Latent Embedded Graphs for Image and Shape Interpolation

Shantanu Vyas, Ting-Ju Chen, Ronak R. Mohanty, Peng Jiang

J. Mike Walker '66 Department of Mechanical Engineering, Texas A&M University

Vinayak R. Krishnamurthy*

J. Mike Walker '66 Department of Mechanical Engineering and Department of Computer Science and Engineering (by Affiliation), Texas A&M University

Abstract

In this paper, we introduce latent embedded graphs, a simple approach for shape and image interpolation using generative neural network models. A latent embedded graph is defined as a topological structure constructed over a set of lower-dimensional embedding (latent space) of points in a high-dimensional dataset learnt by a generative model. Given two samples in the original dataset, the problem of interpolation can simply be re-formulated as traversing through this embedded graph along the minimal path. This deceptively simple method is based on the fundamental observation that a low-dimensional space induced by a given sample is typically non-Euclidean and in some cases may even represent a multi-manifold. Therefore, simply performing linear interpolation of the encoded data may not necessarily lead to plausible interpolation in the original space. Latent embedded graphs address this issue by capturing the topological structure within the spatial distribution of the data in the latent space, thereby allowing for approximate geodesic computations in a robust and effective manner. We demonstrate our approach through variational autoencoder (VAE) as the method for learning the latent space and generating the topological structure using k-nearest-neighbour graph. We then present a systematic study of our approach by applying it to 2D curves (superformulae), image (Fashion-MNIST), and voxel (ShapeNet) datasets. We further demonstrate that our approach performs better than the linear case in preserving geometric and topological variations during interpolation.

Keywords: Geometric Interpolation, Variational Autoencoders, Generative Neural Networks, Image and Shape Morphing.

1. Introduction

Interpolating high dimensional data, especially images and shapes, is a fundamental task with applications in computer vision as well as geometric modeling such as 3D shape morphing [1], lip image sequence interpolation [2], magnetic resonance imaging (MRI) sequence interpolation [3], and video frame interpolation [4]. In the past decade or so, advances in generative neural network models, such as variational autoencoders [5] (VAEs) and generative adversarial networks (GANs) [6], have frequently used image and shape interpolations to demonstrate their efficacy to capture features from high-dimensional data. Recently, few works [7, 8, 9] have shed light on a fundamental issue with such interpolation tasks: the low-dimensional (latent) feature space captured by the generative model is seldom (if at all) linear. Therefore, a logical strategy proposed by these works is to use non-Euclidean metrics to interpolate between the data points in the latent space. However, these approaches do not explicitly account for the non-uniform, non-linear, and sometimes multi-manifold distribution of points in the latent space learnt by a given generative model.

In this paper, we first demonstrate, with examples, non-linear and multi-manifold behavior in a generative latent space. Subsequently, we introduce and systematically evaluate a simple approach for image and shape interpolation that imposes a topological structure (a graph) on the latent vectors and simply redefines interpolation as a graph traversal process between two points. We show that this deceptively simple approach results in a *smooth* (perceptually consistent) interpolation between two points sampled from the underlying high-dimensional space of the data-set. Our approach shows promising preliminary results for interpolations that require topological variations (e.g. change of genus from one shape to another).

1.1. The Problem in Detail

The problem of interpolating data (such as images) is typically translated into that of dimensionality reduction. Given some samples in an arbitrary space, the basic idea involves computing low-dimensional codes that capture the core features of the sample assuming the existence of some common metric that defines *similarity* or *closeness* of the original samples. Once projected on to the low-dimensional space, the samples can be treated as vectors amenable to simple (typically linear) algebraic operations. Given two samples in some high-dimensional space, a typical interpolation workflow involves the following pipeline: (1) Project data from the original space to the latent space; (2) Compute interpolated points linearly between the low-dimensional embedding of the data; (3) Project each interpolated point from the low-dimensional space back to the original space. We make two observations here.

Observation 1: . The data sampled from an underlying high-dimensional space concentrates close to a non-linear low-dimensional manifold [10]. This observation is supported by the

*Corresponding author

Email addresses: svyas@tamu.edu (Shantanu Vyas), carol0712@tamu.edu (Ting-Ju Chen), ronakmohanty@tamu.edu (Ronak R. Mohanty), maskjp@tamu.edu (Peng Jiang), vinayak@tamu.edu (Vinayak R. Krishnamurthy*)

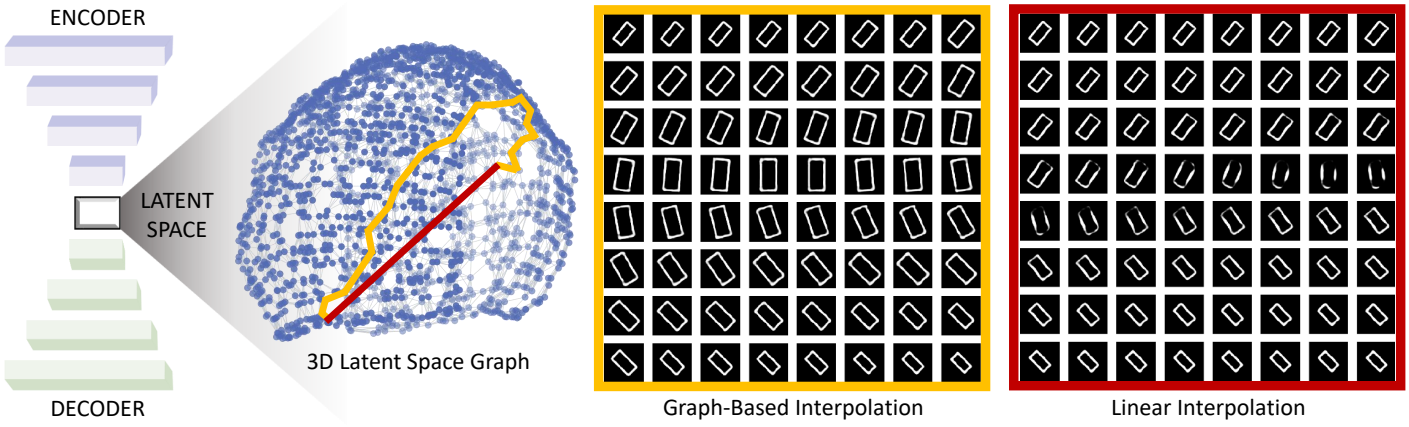

Figure 1: We showcase an outline of the Latent Embedded Graphs of a rectangle dataset for graph-based and linear interpolation techniques. We illustrate this for a set of images containing rectangles rotated by different angles in different frames. Two images are specified at the start and end of the interpolation path. Subsequently, we train a variational autoencoder and construct a graph on the datapoints (nodes) in the learned latent space. Nodes are represented in steel blue; and graph edges in light grey.

well-known manifold assumption that forms the basis of manifold learning techniques [1], also, attributes to the success of deep learning techniques [10]. In principle, manifold learning techniques are algorithms that "*recover a low-dimensional manifold embedded in a high-dimensional ambient space*" [1]. The primary assumption in these methods is that the data can be represented in a Euclidean sense after learning them. Generative neural network models, such as Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) [6], are able to learn a low dimensional latent distribution (latent space) and sample from this latent space back to the input space to generate or reconstruct data. These models usually follow a standard process for data reconstruction — the encoder first maps the high dimensional input data into a low-dimensional latent distribution; the decoder further samples from this latent distribution to reconstruct back into the input space. The data distribution in the latent space can also be controlled by designing special loss functions [10, 5, 11] that forces it to follow simple distributions such as Gaussian (normal) or uniform. This allows us to generate a data sequence that reflects the distribution of the data set in latent space by sampling from these low dimension simple distribution. However, real-world datasets do not follow simple distributions and the latent space might not accurately represent the real data space. In this paper, we will focus on using variants of VAEs to learn low dimensional latent distributions.

Observation 2: . While the intrinsic dimension of data is difficult to estimate and current estimators are not very accurate [12], it is a common practice to use a *sufficiently large* dimension for the latent space (e.g., the bottleneck layer in an autoencoder) to avoid missing many features. Even if the latent space dimension is the same as the intrinsic dimension of the data, it is not necessarily true that interpolations using linear algebraic operations will necessarily reflect the structure underlying the dataset. This is simply because the underlying structure of the data cannot be assumed to be Euclidean. Recent works support this idea [9, 8, 9] wherein the latent space learnt by generative models is modeled after some Riemannian Manifold such that distance between samples is computed as a geodesic path on the manifold. However, studies and one of our own experiments show that a dataset, when projected onto some latent space, may exhibit multi-manifold property [13].

1.2. Proposed Approach

We propose a simple approach which makes only one assumption: *the topological structure imposed on the points in the latent space must be fully connected such that every point has a non-empty neighborhood*. We call this structure the *latent embedded graph* to suggest a "*framework*" in the latent space that facilitates traversal from one point to another. Given two samples in the original dataset, the interpolation between the samples is implemented as computing and traversing the minimal-length path on this embedded graph. In order to systematically investigate this approach, we present several concrete examples using different variants of VAEs in order to map the high dimensional data onto a lower dimension latent space distribution. We specifically start with a case that results in a multi-manifold point distribution in the latent space. To capture the (potentially multi-manifold) structure of the space underlying the data distribution in latent space, we construct a graph in the latent space. In our experiments, we used the k nearest neighbor (k-NN) graph. k-NN graphs are generally known to robustly capture neighborhoods of point distributions [14, 15, 16]. Additionally, we constrain the choice of the specific value of k to allow for the graph being connected (to ensure a path between any pair of points). In addition, we study an adaptive graph construction method as outlined by Lin et al [17]. Finally, we demonstrate several cases of shape and image interpolation.

We make three contributions in this paper. First, we propose a general computational framework for interpolation that does not assume the structure of the data to be either Euclidean or single manifold. Second, we systematically investigate our interpolation method through two different topological structures (k-NN and adaptive graphs). Finally, we introduce a new measure to compute the smoothness of interpolation by leveraging the variation of image and shape gradients between any two consecutive interpolated data points. Using our measure and the well-known earth-mover's distance metric, we conduct a series of evaluations of our method in comparison to linear interpolation. Our results demonstrate that this method results in better interpolation in comparison to the linear method, especially when capturing topological changes between the interpolated samples.

2. Background

2.1. Image and Shape Morphing in Computer Graphics

The problem is interpolation is generally identified with image and mesh morphing in computer graphics and vision literature. Traditional image morphing algorithms are based on intensity blend, warp generation, feature specification, and transition control methods [18, 19]. More recently, image interpolation techniques such as nearest neighbors, bi-linear and bi-cubic interpolation have been used for image magnification applications [20]. Other recent works in image morphing improve on traditional methods by finding different ways to automate the morphing process (by automatically finding mappings between corresponding image elements) [21, 22]. However, in most cases, the demonstration of interpolation is typically on images that share strong structural similarity (for example: interpolating human faces or flowers). Even methods that can interpolate between dissimilar images [22], result in morphed images that contain elements from only the two input images.

In the case of mesh morphing, the traditional methods involve the process of finding correspondence between meshes, generating new mesh connectivity and finding vertex paths between the meshes [23]. Turk et al. represent the shapes as implicit functions, and conduct the interpolation between the two functions [24]. Whereas, Zefran et al. are able to generate smooth rigid body motions by interpolating between the rigid body displacements in 3D space [25]. More advanced approaches include shape interpolation methods which are based on the Poisson equation [26] where the traversal path is treated as a set of Poisson Equations on a manifold mesh. Recent works that take a data-driven approach [27, 28] improve upon the quality of the morphs. However, similar to image morphing, these approaches focus on a single class of shapes/meshes, such as human poses, surfaces homeomorphic to spheres in prior works [29], and so on. In contrast, our approach for interpolation is *generative* and leverages recent developments in deep neural networks. This statistical data-driven strategy allows for generalizing beyond morphing approaches that either rely on the data being from a specific class (faces etc.) or require specialized algorithms. Furthermore, we are specifically interested in improving upon the weaknesses of the current generative deep learning based approaches.

2.2. Manifold Learning for Interpolation

Manifold learning techniques have been used in a wide range of applications [1] such as shape morphing [30, 1], image recognition [31, 32, 33, 34], and motion analysis [35, 36]. There are several works that borrow from the data manifold assumption [10] to achieve interpolation for a given data-set. One of the early works by Bergler et al. [2] showcase a projected image of a lip onto a 10-dimensional linear subspace and then induce a non-linear interpolation algorithm by traversing on this manifold. Similarly, Souvenir et al. [3] developed a method for biomedical image deformation analysis which incorporates multiple video data simultaneously in order to facilitate inter-image interpolation by utilizing an image-based manifold. Similarly, for works discussing 2D shape analysis, Ma et al. [1] represented a class of closed curves using functions [30], thus, imposing a Riemannian manifold structure on the curve collection using an elastic metric. The purpose of the metric was to conduct interpolation using a modified De Casteljau or Aitken-Neville algorithm. There are works discussing the facial surface representation as an indexed collection of facial curves. The geodesic distance on the facial shape manifold was specifically designed and utilized Local Binary Pattern (LBP) histograms and image graph to generate face animations from image collections of the same person [37].

2.3. Interpolation with Generative Models

Generative neural network models such as variational autoencoders [5] and generative adversarial networks (GANs) [6] provide a rich toolbox to address some of the problems with the previous approaches. These models are able to generalize to arbitrary shapes and images across a variety of classes. They promisingly showcase the ability to learn low dimensional latent codes and map them back to their input space, thus, allowing a smooth interpolation of shapes and images by interpolating in their latent codes and subsequently decoding the interpolated latent codes to obtain the interpolants in the original space. For example, Nguyen et al. [4] use an auto-encoder structure to create a deep and locally linear embedding to interpolate between the video frames. Zhu et al. [38] discuss image manipulation methods by learning a manifold using GANs and constraining the manipulations on the learnt manifold to preserve realistic effects. More recently, smooth and realistic interpolation has been achieved by traversing the latent space non-linearly by using the Wassertein Barycenter Problem to generate the traversal path [39].

In a way, an autoencoder can be considered as a dimensionality reduction method [40] using artificial neural networks. In that sense, one can observe the similarity between generative models and classical manifold learning techniques. An advantage of using neural network models is that by combining different kinds of layers, the network can be adapted to different kinds of data. For example, convolutional layers [41, 42] are highly effective in detecting features in images, and recurrent layers can be used for sequential data [43, 44]. Further, there are many variations of autoencoders with varying approaches towards a latent space distribution. For instance, Variational Autoencoder [5] learns a latent variable model for its input data and captures a more compact latent space than regular autoencoders. On the other hand, Wasserstein autoencoder minimizes a penalized form of the Wasserstein distance between the model distribution and the target distribution [45]. Alternatively, adversarial autoencoders combine the adversarial network and autoencoders, and have the ability to get better regularized latent code [46]. These variations of autoencoders can be used to learn a lower dimensional manifold of arbitrary high dimensional data, and has been discussed extensively in existing works for the geometric modeling domain [47, 48, 49, 50, 51, 52] . One of the seminal works in this area by Yumer et al. [51] demonstrates how VAEs can be used to encode parametric shapes in order to enable a new class of procedural geometric modeling workflow. We observe that in principle, learning a lower-dimensional embedding for a higher dimensional data allows for one to perform algebraic operations in the lower-dimensional space (the latent space) and project the outputs of these operations back to the original space. Interpolation can be regarded as one of such algebraic operations.

2.4. Our Approach

Our work draws inspiration from recent works [7, 9, 8] that discuss and treat the learned manifolds of generative neural network models as Riemannian manifolds. They further, develop algorithms to find the geodesics between two points on the manifold, thus, enabling interpolation along this path. The key difference between these works and ours is that in addition to considering the non-linearity of the latent spaces learnt by generative

models, we also remove the single manifold assumption made in prior works.

3. Methodology

In our work, we aim to systematically explore the interpolation on the latent embedded graphs of a variety of datasets using generative neural network models to learn their latent space. A variety of generative neural network models exist [53] such as variants of autoencoders, generative adversarial networks (GANs) and so on. While our approach is agnostic to a specific generative model, we chose the autoencoder as our overarching architecture for this work. In contrast to GANs, autoencoders allows us to directly query the latent embedding for a given sample in the original space using the encoder. Therefore, they offered an intuitive way to conduct systematic studies for a wide range of inputs. In this section, we describe our methodology for latent embedded graph based interpolation and define the metrics for the experimental evaluation of our approach.

3.1. Overview

Given a dataset in its original dimension, $X = \{x_0, x_1, ..., x_n\}$, and a pair of specified data points (x_s, x_t) from the dataset we compute the interpolated points in the following steps:

Step 1. We construct a probabilistic Autoencoder, consisting of an encoder ($\phi : \mathcal{X} \to \mathcal{F}$ and a decoder $\psi : \mathcal{F} \to \mathcal{X}$. The encoder ϕ maps the dataset X into a low-dimensional latent space distribution. We refer the encoded data set as $Z = \{z_0, z_1, ..., z_n\}$ and the encoded data points as (z_s, z_t).

Step 2. We construct a neighborhood graph to capture the geometric structure of the encoded dataset $G(V, E)$, where V represents the set of vertices and E represents the set of edges between two vertices.

Step 3. We find the shortest path $P((v_s, v_t)) = \{\bar{e}_{s1}, \bar{e}_{12}, ..., \bar{e}_{pt}\}$ between the encoded pair of data points (z_s, z_t).

Step 4. We linearly interpolate on each $\bar{e}_{ij}$ of the path and get a set of interpolated points $\bar{Z} = \{\bar{z}_0, ..., \bar{z}_s\}$.

Step 5. We use the decoder to reconstruct the interpolated points to the original space X

The method used in each step is further discussed in detail in subsequent sections. We specifically look into the methods that help us construct: (1) the *low-dimensional latent space* from the original dataset, (2) the latent embedded graph that captures the structure of the encoded data, and finally, (3) the *path and interpolants* which represent smooth transition between any two nodes from the constructed graph.

3.1.1. ***Learning the Latent Space***

We design different architectures of autoencoders for different datasets based on the representation and content of the dataset. For instance, we developed a synthetic dataset of geometric curves (wherein the representation is geometric) for which we used fully-connected dense layers. On the other hand, for image data, we explored both dense layers (for vector representations of images) and 2D convolutional layers which is an effective and memory efficient way to train images [54]. A similar approach was used for 3D voxel data using 3D convolutional layers. Specific architecture details for each experiment we conducted are provided in the following sections (Section. 4).

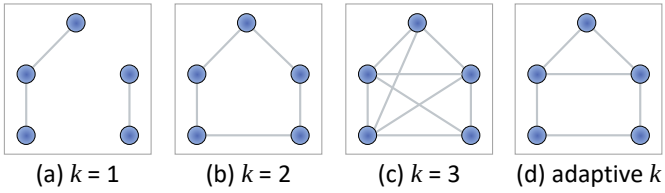

Figure 2: Illustration of k nearest neighbor (k-NN) graphs with (a) $k = 1$; (b) $k = 2$ (k_{min}); (c) $k = 3$; and (d) adaptive k.

In addition to the architecture of the autoencoder, the choice of the dimension of the latent space is also important. Ideally, it should be equal to the intrinsic dimension of the data in order to capture the inherent structure [12]. However, intrinsic dimension estimate is still an open research problem, with many methods being introduced lately [12, 55]. In our experiments, for the datasets generated by us, we already know the intrinsic dimension (*id*) of the datasets, and the estimated latent space dimension is chosen as a number that is equal or larger to it (eg. the *id* for the 2D superformula data is 3). For cases where we did not know the *id*, we developed the model based on a trial and error method.

3.1.2. ***Graph Construction***

Graph construction plays a significant role in our method because it is the basis for generating the interpolated points. The most widely used neighborhood graph construction methods include the k-nearest neighbors algorithm and the ϵ-close graph algorithm. ϵ-close graph algorithm can always construct a symmetric graph, but it also ends up creating too many edges when the data is densely distributed. On the other hand, the k-nearest neighbor algorithm does not always produce a symmetric graph, but the number of edges can be controlled. Therefore, in this work, we choose k-nearest neighbor algorithm to construct the graph in the latent space to make sure the model is capturing the distribution of the data while maintaining a reasonable amount of the edges being produced. Here, the selection of k is important for two main reasons. First, if k is too large, it may lead to smoothing or elimination of small-scale structures in the manifold, which hurts the graph's ability to capture the geometric structure of the distribution thus makes the interpolations to be non-sample-aware. Second, if k is too small, it results in disjoint graphs [56]. To this end, Carey [57] proposed metrics to evaluate the graphs used in manifold learning techniques and further pointed out that the diameter of the graph suddenly changes when k becomes too large. We explore two different strategies to resolve the issue of the selection of k as mentioned below.

Strategy 1: k_{min}-NN Graphs. The first strategy is when the value of $k \in \mathcal{Z}^+$ is fixed for the entire graph. Here, we simply begin with our original assumption that the graph must be connected. Therefore, our choice of k is the minimum possible positive integer for which the graph is connected. We will call this the k_{min}-NN graph.

Strategy 2: Adaptive k-NN Graphs. The second (and more involved) strategy is to select k adaptively so as to avoid long edges and extraneous connections between points where using Euclidean distance would not be appropriate. Typically, manifold-based methods have been introduced to achieve this [58, 17, 59]

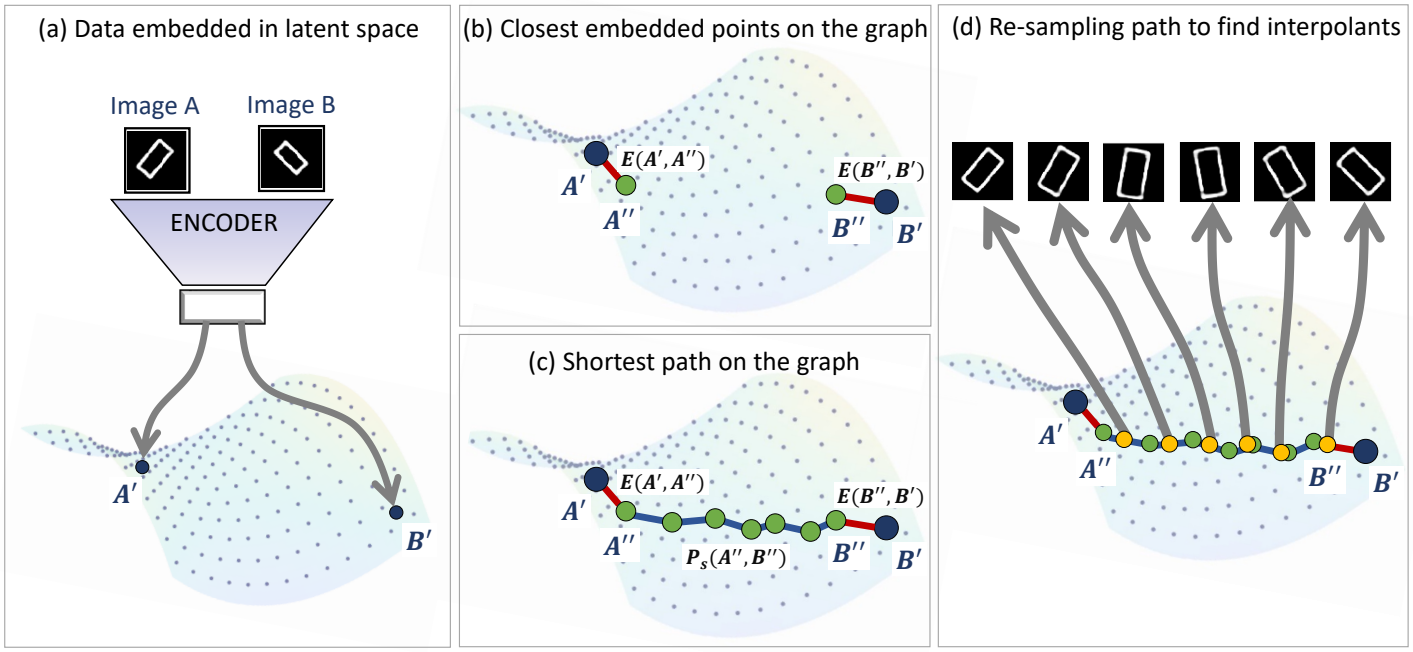

Figure 3: Our interpolation pipeline: (a) Image A and Image B are the inputs to the encoder to obtain the respective embeddings A' and B' in the latent space; (b) Two vertices A'' and B'' on the latent embedded graph that are closest to A' and B', respectively, are found (red path highlights the edges $E(A', A'')$ and $E(B'', B')$) (c) shortest path $P_S(A'', B'')$ (blue path) between them (A'' and B'') is computed using a piece-wise linear approach along the constructed graph. (Green points are existing vertices of the graph); (d) The shortest path is re-sampled to obtain the interpolants (yellow points) which are provided to the decoder to obtain interpolated images

and to improve the quality of the graph. It is found that the adaptive graph decreases the number of edges while capturing the distribution properties of the data (Figure 2). Therefore, we utilize and follow the adaptive graph construction strategy described by Lin at al. [17] in this work, to build the *Latent Embedded Graph*. The main steps described in their construction procedure are as follows:

Step 1. Search for k-NNs for our original dataset $X = \{x_0, x_1, ..., x_n\}$, denoted as k-NN (x_i).

Step 2. Construct a k-NN graph using $k \geq k_{min}$ to ensure that each x_i is connected to each of its k-NN.

Step 3. Identify the "visible" neighbors $VN(x_i) = \{y \in k{-}NN(x_i) | \langle x_i - z, y - z \rangle \geq 0, \vee z \in k{-}NN(x_i)\}$ of x_i and maintain connections only between them; eliminating any connections between the "invisible" neighbors. Here, y is a visible neighbor of x_i if there is no other point z between them i.e. in the "visible" neighborhood of x_i.

Step 4. Remove the *short-circuit* neighbors, which usually produce longer edges than others, thus, obtaining a safe neighborhood devoid of edges abruptly affecting the k-NN graph topology. Using PCA, eliminate the neighbors that affect the local linearity.

It is important to note here that adaptive graph construction method takes relatively more time than constructing a fixed connected graph. Specifically, the complexity for constructing a fixed k-NN graph is $O(|V|^2(n+k))$, whereas for the adaptive graph it is $O(|V|^3(K + \log(V))$ if using the graph diameter to chose the reasonable value of k. Here, $|V|$ is the number of vertices, and n is the dimension of the data. Our approach adopted from Lin et al. [17] has the complexity of roughly $O(|V|^2(n + K))$.

3.1.3. *Interpolation Strategy*

Finding the shortest path between two nodes in a manifold is an essential step in traversing and hence, interpolating on the manifold structure [1]. Even though we do not assume a single manifold structure in this work, the shortest path strategy for interpolation still holds. For a discrete topological structure such as our latent embedded graph, the shortest graph path between two nodes serves as a counterpart for the geodesic. Many methods have been introduced to compute such a path efficiently. For example, a widely used method, Floyd-Warshall algorithm [60], is suitable for a dense graph where the number of edges is close to the maximum number of vertices. In our work, we adopt Dijkstra's algorithm [61] for one main reason: the graph representation of the manifold is always a sparse graph since the k of our adaptive k-NN graph is usually much smaller than the total number of vertices.

In order to interpolate between two new data points, say A and B which are not present in the original dataset, we first use the encoder to compute their latent embeddings A' and B' respectively (Figure 3(a)). We then find two vertices A'' and B'' on the latent embedded graph that are closest to the newly embedded points A' and B' respectively (Figure 3(b)). In our next step, we compute the shortest path, $P_S(A', B')$, between A'' and B'' by sequentially concatenating the edge $E(A', A'')$ with path $P_S(A'', B'')$, followed by edge $E(B'', B')$ (Figure 3(c)). Here, $P_S(A'', B'')$ is the shortest path between A'' and B'' that is readily available from the latent embedding graph. As a result, $P_S(A', B')$ is a piece-wise linear approximation of the geodesic between A' and B' (our new embedded points from the data points A and B). Note that, once computed, $P_S(A', B')$ can be arbitrarily re-sampled based on however many interpolants that are needed. In our work, we perform equidistant re-sampling of $P_S(A', B')$ (Figure. 3(d)). Alternately, it is also possible to use $P_S(A', B')$ to generate high-

resolution polynomial curves [62, 63].

3.2. Evaluation Metrics

While there are a few metrics that have been proposed to quantitatively measure the quality of an interpolation, there are some key considerations that we need to address in our case. First and foremost, we note that we are specifically interested in how the interpolation affects the shape (both geometry and topology). Second, we desire the interpolations to be smooth in the sense that there are minimal abrupt changes or transitions between two consecutive interpolated images. More importantly, it is necessary for the interpolations to be complete, which means that for two input images with the same topology (e.g. both homeomorphic to a disc), the topology of the interpolated images should be preserved. Finally, in order to have meaningful interpolations, we require perceptually consistent transitions between the interpolated images. We experimented with several metrics available in literature [7, 8, 9] and chose the Earth Mover's Distance [64] based on its consistency. We also developed our own evaluation metric, the Gradient Projection Metric (GPM), that encodes shape changes across the interpolated images.

The Earth Mover's Distance (EMD), also known as the Wasserstein Distance, calculates the distance between probability distributions and is used to compare visual similarities between images and voxels [64, 65]. We specifically use the EMD variants described in [66, 67], between every two consecutive frames of the interpolations. For smooth transitions, the EMD values will be fairly small, and will show larger values for abrupt transitions. It will also show larger values when there's loss of information between frames, hence satisfying our desired characteristic of complete interpolations. Lack of persistent transitions may correspond to consecutive EMD values to be very similar, showing a smooth curve when plotted over all the interpolations.

Consider a pair of images I_t and I_{t+1} at time frames t and $t+1$, respectively, each having a size of $m \times n$ pixels. The **Gradient Projection Metric (GPM)** is computed as follows:

Step 1. Compute gradient vectors $V_t(i, j)$ and $V_{t+1}(i, j)$ at pixel (i, j).

Step 2. Compute the difference between the product of the vector magnitudes and their dot products.

$$G(i, j) = (\mid V_t(i, j) \mid \times \mid V_{t+1}(i, j) \mid) - s_t^{t+1}(i, j)$$

where $s_t^{t+1}(i, j)$ is the dot product of $V_t(i, j)$ and $V_{t+1}(i, j)$.

Step 3. Calculate the root mean square (RMS) of this difference and the value we obtain from this, is our evaluation metric.

$$G_{RMS} = (\frac{1}{N}(\sum_{i=0}^{m}\sum_{j=0}^{n} G(i, j))^2)^{\frac{1}{2}}$$

where, $N = m \times n$

The aforementioned metric is used primarily for image-based datasets (§ 4.2.2, § 4.3). However, for 3D voxel datasets (§ 4.4), we compute 3D gradient vectors in the first step. GPM is based on the rationale that it will produce smaller values for smooth transitions between consecutive interpolations. This is due to the reduced movements (rotations) of pixels/voxels between the frames. We expect larger values for cases with abrupt transitions between two interpolations. This will help us highlight incomplete transitions when the pixel/voxel values in consecutive interpolations change and affect the gradient vector values, thus, being a good indicator for a complete and perceptually smooth interpolation process. For our image and 3D voxel based experiments, GPM is calculated over hundred interpolations for each interpolation method in the experiments, thus, allowing us to observe the transitions in greater detail.

4. Experiments and Results

We conduct several experiments on different image and shape-based datasets, to evaluate our latent embedded graph methods compared to linear interpolation (current standard in latent space interpolation for generative models). To better understand the distribution of data in the latent space, we generate two types of shape datasets using the parametric equation for Superformula [68]. This allows us to generate a variety of shapes parametrically, making the parameters a learnable feature of the dataset. Additionally, we conducted experiments on the Fashion-MNIST [69] and ShapeNet [70] datasets, which provide a wide variety of data to evaluate our interpolation methods.

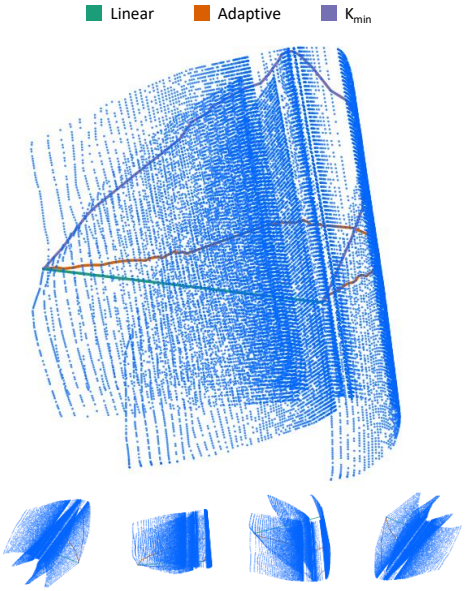

Figure 4: Latent space visualization for the the curve-based superformula dataset generated in *§ 4.1*, with the three interpolation paths. Multi-manifold property of the latent space can be observed in this figure. Also, different views of the structure in the bottom row.

4.1. 2D Superformula: Geometric Shape Interpolation

Superformula [68] is a generalization of the superellipse and its equation is given below:

$$r(\phi) = \left(\left|\frac{\cos(\frac{m_1\phi}{4})}{a}\right|^{n_2} + \left|\frac{\sin(\frac{m_2\phi}{4})}{b}\right|^{n_3}\right)^{\frac{1}{n_1}}$$

$$x = s \times r(\phi) \times \cos(\phi)$$

$$y = r(\phi) \times \sin(\phi)$$

Experiments	Superformula (Polygonal)	Superformula (Image)		Fashion-MNIST	ShapeNet
Training Dataset	28,282	4,000	28,282	60,000	3991
Data Representation	Polygons (1000 2D points per polygon)	Image (128×128 pixels)	Image (128×128 pixels)	Image (28×28 pixels)	Voxels ($32 \times 32 \times 32$ voxels)
Autoencoder Model	AE	CVAE	CVAE	AAE	CVAE
Encoder Arch.	4 FC	5 Conv, 4 FC	5 Conv, 4 FC	3 Conv, 1 FC	4 Conv, 2 FC
Latent Space Dim.	3	3	3	100	100
Decoder Arch.	4 FC	4 FC, 6 Conv	4 FC, 6 Conv	1 FC, 3 Conv	2 FC, 4 Conv

Table 1: A summary of all experiments conducted in this work. The table highlights: Size of the Training Datasets, Type of Data Representation, Type of Autoencoder models used, Encoder Architecture, Latent Space Dimension, and Decoder Architecture. (**Abbreviations:** AE: Autoencoder; CVAE: Convolutional Variational Autoencoder; AAE: Adversarial Autoencoder; FC: Fully Connected Layers; Conv: Convolutional Layers)

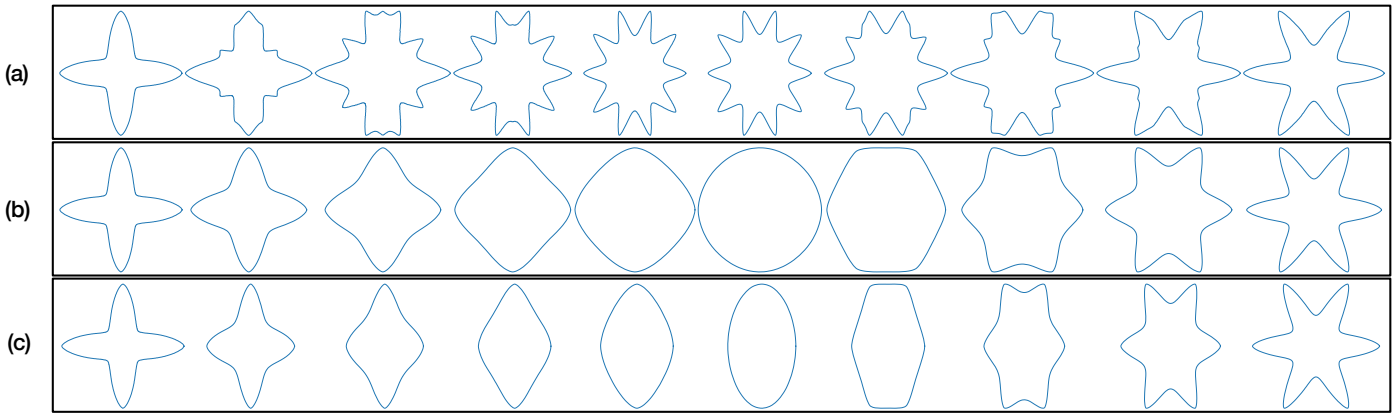

Figure 5: Interpolation results for experiments on curve-based superformula dataset: (a) Linear interpolation between two given curves is jittery; Interpolation along the: (b) k-NN graph with k_{min} generates smooth and wide intermediate curves; (c) the adaptive k-NN graph generates smooth and narrower intermediate curves

Here, in the above equation, r and ϕ are the radius and angle of the superformula shape respectively. The parameter m defines the number of corners of the superformula shape and is set to an integer value to obtain complete polygons. Parameters n_2 and n_3 determine if the shape is inscribed or circumscribed and parameter n_1 can make the corners sharper or flatter and the edges straight or curved. Parameters a and b are usually kept constant and equal to 1, giving us circular shapes. We further introduce parameter s that can make the shapes narrower or wider. Many shapes can be generated by changing the parameter values, and we use this property of the superformula to generate multiple datasets by varying different parameters. This allows us to observe and have a better understanding of the differences in the latent space structures for the different datasets. We discuss these experiments in further detail in the following sections.

Dataset. For this experiment, we generate a new dataset of $28,282$ superformula shapes, represented as curves. Each curve is a sequence of points (coordinates) and is assumed to be closed. Here, the shapes are generated by changing the parameters s, n_1 and m (here, $m_1 = m_2 = m$). Specifically, the value of s is varied from 0.1 to 1.0 and n_1 from 0.1 to 10. Parameter m is allotted four values: $4, 6, 8$ and 10. The other parameters are kept constant and are given the following values: $a = b = 1.0$; $n_2 = n_3 = 1.7$.

Model Architecture. We train a simple autoencoder on our dataset, where the encoder contains four fully connected layers, and the decoder mirrors this structure. Since three parameters are being changed to generate the dataset, its intrinsic dimension is 3 and therefore we set the latent space dimension to 3. We use a simple autoencoder for this experiment due to the simplistic nature of the dataset.

Findings. Since the value of m is set to 4 discrete values, the latent space structure of the dataset should ideally be discrete too. However, when s and n_1 are large enough (i.e., the value of s is close to 1.0 and n_1 is close to 10.0), the generated shapes are ellipsoidal for all values of m. Therefore, each class of the shapes may correspond to a single manifold structure and they

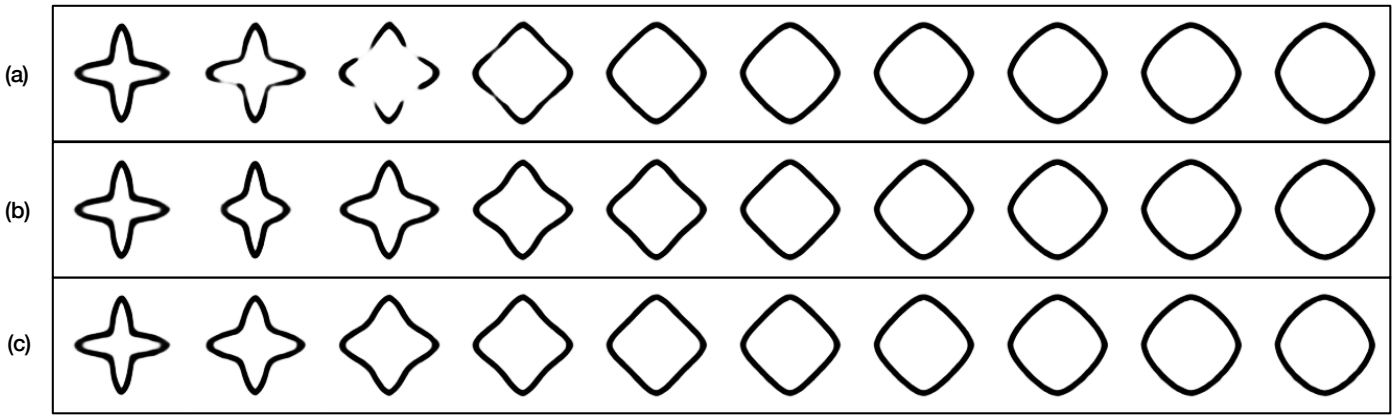

Figure 6: Interpolation results for experiments on image-based superformula dataset (with 2 variables): (a) Linear interpolation generates a few incomplete images; Interpolation along: (b) k-NN graph with k_{min} generates complete but narrow images in the beginning; (c) adaptive k-NN graph generates complete and smooth images

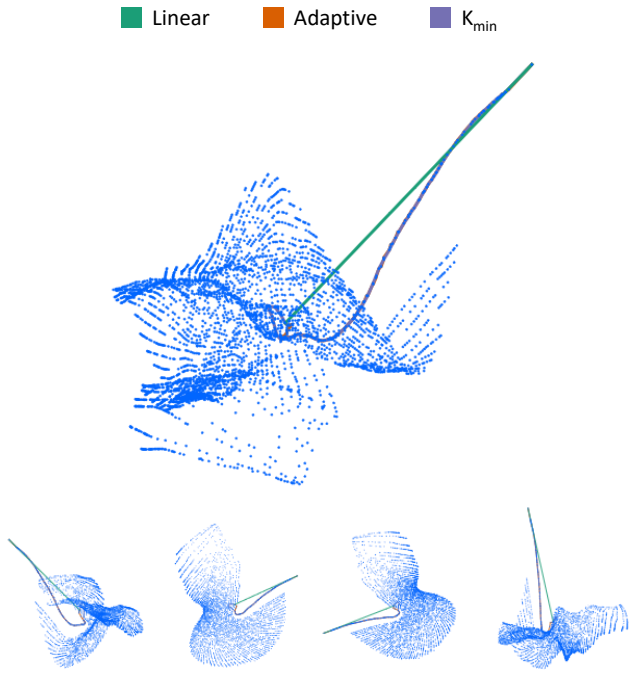

Figure 7: Latent space visualization for the image-based superformula dataset (with 2 variables) generated in *§ 4.2.1*. The three interpolation paths are shown along with the different views of the structure

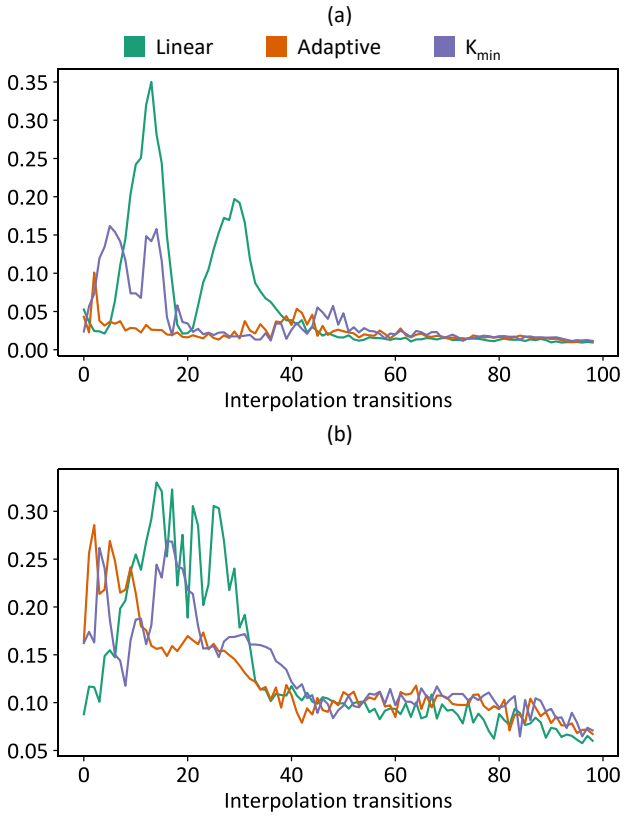

Figure 8: Evaluation of the interpolated images generated in *§ 4.2.1*: We observe higher peaks in (a) Earth Mover's Distance and (b) Gradient Projection Metric for linear interpolation due to the incomplete interpolated images.

all connect at a seam, where the shapes resemble circles, forming a *multi-manifold structure* (Figure 4).

We generated interpolations with three different methods on this multi-manifold structure, namely, linear, graph based on k-NN graph with k_{min}, and graph based on adaptive k-NN graph. The linear interpolation path between two points on alternating manifolds cuts through an inner manifold (green line in Figure 4), and this intersection corresponds to a different class of shape. We observe this in the intermediate interpolated shapes, where the initial four cornered shape interpolates to a ten cornered shape and finally to the six cornered shape at the end (Figure 5(a)). The jittery appearance of thelinearly interpolated shapes is likely because of the fact that the straight line joining the two end points largely does not pass through the multi-manifold thereby missing most of the latent space. In comparison, the intermediate shapes for our graph based methods are smoother boundaries. More importantly, both the k_{min} and adaptive k-NN based interpolations traverse the embedded graph structure thereby passing the seam at which the four classes of shapes meet (Figure 4). As a result, we note that the interpolated shapes in the middle of the sequence are ellipsoidal (Figure 5(b) and (c)) as should be expected. Furthermore, notice that the difference in the path traversed by the two interpolation methods results in the different aspect ratios of

the interpolated shapes, where the interpolated shapes for the k_{min} approach are comparatively wider than adaptive k-NN.

4.2. 2D Superformula: Image Interpolation

We generated two image-based datasets of the superformula shapes and conducted experiments on them.

4.2.1. Parametric Variation of s, n_1

Dataset. For our first experiment on the shape images, we generate a new image-based dataset of $4,000$ superformula shapes by varying the parameters s and n_1 from 0.1 to 1.0, individually. The other parameters are kept constant and are given the following values: $a = b = 1.0$; $m_1 = m_2 = 4.0$; $n_2 = 1.7$; $n_3 = 1.7$.

Model Architecture. We use the generated dataset to train a convolutional variational autoencoder. The architecture of the encoder includes five convolutional layers followed by four fully-connected layers, while the decoder contains four fully-connected layers followed by six convolutional layers, all but last of which are followed by up-sampling layers. The latent space dimension is set to 3, as the data is being varied over two parameters, allowing us to visualize it in 3D space.

Findings. For this experiment, we intentionally chose our starting and ending images such that the latent code corresponding to one of the images lies far from the majority of the other data points (Figure 7). A major portion of the linear interpolation path, traverses in regions containing very few data points towards one end of the interpolation. This likely results in the intermediate interpolated images to be missing pixels at the concave parts of the star shapes (Figure 6(a)). In comparison, we observe that the interpolations for both the graph based methods do not have the aforementioned abrupt artifacts. We believe that this is a direct outcome of graph traversal. Furthermore, in this specific example, we also note that the k_{min} and adaptive k-NN graph result in very similar interpolations. This is can be attributed to the observation that the minimal paths for both these graphs have an extensive overlap (purple and orange lines in Figure 7). The evaluation metrics support our results as both EMD and GPM show sharp peaks for the linear interpolation method that corresponds to the incomplete shapes (Figure 8). The EMD plot for the interpolations on the adaptive k-NN graph remains fairly flat through the interpolation transitions and shows a downward decline in the GPM plot signifying smooth transitions. The plots for the interpolation on the k-NN graph with k_{min} follow a similar trajectory, however, a few short sharp peaks can be observed that maybe due to transitions from one shape to another.

4.2.2. Parametric Variation of s, n_1, m

Dataset. We generated a new dataset of $28,282$ images of superformula shapes, by changing the parameters s, n_1 and m similar to the dataset preparation in *Section 4.1*. The value of s was varied from 0.1 to 1.0, n_1 from 0.1 to 10 and the values allotted to m were: $4, 6, 8$ and 10.

Model Architecture. We trained a convolutional variational autoencoder on this dataset with the same architecture as the previous experiment (Section 4.2.1). We set the latent dimension to 3 corresponding to the 3 parameters generating the dataset.

Findings. The two shape images we choose to interpolate between are similar to the two shapes we chose in *Section 4.1*, allowing us to observe the differences between the image-based and geometric-based interpolations. The path traversed by the linear interpolation does not pass through any surfaces of the latent space structure (green line in Figure 10), and hence we observe that the intermediate interpolated images do not resemble any other specific shapes. Instead, the transition between the two given images is quite fuzzy and abrupt, and some of the intermediate images are incomplete (Figure 9(a)). Our graph based approaches however, traverse a much longer path along the embedded graph structure, (orange and purple lines in Figure 10) and result in smoother, complete and consistent transition from the initial to the final image (Figure 9(b) and (c)). We also observe that results of the graph based methods, are quite similar to the ones in *Section 4.1* in terms of transition, as they interpolate from a four cornered shape to a circular shape and then finally to a six cornered shape. We do not see a similar comparison between the two linear interpolations, as they share no similarities in terms of the interpolated shapes.

We further evaluate the quality of the interpolations using the EMD and GPM (Figure 11). It is shown that linear interpolation method tends to peak near the halfway point (Figure 11, green curve), which corresponds with the incomplete interpolated images that are observed during the transition from one shape to another. Apart from these peaks, the curve remains quite flat in both the evaluation methods, as there is very little transition between the shapes. The graph based methods show a different trend: they peak towards the initial and final transitions in the EMD plot, and remain fairly flat during the middle transitions. This occurs due to the fairly quick transitions in the initial and final phases of interpolation, i.e. from a 4 cornered star to a square initially and from a hexagon to six cornered star towards the end. The plot for GPM shows the graph based methods to have a fairly flat slope overall as the values range from 0.15 to 0.25, though, on average they have greater values than linear interpolation as there is a continuous transition in the graph based approach, which leads to reduced similarity between consecutive interpolations.

In our experiments so far, we observed that for cases where there was a considerable difference between the path lengths for the k_{min} and adaptive k-NN graphs (Figures. 4 and 10), the adaptive k-NN graphs typically admitted shorter paths. While we do not currently have a theoretical basis for this, we will use the adaptive k-NN approach in the rest of the experiments.

4.3. Fashion-MNIST

We conduct experiments on the Fashion-MNIST dataset [69] to evaluate performances of the different interpolation methods.

Dataset. Fashion-MNIST is a database consisting of fashion products, offering more complexity than the commonly used MNIST dataset [71]. It consists of a training set of $60,000$ examples and test set of $10,000$ examples. Each example is a grayscale image of size 28×28 and belongs to one of ten labeled classes.

Model Architecture. For our experiment on the Fashion-MNIST dataset, we construct and train an adversarial autoencoder [46], where the encoder-decoder architecture behaves as the generator and we have a discriminator that differentiates between the real and fake images. The latent space distribution here is guided by the adversarial process, unlike the KL-Divergence in Variational Autoencoder. This allows us to generate higher quality interpolated images. The generator section is trained first and

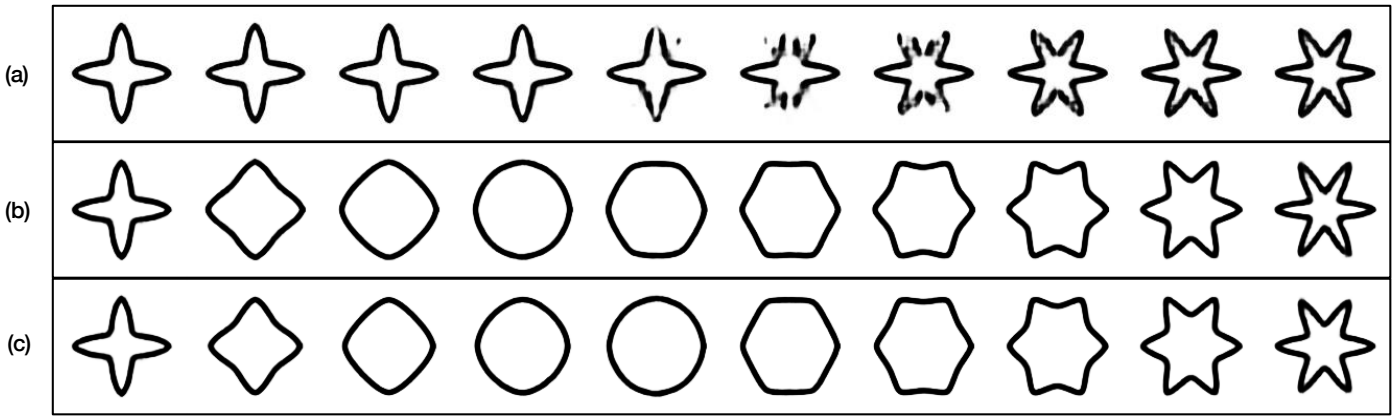

Figure 9: Interpolation results for experiments on image-based superformula dataset (with 3 variables): (a) Linear interpolation generates fuzzy intermediate images during transition between shapes; Interpolation along: (b) k-NN graph with k_{min} and (c) adaptive k-NN graph generate wide variety of complete shape images

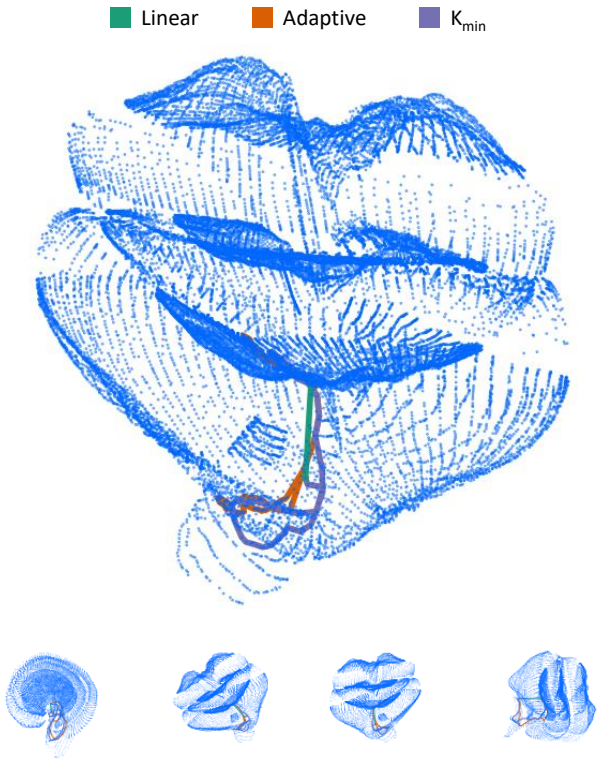

Figure 10: Latent space visualization for the image-based superformula dataset (with 3 variables) generated in *§ 4.2.2*. The three interpolation paths are shown along with the different views of the structure

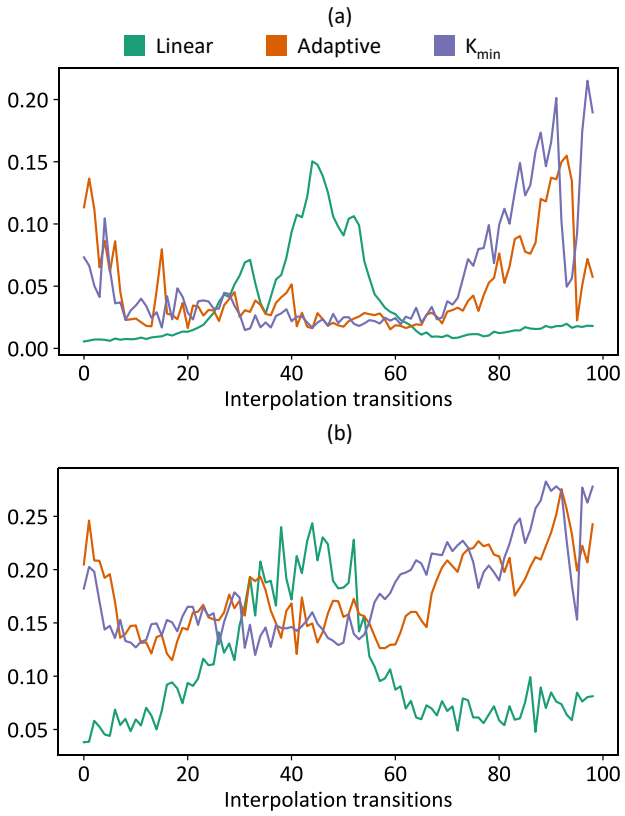

Figure 11: Evaluation of the interpolated images generated in *§ 4.2.2*: We observe high peaks in (a) Earth Mover's Distance, and (b) Gradient Projection Metric plots for graph-based approaches at the start and end of the interpolation transitions due to the quick transitions to new shapes. Peaks for linear interpolation occurs at the fuzzy transition between shape.

the discriminator section is trained next. The encoder consists of three convolutional layers, each followed by a batch normalization layer, and ends in a fully-connected layer. The decoder mirrors this structure. The discriminator consists of four convolutional layers. The latent space dimension is arbitrarily set to 100, owing to the higher complexity of the dataset compared to the superformula shape datasets.

Findings. Our experiment with Fashion-MNIST showcases a very critical aspect of interpolation, namely, *topological transition across the interpolants*. When comparing linear and our graph-based interpolation, we observe a clear difference between the two. For linear interpolation (Figure 12(a)), we observe virtually no initial change in the shape of the handbag (row 1) followed by *a pixel-by-pixel blending* (rows 2-3) between the handbag (genus-1 — a topological annulus) and the shoe (genus-0 — a topological disc) culminating in an almost quasi-static change

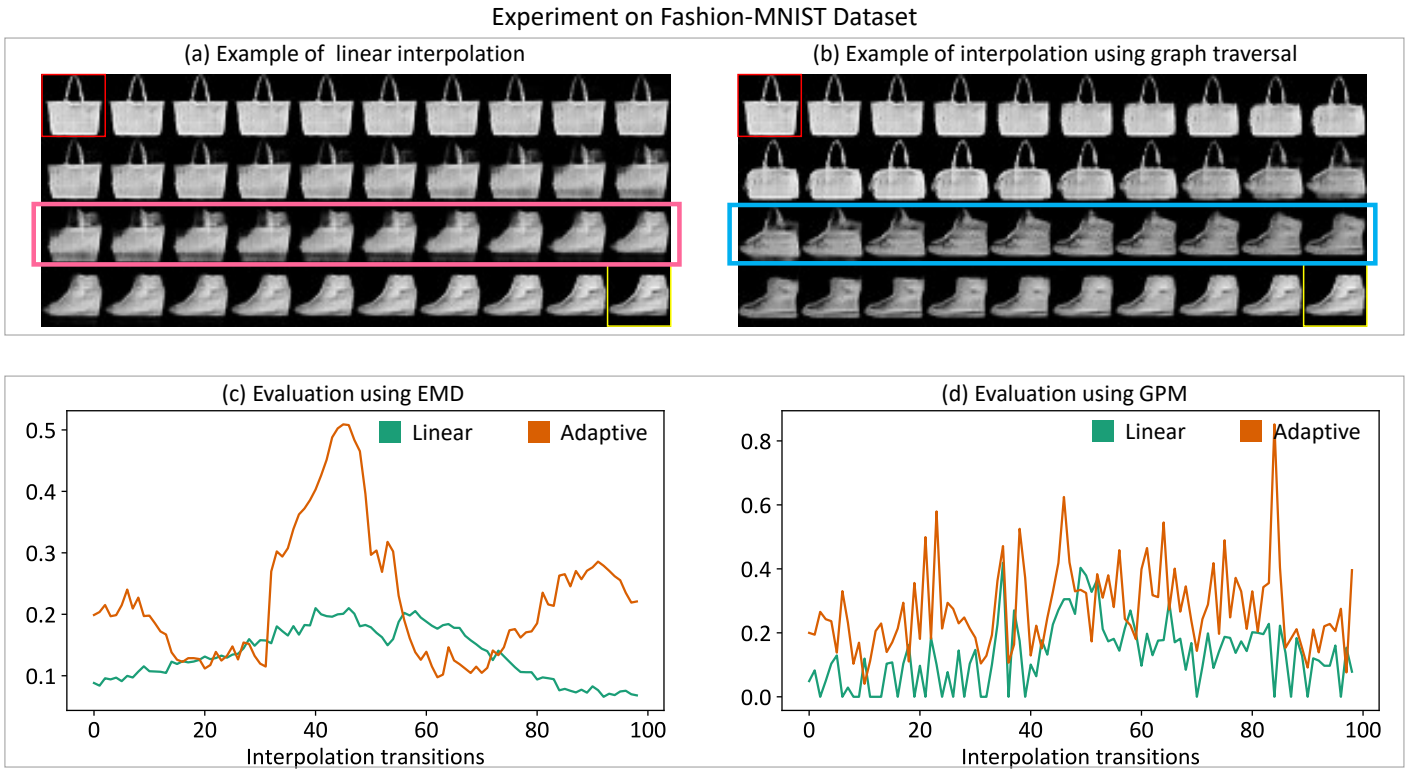

Figure 12: Interpolation and evaluation results for the Fashion-MNIST dataset. Top row: (a) linear interpolation generates fuzzy and slow transition between the handbag and shoe (pink box) and (b) adaptive k-NN graph-based approach generates topological transition from the handle of the hand bag to the tongue of the shoe (cyan box). Bottom row: Evaluation using (c) EMD and (d) GPM plots favor linear interpolation due to non-consistent transitions.

in the shoe (row 4). The qualitative observations are supported by the EMD metric (Figure 12(c)) wherein we observe a low-slope increase in the values (signifying row 1 of Figure 12(a)) followed by a dip (rows 2 and 3 of Figure 12(a)) and culminating in a gradual descent (row 4 of Figure 12(a)).

On the other hand, the graph-based interpolation (Figure 12(b)) shows a discernible change of shape in the body of the handbag (rows 1 and 2) while maintaining the genus-1 topology followed by a *smooth topological transition* seen as the contraction of the handle of the handbag (row 3) culminating in a gradual and discernible change of shape of the shoe (row 4). These qualitative observations are also supported by the EMD metric (Figure 12(c)) wherein we observe a dip in the values (signifying rows 1 and 2 of Figure 12(b)) followed by a sharp rise and dip during the topological transition (row 3 of Figure 12(b)) and culminating in a gradual ascent (row 4 of Figure 12(b)). To the best our knowledge the topological transition of genus-1 to genus-0 (handle contraction in row 3 of Figure 12(b)) is currently not possible in any known generative neural network based interpolation.

We note here that there is a need for a deeper analysis of such topological transitions through a controlled study of carefully constructed datasets. Secondly, the evaluation of topological shifts also needs extensive future research. The EMD metric (Figure 12(c)) relies on the pixel-wise similarity between two consecutive images in the sequence. Secondly, the plots for GPM are quite similar for both of the interpolation methods with the graph-based method showing higher values and sharper peaks, indicating a consistent transition between shapes (Figure 12(d)). In fact, we specifically note here that the EMD and GPM metrics are likely not suitable for evaluating topological shifts in images and further research is needed on new metrics.

4.4. *ShapeNet*

We conduct experiments on the ShapeNet dataset to evaluate the performance of our interpolation method on 3D shapes.

Dataset. ShapeNet [70] is a large-scale dataset of 3D models from a wide variety of categories. The complete dataset consists of more than 3,000,000 models and 3135 categories. For our experiments, we used a subset of the ShapeNet dataset consisting of 3991 shapes over 10 categories which include: chairs, tables, beds, bathtubs, bookshelves, dressers, monitors, nightstands, sofas and toilets. The voxel data of size $32 \times 32 \times 32$ for each of the shape was used to train our models.

Model Architecture. We constructed a convolutional variational autoencoder to train on our ShapeNet dataset. The encoder architecture included four 3D convoluted layers, and a fully connected layer, each of which was followed by a batch normalization layer. The decoder consisted of two fully connected layers, followed by four 3D transposed convolutional layers, and one 3D convolutional layer at the end, with a batch normalization layer after each of these layers. The latent space dimension was set to 100 owing to the complexity of the dataset.

Findings. We show two examples comparing the linear interpolation results with our graph based based approach on the learned latent space. In the first example, we interpolate between a table and a sofa (Figure. 13). We observe that the results from the linear interpolation method disintegrate from the initial to the final shape without any perceptually meaningful intermediate shapes. While the EMD and GPM plots favor this due to the lack of change in the interpolated shapes, we do not generate any

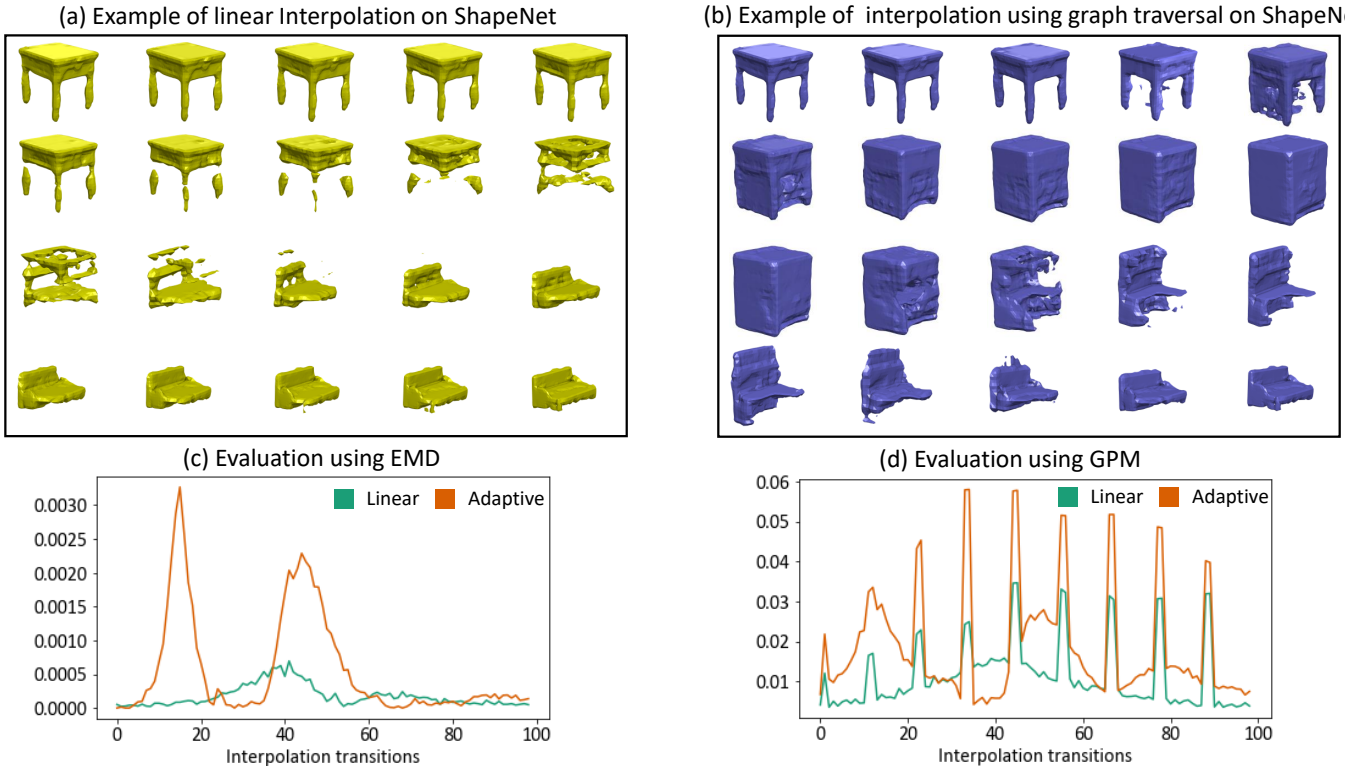

Figure 13: ShapeNet Example 1: Interpolation and evaluation results for the ShapeNet dataset. Top row: (a) linear interpolation generates incomplete and less meaningful interpolations between the table and sofa and (b) adaptive k-NN graph-based approach generates consistent and meaningful transitions from the table to a dresser, to a bookshelf and finally the sofa. Bottom row: Evaluation using (c) Earth Mover's Distance and (d) Gradient Projection Metric. The EMD plot shows two sharp peaks for the graph-based approach due to the transitions to the dresser and the sofa. GPM plot favors linear interpolation as its peaks are comparatively shorter.

intermediate shapes with either of the features of the table or the sofa. In our graph based approach, however, we observe the table changing to an intermediate shape that looks like a dresser, further changing to a form that vaguely resembles a book shelf and finally to a sofa. Our approach generates different and more meaningful interpolated shapes compared to the linear interpolation methods. The peaks in the EMD plot indicate the shift between different classes of shapes (e.g. sofa to night-stand).

In our second example, we show an L-shaped sofa interpolating to an office chair, using both of the interpolation methods (Figure. 14). Similar to the previous example, we can see the linear interpolation method does not produce any perceptually meaningful results. The sofa simply loses its shape to an unrecognizable form, which then turns into the shape of a chair in a disorderly fashion. Our approach on the other hand, interpolates the sofa to a small desk like shape, then to a smaller chair and finally to the office chair. This shows a perceptually meaningful and consistent transition between shapes that is lacking in the linear interpolation method. As our interpolation approach traverses along the latent embedded graph, we can see how the traversal path explores the perceptually similar shapes, such as the changing sizes of the chair, and the flattening and lengthening of the sofa. The EMD plot shows a sharp peak for our approach when the shape of the sofa becomes flat and gradually converts to a chair. The peak for the linear interpolation method coincides with the fuzzy transitions between the shapes of the sofa and the chair.

In both examples, it is important to note that while both interpolation methods lead to shapes with several disconnected voxels (row 3 in Figures 13(b) and 14(b)), latent embedded graph still offers meaningful interpolations compared to the linear interpolation approach.

5. Limitations & Future Work

5.1. Evaluating Interpolation Quality

A visual inspection (as is nominally practiced in several similar works) of the results show that the latent embedded graph method is effective in generating reasonable interpolations. However, the metrics that we used, either from existing literature [64, 65] or our GPM metric, to quantify the interpolation did not always reflect these visual inspections specifically for interpolations that involved topological shifts. We specifically observe this in the Fashion-MNIST experiment, where the EMD plot for the adaptive k-NN graph shows a sharp peak during the transition from a handbag to a shoe (Figure 12). The sharp peak of the plot corresponds to the interpolated images where topological change occurs from the handle of the handbag to the tongue of the shoe (Figure 12). Linearly interpolated images, however, do not seem to show any topological and perceptual transitions, but score well in the evaluation metrics as the interpolated images remain fairly similar to one another with gradual appearance and disappearance of gray pixels that do not show any meaningful interpolations. Lastly, while our polygonal superformula dataset showed some interesting comparisons between linear and graph-based interpolations (*Section 4.1*), we did not conduct a quantitative evaluation to compare the two cases. While there are several potential methods that could be used (e.g. deformation

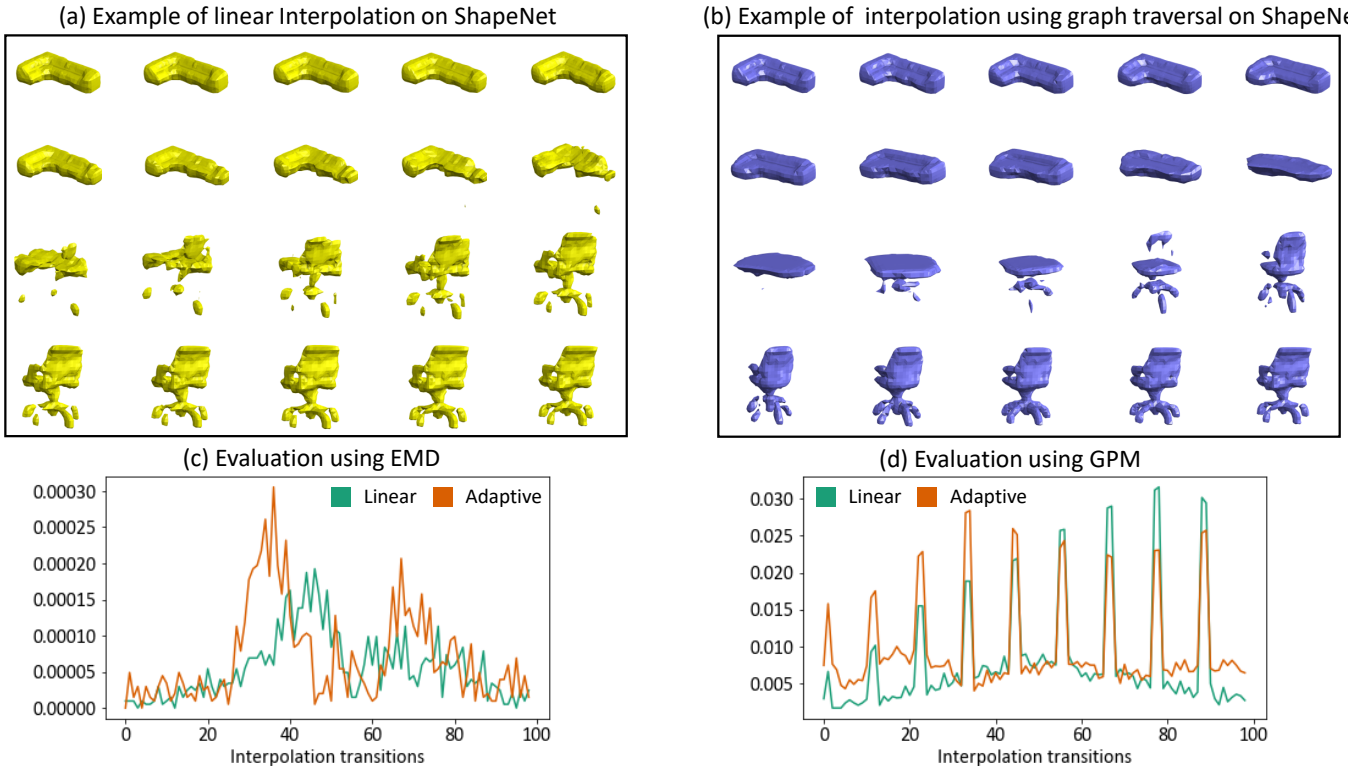

Figure 14: ShapeNet Example 2: Interpolation and evaluation results for the ShapeNet dataset. Top row: (a) linear interpolation generates incomplete and fuzzy interpolation transitions between the L-shaped sofa and office chair; (b) adaptive k-NN graph-based approach generates a perceptually consistent transition from the sofa to a small table, then to a small chair and finally to the office chair. Bottom row: Evaluation using (c) Earth Mover's Distance and (d) Gradient Projection Metric shows the EMD plot favoring linear interpolation due to lack of continuous transitions. The GPM plot shows similar results for both approaches.

fields, curve energy etc), it is currently difficult to assess which metrics would capture perceptual preferences for transitions from one shape to another.

5.2. Capturing Topological Shifts in Interpolation

While our method shows promising results for capturing topological shifts, we note that this was incidental, in that we had not specifically trained our generative models to capture topological shifts. Having said that, our latent embedded graph approach captured the topological transitions well atleast for the Fashion-MNIST experiment. However, we did not achieve the same success for voxel interpolation. In the case of the ShapeNet dataset, we still observe artifacts such as disconnected components while traversing between two classes of shapes (for example in row 3 in Figure. 14 (b)). Perhaps it maybe useful to explore disentagled representations and class-conditional training for such datasets. In general, a deeper investigation of smooth topological transitions is needed similar to our superformula experiment, wherein, more properly curated datasets could be generated and tested specifically to characterize topology shifts while training the generative model itself.

5.3. Graph Construction Strategies

Our choice for using k-NN based graphs was generally influenced by their use in current manifold learning methods. However, there are several other graph construction strategies (e.g. threshold-graphs, rapidly-exploring random trees, etc.) that may be useful for specific data distributions. For instance, a novel bio-inspired method based on space colonization (leaf venation)[72] could be used for graph construction where the similarity-flow along the paths is of importance. There are other options such as rapidly-exploring random trees [73] that belong to the category of incremental graph generation and could be useful for certain applications.

There are several interesting factors that may affect the graph construction strategy. The first is the density of the points sampled in the latent space. Imposing the k-NN graph (or, for that matter, any reasonable discrete topological structure on the point set) allows us to approximate the geodesic in a piece-wise linear manner. Therefore, in order for any graph-based approach to work, a reasonable density is required to allow for the use of Euclidean distance in a piece-wise linear manner. The second factor is the existence of clear and separate clusters in the latent space. This affects the assumption of the graph being connected. In our case, the assumption of a connected graph was a conscious choice we made in order to allow interpolation between any two data samples. However, allowing multiple disjoint graphs may have it's own advantages. For example, it could be used to avoid interpolation between shapes that are inherently incompatible (more so in terms of their genus). Additional theoretical and computational investigation of graph construction strategies is essential. Our work paves the way for further exploration of these research directions.

5.4. Exploring Generative Models

Given that we wanted to systematically investigate interpolation through generative models, we chose to use autoencoders as our basic architecture. Our choice was based on the fact that starting from a data sample in the original space and subsequently generating the interpolants is direct with autoencoders — they provide an explicit encoding mechanism for a data sample in the original space. There are several other model architectures such as GANs that may provide better latent space distributions. However, we did not choose to use these since they do not allow for picking the inputs in the original space. Here, we are encouraged to explore other generative models like variants of GANs that give us more control over the samples to interpolate between, such as InfoGAN [74] that learns latent codes to represent unlabeled data, and ClusterGANs [75] that achieve clustering in the latent spaces of the GAN models. Given that our methodology itself is not restricted to autoencoders, we believe there is a good scope for future investigations of other more powerful generative model architectures.

6. Conclusion

We presented latent embedded graph, a framework for image and shape interpolation by combining dimensionality reduction enabled by generative neural networks with a simple graph-traversal based strategy. We demonstrated our approach through several examples by using VAEs in conjunction with k-NN and adaptive k-NN graph traversal. Our results clearly show that this strategy results in visually smooth interpolations. What is more interesting in our results is that our methodology is able to (1) enable good interpolation even for multi-manifold data distribution and (2) respects topological changes in the shapes that are not explicit in image based representations. We believe these outcomes reveal a rich space to explore several directions including the development of new efficient graph construction methods and new evaluation metrics to evaluate persistent topological and shape interpolations.

Acknowledgments

Removed for anonymization.

References

[1] Y. Ma, Y. Fu, Manifold Learning Theory and Applications, 1st Edition, CRC Press, Inc., Boca Raton, FL, USA, 2011.

[2] C. Bergler, S. M. Omohundro, Nonlinear Image Interpolation using Manifold Learning, in: Advances in Neural Information Processing Systems, Vol. 7, MIT Press, 1994, pp. 973–980.

[3] R. Souvenir, Q. Zhang, R. Pless, Image manifold interpolation using free-form deformations, in: Proceedings - International Conference on Image Processing, ICIP, IEEE, 2006, pp. 1437–1440. doi:10.1109/ICIP.2006.312700.

[4] A.-D. Nguyen, W. Kim, J. Kim, S. Lee, Video frame interpolation by plug-and-play deep locally linear embedding, arXiv preprint arXiv:1807.01462.

[5] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2014, pp. 2672–2680.

[7] G. Arvanitidis, L. K. Hansen, S. Hauberg, Latent space oddity: on the curvature of deep generative models, arXiv preprint arXiv:1710.11379.

[8] N. Chen, A. Klushyn, R. Kurle, X. Jiang, J. Bayer, P. Smagt, Metrics for deep generative models, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2018, pp. 1540–1550.

[9] H. Shao, A. Kumar, P. Thomas Fletcher, The riemannian geometry of deep generative models, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 315–323.

[10] N. Lei, D. An, Y. Guo, K. Su, S. Liu, Z. Luo, S.-T. Yau, X. Gu, A geometric understanding of deep learning, Engineering 6 (3) (2020) 361–374.

[11] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein GAN, CoRR abs/1701.07875. arXiv:1701.07875.
URL http://arxiv.org/abs/1701.07875

[12] P. Campadelli, E. Casiraghi, C. Ceruti, A. Rozza, Intrinsic Dimension Estimation: Relevant Techniques and a Benchmark Framework (oct 2015). doi:10.1155/2015/759567.

[13] A. Goldberg, X. Zhu, A. Singh, Z. Xu, R. Nowak, Multi-manifold semi-supervised learning, in: Artificial Intelligence and Statistics, PMLR, 2009, pp. 169–176.

[14] J. B. Tenenbaum, V. De Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323. arXiv:arXiv:1011.1669v3, doi:10.1126/science.290.5500.2319.

[15] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326. arXiv:arXiv:1011.1669v3, doi:10.1126/science.290.5500.2323.

[16] V. Premachandran, R. Kakarala, Consensus of k-nns for robust neighborhood selection on graph-based manifolds, in: 2013 IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1594–1601. doi:10.1109/CVPR.2013.209.

[17] T. Lin, H. Zha, Riemannian manifold learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (5) (2008) 796–809. doi:10.1109/TPAMI.2007.70735.

[18] G. Wolberg, Image morphing: a survey, The visual computer 14 (8) (1998) 360–372.

[19] P. Gao, T. W. Sederberg, A work minimization approach to image morphing, The Visual Computer 14 (8) (1998) 390–400.

[20] D. Han, Comparison of commonly used image interpolation methods, in: Proceedings of the 2nd international conference on computer science and electronics engineering, Atlantis Press, 2013, pp. 1556–1559.

[21] J. Liao, R. S. Lima, D. Nehab, H. Hoppe, P. V. Sander, J. Yu, Automating image morphing using structural similarity on a halfway domain, ACM Transactions on Graphics (TOG) 33 (5) (2014) 1–12.

[22] E. Shechtman, A. Rav-Acha, M. Irani, S. Seitz, Regenerative morphing, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 615–622.

[23] M. Alexa, Recent advances in mesh morphing, in: Computer graphics forum, Vol. 21, Wiley Online Library, 2002, pp. 173–198.

[24] G. Turk, J. F. O'Brien, Shape transformation using variational implicit functions, in: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999, pp. 335–342. doi:10.1145/311535.311580.
URL http://dx.doi.org/10.1145/311535.311580

[25] M. Žefran, V. Kumar, Interpolation schemes for rigid body motions, Computer-Aided Design 30 (3) (1998) 179 – 189, motion Design and Kinematics. doi:https://doi.org/10.1016/S0010-4485(97)00060-2.
URL http://www.sciencedirect.com/science/article/pii/S0010448597000602

[26] D. Xu, H. Zhang, Q. Wang, H. Bao, Poisson shape interpolation, in: Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling, SPM '05, ACM, New York, NY, USA, 2005, pp. 267–274. doi:10.1145/1060244.1060274.
URL http://doi.acm.org/10.1145/1060244.1060274

[27] L. Gao, S.-Y. Chen, Y.-K. Lai, S. Xia, Data-driven shape interpolation and morphing editing, in: Computer Graphics Forum, Vol. 36, Wiley Online Library, 2017, pp. 19–31.

[28] M. Aydınlılar, Y. Sahillioğlu, Part-based data-driven 3d shape interpolation, Computer-Aided Design 136 (2021) 103027.

[29] M. Alexa, D. Cohen-Or, D. Levin, As-rigid-as-possible shape interpolation, in: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 2000, pp. 157–164.

[30] E. Klassen, A. Srivastava, M. Mio, S. H. Joshi, Analysis of planar shapes using geodesic paths on shape spaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (3) (2004) 372–383. doi:10.1109/TPAMI.2004.1262333.

[31] A. Elgammal, C.-S. Lee, Inferring 3d body pose from silhouettes using activity manifold learning, in: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., Vol. 2, 2004, pp. II–II. doi:10.1109/CVPR.2004.1315230.

[32] H. Murase, S. K. Nayar, Visual learning and recognition of 3-d objects from appearance, International Journal of Computer Vision 14 (1) (1995) 5–24. doi:10.1007/BF01421486.
URL https://doi.org/10.1007/BF01421486

[33] T. Cootes, C. Taylor, D. Cooper, J. Graham, Active shape models-their training and application, Computer Vision and Image Understanding 61 (1) (1995) 38 – 59. doi:https://doi.org/10.1006/cviu.1995.1004.
URL http://www.sciencedirect.com/science/article/pii/S1077314285710041

[34] R. Urtasun, D. J. Fleet, P. Fua, 3d people tracking with gaussian process dynamical models, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), Vol. 1, 2006, pp. 238–245. doi:10.1109/CVPR.2006.15.

[35] V. I. Morariu, O. I. Camps, Modeling correspondences for multi-camera tracking using nonlinear manifold learning and target dynamics, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), Vol. 1, 2006, pp. 545–552. doi:10.1109/CVPR.2006.189.

[36] K. Moon, V. Pavlovic, Impact of dynamics on subspace embedding and tracking of sequences, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), Vol. 1, 2006, pp. 198–205. doi:10.1109/CVPR.2006.148.

[37] I. Kemelmacher-Shlizerman, E. Shechtman, R. Garg, S. M. Seitz, Exploring photobios, ACM Trans. Graph. 30 (4) (2011) 61:1–61:10. doi:10.1145/2010324.1964956.
URL http://doi.acm.org/10.1145/2010324.1964956

[38] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, A. A. Efros, Generative visual manipulation on the natural image manifold, in: European conference on computer vision, Springer, 2016, pp. 597–613.

[39] D. Simon, A. Aberdam, Barycenters of natural images constrained wasserstein barycenters for image morphing, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7910–7919.

[40] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507. arXiv:20, doi:10.1126/science.1127647.

[41] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1097–1105.

[42] P. Y. Simard, D. Steinkraus, J. C. Platt, Best practices for convolutional neural networks applied to visual document analysis, in: Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2, ICDAR '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 958–.
URL http://dl.acm.org/citation.cfm?id=938980.939477

[43] D. Ha, D. Eck, A neural representation of sketch drawings, CoRR abs/1704.03477. arXiv:1704.03477.
URL http://arxiv.org/abs/1704.03477

[44] X. Zhou, B. Hu, Q. Chen, X. Wang, An auto-encoder for learning conversation representation using lstm, in: S. Arik, T. Huang, W. K. Lai, Q. Liu (Eds.), Neural Information Processing, Springer International Publishing, Cham, 2015, pp. 310–317.

[45] I. O. Tolstikhin, O. Bousquet, S. Gelly, B. Schölkopf, Wasserstein auto-encoders, CoRR abs/1711.01558.

[46] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey, Adversarial autoencoders, arXiv preprint arXiv:1511.05644.

[47] W. Chen, M. Fuge, Synthesizing designs with inter-part dependencies using hierarchical generative adversarial networks, Journal of Mechanical Designdoi:10.1115/1.4044076.

[48] W. Chen, M. Fuge, N. Chazan, Design manifolds capture the intrinsic complexity and dimension of design spaces, Journal of Mechanical Design 139 (5) (2017) 051102–051102–10. doi:10.1115/1.4036134.

[49] W. Chen, N. Chazan, M. Fuge, How designs differ: Non-linear embed-

dings illuminate intrinsic design complexity, in: ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, 2016, pp. V02AT03A014–V02AT03A014. doi:10.1115/DETC2016-60112.

[50] W. Zhang, Z. Yang, H. Jiang, S. Nigam, S. Yamakawa, T. Furuhata, K. Shimada, L. B. Kara, 3d shape synthesis for conceptual design and optimization using variational autoencoders, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 59186, American Society of Mechanical Engineers, 2019, p. V02AT03A017.

[51] M. E. Yumer, P. Asente, R. Mech, L. B. Kara, Procedural modeling using autoencoder networks, in: Proceedings of the 28th Annual ACM Symposium on User Interface Software Technology, UIST '15, ACM, New York, NY, USA, 2015, pp. 109–118.

[52] A. Dubrovina, F. Xia, P. Achlioptas, M. Shalah, R. Grvoscot, L. Guibas, Composite shape modeling via latent space factorization, International Conference Computer Vision.

[53] S. Bond-Taylor, A. Leach, Y. Long, C. G. Willcocks, Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models, arXiv preprint arXiv:2103.04922.

[54] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[55] E. Facco, M. d'Errico, A. Rodriguez, A. Laio, Estimating the intrinsic dimension of datasets by a minimal neighborhood information, Scientific reports 7 (1) (2017) 1–8.

[56] O. Samko, A. Marshall, P. Rosin, Selection of the optimal parameter value for the Isomap algorithm, Pattern Recognition Letters 27 (9) (2006) 968–979. doi:10.1016/J.PATREC.2005.11.017.

[57] C. CAREY, Graph construction for manifold discovery, Ph.D. thesis, University of Massachusetts Amherst (2017).

[58] D. Freedman, Efficient simplicial reconstructions of manifolds from their samples, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (10) (2002) 1349–1357. doi:10.1109/TPAMI.2002.1039206.

[59] Z. Zhang, J. Wang, H. Zha, Adaptive manifold learning, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2) (2012) 253–265. doi:10.1109/TPAMI.2011.115.
URL http://dx.doi.org/10.1109/TPAMI.2011.115

[60] R. W. Floyd, Algorithm 97: shortest path, Communications of the ACM 5 (6) (1962) 345.

[61] E. W. Dijkstra, et al., A note on two problems in connexion with graphs, Numerische mathematik 1 (1) (1959) 269–271.

[62] P. Giblin, The essentials of cagd, by gerald farin and dianne hansford. pp. 229.£ 33.00 2000. isbn 1 56881 123 3 (ak peters)., The Mathematical Gazette 86 (505) (2002) 183–184.

[63] T. Popiel, L. Noakes, Bézier curves and c2 interpolation in riemannian manifolds, Journal of Approximation Theory 148 (2) (2007) 111 – 127. doi:https://doi.org/10.1016/j.jat.2007.03.002.
URL http://www.sciencedirect.com/science/article/pii/S0021904507000469

[64] Y. Rubner, C. Tomasi, L. J. Guibas, The earth mover's distance as a metric for image retrieval, International journal of computer vision 40 (2) (2000) 99–121.

[65] R. Szeliski, Computer vision: algorithms and applications, Springer Science & Business Media, 2010.

[66] O. Pele, M. Werman, A linear time histogram metric for improved sift matching, in: Computer Vision–ECCV 2008, Springer, 2008, pp. 495–508.

[67] O. Pele, M. Werman, Fast and robust earth mover's distances, in: 2009 IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 460–467.

[68] J. Gielis, A generic geometric transformation that unifies a wide range of natural and abstract shapes, American Journal of Botany 90 (3) (2003) 333–338. arXiv:https://bsapubs.onlinelibrary.wiley.com/doi/pdf/10.3732/ajb.90.3.333, doi:10.3732/ajb.90.3.333.
URL https://bsapubs.onlinelibrary.wiley.com/doi/abs/10.3732/ajb.90.3.333

[69] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017). arXiv:1708.07747.

[70] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, F. Yu, ShapeNet: An Information-Rich 3D Model Repository, Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015).

[71] Y. LeCun, C. Cortes, C. Burges, Mnist handwritten digit database, ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist 2.

[72] A. Runions, B. Lane, P. Prusinkiewicz, Modeling trees with a space colonization algorithm, in: Proceedings of the Third Eurographics Conference on Natural Phenomena, NPH'07, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2007, pp. 63–70. doi:10.2312/NPH/NPH07/063-070.
URL http://dx.doi.org/10.2312/NPH/NPH07/063-070

[73] S. M. LaValle, J. J. Kuffner Jr, Randomized kinodynamic planning, The international journal of robotics research 20 (5) (2001) 378–400.

[74] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, arXiv preprint arXiv:1606.03657.

[75] S. Mukherjee, H. Asnani, E. Lin, S. Kannan, Clustergan: Latent space clustering in generative adversarial networks, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 33, 2019, pp. 4610–4617.