{-\lambda\|x-x_0\|_2^2}$  for some  $\lambda > 0$ , and thus our global reconstruction map is well-defined as the following:

$$\rho(z) \doteq z + \xi(z) \left( x_0 - \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}+x_c}\{x_0\} + \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{z - x_0\}, \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{z - x_0\}) \right). \tag{13}$$

While  $\psi(x) = e^{-\lambda\|x-x_0\|_2^2}$  has practicality through Lemma 7 and common usage in deep learning methodology, it is important to note that  $\psi$  is no longer a proper partition of unity, as even though  $\psi$  has a point of rapid decay, we have  $\psi > 0$  everywhere. Such positivity would be lethal for function approximation-type problems in differential geometry where information sharing between neighborhoods through these ‘‘approximate’’ partitions of unity would result in an inaccurate function approximation. However, the above approximations in this work still hold for a function  $\psi$  that

is only approximately a partition of unity. Practically, this choice allows us to avoid creating sharp cusps in the resulting manifold, making downstream flattening and reconstruction more feasible.

One important interpretation of the  $\lambda$  parameter in the partition of unity is that it controls the “radius” of the partition. In particular, the function  $r \mapsto e^{-\lambda r^2}$  is monotonically decreasing in  $r \geq 0$ . If we want to find the radius  $r$  for which  $e^{-\lambda r^2} \leq \varepsilon$ , where  $\varepsilon > 0$  is some small constant, we obtain the following:

$$r \geq \sqrt{\frac{\log(1/\varepsilon)}{\lambda}} = O(\lambda^{-1/2}). \quad (14)$$

Thus we can think of the “radius” of  $\psi$  as proportional to  $\lambda^{-1/2}$ .

Finally, recall that we defined the local average  $x_c = \frac{\int_{\mathcal{M}} x\pi(x) dx}{\int_{\mathcal{M}} \pi(x) dx}$  for some smooth function  $\pi: \mathbb{R}^D \rightarrow [0, 1]$  such that  $\pi(x) \approx 1$  for all  $x \in N_0$  and  $\pi$  decays rapidly outside  $N_0$ . In order to achieve these two criteria while making sure that  $\pi$  is smooth and  $x_c$  is close to  $x$ , we simply choose  $\pi = \psi$ , so that  $x_c = \frac{\int_{\mathcal{M}} x\psi(x) dx}{\int_{\mathcal{M}} \psi(x) dx}$ .

#### 4.4 Estimation from finite samples

In this section we discuss estimation and computation of the various quantities discussed in the prior sub-sections. Generic estimates are labeled with a tilde, i.e.,  $\tilde{x}_c$  or  $\tilde{T}_{x_0}\mathcal{M}$ , while optimal estimates are labeled with a hat, i.e.,  $\hat{x}_c$  or  $\hat{T}_{x_0}\mathcal{M}$ .

##### 4.4.1 ESTIMATING THE LOCAL AVERAGE, TANGENT SPACE AND SECOND FUNDAMENTAL FORM

Suppose that we already know the intrinsic dimension  $d$  of the manifold, and that we already know the parameters for the partition of unity  $\psi$  (these will be estimated automatically, as we describe momentarily). Then the only things left to estimate are the local average  $x_c$ , the tangent space  $T_{x_0}\mathcal{M}$  (and technically also the normal space  $N_{x_0}\mathcal{M} = (T_{x_0}\mathcal{M})^\perp$ ), and the second fundamental form  $\mathbb{I}_{x_0}$ .

The local average  $x_c$  with respect to a particular partition of unity  $\psi$  can be efficiently approximated by the sample average:

$$x_c \approx \hat{x}_c \doteq \frac{\sum_{i=1}^N x_i \psi(x_i)}{\sum_{i=1}^N \psi(x_i)}. \quad (15)$$

However, the local tangent space  $T_{x_0}\mathcal{M}$  is harder to approximate from a finite set of samples. A popular method to estimate  $T_{x_0}\mathcal{M}$  from finite data is local PCA (Hoppe et al., 1993; Oue, 1996); however, these methods rely on an assumption that the manifold  $\mathcal{M}$  is close to linear around the base point  $x_0$ , which holds less strongly if  $\mathcal{M}$  has non-negligible curvature. While improvements have been made past local PCA (Zhang et al., 2011), there are still issues when the data has both extrinsic curvature and extrinsic noise.

We may estimate the tangent space  $T_{x_0}\mathcal{M}$  from finite samples by minimizing the autoencoding loss, which is a sum of terms of the form

$$\begin{aligned} \|x_i - \rho(\phi(x_i))\|_2^2 &= \left\| \psi(x_i) \left( -\mathcal{P}_{N_{x_0}\mathcal{M}}\{x_i - x_0\} + \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{T_{x_0}\mathcal{M}}\{x_i - x_0\}, \mathcal{P}_{T_{x_0}\mathcal{M}}\{x_i - x_0\}) \right) \right\|_2^2, \\ &= \psi(x_i)^2 \left\| \frac{1}{2}\mathbb{I}_{x_0}(\mathcal{P}_{T_{x_0}\mathcal{M}}\{x_i - x_0\}, \mathcal{P}_{T_{x_0}\mathcal{M}}\{x_i - x_0\}) - \mathcal{P}_{N_{x_0}\mathcal{M}}\{x_i - x_0\} \right\|_2^2, \end{aligned}$$

where the first equality is from (10). For the sake of optimization, we parameterize  $T_{x_0}\mathcal{M}$  by an orthogonal matrix  $U \in \mathcal{O}(D, d)$  such that  $UU^\top = \mathcal{P}_{T_{x_0}\mathcal{M}}$ . This yields the following:

$$\|x_i - \rho(\phi(x_i))\|_2^2 = \psi(x_i)^2 \left\| \frac{1}{2} \mathbb{I}_{x_0}(UU^\top(x_i - x_0), UU^\top(x_i - x_0)) - (I - UU^\top)(x_i - x_0) \right\|_2^2. \quad (16)$$

Finally, we parameterize the second fundamental form (or more specifically the quantity  $\frac{1}{2}\mathbb{I}_{x_0}$ ) by its coordinates with respect to the basis defined by the columns of  $U$ , represented as a multi-array  $V \in \mathbb{R}^{D \times d \times d}$  in the following way:

$$\frac{1}{2} \mathbb{I}_{x_0}(w_1, w_2) = V(w_1, w_2) \doteq \sum_{j=1}^d \sum_{k=1}^d v_{jk} \langle u_j, w_1 \rangle \langle u_k, w_2 \rangle, \quad (17)$$

where  $v_{jk} \in \mathbb{R}^D$  are the slices through the first coordinate of  $V$ . Thus, we can estimate  $U$  and  $V$  through a computationally feasible autoencoding loss:

$$\mathcal{L}_\psi(\tilde{U}, \tilde{V}) \doteq \frac{1}{N} \sum_{i=1}^N \psi(x_i)^2 \left\| \tilde{V}(\tilde{U}\tilde{U}^\top(x_i - x_0), \tilde{U}\tilde{U}^\top(x_i - x_0)) - (I - \tilde{U}\tilde{U}^\top)(x_i - x_0) \right\|_2^2. \quad (18)$$

This yields a global reconstruction problem, whose value we label  $\text{Recon}(d, \psi)$ :

$$\text{Recon}(d, \psi) \doteq \inf_{\substack{U \in \mathcal{O}(D, d) \\ V \in \mathbb{R}^{D \times d \times d}}} \mathcal{L}_\psi(U, V). \quad (19)$$

The solutions to this problem are our estimates for the tangent space basis  $\hat{U}$  and second fundamental form matrix  $\hat{V}$ , which translate into estimates for the *actual* tangent space  $\hat{T}_{x_0}\mathcal{M}$  and second fundamental form  $\hat{\mathbb{I}}_{x_0}$  in a straightforward way.

While the above problem (19) may look daunting, note that for a fixed  $\tilde{U} \in \mathcal{O}(D, d)$ , solving for the entries of  $\hat{V}$  is a least-squares problem and may be efficiently solved in closed-form. The remaining component of the optimization is to solve for  $\hat{U}$ , which is an  $O(Dd)$ -dimensional optimization over the Stiefel manifold  $\mathcal{O}(D, d)$ . While the Stiefel manifold is a non-convex set, optimization over the Steifel manifold is well studied (Fraikin et al., 2007; Li et al., 2020; Qu et al.; Zhai et al., 2020; Zhai et al.). We reiterate that all associated complexities fulfill our complexity requirement of  $O(NDd^k)$  for some positive integer  $k$ .

There is one final complexity that we have swept under the rug: the second fundamental form  $\mathbb{I}_{x_0}$  has codomain  $N_{x_0}\mathcal{M}$ , which is a  $D - d$  dimensional subspace, but our bilinear  $V$  map has codomain  $\mathbb{R}^D$  in general, and at first glance is not restricted to any  $D - d$  dimensional subspace. The resolution to this dilemma is that if we fix a  $\tilde{U} \in \mathcal{O}(D, d)$  and solve for  $\hat{V}$  using least squares, the resulting bilinear  $\hat{V}$  map will always have codomain equal to  $\text{Im}(\tilde{U})^\perp$  due to the structure of the problem. In particular, if  $\text{Im}(\tilde{U}) = T_{x_0}\mathcal{M}$  then  $\hat{V}$  will have codomain  $N_{x_0}\mathcal{M}$ . We now formally state this result; the proof can be found in Appendix D.

**Proposition 8.** *Fix  $\tilde{U} \in \mathcal{O}(D, d)$ . Further suppose that  $\psi(x_i) > 0$  for all  $x_i \in \mathcal{X}$ . Then any solution  $\hat{V}$  to the problem*

$$\inf_{\tilde{V} \in \mathbb{R}^{D \times d \times d}} \mathcal{L}_\psi(\tilde{U}, \tilde{V}) \quad (20)$$

*has the following properties:*

1.  $\tilde{U}^\top \hat{V}(x_i - x_0, x_i - x_0) = 0$  for all  $x_i \in X$ .

For the following parts: define the matrix  $A \in \mathbb{R}^{N \times \frac{1}{2}(d^2+d)}$  in the following way: let  $B \in \mathbb{R}^{N \times d \times d}$  be the 3-tensor with entry values  $B_{ijk} = \langle u_j, x_i - x_0 \rangle \langle u_k, x_i - x_0 \rangle$ . Let  $B_i \in \mathbb{R}^{d \times d}$  be the  $i^{\text{th}}$  slice of the tensor  $B$ , fixing the first dimension. Finally, let the  $i^{\text{th}}$  row of  $A$  be the vectorized upper-diagonal component of  $B_i$ .

2. If  $A$  has full column-rank, i.e.  $\text{rank}(A) = \frac{1}{2}(d^2 - d)$ , then  $\hat{V}$  is unique.
3. Further, if  $A$  has full column-rank, then  $\tilde{U}^\top \hat{v}_{jk} = 0$  for all  $1 \leq j, k \leq d$ , therefore  $\tilde{U}^\top \hat{V}(w_1, w_2) = 0$  for all  $w_1, w_2 \in \mathbb{R}^D$ .

There are still a few parameters which the problem in (19) relies on: namely, the intrinsic dimension  $d$  of the manifold and the partition of unity  $\psi$ .

#### 4.4.2 ESTIMATING THE LOCAL DIMENSION

One such parameter is the intrinsic dimension  $d$ . Since  $d$  is equivalently the number of basis vectors used to locally represent the data:

*The local dimension  $d$  is chosen such that each encoder layer is as sparse and efficient as possible.*

This directly translates to finding the smallest  $d$  such that we can still locally reconstruct the data up to a desired precision  $\varepsilon_{\text{dim}} > 0$ :

$$\hat{d} = \min_{\tilde{d} \geq 0} \tilde{d} \tag{21}$$

$$\text{s.t. } \text{Recon}(\tilde{d}, \psi_0) \leq \varepsilon_{\text{dim}}. \tag{22}$$

Here, since we have not configured a good partition of unity  $\psi$  yet, we use another, “default” partition of unity  $\psi_0: \mathbb{R}^D \rightarrow \mathbb{R}$  given by

$$\psi_0(x) = e^{-\lambda_0 \|x - x_0\|_2^2}, \tag{23}$$

where  $\lambda_0 > 0$  is a scale parameter which is set as a hyperparameter to the algorithm. Heuristically it controls any set radius about  $x_0$  for which we expect the dataset  $\mathcal{X}$  to at least have samples in every intrinsic direction; we should think of this radius as proportional to  $\lambda_0^{-1/2}$ , as per our previous discussion.

This optimization is in similar spirit to fixed-rank approaches to low-rank matrix completion, such as ALS (Takács and Tikk, 2012) and Riemannian methods (Vandereycken, 2013), where we can adaptively choose the dimension  $\hat{d}$  at each point  $x_0$  by finding the minimal  $\tilde{d}$  such that the optimization problem in (19) achieves some desired threshold loss  $\varepsilon_{\text{dim}}$ .

Indeed,  $\hat{d}$  is designed to model the intrinsic dimension of  $\mathcal{M}$ , since any local flattening of lower dimension would collapse a direction in the tangent space and thus not be locally invertible. Since in practice we expect  $d \ll D$ , we can afford to solve (21) by iteratively solving (19) from  $d = 1$  upwards. Further, since theoretically  $d$  should be the same at every  $x_0 \in \mathcal{M}$ , we can start the search at the next iterate from the previous estimate  $\hat{d}$  to save computation.

**Learning a good partition of unity.** Once we estimate the dimension  $\hat{d}$ , the only parameter left is the choice of  $\lambda$  in the partition of unity:

$$\psi(x) = \psi_\lambda(x) = e^{-\lambda \|x - x_0\|_2^2}. \tag{24}$$



Again, we know that  $\lambda^{-1/2}$  is proportional to the radius of our partition, which is really the radius of the neighborhood in which our quadratic model approximately holds.

In similar spirit to  $d$ , we choose  $\lambda$  to make the individual layer as effective as possible, by maximizing the radius of the partition and thus the affected radius of the layer:

*The radius parameter  $\lambda^{-1/2}$  is chosen such that each encoder layer is as productive as possible.*

While  $d$  is optimized to use the fewest number of parameters within an individual layer (analogous to *network width*),  $\lambda$  is chosen to minimize the number of needed layers (analogous to *network depth*) by ensuring each layer makes as much progress towards a flattening as possible. Analogously to  $d$ , then, we can formalize the above statement into a constrained optimization problem:

$$\hat{\lambda} = \inf_{\tilde{\lambda} > 0} \tilde{\lambda} \quad (25)$$

$$\text{s.t. } \text{Recon}(\hat{d}, \psi_{\tilde{\lambda}}) \leq \varepsilon_{\text{POU}}. \quad (26)$$

Since  $\text{Recon}(\hat{d}, \psi_{\tilde{\lambda}})$  is a monotonic function with respect to  $\tilde{\lambda}$  (and the monotonicity is non-strict only if  $\text{Recon}(\hat{d}, \psi_{\tilde{\lambda}}) = 0$ ), if we define

$$\ell(\tilde{\lambda}) \doteq \text{Recon}(\hat{d}, \psi_{\tilde{\lambda}}), \quad (27)$$

then the above optimization problem amounts to computing

$$\hat{\lambda} = \ell^{-1}(\varepsilon_{\text{POU}}). \quad (28)$$

Since  $\ell$  is a function from  $\mathbb{R}$  to  $\mathbb{R}$ , its inversion can be efficiently computed by e.g. the secant method.

One issue with naively implementing this optimization is that we only have finite samples  $\mathcal{X}$  from  $\mathcal{M}$ . Thus there is a smallest radius (largest  $\tilde{\lambda}$ ) such that, for any smaller radii, there are not enough points in  $\mathcal{X}$  which are at most this distance to  $x_0$  to make the problem in (19) well-conditioned. Thus, we introduce another hyperparameter  $\lambda_{\text{max}}$  which controls the maximum permissible value of  $\tilde{\lambda}$ . The optimization problem becomes

$$\hat{\lambda} = \inf_{\tilde{\lambda} \in (0, \lambda_{\text{max}})} \tilde{\lambda} \quad (29)$$

$$\text{s.t. } \text{Recon}(\hat{d}, \psi_{\tilde{\lambda}}) \leq \varepsilon_{\text{POU}}. \quad (30)$$

Similarly to before, we can exploit the monotonicity property of  $\psi_{\tilde{\lambda}}$  to obtain

$$\hat{\lambda} = \min\{\lambda_{\text{max}}, \ell^{-1}(\varepsilon_{\text{POU}})\}. \quad (31)$$

The issue is that if  $\ell^{-1}(\varepsilon_{\text{POU}}) > \lambda_{\text{max}} = \hat{\lambda}$  then we do *not* have  $\text{Recon}(\hat{d}, \psi_{\tilde{\lambda}}) \leq \varepsilon_{\text{POU}}$ . To fix this, we amend the partition of unity with a multiplicative constant:

$$\psi(x) \doteq \psi_{\hat{\lambda}}(x) = \min \left\{ \alpha_{\text{max}}, \sqrt{\frac{\varepsilon_{\text{POU}}}{\ell(\hat{\lambda})}} \right\} \cdot e^{-\hat{\lambda} \|x - x_0\|_2^2}, \quad (32)$$

where  $\alpha_{\text{max}} \in (0, 1]$  is a user-set hyperparameter which controls the smoothness of the boundary of the partition of unity created by the flattening procedure in each step; in particular, if  $\alpha_{\text{max}} = 1$ , then non-smooth cusps appear in the flattening procedure, so we usually take  $\alpha_{\text{max}} < 1$ .

#### 4.5 A full description of the algorithm

With all these details, we are ready to compose a full description of the algorithm. At each iteration, we estimate the local dimension, compute a good partition of unity, estimate the local mean, tangent space, and second fundamental form, and compose our local flattening map, then we glue it with a partition of unity to get a global flattening map. Finally, we compute its inverse, the global reconstruction map, and append them to our multi-layer flattening and reconstruction maps, iteratively forming the Flattening Network.

More formally, we propose the following algorithm for training Flattening Networks (FlatNets):

---

**Algorithm 1** Main construction of FlatNet.
 

---

```

1: function FLATNETCONSTRUCTION( $\mathcal{X}, L, \varepsilon_{\text{dim}}, \lambda_0, \varepsilon_{\text{POU}}, \lambda_{\text{max}}, \alpha_{\text{max}}$ )
2:   Initialize  $\phi_{\text{net}}, \rho_{\text{net}} \leftarrow \text{id}_{\mathbb{R}^D}$ 
3:   for  $\ell \in [L]$  do
4:     Sample a random point  $x_0 \in \mathcal{X}$ 
5:     % Estimate the local dimension
6:     Define  $\psi_0: x \mapsto e^{-\lambda_0 \|x-x_0\|_2^2}$  ▷ Equation (23)
7:     Compute  $\hat{d} \leftarrow \min\{\tilde{d} \in [D]: \text{Recon}(\tilde{d}, \psi_0) \leq \varepsilon_{\text{dim}}\}$  ▷ Equations (21) and (22)
8:     % Compute the partition of unity
9:     For each  $\tilde{\lambda}$ , define  $\psi_{\tilde{\lambda}}: x \mapsto e^{-\tilde{\lambda} \|x-x_0\|_2^2}$  ▷ Equation (24)
10:    Define  $\ell: \tilde{\lambda} \mapsto \text{Recon}(\hat{d}, \psi_{\tilde{\lambda}})$  ▷ Equation (27)
11:    Compute  $\hat{\lambda} \leftarrow \min\{\lambda_{\text{max}}, \ell^{-1}(\varepsilon_{\text{POU}})\}$  ▷ Equation (28)
12:    Define  $\psi: x \mapsto \min\left\{\alpha_{\text{max}}, \sqrt{\varepsilon_{\text{POU}}/\ell(\hat{\lambda})}\right\} \cdot e^{-\hat{\lambda} \|x-x_0\|_2^2}$  ▷ Equation (32)
13:    % Estimate the local average, tangent space, and second fundamental form
14:    Compute  $\hat{x}_c \leftarrow \frac{\sum_{i=1}^N x_i \psi(x_i)}{\sum_{i=1}^N \psi(x_i)}$  ▷ Equation (15)
15:    Compute  $\hat{U}, \hat{V} \leftarrow \text{solutions of } \text{Recon}(\hat{d}, \psi)$  ▷ Equation (19)
16:    % Compute local and global flattening maps and and global reconstruction
    map
17:    Define  $\phi_{\text{loc}}: x \mapsto \hat{U}\hat{U}^\top(x - \hat{x}_c) + \hat{x}_c$  ▷ Equation (4)
18:    Define  $\phi: x \mapsto \psi(x)\phi_{\text{loc}}(x) + (1 - \psi(x))x$  ▷ Equation (9)
19:    Compute  $\xi$  such that  $\xi(\phi(x)) = \psi(x)$  for all  $x \in \mathbb{R}^D$  ▷ Theorem 7
20:    Define  $\rho: z \mapsto z + \xi(z) \left(x_0 - \hat{U}\hat{U}^\top(x_0 - \hat{x}_c) + \hat{x}_c + \hat{V}(\hat{U}\hat{U}^\top(z - x_0), \hat{U}\hat{U}^\top(z - x_0))\right)$  ▷ Equation (13)
21:    % Compose with previous layers
22:    Redefine  $\phi_{\text{net}} \leftarrow \phi \circ \phi_{\text{net}}$ 
23:    Redefine  $\rho_{\text{net}} \leftarrow \rho_{\text{net}} \circ \rho$ 
24: return  $\phi_{\text{net}}, \rho_{\text{net}}$ 

```

---

In Appendix E.1, we show that Flattening Networks meet the scalability requirements detailed in Section 1.1.

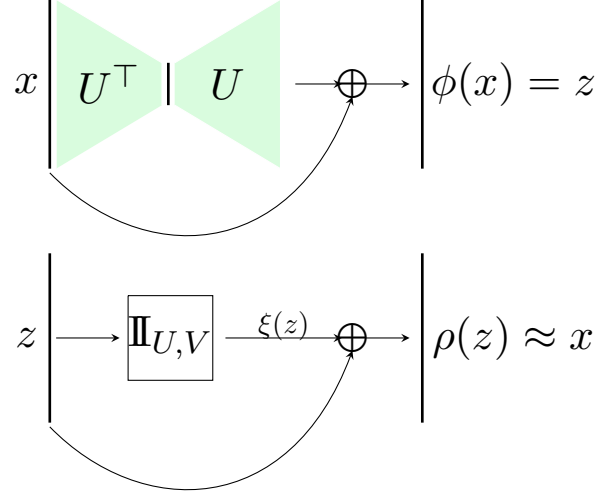


Figure 5: Network diagrams for each individual layer pair, with the encoder  $\phi_\ell : \mathbb{R}^D \rightarrow \mathbb{R}^D$  depicted on the top and the decoder  $\rho_\ell : \mathbb{R}^D \rightarrow \mathbb{R}^D$  depicted on the bottom. Note the encoder’s structure resembles a multi-layer perceptron layer with a residual connection; here though, the depicted sum is not a plain sum, but a weighed sum based on the Boltzmann distribution  $\psi(x)$ . We have omitted the bias terms that arise from constants in the expressions of  $\phi, \rho$ . Note that in generalizing this framework,  $\mathbb{I}_{U,V}$  can be replaced with any local, interpretable model of the data.

## 5. Algorithm intuition and convergence analysis

In this section, we provide the reader with some understandings and interpretations of the presented Algorithm 1.

### 5.1 Network diagram

Recall that our method builds a flattening pair  $\phi_{\text{net}}, \rho_{\text{net}}$  by repeatedly constructing and composing forward and backwards maps:  $\phi_{\text{net}} = \phi_L \circ \dots \circ \phi_1$  and  $\rho_{\text{net}} = \rho_1 \circ \dots \circ \rho_L$ , where each layer pair  $\phi_\ell, \rho_\ell$  takes the following form:

$$\begin{aligned} \phi_\ell(x) &= \psi(x)(UU^\top(x - x_c) + x_c) + (1 - \psi(x))x, \\ &= \psi(x)(UU^\top x + (I - UU^\top)x_c) + (1 - \psi(x))x, \end{aligned} \quad (33)$$

$$\begin{aligned} \rho_\ell(z) &= z + \xi(z) \left( x_0 - \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M} + x_c}\{x_0\} + \frac{1}{2}V(\mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{z - x_0\}, \mathcal{P}_{\mathbb{T}_{x_0}\mathcal{M}}\{z - x_0\}) \right), \\ &= z + \xi(z) \left( (I - UU^\top)(x_0 - x_c) + \frac{1}{2} \sum_{j=1}^d \sum_{k=1}^d v_{jk} \langle u_j, z - x_0 \rangle \langle u_k, z - x_0 \rangle \right), \end{aligned} \quad (34)$$

where  $U \in \mathbb{R}^{D \times d}$  with  $U^\top U = I$ ,  $V \in \mathbb{R}^{D \times d \times d}$ ,  $x_0 \in \mathcal{X}$ ,  $\psi(x) = \alpha e^{-\lambda \|x - x_0\|_2^2}$  for algorithmically chosen  $\alpha, \lambda > 0$ ,  $x_c = \frac{\sum_{i=1}^N x_i \psi(x_i)}{\sum_{i=1}^N \psi(x_i)}$ , and  $\xi(z)$  is the computed scalar function such that  $\xi(\psi(x)) = \psi(x)$ . These maps are graphically depicted in Figure 5.

## 5.2 What does each iteration look like?

To build some intuition for what each iteration of the network construction algorithm is doing, in Figure 6 we provide an example run of the algorithm on data sampled from a three-quarters-circle curve, and graph the output of the flattening map  $\phi_{CC}$  after 0, 20, 40, 60, and 80 layers have been constructed. More formally, let  $\phi_{1:\ell}$  be the flattening map composed of the first  $\ell$  layers of  $\phi_{CC}$ . Then define the following:

$$\mathcal{X}_\ell = \phi_{1:\ell}(\mathcal{X}), \quad (35)$$

$$\mathcal{M}_\ell = \phi_{1:\ell}(\mathcal{M}). \quad (36)$$

Note that we build layers of a FlatNet in a “closed-loop fashion” (Ma et al., 2022): the directions to flatten the data manifold  $\mathcal{M}$ , i.e., the estimator of the tangent space used by each individual layer of  $\phi_{CC}$ , are only determined by repeatedly going back to the data through the corresponding reconstruction map and finding the optimal flattening direction.

## 5.3 Convergence analysis

A complete convergence proof is outside of the scope of this paper, as proving a full theorem for convergence would require utilizing structure from the sampling size and density, noise level, structure of  $\mathcal{M}$  (something akin to low-curvature), and more. We leave a complete theorem to future work.

However, there are still many theoretical ideas for convergence that both illuminate the design choices for our architecture and characterize typical optimization behaviors: namely, what does hitting bad/good local minima look like. To this end, we give theoretical formality to the convergence of our main algorithm. We first detail the proper “norm” to use when detailing how close a manifold along the algorithm’s process  $\mathcal{M}_k$  to converging. We then give an outline for a convergence theorem using the proposed norm, characterizing both why we expect our main algorithm to converge to a good minimum and what happens when a bad local minimum is hit.

**A potential function for convergence.** Recall from Figure 4 that we use the difference between the convex hull  $\text{conv}(\mathcal{M})$  and the manifold  $\mathcal{M}$  itself to heuristically measure how close  $\mathcal{M}$  is to being flattened. Thus, we can first design a scalar-valued *potential function*  $\mathcal{H}$  that measures the difference between  $\text{conv}(\mathcal{M})$  and  $\mathcal{M}$ . For the sake of simplicity, we use the well-studied *Hausdorff metric* (Birsan and Tiba, 2005) between  $\mathcal{M}$  and  $\text{conv}(\mathcal{M})$ , which, since  $\mathcal{M} \subseteq \text{conv}(\mathcal{M})$  and  $\mathcal{M}$  is compact, reduces to the following:

$$\mathcal{H}(\mathcal{M}) \doteq \max_{x' \in \text{conv}(\mathcal{M})} \min_{x \in \mathcal{M}} \|x - x'\|_2. \quad (37)$$

Now the following are equivalent:

- $\mathcal{H}(\mathcal{M}) = 0$ ;
- $\mathcal{M} = \text{conv}(\mathcal{M})$ ;
- the second fundamental form of  $\mathcal{M}$  is identically 0;

and the last definition shows that  $\mathcal{H}$  is a metric determining how non-flat a manifold is. However, such a potential function does not reveal anything about the convergence of Algorithm 1, since there exist manifolds such that  $\mathcal{M} \subseteq T_x \mathcal{M}$  at all  $x \in \mathcal{M}$  but still have extrinsic curvature. Such tangent space-contained manifolds  $\mathcal{M}$  are “fixed points” of Algorithm 1 in the sense that any

constructed flattening maps will evaluate to the identity, but  $\mathcal{M}$  is not convex or flat. We then shift our potential function in a manner that more closely resembles Algorithm 1, i.e., measuring convexity of local subsets of a fixed radius  $\eta > 0$ :

$$\Omega_\eta(\mathcal{M}) = \int_{\text{int}_\eta \mathcal{M}} \mathcal{H}(\mathcal{M} \cap B_\eta(x)) \, dx, \quad (38)$$

$$\text{where } \text{int}_\eta \mathcal{M} \doteq \{x \in \mathcal{M} : \partial \mathcal{M} \cap B_\eta(x) = \emptyset\}, \quad (39)$$

where  $\partial \mathcal{M}$  is the boundary of  $\mathcal{M}$ . If  $\mathcal{H}(\mathcal{M}) = 0$ , then  $\mathcal{H}(\mathcal{M} \cap B_\eta(x)) = 0$  at each  $x$  ensures that the tangent space  $T_x \mathcal{M}$  doesn't change locally, and thus  $T_x \mathcal{M}$  cannot change globally, but  $\mathcal{M}$  does not necessarily have to be convex, since the above integral is only taken in the interior of  $\mathcal{M}$ . We can finally discretize the integral into a finite sum over the dataset  $\mathcal{X}$ :

$$\tilde{\Omega}_\eta(\mathcal{M}) = \sum_{i=1}^n \mathcal{H}(\mathcal{M} \cap B_\eta(x_i)). \quad (40)$$

The role of  $\eta$  becomes more apparent in the above formulation: if  $\mathcal{X}$  is an  $\eta$ -cover for  $\mathcal{M}$  (see Theorem 2), then  $\tilde{\Omega}_\eta(\mathcal{M}) = 0$  implies  $\mathcal{M} \subseteq T_x \mathcal{M}$  for all  $x \in \mathcal{M}$ , and thus implies a good convergence property for Algorithm 1. As neighborhoods in (40) may contain boundary points, some case-work is needed in order to handle behavior at boundary points, where Algorithm 1 may converge while the local loss remains nonzero.

**Convergence analysis via modelable radius.** While  $\mathcal{X}_\ell$  and  $\mathcal{M}_\ell$  are obviously important quantities to track for convergence, we present one more quantity of interest. Let  $r_{Q,\ell,\varepsilon}(x) \geq 0$  be the maximum radius  $r$  around  $x$  such that  $\mathcal{M}_\ell$  is modelable by a quadratic around  $x$  of radius  $r$  with at most  $\varepsilon$  error:

$$\max_{\substack{y \in \mathcal{M} \\ \|y-x\|_2 \leq r}} \max_{\substack{z \in \hat{Q} \\ \|z-x\|_2 \leq r}} \|y-z\|_2 \leq \varepsilon, \quad (41)$$

where  $\hat{Q}$  is a local quadratic model given by (1) for some proposed tangent space  $\hat{T}_x \mathcal{M}_k \subseteq \mathbb{R}^D$  of dimension  $d$  and second fundamental form  $\hat{\mathbb{I}}_x: T_x \mathcal{M} \times T_x \mathcal{M} \rightarrow N_x \mathcal{M}$ . If no such upper bound exists, we say  $r_{Q,\ell,\varepsilon}(x) = \infty$ .

In particular,  $r_{Q,\ell,\varepsilon}$  plays a crucial role in studying the convergence of Algorithm 1, as it completely characterizes whether or not we converge to a good or bad fixed point. This idea can be expressed concretely by the following dichotomy:

1. Suppose that  $\mathcal{M}_\ell \subseteq T_x \mathcal{M}_\ell$  for some  $\ell \in \mathbb{N}$ , thus halting Algorithm 1 at a good fixed point. Then  $r_{Q,\ell,\varepsilon}(\phi_{1:\ell}(x_i)) = \infty$  for all  $i \in [N]$  by setting  $\hat{\mathbb{I}}_{x_i} = 0$ , and thus the sequence  $(r_{Q,\ell,\varepsilon}(\phi_{1:\ell}(x_i)))_{\ell \in \mathbb{N}}$  has no upper bound for any  $i \in [N]$ .
2. Suppose on the contrary there exists some point  $x_i \in \mathcal{X}$  such that the sequence  $(r_{Q,\ell,\varepsilon}(\phi_{1:\ell}(x_i)))_{\ell \in \mathbb{N}}$  has an upper bound. Thus,  $\mathcal{M}_\ell \not\subseteq T_{x_j} \mathcal{M}_\ell$  for all  $\ell \in [L]$ , and  $x_j \in \mathcal{X}$ . Then if  $\mathcal{M}_\ell$  reaches a fixed point at some timestep  $\ell$ , there must exist some  $x_j$  such that  $r_{Q,\ell,\varepsilon}(\phi_{1:\ell}(x_j)) < \eta$ , since otherwise there exists an atlas of  $\mathcal{M}_\ell$  where each chart/neighborhood is completely flat, contradicting the statement that  $\mathcal{M}_\ell \not\subseteq T_x \mathcal{M}_\ell$ .

From the above dichotomy, we see the role of  $r_{Q,\ell,\varepsilon}$ : this quantity will diverge if Algorithm 1 converges to a good local minimum, and converge below the ‘‘threshold level’’  $\eta$  if a bad local minima is reached. However, the above analysis still *assumes* that Algorithm 1 converges. Extending

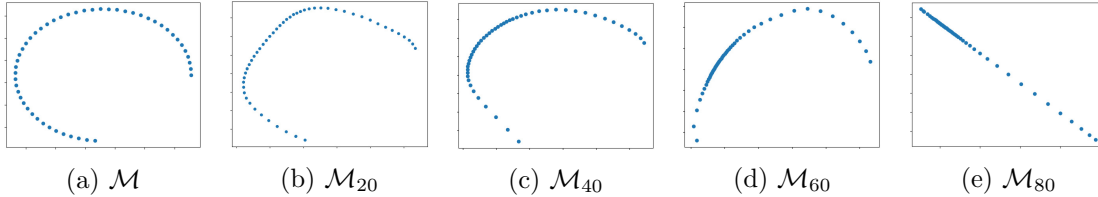


Figure 6: Plots of samples from the manifolds  $\mathcal{M}_k$  after a given number of layers  $k$  have been constructed. Note that the flattening is not isometric, so some intrinsic distortion is expected. However, this distortion is controlled by the requirement of each flattening to be invertible.

beyond this assumption is feasible, but requires detailed numerical analysis to account for the specific problem structure. The following is a dichotomy on the trajectory  $(\mathcal{M}_\ell)_{\ell \in \mathbb{N}}$  that does not require assumptions on convergence:

1. Suppose that for some  $\ell \in \mathbb{Z}$  and  $x_i \in \mathcal{X}$ , we have  $r_{Q,t,\varepsilon}(\phi_{1:t}(x_i)) < \eta$  for all  $t > \ell$ . Then Algorithm 1 either does not converge or converges to a bad local minima where  $\mathcal{M}_\ell \not\subseteq T_x \mathcal{M}_\ell$ .
2. Suppose that for all  $\ell \in \mathbb{Z}$  and  $x_i \in \mathcal{X}$ , there exists  $t > \ell$  such that  $r_{Q,t,\varepsilon}(\phi_{1:t}(x_i)) \geq \eta$ . Then Algorithm 1 can always make some progress on every  $x_i$  at iteration  $t$ .

Formalizing the above “progress” into positive progress via  $\lim_{\ell \rightarrow \infty} \tilde{\Omega}(\mathcal{M}_\ell) = 0$  is the main missing ingredient for a full convergence theorem. While it is clear that an iteration of Algorithm 1 at point  $x_i$  decreases the component of  $\tilde{\Omega}_\eta(\mathcal{M}_\ell)$  coming from  $x_i$  in (40), namely  $\mathcal{H}(\mathcal{M}_\ell \cap B_\eta(\phi_{1:\ell}(x_i)))$ , it’s not clear that the amount this iteration inadvertently increases  $\mathcal{H}(\mathcal{M}_\ell \cap B_\eta(\phi_{1:\ell}(x_j)))$  for other  $x_j \in \mathcal{X}$  would not lead to an overall increase in  $\tilde{\Omega}_\eta(\mathcal{M}_\ell)$ . Proving this relation requires careful analytical work on the global structure of  $\mathcal{M}$  and  $\mathcal{X}$ , which we leave for future work.

## 6. Experiments

To evaluate our method, we test our Flattening Networks (FlatNet) on synthetic low-dimensional manifold data, randomly generated high-dimensional manifolds generated by Gaussian processes (Lahiri et al., 2016; Lawrence and Hyvärinen, 2005), and popular real-world imagery data sets. The following are experimental observations that set our method apart from what is currently available:

1. *There is no need to select the intrinsic dimension  $d$* , as this is automatically learned from the data using (21). This is in contrast to neural network-based representation learners, e.g. variational autoencoders (Kingma and Welling, 2013), which need a feature dimension selection beforehand, or many manifold learning methods that need an intrinsic dimension specified. This is important for practical data manifolds, since the intrinsic dimension of the underlying data manifold is hardly ever known.
2. *There is no need to select a stopping time*, as our convexifying geometric flow converges once the learned representation is already flat.
3. *Learned manifolds from noisy data are smooth*, as depicted in figures in Section 6.1. This is likewise important for practical data manifolds, as samples have off-manifold noise.

Code for both FlatNet itself and the below experiments is publicly available<sup>1</sup>.

1. <https://github.com/michael-psenka/manifold-linearization>

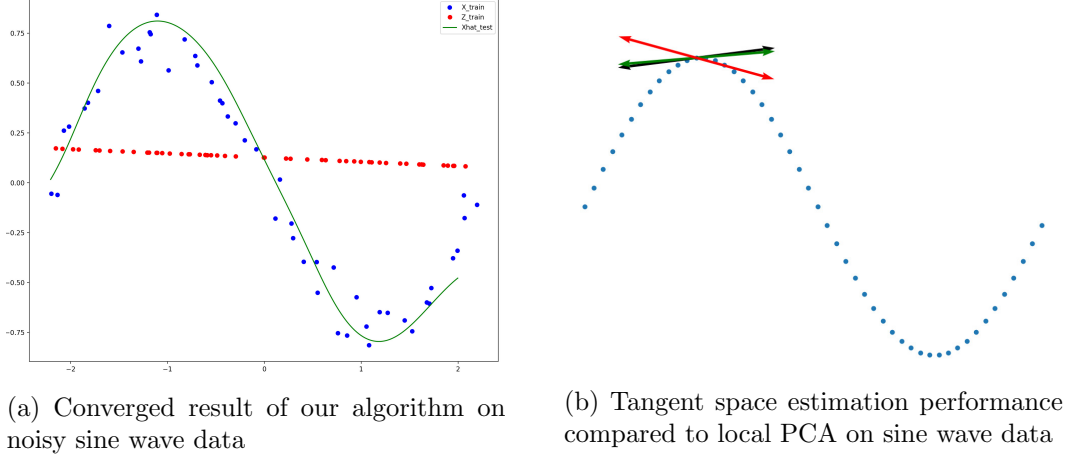


Figure 7: Results of FlatNet on data sampled uniformly from the graph of a sine wave embedded in  $\mathbb{R}^2$ . In the left figure: the blue points are the training data  $X$ , the red points are the flattened data  $Z$ , and the green line is the reconstruction of interpolated points  $\hat{X}_{\text{test}}$ . On the right, each double sided arrow is a tangent space: black is ground truth, green is our estimation method, and red is local PCA.

### 6.1 Low-dimensional manifold data

We test on three types of low-dimensional manifolds:

- Data sampled from a (noisy) sine wave,
- Data sampled from a Gaussian process manifold (Lahiri et al., 2016; Lawrence and Hyvärinen, 2005) with intrinsic dimension  $d = 1$  in Euclidean space of extrinsic dimension  $D = 2$ ,
- Data sampled from a Gaussian process manifold with  $d = 2$  and  $D = 3$ .

Since the Gaussian process manifold data generation process is not clear at first glance, we describe it here. We first lay out  $N$  vectors of intrinsic coordinates in a matrix  $C \in \mathbb{R}^{d \times N}$ ; in our experiments, these are uniformly generated. Then for each  $i \in \{1, \dots, D\}$ , we set up a correlation matrix  $\Sigma_i \in \mathbb{R}^{N \times N}$  whose coordinates  $(\Sigma_i)_{pq} = \frac{L_i}{D} e^{-\rho_{pq}/2}$ , where  $L_i > 0$  is a hyperparameter (set to be 1 in our quantitative experiments) and  $\rho_{pq} = \|c_p - c_q\|_2^2$ , where  $c_p$  and  $c_q$  are the columns of  $C$ . Then the  $i^{\text{th}}$  row of  $X$  (i.e. the  $i^{\text{th}}$  coordinate of all datapoints, a vector in  $\mathbb{R}^N$ ) is sampled (independently of all other rows) from the Gaussian  $\mathcal{N}(0_N, \Sigma_i)$ . Repeating this for all  $i \in \{1, \dots, D\}$  obtains the full data matrix  $X = [x_1 \ \dots \ x_N] \in \mathbb{R}^{D \times N}$ .

We measure the performance of our algorithm in a few ways:

- Given data  $X \in \mathbb{R}^{D \times N}$ , we can plot the features  $Z \doteq \phi_{\text{net}}(X) \in \mathbb{R}^{D \times N}$  and reconstructions  $\hat{X} \doteq \rho_{\text{net}}(Z) \in \mathbb{R}^{D \times N}$ . Then the features  $Z$  should form an affine subspace and the reconstructions  $\hat{X}$  should be close to  $X$  sample-wise. In order to understand how our method generalizes, we may linearly interpolate between pairs of features in  $Z$  to get a large new matrix  $Z_{\text{test}} \in \mathbb{R}^{D \times M}$  for  $M \gg N$ , then reconstruct that as  $\hat{X}_{\text{test}} := \rho_{\text{net}}(Z_{\text{test}}) \in \mathbb{R}^{D \times M}$ . Ultimately, we plot  $X$ ,  $Z$ , and  $\hat{X}_{\text{test}}$ .
- If we know an explicit formula for the ground truth curve that generates the data, we may compute the tangent space at each point. We may thus evaluate how close our tangent space

estimator used in the algorithm is to the ground truth tangent space. We do this by plotting both the estimated tangent space and the true tangent space; they should greatly overlap.

In Figure 7, we demonstrate the performance of our algorithm FlatNet on data generated from the graph of a (noisy) sine wave. More precisely, set  $N = 50$ ,  $D = 2$ ,  $d = 1$ , and  $M = 5000$ . In the left figure we test linearization, reconstruction, and generalization, while in the right figure we demonstrate tangent space estimation.

In Figure 8, we demonstrate the performance of our algorithm FlatNet on data generated on a manifold constructed via Gaussian processes (Lahiri et al., 2016; Lawrence and Hyvärinen, 2005). Note that for such manifolds, we do not know ground truth tangent spaces, so we constrain our experiments to demonstrating reconstruction and generalization performance. We set  $N = 50$ ,  $D = 2$ ,  $d = 1$ , and  $M = 5000$  like before. We compare the reconstruction and generalization performance of FlatNet against three types of variational autoencoders (VAEs): (1) the vanilla VAE (Kingma and Welling, 2013), (2)  $\beta$ -VAE ( $\beta = 4$ ) (Higgins et al., 2017), and (3) FactorVAE ( $\gamma = 30$ ) (Kim and Mnih, 2018). Each VAE encoder and decoder is a 5 layer multi-layer perceptron, with dimension 100 at each layer, and latent dimension  $d = 1$ . Each VAE is trained for 100 epochs with Adam optimizer and  $10^{-3}$  learning rate. The experiment demonstrates that FlatNet is convincingly better at learning to reconstruct low-dimensional structures, and has better generalization performance, than the VAEs we compare against.

In Figure 9, we demonstrate the performance of FlatNet on data generated on a manifold constructed via Gaussian processes as before. This time, we set  $N = 50$ ,  $D = 3$ ,  $d = 2$ , and  $M = 5000$ . We do the same comparisons to VAEs as before; this time the VAEs have latent dimension  $d = 2$ . The experiment again demonstrates that FlatNet is better at learning to reconstruct low-dimensional structures and has better generalization performance.

## 6.2 High-dimensional manifold data

In this section we test our algorithm on random Gaussian process manifolds (Lahiri et al., 2016; Lawrence and Hyvärinen, 2005) of varying intrinsic dimension  $d$  in extrinsic dimension  $D = 100$  Euclidean space. We typically use  $N = 1000$ .

There are two quantitative ways we measure the performance of FlatNet in high dimensions: reconstruction quality and ability to estimate the dimension of the manifold. Note that the latter is essentially trivial in the low dimensional regime, yet in the high-dimensional regime it can be thought of as a rough certificate for the accuracy of each local flattening iteration.

In Figure 10, we empirically evaluate the reconstruction error on our finite samples  $\{x_1, \dots, x_N\}$  as the following:

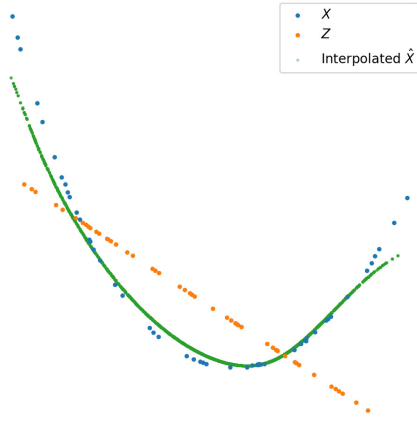
$$\text{reconstruction error} \approx \frac{1}{N} \sum_{i=1}^N \|x_i - \hat{x}_i\|_2^2, \quad \text{where} \quad \hat{x}_i := \rho_{\text{net}}(\phi_{\text{net}}(x_i)). \quad (42)$$

We estimate the error across 3 trials of the experiment, and plot the mean curve and standard deviation, for each  $d \in \{5, 10, 15, 20\}$ . We compare against FactorVAE, which has the same configuration as previously discussed except for the latent dimension which is set to  $d$ .<sup>2</sup> Overall, the experiment demonstrates again that FlatNet is convincingly better at learning to reconstruct low-dimensional structures than the VAEs we compare against.

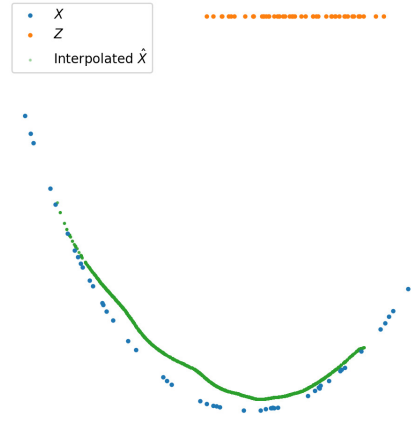
In Figure 11, we empirically evaluate the dimension estimation capability. Given a trained FlatNet encoder and decoder, we estimate the global intrinsic dimension of  $\mathcal{M}$  using the follow-

2. We also compared against the vanilla VAE and  $\beta$ -VAE, but those had nearly identical performance to FactorVAE and so are omitted from the chart for visual clarity.

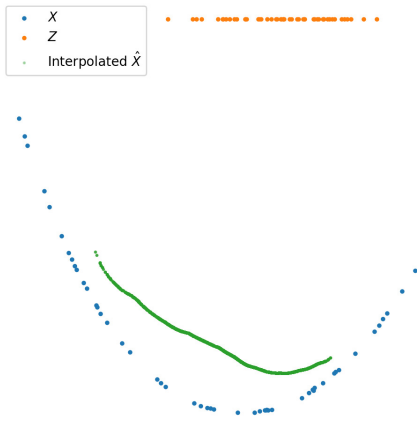
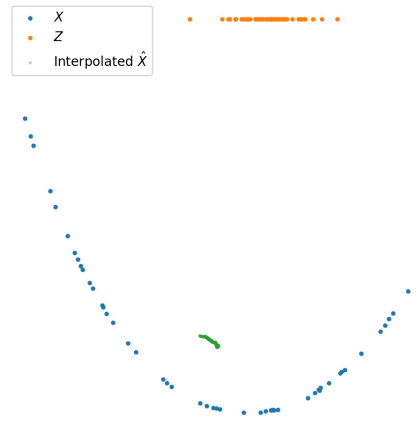




(a) FlatNet.

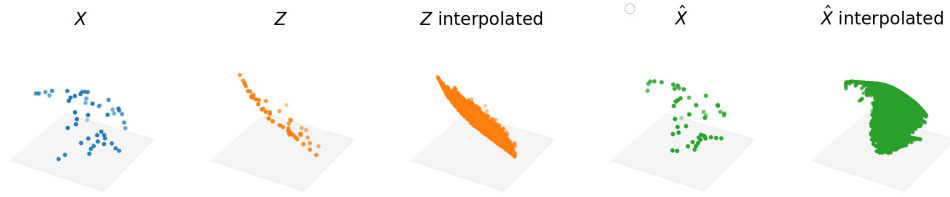


(b) VAE.

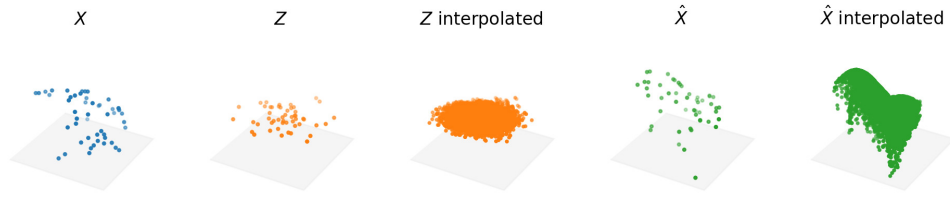

 (c)  $\beta$ -VAE.


(d) FactorVAE.

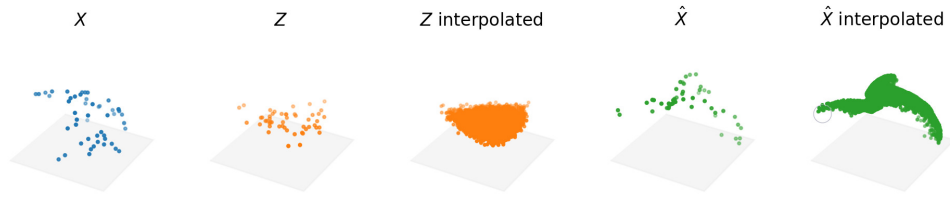
Figure 8: Results of FlatNet contrasted with VAE,  $\beta$ -VAE, and FactorVAE. Data is  $N = 50$  points sampled from a random  $d = 1$  dimensional Gaussian process manifold in  $\mathbb{R}^D = \mathbb{R}^2$ . FlatNet is the only presented method that retains the smoothness of the original data manifold, an expected result from explicitly utilizing the data geometry.



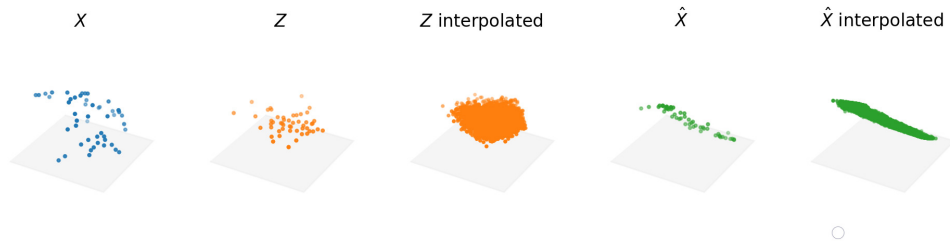
(a) FlatNet.



(b) VAE.



(c)  $\beta$ -VAE.



(d) FactorVAE.

Figure 9: Results of FlatNet contrasted with VAE,  $\beta$ -VAE, and FactorVAE. Data is  $N = 50$  points sampled from a random  $d = 2$ -dimensional Gaussian process manifold in  $\mathbb{R}^D = \mathbb{R}^3$ . Still in nontrivial settings, FlatNet uniquely learns the correct geometry of the underlying manifold compared to the other tested methods, and thus generalizes better. As a visual note, the feature space for FlatNet is really a 2-dimensional affine subspace, as desired; this would be clear after rotating the figure. Tools to perform interactive visualization are present in the attached code. 26

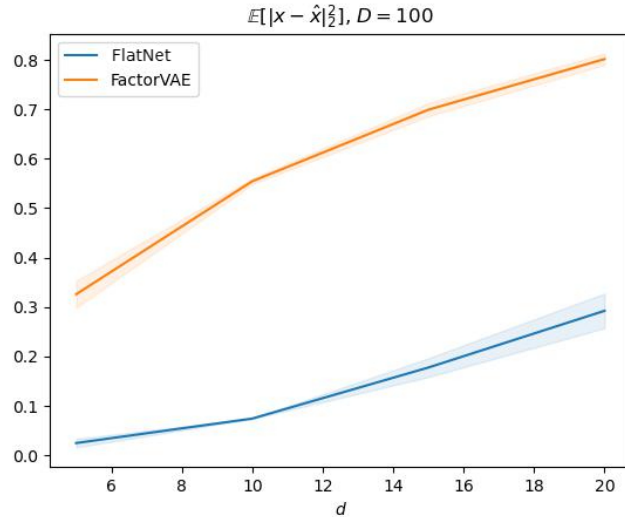


Figure 10: Reconstruction error of FlatNet and FactorVAE. Data is  $N = 1000$  points sampled from a random  $d \in \{5, 10, 15, 20\}$  dimensional Gaussian process manifold in  $\mathbb{R}^D = \mathbb{R}^{100}$ . Error margins are  $\pm 1$  standard deviation across 3 trials.

ing procedure. Each iteration  $\ell$  of the training process estimated an intrinsic local dimension  $\hat{d}_k$  for some neighborhood of the manifold; to estimate the global dimension we simply compute  $\hat{d} \doteq \text{mode}(\hat{d}_\ell: \ell \in [L])$ , i.e., the most commonly estimated local dimension. We estimate the intrinsic dimension across 3 trials of the experiment, and plot the mean curve and standard deviation, for each  $d \in \{5, 10, 15, 20\}$ . We compare with the popular intrinsic dimension estimation algorithms MLE (Levina and Bickel, 2004a) and TwoNN (Facco et al., 2017). Overall, the experiment demonstrates that FlatNet is competitive with the state of the art at dimension estimation, even when not explicitly designed for this task.

### 6.3 Real-world imagery data

Real-world imagery data such as MNIST tend to lie on high-curvature or non-differentiable pathological manifolds (Wakin et al., 2005). In order to make such data tractable for the use of FlatNet, we use the (vectorized) Fourier transform of the image, say  $\mathcal{F}\{x_i\}$ , instead of the image  $x_i$  itself, as the input to FlatNet, where it becomes easier to smooth out domain transformations (e.g. convolution with a Gaussian kernel).

We begin with a constructed manifold from MNIST to visualize how well FlatNet recovers intrinsic features from 2D vision data. To this end, we take a single image from MNIST, and both rotate it by various angles and translate it along the  $y$ -axis by various amounts. The result is a 2-dimensional manifold embedded in pixel space (here,  $D = 32 \times 32 = 1024$ ). In Figure 15, we present the results of running FlatNet on the constructed manifold, with both the features it recovers and reconstruction accuracy.

We now move to the original MNIST data set. In Figure 12, we demonstrate the empirical reconstruction performance of FlatNet on MNIST data. Notice that the reconstruction is not pixel-wise perfectly accurate; the reconstructed samples appear closer to an archetype of the digit

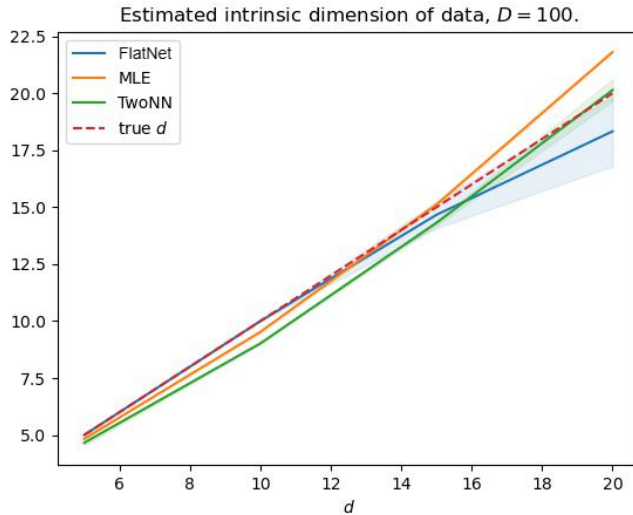


Figure 11: Global intrinsic dimension estimation of FlatNet, MLE, and TwoNN. Data is  $N = 1000$  points sampled from a random  $d \in \{5, 10, 15, 20\}$  dimensional Gaussian process manifold in  $\mathbb{R}^D = \mathbb{R}^{100}$ . Error margins are  $\pm 1$  standard deviation across 3 trials. The number of samples was kept the same through experiments to simulate the relative data sparsity for real data manifolds, so dimension estimates are expected to vary more at higher intrinsic dimensions.

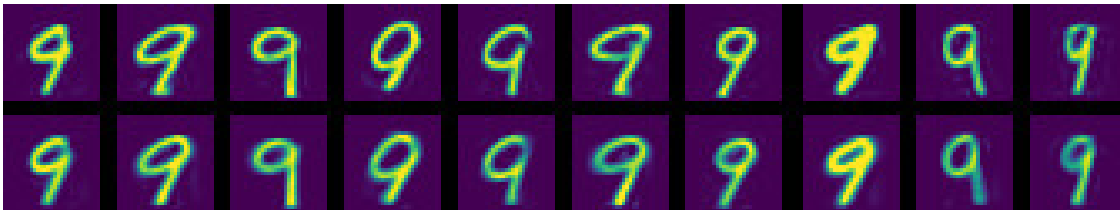


Figure 12: Original  $x_i$  (top) vs. reconstruction  $\hat{x}_i \doteq \mathcal{F}^{-1}\{\rho_{\text{net}}(\phi_{\text{net}}(\mathcal{F}\{x_i\}))\}$  (bottom), using FlatNet flattening  $\phi_{\text{net}}$  and reconstruction  $\rho_{\text{net}}$  on the MNIST data set.

than the original image. This example hints that the encoder function  $\phi_{\text{net}}$  naturally compresses semantically non-meaningful aspects of the input image, which is a useful property for encoders.

In Figure 13, we demonstrate that the feature space of FlatNet is linear in that it corresponds to an affine subspace. In particular, we demonstrate that linear interpolation in feature space corresponds to semantic interpolation in image space. In Figure 14, we demonstrate that we can sample from the linear (structured) feature space to generate semantically meaningful data in image space.

#### 6.4 Low-dimensional counterexamples

We present two low-dimensional counterexamples to show potential limitations of the method, as well as what the algorithm looks like when it hits a failure mode. We present two illustrative results.

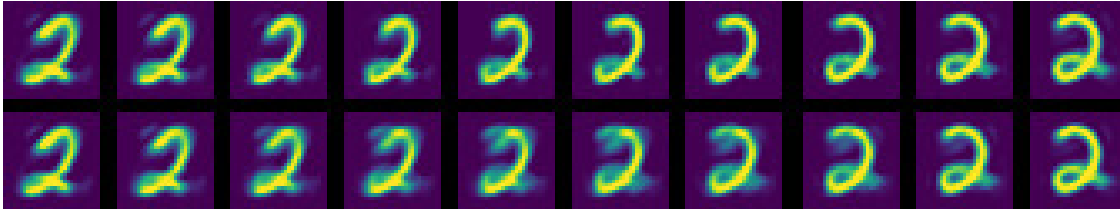


Figure 13: Comparison of interpolation: FlatNet (top), i.e.,  $\hat{x} \doteq \mathcal{F}^{-1}\{\rho_{\text{net}}(\theta\phi_{\text{net}}(\mathcal{F}\{x_1\}) + (1 - \theta)\phi_{\text{net}}(\mathcal{F}\{x_2\}))\}$ , versus linear interpolation (bottom), i.e.,  $\hat{x} \doteq \theta x_1 + (1 - \theta)x_2$ .

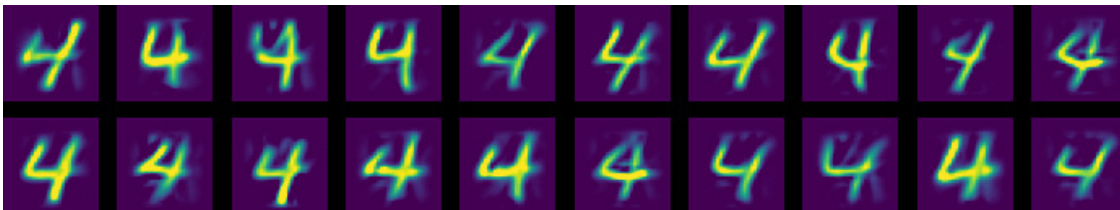


Figure 14: Sampling using FlatNet:  $\hat{x} = \mathcal{F}^{-1}\{\rho_{\text{net}}(z)\}$  where  $z$  is a Gaussian random variable supported on the affine subspace  $\phi_{\text{net}}(\mathcal{F}\{\mathcal{M}\})$ .

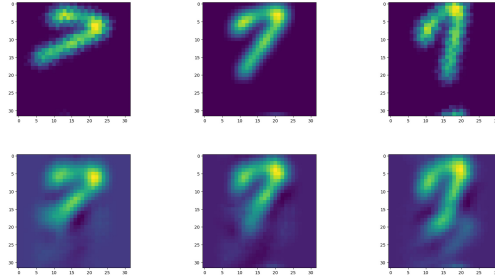
**Closed circle.** The first is data sampled from a closed curve, a circle, with noise. This represents a manifold that is not even theoretically flattenable. While this represents a theoretical boundary for FlatNet, note that any continuous autoencoding pair, regardless of the model, likewise cannot flatten these manifolds. See Figure 16 for a visualization on how FlatNet behaves on such manifolds: it will start to form flat patches until it cusps and is not flattenable or modelable anymore.

**Unaugmented Swiss roll.** The second is data sampled from the “Swiss roll” data set. While this manifold is theoretically flattenable, it is quite “crammed” in its ambient space: not only is it a hypersurface ( $d = D - 1$ ), its self-inward curling makes compression-based flattening methods like FlatNet fold the manifold into itself before. If FlatNet is run on the pure Swiss roll, it runs into the same problems as the closed circle: it quickly cannot make any invertible flattening progress, and thus halts without changing the manifold much at all.

*However*, if a simple nonlinear feature is appended ( $(x_1, x_2, x_3) \mapsto (x_1, x_2, x_3, x_1^2 + x_3^2)$ ), then FlatNet is able to both efficiently flatten the manifold and give an approximate reconstruction. While the approximate reconstruction has noticeable differences from the original manifold<sup>3</sup>, the learned features correspond closely to the intrinsic coordinates that generated the Swiss roll data set. See Figure 17 for plots of the results.

As our method is compressive in nature (derived from “compressing out” extrinsic curvature), the results here make sense: the manifold will more likely “run into itself” when it does not have as much room to move (e.g. for hypersurfaces). However, (a) we expect this to happen less in data manifold scenarios, where the codimension is much higher, and (b) our simple solution to make Swiss roll work for FlatNet highlights an interesting direction for future work: nonlinear liftings of the manifold before flattening to allow the manifold more room to compress into itself.

3. As we are also forcing the reconstruction to fit the artificially constructed feature  $x_4 = x_1^2 + x_3^2$ , we can expect a decrease in the reconstruction quality as a cost for easier flattening.



(a) Original rotated/translated images (top) compared to reconstruction  $\rho_{\text{net}}(\phi_{\text{net}}(x))$  from FlatNet (bottom)

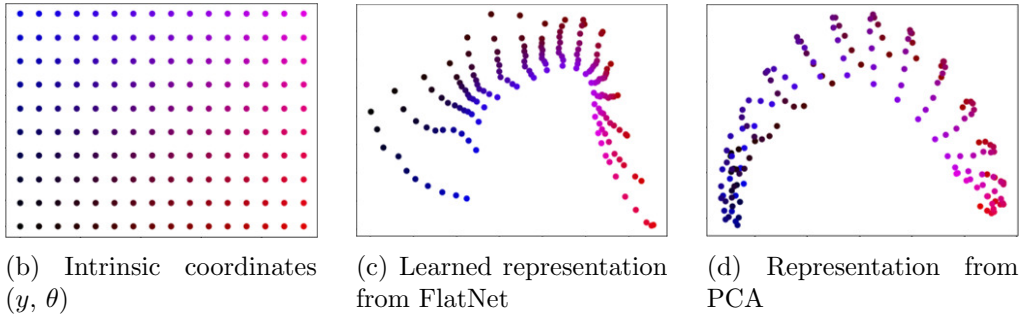


Figure 15: Results of FlatNet on the following manifold: a base point is selected from MNIST (here a “7”), and various  $y$ -translations between  $[-7, 7]$  pixels and rotations between  $[-30, 30]$  degrees. The representation from FlatNet was run until convergence: it determined the data set was of dimension 2 automatically. Note that while the features look distorted compared to the intrinsic features, FlatNet learns a representation that is relatively smoothly deformable into the canonical coordinates. However, the PCA representation obtained from taking the top two principal components (which requires knowledge of  $d = 2$  a-priori), yields a “tangled” representation: mapping the PCA representation back into the intrinsic coordinates would require a more complex function.

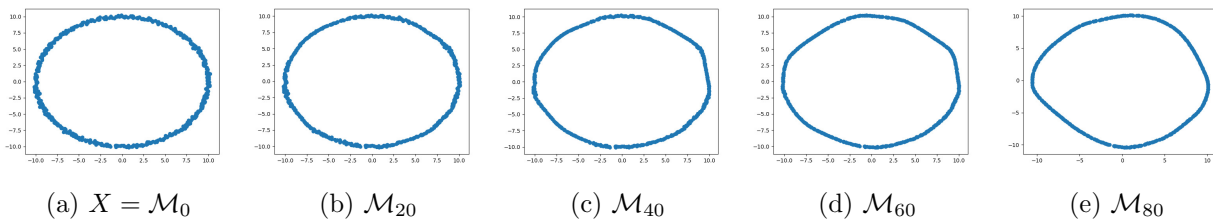


Figure 16: Plots of resulting features  $\mathcal{M}_\ell \doteq \phi_{1:\ell}(X)$  through the first  $\ell$  layers of the flattening map  $\phi_{\text{net}}$ , for data sampled from a noisy circle. We observe denoising of the data set onto a closed curve which oscillates but ultimately does not converge to a linear representation. While there is no globally continuous function that would be able to linearize such a structure (including many deep learning autoencoders), it remains an interesting direction of work on how to canonicalize such structures with closed loops.

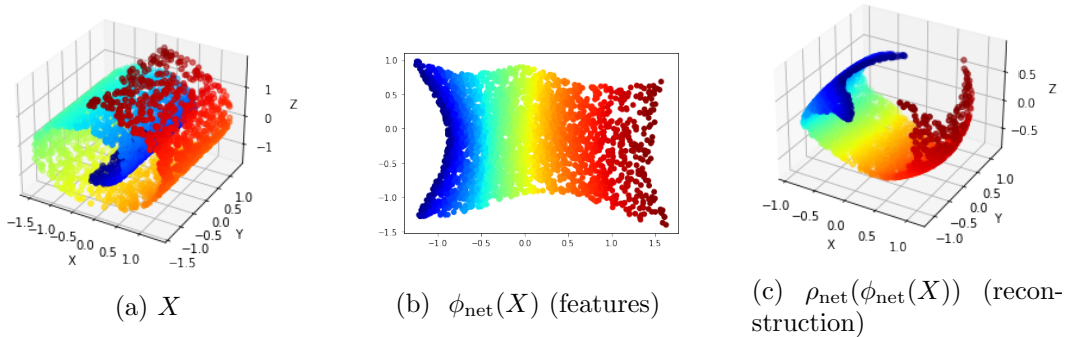


Figure 17: Plots of (a) the Swiss roll data set ( $N = 3000$ ), (b) the learned features after a converged FlatNet flattening on lifted 4D features  $(x_1, x_2, x_3, x_1^2 + x_3^2)$ , and (c) the learned reconstruction from the FlatNet. Reconstruction accuracy suffers as we are likewise adding extraneous coordinates to the reconstruction problem; however, the representations learned by FlatNet are noticeably close to the original generating coordinates.

## 7. Conclusion and future work

In this work, we propose a computationally tractable framework and algorithm for flattening and reconstructing data manifolds, with the objective of explicitly learning representations of data more amenable to downstream tasks by utilizing their interpolated geometry. In particular, our method generates an autoencoding pair in a forward, layer-wise fashion. Of primary practical benefit for this methodology is the automation of network design: the network’s width and height for example are both chosen automatically to be as minimal as possible. For researchers training a new autoencoder model from scratch, this can pose a large practical benefit. When more explicit geometric structure of the data is utilized, a much larger amount of the network can be chosen automatically.

Of course, one does not necessarily need to use a local quadratic model for modelling the geometry of a data set. Indeed, most of the theory in this paper still holds if we replace the local model  $\mathbb{I}_{x_0}$  with something besides a quadratic, such as local domain deformation models for image manifolds. This opens up a broader space of possibilities for both analyzing and manipulating geometries found in data sets and data manifolds.

Even within the realm of manifold learning, this work focuses entirely on the injective flattening and removal of extrinsic curvature, without considerations of intrinsic curvature or true metrics. However, the resulting flattened space of the manifold could make intrinsic manifold learners more efficient and effective in the resulting space, making flattening networks a viable component for a full intrinsic manifold learning algorithm.

Outside of the flattenability assumptions (which excludes any closed loops in the manifold, like for example rotation groups for computer vision), it is still a strong assumption that the entire dataset is a single, connected manifold of a single dimension. One common example is a “multiple manifolds hypothesis” (Brown et al., 2022; Vidal et al., 2005; Yu et al., 2020), but there is still more work to do for a geometric model that captures realistic, hierarchical structures. Nonetheless, we feel the presented work builds grounds for geometric-based learning methods in more practical environments which scale with modern data requirements.

### **Acknowledgements**

We thank Sam Buchanan of TTIC for helpful discussions around existing approaches to manifold flattening and learning. Michael Psenka acknowledges support from ONR grant N00014-22-1-2102. Yi Ma acknowledges support from ONR grant N00014-22-1-2102 and the joint Simons Foundation-NSF DMS grant #2031899. This work was partially supported by NSF 1704458, the Northrop Grumman Mission Systems Research in Applications for Learning Machines (REALM) initiative, NIH NIA 1R01AG067396, and ARO MURI W911NF-17-1-0304.



## Appendix A. Additional experiments

We provide below additional figures to help the reader visualize the performance of FlatNet.

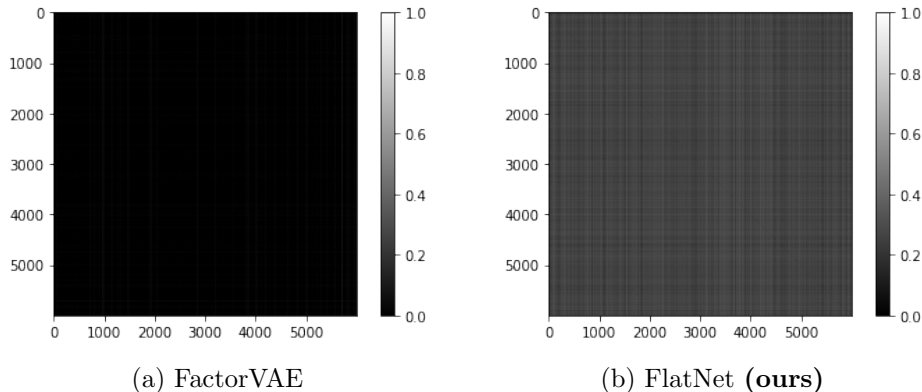


Figure 18: Intrinsic distortion of FactorVAE vs. FlatNet (*brighter picture is better*, i.e. less distorted), on a random manifold of embedding dimension  $D = 100$ , intrinsic dimension  $d = 10$ , and training set size  $N = 6000$ . We measure the distortion rate by the elementwise ratio of matrices  $\text{EDM}(\phi_{\text{net}}(X))/\text{EDM}(C)$ , where  $\text{EDM}(\cdot) \in \mathbb{R}^{N \times N}$  is the Euclidean distance matrix of a matrix of samples,  $X \in \mathbb{R}^{D \times N}$  is the training data, and  $C \in \mathbb{R}^{d \times N}$  are the intrinsic coordinates that generated  $X$ . Depicted results are scaled by the maximum of the EDM ratio.

We also test the intrinsic distortion of FactorVAE of a random manifold over various stopping times (Figure 19).

## Appendix B. Differential geometry overview

For the sake of being self-contained, we give a brief introduction to some fundamental definitions and constructs in differential geometry needed for this paper, as well as provide the geometric definitions used for this paper. These definitions are not meant to be mathematically complete, but communicate the main ideas used in this paper. Please see the appendix for a more detailed overview, and see Boumal (2020); Lee (2012, 2018) for further introductions to differential geometry and embedded submanifolds.

### B.1 Embedded submanifolds

While general differential geometry studies general manifolds, our focus for this paper are manifolds that pertain to the *manifold hypothesis*; that is, lower dimensional manifolds explicitly embedded in higher-dimensional Euclidean spaces. The following is a typical definition for such manifolds:

**Definition 9.** *An embedded submanifold (or a manifold)  $\mathcal{M}$  is a subset of some Euclidean space  $\mathbb{R}^D$  such that for every  $x \in \mathcal{M}$ , there is some nonempty neighborhood  $N_\varepsilon(x) = \mathcal{M} \cap B_\varepsilon(x)$  that is diffeomorphic with a subset of  $\mathbb{R}^d$ . The integer  $d$  is called the dimension of the manifold, denoted  $\dim(\mathcal{M})$ .*

Intuitively, an embedded submanifold is a continuous structure that is at every point locally invertible to a  $d$ -dimensional vector representation. For this paper, we will further require manifolds

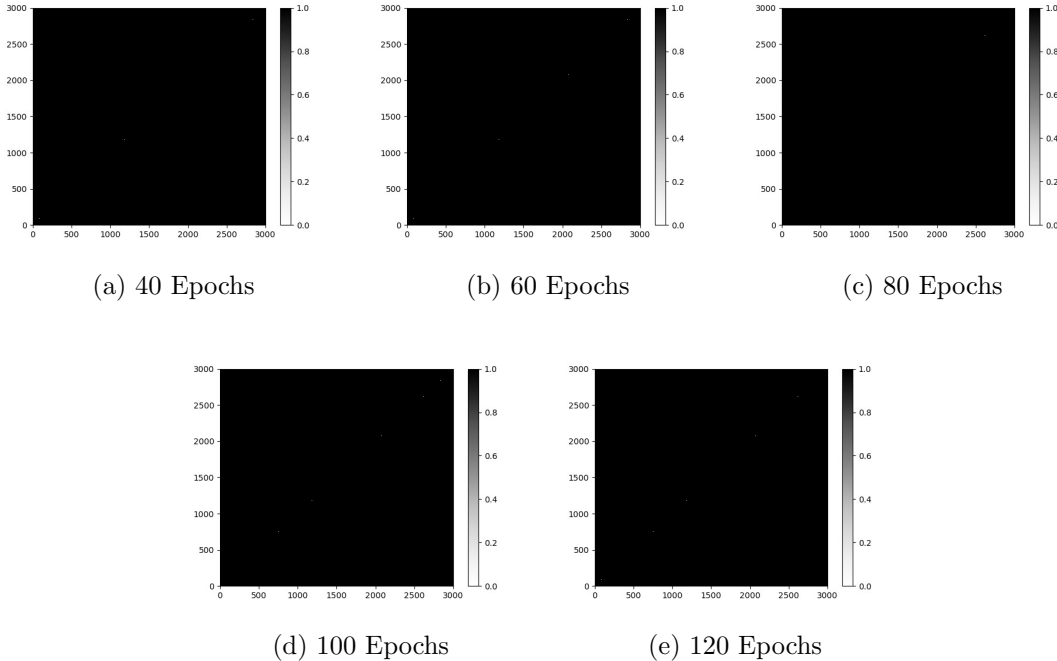


Figure 19: Intrinsic distortion of FactorVAE, on a random manifold of embedding dimension  $D = 100$ , intrinsic dimension  $d = 4$ , and training set size  $N = 3000$ . We also observed similar performance with BetaVAE and VanillaVAE.

to be *smooth* in order to discuss geodesics and curvature. Please see Lee (2012) for a comprehensive definition.

### B.2 Tangent space, extrinsic curvature

Akin to how smooth functions can be locally approximated by a linear function via its derivative, smooth manifolds can be locally approximated by a linear subspace using its *tangent space*; this is a construct commonly used in classical manifold learning. There are many ways to define the tangent space; we provide one here.

**Definition 10.** Let  $\mathcal{M} \subset \mathbb{R}^D$  be a manifold of dimension  $d$ , and  $x \in \mathcal{M}$ . The tangent space at  $x$ , denoted  $T_x\mathcal{M}$  is defined as the collection of all velocity vectors of smooth curves  $\gamma$  on  $\mathcal{M}$  from  $x$ :

$$T_x\mathcal{M} \doteq \left\{ \frac{d}{dt}\gamma(0) \mid \gamma : [0, 1] \rightarrow \mathbb{R}^D, \gamma(t) \in \mathcal{M}, \gamma(0) = x \right\}. \tag{43}$$

This is a linear subspace of  $\mathbb{R}^D$  of dimension  $d$ , and we call its orthogonal compliment  $N_x\mathcal{M} \doteq \{v \mid \langle v, w \rangle = 0 \text{ for all } w \in T_x\mathcal{M}\}$  the *normal space*. Thus  $T_x\mathcal{M}$  encodes a linear approximation to local movement around  $x$  on the manifold.

We now introduce curvature. Recall that for smooth functions, if the derivative is constant everywhere, i.e.  $\frac{d}{dx}f(x) = c$  for some  $c$ , then  $f$  is a globally linear function. We can equivalently conclude this if  $\frac{d^2}{dx^2}f(x) = 0$  everywhere. If the tangent space is analogous to the derivative, then the *extrinsic curvature* is analogous to the second derivative, and heuristically measures how much the tangent space is changing locally. There are again many definitions of extrinsic curvature, and one commonly used definition is through the *second fundamental form*:

**Definition 11.** Let  $\mathcal{M} \subset \mathbb{R}^D$  be a manifold of dimension  $d$ , and  $x \in \mathcal{M}$ . The second fundamental form at  $x$ , denoted  $\mathbb{I}_x$ , is the bilinear map  $\mathbb{I}_x: \mathbb{T}_x\mathcal{M} \times \mathbb{T}_x\mathcal{M} \rightarrow \mathbb{N}_x\mathcal{M}$  defined as the following:

$$\mathbb{I}_x(v, w) \doteq D_v \mathcal{P}_{\mathbb{T}_x\mathcal{M}}\{w\}, \quad (44)$$

where  $\mathcal{P}_{\mathbb{T}_x\mathcal{M}}$  is the orthogonal projector from  $\mathbb{R}^D$  onto  $\mathbb{T}_x\mathcal{M}$ , and  $D_v \mathcal{P}_{\mathbb{T}_x\mathcal{M}}\{w\}$  is the differential of the base point  $x$  along  $v$ .

Please see (Boumal, 2020, eq. (5.37)) for a more complete and rigorous version of the above definition. Intuitively,  $\mathbb{I}_x$  measures how much the tangent space changes locally. Indeed, if  $\mathbb{I}_x$  is identically 0 everywhere on  $\mathcal{M}$ , then the tangent space  $\mathbb{T}_x\mathcal{M}$  doesn't change anywhere, and  $\mathcal{M}$  is a globally linear subspace of dimension  $d$ . This motivates our use of  $\mathbb{I}$  to characterize the nonlinearity of a manifold  $\mathcal{M}$ , and motivates our definition of *flatness* and the following immediate corollary:

**Definition 12.** A manifold  $\mathcal{M} \subset \mathbb{R}^D$  is said to be flat if  $\mathbb{I}_x(v, w) = 0$  for all  $x \in \mathcal{M}$  and all  $v, w \in \mathbb{T}_x\mathcal{M}$ .

**Corollary 13.** If a manifold  $\mathcal{M} \subset \mathbb{R}^D$  of dimension  $d$  is flat, then  $\mathcal{M}$  is globally contained within an affine subspace of dimension  $d$ , i.e.,  $\mathcal{M} \subset \mathbb{T}_x\mathcal{M} + x$  for any  $x \in \mathcal{M}$ .

Finally, we introduce geodesics. One primary challenge of nonlinear datasets is the inability to interpolate via convex interpolation; indeed, for manifolds, the linear chord  $\gamma(t) \doteq (1-t)x + ty$  for  $x, y \in \mathcal{M}$  rarely stays in the manifold for all  $t \in [0, 1]$ . While there are typically many interpolations  $\gamma(t)$  between  $x$  and  $y$  that stay within  $\mathcal{M}$ , there is at most one that minimizes the path length: this is called the *geodesic* between  $x$  and  $y$ .

**Definition 14.** Let  $\mathcal{M} \subset \mathbb{R}^D$  be a manifold of dimension  $d$ . For any two  $x, y \in \mathcal{M}$ , if there exists a smooth curve  $\gamma: [0, 1] \rightarrow \mathcal{M}$  such that  $\gamma(0) = x$  and  $\gamma(1) = y$ , the geodesic between  $x$  and  $y$  is the interpolating curve of minimal arc length:

$$\gamma^* \doteq \underset{\substack{\gamma: [0,1] \rightarrow \\ \gamma(0)=x, \gamma(1)=y}}{\operatorname{argmin}} \int_0^1 \left\| \frac{d}{dt} \gamma(t) \right\|_2 dt. \quad (45)$$

For this paper, we assume for all manifolds  $\mathcal{M}$  that all pairs of points have geodesics between them; this is akin to assuming the manifold is connected.

## Appendix C. Geometric flows for manifold flattening

We now draw a relation between Algorithm 1 and a more traditional methods of manifold flattening: *geometric flows*. A common practice in differential geometry for manifold manipulation is to define a differential equation and study the evolution of  $\mathcal{M}$  through the corresponding dynamics. Such equations are often called *geometric evolution equations*, or *geometric flows*, and are typically designed to evolve complicated manifolds into simpler, more uniform ones. They are responsible for some powerful geometric theorems, such as the uniformization theorem, and are a heavily studied area in differential geometry.

Arguably the most well-known of the geometric flows is the *Ricci flow* (Hamilton, 1982). While commonly used in theoretical work, the Ricci flow is ill-suited for embedded submanifold flattening, as it is an open problem as to whether a realization of the Ricci flow even exists in the embedding space (Coll et al., 2020). There are also geometric flows that minimize extrinsic curvature (as opposed to intrinsic curvature), such as the *curve-shortening flow* (Abresch and Langer, 1986) and

*mean curvature flow* (Huisken, 1984). While there is rich theory behind both of these flows, each requires restricted settings which are non-ideal for data manifolds, with the former being defined only on curves ( $d = 1$ ) and the latter on hypersurfaces ( $d = D - 1$ ).

In this section, we introduce a new geometric flow that is well-defined on general embedded submanifolds of a Euclidean space, thus extending to more realistic settings for data manifolds. Our flow will flatten the input manifold  $\mathcal{M}$ . The encoding map  $\phi_{CC}: \mathbb{R}^D \rightarrow \mathbb{R}^D$  we seek to learn will then act on a point  $x_0 \in \mathbb{R}^D$  by approximating the flow starting at  $x_0$ .

### C.1 Convexification flow

Recall from Theorem 6 that flatness and convexity are intimately connected. Thus, our flow focuses on minimizing the volume of the difference between convex hull of the manifold and the manifold itself. Accordingly, we call it the *convexification flow*.

We now discuss the mechanics of how the flow should behave. In order to compress  $\mathcal{M}$  until it becomes convex, it should point the velocity vector of all points  $x_0 \in \mathcal{M}$  towards the boundary  $\partial \text{conv}(\mathcal{M})$ . However,  $\text{conv}(\mathcal{M})$  is challenging to compute in high dimensions (Erickson, 1996; Gale, 1963, Chp. 3), so the flow should only use local information that computes a direction from  $x_0$  towards  $\partial \text{conv}(\mathcal{M})$ .

Fix a smooth integrable function  $\psi: \mathbb{R}^D \rightarrow [0, 1]$ . Recall that the local average over  $\mathcal{M}$ , defined as

$$A_{\mathcal{M}}(x_0) \doteq \frac{\int_{\mathcal{M}} x \psi(x) \, dx}{\int_{\mathcal{M}} \psi(x) \, dx} \quad (46)$$

is not necessarily contained in  $\mathcal{M}$ , but when there is some nonzero curvature (i.e., the manifold is not locally flat) it is contained in the set difference  $\text{conv}(\mathcal{M}) \setminus \mathcal{M}$ . Thus, it seems that we can use local averages to direct the flow velocity towards the boundary of the convex hull.

More precisely, we define the following flow, which (as previously stated) we call the *convexification flow*.

**Definition 15** (Convexification Flow). *Let  $x_0 \in \mathcal{M}$ . The trajectory under the convexification flow  $x: [0, \infty) \rightarrow \mathbb{R}^D$  of  $x_0$  is given by the initial value problem (IVP):*

$$\frac{d}{dt} x(t) = \bar{x}(t) - x(t), \quad (47)$$

$$x(0) = x_0. \quad (48)$$

Here  $\bar{x}(t)$  is a short-hand for the local average of  $x(t)$  at time  $t$ :

$$\bar{x}(t) \doteq A_{\mathcal{M}(t)}(x(t)), \quad (49)$$

where  $\mathcal{M}(t)$  is the transformation of the original manifold  $\mathcal{M}$  under the flow until time  $t$ :

$$\mathcal{M}(t) \doteq \{x(t) \mid \text{eq. (47) holds for } x \text{ on } [0, t], x(0) \in \mathcal{M}\}. \quad (50)$$

Fix  $t \geq 0$ . Suppose that  $\mathcal{M}(t)$  is convex. Then if  $x(t)$  is in the interior  $\mathcal{M}(t)^\circ$ , we have  $x(t) = \bar{x}(t)$ . This suggests that the above flow is stationary, or halts, whenever  $\mathcal{M}(t)$  is convex. However, even if  $\mathcal{M}(t)$  is convex, the equality  $\bar{x}(t) = x(t)$  does not necessarily hold on the boundary  $\partial \mathcal{M}(t)$ . This phenomenon is important; as is the case with many unnormalized geometric flows, this ODE will evolve  $\mathcal{M}(t)$  into a singularity as  $t \rightarrow \infty$ . Thus, we need to normalize this flow.

### C.2 “Normalized” convexification flow

Analyzing the cause of singularity in (47), we see that fatal collapse only starts when  $\frac{d}{dt}x(t)$  points towards  $\mathcal{M}(t)$  itself. This can be alleviated by restricting the velocity vector  $\bar{x}(t) - x(t)$  to belong to the normal space  $N_{x(t)}\mathcal{M}(t)$  at  $x(t)$ . Thus, we define the following flow, which we call the *normalized convexification flow*.

**Definition 16** (Normalized Convexification Flow). *Let  $x_0 \in \mathcal{M}$ . The trajectory under the normalized convexification flow  $x: [0, \infty) \rightarrow \mathbb{R}^D$  of  $x_0$  is given by the following IVP:*

$$\frac{d}{dt}x(t) = \mathcal{P}_{N_{x(t)}\mathcal{M}(t)}\{\bar{x}(t) - x(t)\}, \quad (51)$$

$$x(0) = x_0. \quad (52)$$

Here  $\mathcal{P}_{N_{x(t)}\mathcal{M}(t)}$  is the orthogonal projection operator onto the subspace  $N_{x(t)}\mathcal{M}(t)$ , and  $\bar{x}(t)$  and  $\mathcal{M}(t)$  are defined analogously to before:

$$\bar{x}(t) := A_{\mathcal{M}(t)}(x(t)), \quad (53)$$

$$\mathcal{M}(t) := \{x(t) \mid \text{eq. (51) holds for } x \text{ on } [0, t], x(0) \in \mathcal{M}\}. \quad (54)$$

This flow does not completely avoid singularities, noting its similarities still to the unnormalized curve-shortening flow; we resolve this issue shortly when we discretize the flow, and in the next section when we make some algorithmic tweaks.

### C.3 Discretizing the normalized convexification flow

We now show how to recover a simplified version of Algorithm 1 by discretizing the convexification flow. Fix  $h > 0$  to be our discretization interval, and let  $k \geq 0$  be a non-negative integer. From here on, we will write  $x[k]$  to denote  $x(kh)$ , and similarly for  $\bar{x}$  and  $\mathcal{M}$ .

Let  $x: [0, \infty) \rightarrow \mathbb{R}^D$  evolve according to the normalized convexification flow in (51), (52). Our discretization will write  $x[k+1] = x((k+1)h)$  (approximately) in terms of  $x[k] = x(kh)$ . We use a first-order Taylor approximation around  $x[k+1]$ :

$$x[k+1] = x((k+1)h), \quad (55)$$

$$\approx x(kh) + h \left[ \frac{d}{dt}x(t) \right]_{t=kh}, \quad (56)$$

$$= x(kh) + h\mathcal{P}_{N_{x(kh)}\mathcal{M}(kh)}\{\bar{x}(kh) - x(kh)\}, \quad (57)$$

$$= x[k] + h\mathcal{P}_{N_{x[k]}\mathcal{M}[k]}\{\bar{x}[k] - x[k]\}, \quad (58)$$

$$= x[k] + h(\text{id}_{\mathbb{R}^D} - \mathcal{P}_{T_{x[k]}\mathcal{M}[k]})\{\bar{x}[k] - x[k]\}, \quad (59)$$

$$= x[k] + h\bar{x}[k] - hx[k] - h\mathcal{P}_{T_{x[k]}\mathcal{M}[k]}\bar{x}[k] + h\mathcal{P}_{T_{x[k]}\mathcal{M}[k]}x[k], \quad (60)$$

$$= (1-h)x[k] + h(\bar{x}[k] + \mathcal{P}_{T_{x[k]}\mathcal{M}[k]}\{x[k] - \bar{x}[k]\}), \quad (61)$$

$$= (1-h)x[k] + h\mathcal{P}_{T_{x[k]}\mathcal{M}[k]+\bar{x}[k]}\{x[k]\}, \quad (62)$$

with exact equality achieved as  $h \rightarrow 0$ . If we extend this step in (62) to a neighborhood around  $x[k]$ , fixing the projector  $\mathcal{P}_{T_{x[k]}\mathcal{M}[k]+\bar{x}[k]}$  with respect to the central point  $x[k]$ , then (62) matches the forward map defined in (4) and (9); in particular, the quantity  $h$  in (62) corresponds to the partition of unity  $\psi(x)$  in (9).

## Appendix D. Proofs

We provide below important proofs for theory included in the main body.

### Proof of Theorem 6

As we are primarily studying the set  $\phi(\mathcal{M})$ , we denote this manifold  $\mathcal{Z}$  for brevity.

1. This comes as an immediate corollary to the following geometric lemma (Lee, 2018, Proposition 8.12):

**Lemma 17.** *An embedded submanifold  $\mathcal{M} \subset \mathbb{R}^D$  is flat if and only if the geodesics of  $\mathcal{M}$  are geodesics in the embedding space  $\mathbb{R}^D$ .*

As the geodesics of Euclidean space are straight lines, it follows that the geodesics of  $\mathcal{Z}$  are likewise straight lines. Since  $h$  is smooth and  $\mathcal{M}$  is compact and connected, it follows that  $\mathcal{Z}$  is compact and connected, and each pair of points  $z_1, z_2$  have a geodesic between them. Thus, all point pairs  $z_1, z_2 \in \mathcal{Z}$  are connected by straight lines, making the set  $\mathcal{Z}$  convex.

2. This claim is proven in two parts: (a)  $\mathcal{Z} \subset z + T_z\mathcal{Z}$  for any  $z \in \mathcal{Z}$ , and (b)  $z_1 + T_{z_1}\mathcal{Z} = z_2 + T_{z_2}\mathcal{Z}$  for all  $z_1, z_2 \in \mathcal{Z}$ .

Fix an arbitrary  $z \in \mathcal{Z}$ . As established in part 1, all pairs of points in  $\mathcal{Z}$  are connected by a geodesic, and all geodesics of  $\mathcal{Z}$  are straight lines. Thus, all points  $z' \in \mathcal{Z}$  can be represented as  $z' = z + tv$  for some  $t \in \mathbb{R}$  and  $v \in T_z\mathcal{Z}$ , and it follows that  $\mathcal{Z} \subset z + T_z\mathcal{Z}$ .

For the second claim, note that  $T_{z_1}\mathcal{Z} = T_{z_2}\mathcal{Z}$  for all pairs of points  $z_1, z_2 \in \mathcal{Z}$ , as the second fundamental form is identically zero everywhere. Denoting the shared tangent space  $T_z\mathcal{Z}$ , it then suffices to show that  $z_1 - z_2 \in T_z\mathcal{Z}$  for all  $z_1, z_2 \in \mathcal{Z}$ . This is indeed the case, as the geodesics of  $\mathcal{Z}$  are of the form  $\gamma(t) = z_1 + t(z_2 - z_1)$ , and  $\gamma'(0) = z_2 - z_1$ .

3. The autoencoding property  $g(f(x)) = x$  for all  $x \in \mathcal{M}$  (and equivalently  $\|g(f(x)) - x\|_2 \leq \epsilon$  when there is error  $\epsilon$ ) holds trivially from construction, so what is left is to show is that there is no pair of functions  $f : \mathbb{R}^D \rightarrow \mathbb{R}^p$  and  $g : \mathbb{R}^p \rightarrow \mathbb{R}^D$  where  $g(f(x)) = x$  for all  $x \in \mathcal{M}$ , and  $p < d$ .

Denote  $Df(x)[v] := \lim_{t \rightarrow 0} \frac{f(x+tv) - f(x)}{t}$ , the differential of  $f$  at  $x$  along  $v$ . This notation is used to emphasize that  $Df(x)$  is a linear map. The following chain rule holds (Boumal, 2020):

$$D(g \circ f)(x)[v] = Dg(f(x))[Df(x)[v]]. \quad (63)$$

Using the autoencoding equality over  $\mathcal{M}$ , and the fact that  $D(\text{id})[v] = v$ , we get the following equality for any fixed  $x \in \mathcal{M}$ :

$$Dg(f(x))[Df(x)[v]] = v, \quad (64)$$

for any  $v \in T_x\mathcal{M}$ . Since  $T_x\mathcal{M}$  is a linear space of dimension  $d$ , it follows that the composite linear map  $Dg(f(x))[Df(x)[v]]$  must have rank of at least  $d$ . Since  $Df(x)[v]$  is a linear map from  $\mathbb{R}^D$  to  $\mathbb{R}^p$ , this implies that  $p \geq d$ .

When  $f, g$  are allowed error  $\epsilon$ , we can still rely on the required injectivity of  $f$  to ensure that  $p \geq d$ . For any  $x \in \mathcal{M}$ , we can use a coordinate chart of  $\mathcal{M}$  at  $x$ , denoted  $\xi : U \rightarrow \mathcal{M}$ , note that  $\xi \circ f$  must be locally invertible in some open subset of  $U$ , and apply the same argument as above.

■

### Proof of Lemma 7

Define  $\beta_z(x) := (1 - e^{-\lambda z} e^{-\lambda x})^2 x$ ,  $\mathcal{P}_{U+x_c}\{z\} := UU^\top(z - x_c) + x_c$ , and  $(I - \mathcal{P}_{U+x_c})\{z\} := z - \mathcal{P}_{U+x_c}\{z\}$ . Since  $\beta_z(x)$  is a strictly monotonically increasing function for  $x, z \geq 0$  (product of strictly monotonically increasing functions on positive input),  $\beta_z(x)$  is an invertible scalar function for all  $z \geq 0$ . Denote  $\eta_1 := \|\mathcal{P}_{U+x_c}\{z\}\|_2^2 = \|\mathcal{P}_{U+x_c}\{x\}\|_2^2$ , and  $\eta_2 := \|(I - \mathcal{P}_{U+x_c})\{z\}\|_2^2 = (1 - \psi(x))^2 \|(I - \mathcal{P}_{U+x_c})\{x\}\|_2^2$ . Further notate the original norms  $\nu_1 := \|\mathcal{P}_{U+x_c}\{x\}\|_2^2$  and  $\nu_2 := \|(I - \mathcal{P}_{U+x_c})\{x\}\|_2^2$ . Note that under this notation,  $\psi(x) = e^{-\lambda(\nu_1 + \nu_2)}$ . Since  $\nu_1 = \eta_1$ , we simply need to find an equation for  $\nu_2$  from  $\eta_1, \eta_2$ .

$$\eta_2 = (1 - \phi(x))^2 \|(I - \mathcal{P}_{U, x_c})x\|_2^2, \quad (65)$$

$$= (1 - e^{-\lambda(\nu_1 + \nu_2)})^2 \nu_2, \quad (66)$$

$$= (1 - e^{-\lambda\eta_1} e^{-\lambda\nu_2})^2 \nu_2, \quad (67)$$

$$= \beta_{\eta_1}(\nu_2). \quad (68)$$

Finally, we get the following equation for our partition of unity as a function of the output features  $z$ :

$$\psi(x) = \xi(z) := \alpha e^{-\gamma \left( \|\mathcal{P}_{U, x_c} z\|_2^2 + \beta^{-1} \frac{1}{\|\mathcal{P}_{U, x_c} z\|_2^2} (\|(I - \mathcal{P}_{U, x_c})z\|_2^2) \right)}. \quad (69)$$

Computing the above function amounts to inverting a scalar function, which reduces to scalar root-finding. ■

### Proof of Proposition 8

1. For convenience, let  $\alpha_{ij} = \langle \tilde{u}_j, \tilde{U}\tilde{U}^\top(x_i - x_0) \rangle = \langle \tilde{u}_j, x_i - x_0 \rangle$ . We write

$$\mathcal{L}_\psi(\tilde{U}, \tilde{V}) \tag{70}$$

$$= \frac{1}{N} \sum_{i=1}^N \psi(x_i)^2 \left\| \tilde{V}(\tilde{U}\tilde{U}^\top(x_i - x_0), \tilde{U}\tilde{U}^\top(x_i - x_0)) - (I - \tilde{U}\tilde{U}^\top)(x_i - x_0) \right\|_2^2, \tag{71}$$

$$= \frac{1}{N} \sum_{i=1}^N \psi(x_i)^2 \left\| \sum_{j=1}^d \sum_{k=1}^d \alpha_{ij} \alpha_{ik} \tilde{v}_{jk} - (I - \tilde{U}\tilde{U}^\top)(x_i - x_0) \right\|_2^2, \tag{72}$$

$$= \frac{1}{N} \sum_{i=1}^N \psi(x_i)^2 \left\| \tilde{U}\tilde{U}^\top \left( \sum_{j=1}^d \sum_{k=1}^d \alpha_{ij} \alpha_{ik} \tilde{v}_{jk} \right) - (I - \tilde{U}\tilde{U}^\top) \left( x_i - x_0 - \sum_{j=1}^d \sum_{k=1}^d \alpha_{ij} \alpha_{ik} \tilde{v}_{jk} \right) \right\|_2^2, \tag{73}$$

$$= \frac{1}{N} \sum_{i=1}^N \psi(x_i)^2 \left\| \tilde{U}\tilde{U}^\top \left( \sum_{j=1}^d \sum_{k=1}^d \alpha_{ij} \alpha_{ik} \tilde{v}_{jk} \right) \right\|_2^2 \tag{74}$$

$$+ \frac{1}{N} \sum_{i=1}^N \psi(x_i)^2 \left\| (I - \tilde{U}\tilde{U}^\top) \left( x_i - x_0 - \sum_{j=1}^d \sum_{k=1}^d \alpha_{ij} \alpha_{ik} \tilde{v}_{jk} \right) \right\|_2^2, \tag{75}$$

$$\geq \frac{1}{N} \sum_{i=1}^N \psi(x_i)^2 \left\| (I - \tilde{U}\tilde{U}^\top) \left( x_i - x_0 - \sum_{j=1}^d \sum_{k=1}^d \alpha_{ij} \alpha_{ik} \tilde{v}_{jk} \right) \right\|_2^2,$$

with equality if and only if  $\tilde{U}\tilde{U}^\top \left( \sum_{j=1}^d \sum_{k=1}^d \alpha_{ij} \alpha_{ik} \tilde{v}_{jk} \right) = 0$  for all  $i \in \{1, \dots, N\}$ , which since  $\tilde{U}$  are full rank is true if and only if  $\tilde{U}^\top \left( \sum_{j=1}^d \sum_{k=1}^d \alpha_{ij} \alpha_{ik} \tilde{v}_{jk} \right) = 0$ .

2. This amounts to noticing the cost function  $\mathcal{L}_\psi(\tilde{U}, \tilde{V})$  can be written as a least squares problem, and accounting for trivial redundancy. We rewrite the formula of  $\mathcal{L}_\psi(\tilde{U}, \tilde{V})$  here for convenience:

$$\mathcal{L}_\psi(\tilde{U}, \tilde{V}) \tag{76}$$

$$= \frac{1}{N} \sum_{i=1}^N \psi(x_i)^2 \left\| \tilde{V}(\tilde{U}\tilde{U}^\top(x_i - x_0), \tilde{U}\tilde{U}^\top(x_i - x_0)) - (I - \tilde{U}\tilde{U}^\top)(x_i - x_0) \right\|_2^2, \tag{77}$$

$$= \frac{1}{N} \sum_{i=1}^N \psi(x_i)^2 \left\| \sum_{j=1}^d \sum_{k=1}^d \tilde{v}_{jk} \langle \tilde{u}_j, x_i - x_0 \rangle \langle \tilde{u}_k, x_i - x_0 \rangle - (I - \tilde{U}\tilde{U}^\top)(x_i - x_0) \right\|_2^2. \tag{78}$$

Define  $B$  again as in the proposition statement:  $B_{ijk} = \langle \tilde{u}_j, x_i - x_0 \rangle \langle \tilde{u}_k, x_i - x_0 \rangle$ . We can flatten out the last two indices to get a matrix  $A \in \mathbb{R}^{N \times d^2}$ , such that  $A_{i, j+d \cdot k} = B_{ijk}$ . If we further stack the vectors  $\tilde{v}_{jk}$  into a matrix  $M_{\tilde{v}} \in \mathbb{R}^{d^2 \times D}$ , where the  $(j + d \cdot k)^{\text{th}}$  row is  $\tilde{v}_{jk}$ , and a matrix  $C \in \mathbb{R}^{N \times D}$  such that the  $i^{\text{th}}$  row of  $C$  is  $(I - \tilde{U}\tilde{U}^\top)(x_i - x_0)$ . Then we can write the following:

$$\mathcal{L}_\psi(\tilde{U}, \tilde{V}) = \|D_{\psi(x_i)} A M_{\tilde{v}} - D_{\psi(x_i)} C\|_F^2, \tag{79}$$

where  $D_{\psi(x_i)} \in \mathbb{R}^{N \times N}$  is the diagonal matrix such that  $D_{\psi(x_i)}_{ii} = \psi(x_i)$ .



An important problem with the above formulation is that  $A$  will never be full rank, since  $B_{ijk} = B_{ikj}$ ; this will lead to trivial duplicate entries in the rows of  $A$ . This is where the proposition's construction comes in: if we instead construct  $A' \in \mathbb{R}^{N \times \frac{1}{2}(d^2+d)}$  such that the  $i^{\text{th}}$  row is only the flattened upper diagonal component of  $B_i \in \mathbb{R}^{d \times d}$ , and further modify  $M'_v \in \mathbb{R}^{\frac{1}{2}(d^2+d) \times D}$  to only contain upper-diagonal entries of the list  $\tilde{v}_{jk}$ , scaling each off-diagonal entry by 2, then we can write the following:

$$\mathcal{L}_\psi(\tilde{U}, \tilde{V}) = \|D_{\psi(x_i)}A'M'_v - D_{\psi(x_i)}C\|_F^2, \quad (80)$$

where the matrix  $A'$  can now feasibly be full column-rank. As the above is a standard least squares problem, the solution  $M'_v$  (and equivalently the solution for  $\hat{V}$ ) is unique only when  $D_{\psi(x_i)}A'$  is of full column-rank, which since  $\psi(x_i) > 0$  for all  $x_i$  means  $\hat{V}$  is unique only when  $A'$  is full rank.

- Note that, using the notation of part 2., part 1. implies that  $A'M'_v\tilde{U} = 0$ . Since  $A'$  is of full column rank, this implies that  $M'_v\tilde{U} = 0$ , which implies  $\tilde{U}^\top \hat{v}_{jk} = 0$  for all  $1 \leq j, k, \leq d$ . Since any output of  $\hat{V}$  is a linear combination of  $\hat{v}_{jk}$ , it follows that  $\tilde{U}^\top \hat{V}(w_1, w_2) = 0$  for all  $w_1, w_2 \in \mathbb{R}^D$ . ■

## Appendix E. Algorithmic analysis

We provide here some algorithmic analysis to give the reader an idea for the computational burden that computing FlatNet entails. While we currently do not have rigorous computational guarantees, we can give ideas for scaling by providing critical point characterization, and time complexity of computing the cost function (including all network evaluations).

### E.1 Time complexity

To give the reader an idea of how FlatNet scales with data size and complexity, we provide some basic asymptotic time complexity analysis on the main bottleneck computation for constructing a FlatNet pair  $f, g$ : the optimization in (18). The following theorem encapsulates this time complexity analysis:

**Theorem 18.** *The loss function given in (18) of the main body can be computed in  $O(NDd^2)$  flops.*

**Proof** For reference, the cost function in question is the following:

$$\mathcal{L}_\psi(\tilde{U}, \tilde{V}) = \frac{1}{N} \sum_{i=1}^N \psi(x_i)^2 \left\| \sum_{j=1}^d \sum_{k=1}^d \tilde{v}_{jk} \langle \tilde{u}_j, x_i - x_0 \rangle \langle \tilde{u}_k, x_i - x_0 \rangle - (I - \tilde{U}\tilde{U}^\top)(x_i - x_0) \right\|_2^2. \quad (81)$$

The outer sum's size from (81) will be at most  $N$ , so we can incur an  $O(N)$  cost and focus our analysis on the summand for fixed  $x \in X$ :

$$\left\| \sum_{j=1}^d \sum_{k=1}^d \tilde{v}_{jk} \langle \tilde{u}_j, x_i - x_0 \rangle \langle \tilde{u}_k, x_i - x_0 \rangle - (I - \tilde{U}\tilde{U}^\top)(x_i - x_0) \right\|_2^2. \quad (82)$$

Since  $U \in \mathbb{R}^{D \times d}$ ,  $U^\top(x_i - x_0)$  is computable in  $O(Dd)$  flops, and the subsequent multiplication of  $U$  requires another  $O(Dd)$  flops. Thus,  $(I - \tilde{U}\tilde{U}^\top)(x_i - x_0)$  is computable in  $O(Dd)$  flops. For the remaining double sum  $\sum_{j=1}^d \sum_{k=1}^d \tilde{v}_{jk} \langle \tilde{u}_j, x_i - x_0 \rangle \langle \tilde{u}_k, x_i - x_0 \rangle$ , we can reduce to a single summand as before and incur a cost of  $O(d^2)$ , reducing time complexity analysis to the following:

$$\tilde{v}_{jk} \langle \tilde{u}_j, x_i - x_0 \rangle \langle \tilde{u}_k, x_i - x_0 \rangle. \quad (83)$$

Since  $\langle \tilde{u}_j, x_i - x_0 \rangle$  have already been computed from  $U^\top(x_i - x_0)$  and  $\tilde{v}_{jk} \in \mathbb{R}^D$ , the resulting expression requires  $O(D)$  flops to compute. Thus, the expression (82) requires  $O(Dd^2)$  flops to compute. The last uncomputed operation is the final  $L^2$  norm of the  $D$ -dimensional difference vector, which requires  $O(D)$  flops. Finally, the overall computation takes  $O(N(Dd^2 + Dd + D)) = O(NDd^2)$  flops. ■

## References

- U. Abresch and J. Langer. The normalized curve shortening flow and homothetic solutions. *Journal of Differential Geometry*, 23(2):175–196, 1986.
- S. Arora, A. Risteski, and Y. Zhang. Do gans learn the distribution? some theory and empirics. In *International Conference on Learning Representations*, 2018.
- M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.
- Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- T. Biršan and D. Tiba. One hundred years since the introduction of the set distance by dimitrie pompeiu. *System Modelling and Optimization*, 199:35–39, 2005.
- N. Boumal. An introduction to optimization on smooth manifolds. *Available online*, Aug, 2020.
- B. C. Brown, A. L. Caterini, B. L. Ross, J. C. Cresswell, and G. Loaiza-Ganem. The union of manifolds hypothesis and its implications for deep generative modelling. *arXiv preprint arXiv:2207.02862*, 2022.
- P. Campadelli, E. Casiraghi, C. Ceruti, and A. Rozza. Intrinsic dimension estimation: Relevant techniques and a benchmark framework. *Mathematical Problems in Engineering*, 2015:1–21, 2015.
- K. M. Carter, R. Raich, and A. O. Hero III. On local intrinsic dimension estimation and its applications. *IEEE Transactions on Signal Processing*, 58(2):650–663, 2009.
- K. Chan, Y. Yu, C. You, H. Qi, J. Wright, and Y. Ma. Redunet: A white-box deep network from the principle of maximizing rate reduction. *Journal of machine learning research*, 23(114), 2022.
- C. P. Chen. A rapid supervised learning neural network for function interpolation and approximation. *IEEE Transactions on Neural Networks*, 7(5):1220–1230, 1996.
- S. Chen, S. Chewi, J. Li, Y. Li, A. Salim, and A. R. Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215*, 2022.

- V. E. Coll, J. Dodd, and D. L. Johnson. Ricci flow on surfaces of revolution: an extrinsic view. *Geometriae Dedicata*, 207(1):81–94, 2020.
- J. A. Costa and A. O. Hero. Determining intrinsic dimension and entropy of high-dimensional shape spaces. In *Statistics and analysis of shapes*, pages 231–252. Springer, 2006.
- L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- J. Dieudonné. Sur les fonctions continues numérique définies dans une produit de deux espaces compacts. *Comptes Rendus Acad. Sci. Paris*, 205:593–595, 1937.
- J. G. Erickson. *Lower bounds for fundamental geometric problems*. University of California, Berkeley, 1996.
- E. Facco, M. d’Errico, A. Rodriguez, and A. Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):1–8, 2017.
- C. Fefferman, S. Mitter, and H. Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.
- C. Fraikin, K. Hüper, and P. V. Dooren. Optimization over the stiefel manifold. In *PAMM: Proceedings in Applied Mathematics and Mechanics*, volume 7, pages 1062205–1062206. Wiley Online Library, 2007.
- D. Gale. Neighborly and cyclic polytopes. In *Proc. Sympos. Pure Math*, volume 7, pages 225–232, 1963.
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
- D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.
- R. S. Hamilton. Three-manifolds with positive ricci curvature. *Journal of Differential geometry*, 17(2):255–306, 1982.
- K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.
- M. Hein and J.-Y. Audibert. Intrinsic dimensionality estimation of submanifolds in rd. In *Proceedings of the 22nd international conference on Machine learning*, pages 289–296, 2005.
- I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.
- G. Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

- J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.
- H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 19–26, 1993.
- G. Huisken. Flow by mean curvature of convex surfaces into spheres. *Journal of Differential Geometry*, 20(1):237–266, 1984.
- A. J. Izenman. Introduction to manifold learning. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(5):439–446.
- A. Jansen, G. Sell, and V. Lyzinski. Scalable out-of-sample extension of graph embeddings using deep neural networks. *Pattern Recognition Letters*, 94:1–6, 2017.
- H. Kim and A. Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.
- D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- I. Kobyzev, S. Prince, and M. Brubaker. Normalizing flows: An introduction and review of current methods, arxiv e-prints. *arXiv preprint arXiv:1908.09257*, 2019.
- M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.
- S. Lahiri, P. Gao, and S. Ganguli. Random projections of random manifolds. *arXiv preprint arXiv:1607.04331*, 2016.
- N. Lawrence and A. Hyvärinen. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of machine learning research*, 6(11), 2005.
- J. Lee. *Introduction to Smooth Manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer-Verlag New York, 2nd edition, 2012. doi: 10.1007/978-1-4419-9982-5.
- J. M. Lee. *Introduction to Riemannian manifolds*, volume 176. Springer, 2018.
- E. Levina and P. Bickel. Maximum likelihood estimation of intrinsic dimension. *Advances in neural information processing systems*, 17, 2004a.
- E. Levina and P. Bickel. Maximum likelihood estimation of intrinsic dimension. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004b. URL <https://proceedings.neurips.cc/paper/2004/file/74934548253bcab8490ebd74afed7031-Paper.pdf>.
- B. Li, Y.-R. Li, and X.-L. Zhang. A survey on laplacian eigenmaps based manifold learning methods. *Neurocomputing*, 335:336–351, 2019. ISSN 0925-2312. doi: <https://doi.org/10.1016/j.neucom.2018.06.077>. URL <https://www.sciencedirect.com/science/article/pii/S0925231218312645>.
- J. Li, L. Fuxin, and S. Todorovic. Efficient riemannian optimization on the stiefel manifold via the cayley transform. *arXiv preprint arXiv:2002.01113*, 2020.

- J.-B. Li, J.-S. Pan, and S.-C. Chu. Kernel class-wise locality preserving projection. *Information Sciences*, 178(7):1825–1835, 2008. ISSN 0020-0255. doi: <https://doi.org/10.1016/j.ins.2007.12.001>. URL <https://www.sciencedirect.com/science/article/pii/S0020025507005658>.
- T. Lin and H. Zha. Riemannian manifold learning. *IEEE transactions on pattern analysis and machine intelligence*, 30(5):796–809.
- J. Lucas, G. Tucker, R. Grosse, and M. Norouzi. Understanding posterior collapse in generative latent variable models. 2019.
- Y. Ma, D. Tsao, and H.-Y. Shum. On the principles of parsimony and self-consistency for the emergence of intelligence. *Frontiers of Information Technology & Electronic Engineering*, 23(9):1298–1323, 2022.
- L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.
- M. G. Monera, A. Montesinos-Amilibia, and E. Sanabria-Codesal. The taylor expansion of the exponential map and geometric applications. *Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales. Serie A. Matematicas*, 108(2):881–906, 2014.
- S. Oue. On asymptotics of local principal component analysis. *Hitotsubashi journal of commerce and management*, pages 1–11, 1996.
- H. Qiao, P. Zhang, B. Zhang, and S. Zheng. Learning an intrinsic-variable preserving manifold for dynamic visual tracking. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(3):868–880.
- Q. Qu, Y. Zhai, X. Li, Y. Zhang, and Z. Zhu. Geometric analysis of nonconvex optimization landscapes for overcomplete learning. In *International Conference on Learning Representations*.
- D. J. Rezende and F. Viola. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.
- R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.
- O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.
- W. F. Schmidt, M. A. Kraaijveld, R. P. Duin, et al. Feed forward neural networks with random weights. In *International conference on pattern recognition*, pages 1–1. IEEE Computer Society Press, 1992.
- A. Schwartz and R. Talmon. Intrinsic isometric manifold learning with application to localization. *SIAM Journal on Imaging Sciences*, 12(3):1347–1391.
- J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

- Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.
- G. Takács and D. Tikk. Alternating least squares for personalized ranking. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 83–90, 2012.
- J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.
- J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394, 2010.
- A. Vahdat, K. Kreis, and J. Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.
- A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.
- L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- B. Vandereycken. Low-rank matrix completion by riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.
- R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (gpca). *IEEE transactions on pattern analysis and machine intelligence*, 27(12):1945–1959, 2005.
- M. B. Wakin, D. L. Donoho, H. Choi, and R. G. Baraniuk. The multiscale structure of non-differentiable image manifolds. In *Wavelets XI*, volume 5914, page 59141B. International Society for Optics and Photonics, 2005.
- J. Wright and Y. Ma. *High-dimensional data analysis with low-dimensional models: Principles, computation, and applications*. Cambridge University Press, 2022.
- J.-W. Xu, A. R. Paiva, I. Park, and J. C. Principe. A reproducing kernel hilbert space framework for information-theoretic learning. *IEEE Transactions on Signal Processing*, 56(12):5891–5902, 2008.
- Y. Yu, K. H. R. Chan, C. You, C. Song, and Y. Ma. Learning diverse and discriminative representations via the principle of maximal coding rate reduction. *Advances in Neural Information Processing Systems*, 33:9422–9434, 2020.
- Y. Yu, S. Buchanan, D. Pai, T. Chu, Z. Wu, S. Tong, B. Haeffele, and Y. Ma. White-box transformers via sparse rate reduction. *Advances in Neural Information Processing Systems*, 36, 2024.
- Y. Zhai, H. Mehta, Z. Zhou, and Y. Ma. Understanding l4-based dictionary learning: Interpretation, stability, and robustness. In *International conference on learning representations*.
- Y. Zhai, Z. Yang, Z. Liao, J. Wright, and Y. Ma. Complete dictionary learning via l 4-norm maximization over the orthogonal group. *The Journal of Machine Learning Research*, 21(1): 6622–6689, 2020.

- J. Zhang, S. Z. Li, and J. Wang. Manifold learning and applications in recognition. *Intelligent multimedia processing with soft computing*, pages 281–300.
- P. Zhang, H. Qiao, and B. Zhang. An improved local tangent space alignment method for manifold learning. *Pattern Recognition Letters*, 32(2):181–189, 2011.
- Z. Zhang and H. Zha. Nonlinear dimension reduction via local tangent space alignment. In *Intelligent Data Engineering and Automated Learning: 4th International Conference, IDEAL 2003, Hong Kong, China, March 21-23, 2003. Revised Papers 4*, pages 477–481. Springer, 2003.
- N. Zheng, J. Xue, N. Zheng, and J. Xue. Manifold learning. *Statistical learning and pattern analysis for image and video processing*, pages 87–119.