Structure-Preserving Deep Autoencoder Based Dimensionality Reduction for Data Visualization

Pasumarthi Sreekar

Advisor: **Dr. Kaustuv Nag**

Department of Computer Science and Engineering
Indian Institute of Information Technology Guwahati

This dissertation is submitted for the degree of
Bachelors of Technology

IIIT Guwahati May 2023

dedicated to my loving parents...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Pasumarthi Sreekar
May 2023

Acknowledgements

I would like to extend my gratitude towards Dr. Kaustuv Nag for the invaluable guidance and support provided throughout this project. For providing their mentorship, my gratitude extends to all the faculty in the Department of Computer Science and Engineering at the Indian Institute of Information Technology Guwahati. My appreciation also goes out to my parents for their unwavering encouragement and support.

Abstract

Autoencoders are typically used for dimensionality reduction. However, a drawback of theirs is that if they have additional representative capacity, they may not be able to preserve the structure of the dataset meaningfully in the lower-dimensional latent space. Regularization can be a way to solve this problem as it encourages the autoencoder to satisfy certain additional properties besides the ability to copy its input to its output. Here, we propose a structure-preserving deep autoencoder-based dimensionality reduction scheme. To this end, we propose a regularizer that helps encourage the autoencoder to learn structure-preserving properties of the dataset, specifically, the ability to preserve the local and global geometry of the dataset. Preserving such geometry is integral to dimensionality reduction as datasets usually lie on a lower dimensional manifold embedded in the original feature space. A chosen reduced dimensionality of two or three for the encoded feature space enables us to visualize the extracted latent representations of the data using scatterplots. The proposed approach, moreover, is unsupervised and has predictability. We use four artificial datasets and four real-world datasets to illustrate the effectiveness of the proposed method. We visualize the datasets along with seven other comparing methods. We also conduct three experiments to validate the predictability of our method and discuss several future research directions.

Table of contents

List of figures

Chapter 1

Introduction

Dimensionality reduction is important to many domains of science and engineering. In studies of vision, speech, and a range of other physical and biological sciences, scientists regularly encounter the problem of finding meaningful low-dimensional representations of high-dimensional data. Although the input dimensionality (e.g., 4096 for 64 pixel by 64 pixel images) may be quite high, the observations result from changes in only a few degrees of freedom, lending a predictable structure to the high-dimensional data. Dimensionality reduction is concerned with finding this underlying structure and preserving it meaningfully in the low-dimensional representation. This may be important for a number of reasons. One possible reason is data visualization. It is difficult to display and understand relationships in dimensions higher than two or three, making the use of low-dimensional transformations of the data appealing. Another possible reason is classification. A typical problem associated with high-dimensional data is the curse of dimensionality. Finding suitable low-dimensional representations mitigates this problem along with the other undesired properties of high-dimensional spaces. Our goal is to discover, given only the high-dimensional data, low-dimensional representations with features that capture the intrinsic degrees of freedom of a data set. In many cases of interest, the observed data are found to lie on an embedded submanifold of the high-dimensional space. The degrees of freedom along this submanifold correspond to the underlying features.

The problem of dimensionality reduction is defined formally as follows. Given a data set $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, where, n is the number of samples and each sample $\boldsymbol{x}_i \in \mathbb{R}^D$. Further, assume that this data set has an intrinsic dimensionality d. In mathematical terms, intrinsic dimensionality d means that the data points in data set $\mathcal{X}$ are lying on or near a submanifold of dimension d that is embedded in

D-dimensional space. Dimensionality reduction aims to find a suitable mapping $\phi\colon \mathbb{R}^D \to \mathbb{R}^d, D > d$, such that, $\mathcal{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n\}$ is a d-dimensional representation of $\mathcal{X}$, retaining the *intrinsic* geometry of the data as much as possible. Here, $\boldsymbol{y}_i = \phi(\boldsymbol{x}_i)$ is the latent representation of sample $\boldsymbol{x}_i$ for $i = 1, 2, \ldots, n$. These notations are consistently employed throughout the remainder of this paper.

There exist several dimensionality reduction techniques that can be used for this purpose. They can broadly be divided into two categories: linear and nonlinear. Linear techniques for dimensionality reduction include principal component analysis (PCA)[7], linear discriminant analysis[3], canonical correlation analysis (CCA)[8], and locality preserving projections [6]. These methods are designed to operate only when the submanifold is embedded linearly, or almost linearly, in the input space, i.e., they are only effective if the data is lying on a high-dimensional hyperplane. As a result, these linear techniques cannot adequately handle complex nonlinear data (nonlinear manifolds). In contrast to the linear techniques, nonlinear methods, such as Isomap[16] and Locally-linear embedding (LLE)[14], have the ability to recover the intrinsic geometric structure of a broad class of nonlinear manifolds. These nonlinear techniques, typically, follow either the local or the global approaches to preserving the intrinsic geometry. Local approaches (LLE, Laplacian Eigenmaps) attempt to preserve the local geometry of the data, i.e., they seek to map nearby points on the manifold to nearby points in the low-dimensional representation. In contrast, global approaches (Isomap) attempt to preserve geometry on all scales, i.e. both the local and global geometry, mapping nearby points on the manifold to nearby points in the low-dimensional representation and faraway points to faraway points. The advantage of using approaches that only preserve the local geometry is that they can be used on a broader range of manifolds, i.e., whose local geometry is close to Euclidean, but whose global geometry may not be. Whereas the approaches that aim to preserve the global geometry as well have the advantage that they can represent the global structure of the data faithfully.

However, a drawback of most of the above *nonlinear* dimensionality reduction techniques is that they may not be able to preserve both the local and global geometry of the data satisfactorily, though they may aim to do so. Another drawback is that they lack predictability. In other words, once the learning phase is over, they cannot project out-of-sample data points, i.e., points not a part of the training set. To address these issues, we propose a structure-preserving deep autoencoder-based dimensionality reduction-based scheme. To this end, we propose a regularizer. Further, we test and compare the method with seven other state-of-the-art dimensionality reduction techniques visually, using four artificial and four real-world data sets. We also conduct experiments to test the predictability of the proposed method. The proposed method offers the following advantages:

1. It preserves both the local and global geometry of the data during the projection satisfactorily.

2. It enables visualization of the data in a two or three-dimensional extracted feature space.

3. It has predictability, i.e, the proposed method can faithfully project unknown (test) data points.

4. The approach is unsupervised.

Chapter 2

Related Works

2.1 Methods That Do Not Use Neural Networks

2.1.1 Linear Techniques

PCA[2] is a very popularly used linear dimensionality reduction technique. In PCA, the principal components of the data are first calculated, i.e the unit vectors that correspond to the directions in which the maximum variance of the data can be explained. It then performs dimensionality reduction by changing the basis of the data to the principal components. MDS [17] is another linear technique for dimensionality reduction. While mapping high-dimensional data to a lower-dimensional feature space, it attempts to preserve the distances between data points (pairwise distances). Classical MDS aims to find the linear mapping M that maximizes the following cost function:

$$E = \sum_{ij} \left\| M\boldsymbol{x}_i - M\boldsymbol{x}_j \right\|^2 \tag{2.1}$$

Classical MDS[17] has demonstrated its efficacy across various practical applications. However, its limitation lies in its disregard for the manifold's topology when computing distances between data points. Consequently, Classical MDS is less suitable for elucidating nonlinear relationships within the data. If high-dimensional data is distributed on or near nonlinear manifolds, Classical MDS can sometimes incorrectly infer that two data points are closer together based on their lower Euclidean distance, even though they may be significantly farther apart on the manifold.

2.1.2 Nonlinear Techniques

Isomap[16] is a technique that aims to resolve this problem of Classical MDS by attempting to preserve the geodesic distances between data points rather than their Euclidean distances. Geodesic distance is the shortest curve length which respects the geometry of the data. In Isomap [16], the geodesic distances between the data points $\boldsymbol{x}_i (i = 1, 2, \ldots, n)$ are computed by first constructing a neighborhood graph G, in which every data point $\boldsymbol{x}_i$ is connected to its k-nearest neighbors. The shortest path between two points in the graph forms an estimate of the geodesic distance between these two points. This can be computed by using either the Dijkstra's[1] or the Floyd's shortest-path algorithm[4]. The low-dimensional representation is then computed by applying classical MDS on the resulting pairwise geodesic distances. Local Linear Embedding (LLE) is similar to Isomap in that it constructs a graph representation of the data points. However, in contrast to Isomap, it attempts to preserve only the local geometry of the data. Similar to LLE is the Laplacian Eigenmaps.

t-SNE is another popular nonlinear dimensionality reduction method. In t-SNE, the Euclidean pairwise distances in the high-dimensional space are converted to symmetrized conditional probabilities $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$, where, $p_{j|i}$ signifies the conditional probability that data point $\boldsymbol{x}_j$ is the neighbor of data point $\boldsymbol{x}_i$. Further, this conditional probability follows a Gaussian Distribution. Similarly, the joint probability q_{ij} of low-dimensional data points $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ is obtained and is assumed to follow the student's t-distribution. Finally, the low-dimensional representations $\boldsymbol{y}_i$s are obtained by minimizing the Kullback-Leibler divergence between the joint probability distributions from these two spaces. Another state-of-the-art dimensionality reduction method is the uniform manifold approximation and projection (UMAP). This method first constructs a fuzzy topological representation of the high-dimensional data by approximating the manifold locally at each point using appropriate local fuzzy simplicial set representations. It then optimizes the low-dimensional representation to have a similar fuzzy topological representation as of the high-dimensional data. This is done by minimizing the cross entropy between the two fuzzy topological representations. Diffusion maps focus on discovering the underlying manifold of data. It maps data into a lower-dimensional space, such that the Euclidean distance between points closely approximates the diffusion distance observed in the original feature space.

2.2 Neural Networks based methods

In denoising autoencoder (DAE)[], the authors proposed a change in the training objective of autoencoders from that of the usual reconstruction loss (MSE) to denoising an artificially corrupted input. This way the neural network is forced to transition from merely learning the identity function to genuinely capturing the intrinsic structure of the data set. Formally, the training objective optimized:

$$\mathcal{L}_{DAE} = \frac{1}{n}\sum_{i=1}^{n} \|f(h(\tilde{\boldsymbol{x}}_i)) - \boldsymbol{x}_i\|^2, \tag{2.2}$$

where h, f are the encoder and decoder functions respectively and $\tilde{\boldsymbol{x}}_i$ is the copy of input $\boldsymbol{x}_i$ corrupted by some form of noise. The encoder output can then be used for the purpose of dimensionality reduction or may be used to pre-train a deeper unsupervised network or a supervised network

Contractive autoencoder (CAE) adds an explicit regularizer to the reconstruction loss function to enable learning the manifold structure of the data. This regularizer encourages the learned encoding of data to be robust and invariant to small variations in the input data. The regularizer defined formally:

$$R = \left\| \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_F^2. \tag{2.3}$$

where f is the encoder function. The training objective is

$$\mathcal{L}_{CAE} = \frac{1}{n}\sum_{i=1}^{n} \|f(h(\boldsymbol{x}_i)) - \boldsymbol{x}_i\|^2 + \alpha R, \tag{2.4}$$

where α is a hyper-parameter controlling the strength of the regularization. In particular, due to its training for robustness against input perturbations, the CAE is encouraged to transform a neighborhood of input points to a smaller neighborhood of output points. The encoded low-dimensional representation is now only sensitive to changes along the manifold directions but is insensitive to changes orthogonal to the manifold.

Generalized autoencoder (GAE)[19] extends the traditional autoencoder in the sense that now each instance $\boldsymbol{x}_i$ reconstructs a set of other instances Ω_i=$\{\boldsymbol{x}_j, \boldsymbol{x}_k, \ldots\}$

rather than just itself. The reconstruction error of sample $\boldsymbol{x}_i$ is now defined as:

$$\sum_{j\in\Omega_i} s_{ij} \left\|\boldsymbol{x}_j - \boldsymbol{x}'_i\right\|^2, \tag{2.5}$$

where $\boldsymbol{x}'_i$ denotes the reconstructed instance of $\boldsymbol{x}_i$ and s_{ij} denotes the weight assigned to the reconstruction error as defined by a relational function of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ on the learned manifold. The total reconstruction error E of n samples is:

$$E = \sum_{i=1}^{n} \sum_{j\in\Omega_i} s_{ij} \left\|\boldsymbol{x}_j - \boldsymbol{x}'_i\right\|^2. \tag{2.6}$$

Minimizing E enables the GAE in capturing the structure of the underlying manifold.

Diffusion nets[10] is a neural network-based approach that uses diffusion maps in their dimensionality reduction scheme. Diffusion nets utilize autoencoders as a part of their framework. The encoder and decoder networks are trained separately. The encoder is trained to learn the diffusion map embedding Ψ of the data set $\mathcal{X}$. A new constraint had further been added to the encoder's training objective to facilitate its output to approximate the embedding. The decoder is trained to learn the inverse mapping between the embedding and the data. The encoder and decoder networks are then stacked together to obtain the autoencoder, where the latent layer computes the diffusion embedding of the data set. The autoencoder can then be used for a number of other applications like out-of-sample extension, outlier detection and denoising.

Chapter 3

Proposed Method

Let there be, as described earlier, a data set $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ consisting of n samples, such that, each sample $\boldsymbol{x}_i \in \mathbb{R}^D$. Further, assume that the data lies on a nonlinear manifold of dimensionality d embedded in the D-dimensional input space. Our objective is to preserve this underlying structure within the data as much as possible in the low-dimensional representation. Specifically, we would like to preserve the intrinsic geometric structure of the underlying manifold by preserving both its local and global geometrical properties. Formally, our goal can be described as to obtain a lower-dimensional embedding $\mathcal{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n\}$ of dimensionality d, where each $\boldsymbol{y}_i$ represents the embedding of the corresponding data point $\boldsymbol{x}_i$, that faithfully preserves the intrinsic geometry of the data by preserving both it's local and global geometrical properties.

To achieve this, we use autoencoders. An autoencoder is a neural network trained to attempt to copy its input to its output. Internally, it has a middle layer that describes a low-dimensional representation of the input. We intend to utilize this low-dimensional representation as our embedding $\mathcal{Y}$ once the autoencoder has been effectively trained. Consequently, the dimensionality of the middle layer becomes d. The network may be viewed as consisting of two parts: an encoder function $f(\cdot)$ and a decoder function $h(\cdot)$. The encoder function f maps the input $\boldsymbol{x}_i$ to a d-dimensional intermediate representation $\boldsymbol{z}_i$ and the decoder function h maps $\boldsymbol{z}_i$ to the reconstruction $r(\boldsymbol{x}_i) = h(f(\boldsymbol{x}_i))$. The autoencoder is trained to minimize the reconstruction error loss (MSE):

$$\mathcal{L}_{MSE} = \frac{1}{n} \sum_{i=1}^{n} \|r(\boldsymbol{x}_i) - \boldsymbol{x}_i\|^2. \tag{3.1}$$

However, a challenge arises. The training loss criterion used doesn't inherently encourage the autoencoder to have geometry-preserving properties. If the autoencoder is sufficiently powerful, the autoencoder may very well learn an intermediate representation that doesn't satisfy our initial objective, i.e. if the encoder and decoder modules are sufficiently powerful, the encoder can map the original data points to an arbitrary low-dimensional representation z_i and the decoder would be able to successfully recover the original data point $\boldsymbol{x}_i$ back from it. Therefore, to ensure that the encoded representation preserves the intrinsic geometry of the data, specifically both the local and global geometrical properties during the projection, we introduce a regularizer. This regularizer explicitly penalizes the autoencoder to preserve the geometry of the data by ensuring that it retains the geodesic distances between data points in the encoded representation. By adding this regularization penalty to the initial training objective (MSE loss), the autoencoder is forced to have structure-preserving properties thereby after successful training the resulting low-dimensional embedding will now align more closely with our intended objective, i.e. $\boldsymbol{z}_i = \boldsymbol{y}_i$.

We begin with a brief description of how the geodesic distances were calculated between data points. The geodesic distances were approximated similarly to as was done in the case of Isomap.

1. We first determined which points are the neighbours of data point $\boldsymbol{x}_i (i \in \{1,2,\ldots,n\})$ based on the Euclidean distances $d_E(\boldsymbol{x}_i,\boldsymbol{x}_j); i < j;$ between pairs of points in the input space.

2. We then connected each point to its k-nearest neighbours. k is a hyper-parameter that controls the number of neighbours connected to each data point. These neighbourhood relations were represented as a weighted graph G over the data points, with edges of weight $d_E(\boldsymbol{x}_i,\boldsymbol{x}_j)$ between the neighbouring points $\boldsymbol{x}_i,\boldsymbol{x}_j$.

3. The geodesic distances between all pairs of points on the manifold were then estimated by computing their shortest path distances $d_G(\boldsymbol{x}_i,\boldsymbol{x}_j)$ in the graph G. For this, we used any of the algorithms, Floyd-Warshall or Dijkstra's. Finally, all the geodesic distances computed were stored in the matrix of graph distances $D_G = \{d_G(\boldsymbol{x}_i,\boldsymbol{x}_j)\}$.

We now define our geometry-preserving regularizer $\mathcal{R}$:

$$\mathcal{R} \quad = \quad \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left| \frac{1}{\sigma_{\mathcal{I}}} \left(d_G \left(\boldsymbol{x}_i, \boldsymbol{x}_j \right) - \mu_{\mathcal{I}} \right) \quad - \quad \frac{1}{\sigma_{\mathcal{H}}} \left(d_E \left(\boldsymbol{y}_i, \boldsymbol{y}_i \right) - \mu_{\mathcal{H}} \right) \right|. \quad (3.2)$$

Here, $\mu_{\mathcal{I}},\sigma_{\mathcal{I}}$ denote the mean and standard deviation of the pairwise geodesic distances set $\Omega_G = \{d_G(\boldsymbol{x}_i, \boldsymbol{x}_j) : i < j; i, j \in B\}$ and $\mu_{D_{\mathcal{H}}}, \sigma_{D_{\mathcal{H}}}$ similarly denote the mean and standard deviation of the pairwise Euclidean distances set $\Omega_E = \{d_E(\boldsymbol{y}_i, \boldsymbol{y}_j) : i < j; i, j \in B\}$. B refers to the batch set used during training and n is its cardinality.

The training objective now optimized by the autoencoder is

$$\mathcal{L}_{reg} = \mathcal{L}_{MSE} + \lambda \mathcal{R}, \tag{3.3}$$

where λ is the regularization coefficient controlling the strength of the regularization.

We now justify how training the autoencoder with the loss function $\mathcal{L}_{reg}$ helps in preserving the intrinsic geometry of the data, specifically the local and global geometrical properties. When the training ends successfully, $\mathcal{L}_{reg}$ should have a small positive value. It means, at this point, that the regularizer loss $\mathcal{R}$ has a small value. In other words, the mean of $\sum_{i<j} \left| d'_G(\boldsymbol{x}_i, \boldsymbol{x}_j) - d'_E(h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)) \right|$ has become small. This would imply that $\left| d'_G(\boldsymbol{x}_i, \boldsymbol{x}_j) - d'_E(h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)) \right|$ has become small for each i, j. Consequently, it would mean that the Euclidean distances between each pair of points in the extracted feature space have approximated the geodesic distances in the input space. This could be said as standardization is a typical way of comparing variables on different scales. Now this would mean that the points nearby on the manifold are also nearby in the latent representation, i.e. the local geometry is preserved and that the points farther on the manifold are also farther in the latent representation, i.e. the global geometry is preserved, thereby fulfilling our initial objective.

Chapter 4

Experiments, Results, and Discussions

4.1 Data Set Description

To test the scheme visually, we use four artificial and four real-world data sets. The artificial data sets are C-Curve, S-Curve, Helix, and Swiss Roll. Each include 2000 samples and have a dimensionality of 3. To generate the Helix data set, we use the equations: $x_3 = \frac{t}{\sqrt{2}}$; x_1 = cos x_3; and x_2 = sin x_3, with t varying from -20 to 20 in steps of 0.02. The four real-world data sets used are: Iris, Frey Face[15], USPS[15], and COIL[11]. Frey Face data set consists of 1965 grayscale images each of size 20×28 pixels, i.e., the input dimension is 560. These are the images of a single person's face with different face orientations and facial emotions. The COIL data set consists of images of 20 different objects which have been captured from 72 equally spaced angles. We have used only the first object in the data set for our experiments. Each image is of resolution 128×128. Thus, the input dimension is 16384 and the number of instances is 72. USPS handwritten digits data set contains 1100 images for each of the handwritten digits: $0 - 9$. Each image is of size 16×16, resulting in inputs of dimension 256. We have considered the images of 0 only for our experiments.

4.2 Experimental Setting

To implement our proposed method, we use two deep multilayer perceptrons with the architectures $a_h = (\text{D,100,100,100,d})^T$ and $a_f = (\text{D,100,100,100,d})^T$ as the encoder $h(\cdot)$ and decoder $f(\cdot)$ of the autoencoder respectively. We use linear, leaky-ReLU, and sigmoid activation functions in the input, hidden, and output layers, respectively. We have implemented the proposed method in Python using `PyTorch` (version 1.2). We also make use of batch-normalization in each layer. Throughout the experiment, we train for 1000 epochs, use a batch size of 32, a learning rate of $1e-3$ and regularization coefficient λ of 1. We use the `Adam` optimizer. We set the seeds of random number generators of Python, `NumPy`, and `PyTorch` to 0, 1, and 2, respectively at the beginning of each experiment. To initialize the weights associated with sigmoidal nodes and leaky-ReLU nodes, we, respectively, use `xavier_uniform_()` and `kaiming_uniform_()` weight initializing methods. We have chosen these parameter settings based on our initial ad-hoc experiments. Every other parameter has the default values as in the library. For storing real values we use the `torch.float64` data type. We use the same autoencoder model for experiments requiring the autoencoder without regularization but with just the loss function changed to the MSE criterion.

4.2.1 Qualitative

We compare the results obtained by the proposed method on the data sets with seven other data visualization methods: Autoencoder (without regularization), Sammon mapping, Isomap, LLE, *t*-SNE, UMAP and diffusion net. All the aforementioned data sets have been preprocessed to linearly scale between $[0.05, 0.95]$. The minimum and the maximum values in the data set have been respectively used. We compare the predictability of the proposed method with that of the out-of sample extension of Isomap and LLE, AE, and simplified NPPE (SNPPE)[13]. For calculating the geodesic distances, we set the value of k to 1% of the training data. Only for the COIL data set, k has been set to 5.

We use the implementations provided in Matlab Toolbox for Dimensionality Reduction (drtoolbox) by Laurens van der Maaten[18] for visualization with Isomap, LLE, and Sammon. *t*-SNE was implemented using the class "manifold.TSNE" of the "scikit-learn"(sklearn) package (version 1.2)[12]. UMAP is implemented using the python API “umap”[9] The diffusion net is implemented with the available code[5]. For LLE, Isomap, UMAP, and diffusion net, we use the same neighborhood size (number of nearest neighbors) as was used by the proposed method. For *t*-SNE, there is no provision of setting the neighborhood size directly. The “perplexity” parameter of *t*-SNE provides an indirect measure of the number of neighbors. So, we set the “perplexity” parameter to the number of nearest neighbors used in the proposed method. For these methods, all other parameters were kept at their default values as was supported by their respective routines. For out-of-sample testing, except for SNPPE, the other three comparing methods are realized using the same MATLAB Toolbox (drtoolbox) provided by Laurens van der Maaten. The results of SNPPE are generated by the code provided by the authors of SNPPE[13]. In these cases also, the neighborhood size, when required, is set to the same value as used by the proposed method. All the other parameters are kept at their default values.

4.3 Results and Discussions

4.3.1 Results and Comparisions

Qualitative

From figures S-1b, S-2b, 4.1b and 4.2b, we observe that the proposed method is able to preserve the global geometry as the points farther on the manifold are farther in the latent representation. We also observe that it is able to protect the local geometry; nearby points on the manifold are mapped to nearby points in the latent representation. Therefore, we conclude that the proposed method is able to faithfully represent the manifold on which the data lie thereby achieving our objective of dimensionality reduction. From figures S-1c, S-2c, 4.1c and 4.2c, we also observe that the autoencoder without regularization (AE) wasn't able to preserve both the local and global geometry of the data sets. Similarly, Sammon's mapping [see figs. S-1g, S-2g, 4.1g and 4.2g] fails in preserving the global geometry of the data sets. Isomap [see figs. S-1e, S-2e, 4.1e and 4.2e] is able to preserve the geometry of the data sets but a drawback of Isomap is that larger distances are given more weightage for preservation than the smaller distances as a result in the case of the Helix data set, points which are close by haven't been properly projected. Observe that points with similar colours haven't been placed linearly. Thus, preserving the local geometry becomes a concern. LLE [see figs. S-1f, S-2f, 4.1f and 4.2f] is able to preserve the geometry to a large extent but a drawback of LLE is that it aims to preserve only the local geometry of the data set, thus it can't take care of the global geometry of the data set as evident in the results with S-Curve. Similar obervations hold true for *t*-SNE [see figs. S-1h, S-2h, 4.1h and 4.2h], UMAP [see S-1i, S-2i, 4.1i and 4.2i] and diffusion net [see figs. S-1d, S-2d, 4.1d and 4.2d] as well.

The proposed method can represent the real-world data sets in two dimensions faithfully, i.e. can represent the 2D manifold on which the data lies. In figure 4.3, the two-dimensional embedding of the Frey-Face data set is shown. Figure 4.3 reveals that the gradual change in the face orientation from left to right is roughly related to the decrease in the projected feature y_2 on the vertical axis. Similarly, the change in the expressed emotion varies roughly with the increase in the projected feature y_1. Smiling, neutral, and annoyed faces are mapped in the left, middle, and right regions, respectively.

Sammon's projection and the autoencoder without regularization (AE) have placed faces expressing different emotions in comparatively overlapping regions as seen in figures S-4 and 4.4. Other neighbourhood-preserving methods have been able to unfold the manifold. Comparing the results in figures S-5, S-3, S-6, S-7 and S-8 with the result in figure 4.3, we can infer that the proposed method has mapped the variations of the Frey-Face data set in a more consistent and desired manner.

For the COIL data set, the expected outcome was obtained by the proposed method (fig. S-17) as well as by Isomap (fig. S-19) and diffusion net (fig. S-24). Results obtained from Sammon's projection, AE. LLE, *t*-SNE and UMAP are shown in figures S-20, S-18, S-21, S-22, and S-23, respectively. They show that these methods cannot represent the manifold faithfully in the two-dimensional space. This can be said because the manifold is a circle like structure embedded in the higher dimensional space which hasn't been similarly represented in the 2D space. In the case of USPS, the proposed method continuously projects zeros with different orientations and line widths onto different regions. The projections by the other eight methods reveal that AE, LLE, *t*-SNE, UMAP, and diffusion net could not represent the data in a way so that orientations and line widths of the character vary smoothly in the projected space. In all three cases of the real-world data sets, the proposed method successfully learns the approximate two-dimensional manifold on which the original data lies.

In the case of Iris, from figs. 4.12a, 4.12b, we can visually observe that the proposed method was able to preserve the global geometric (class) structure and the local geometric (intra-class) structure as well.

4.4 Validation of Predictability for Out-of-Sample Points

As mentioned, to compare the predictability of the proposed method, we consider the following four methods: out-of-sample extension of Isomap, LLE and two parametric methods: AE and SNPPE [13]. Note that SNPPE involves a polynomial mapping. Following [13], we use the second-order and third-order polynomial-based SNPPE. To assess the predictability of the system, we have designed the following three experiments:

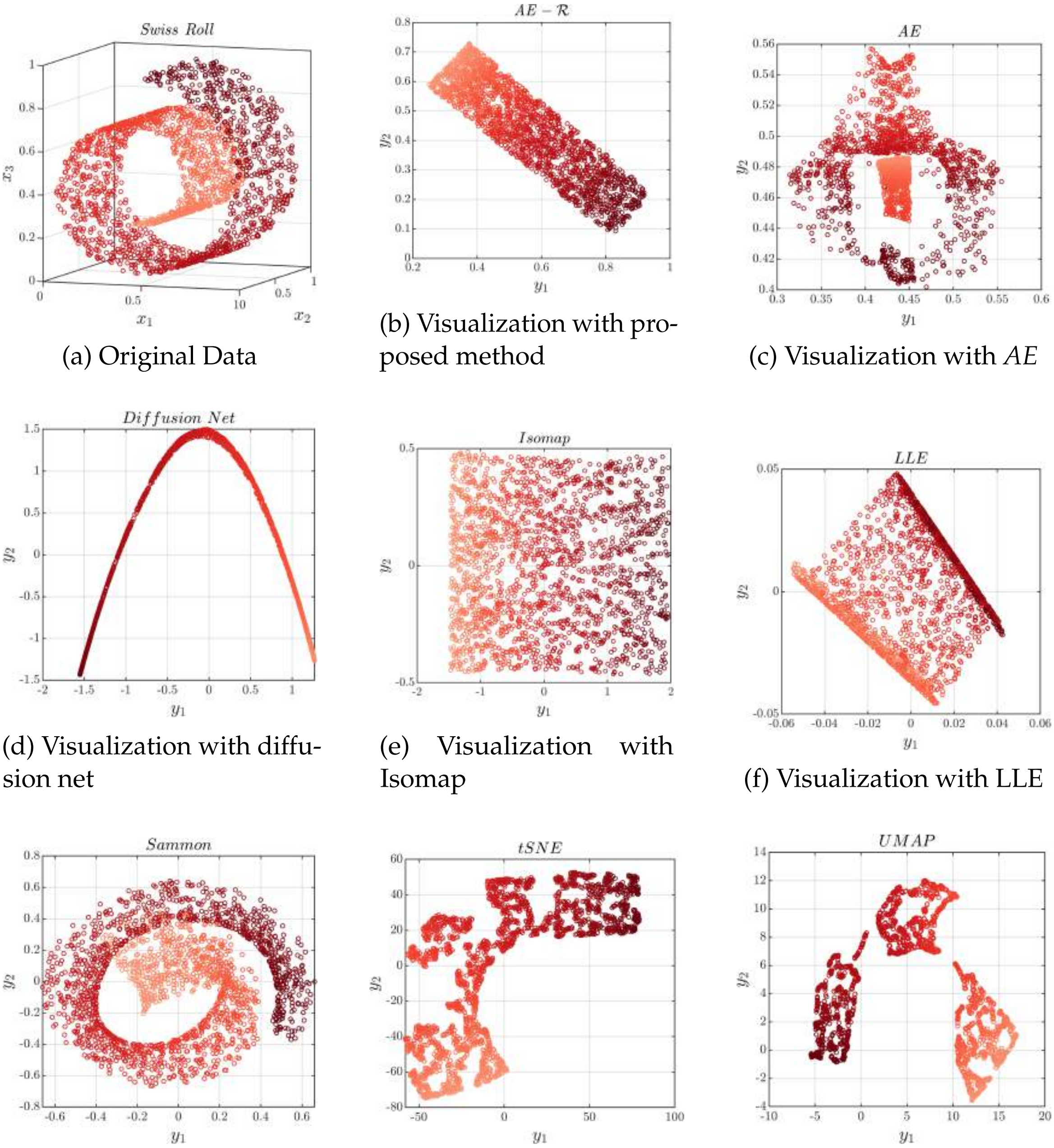
1. randomly dividing the data set into training and test sets;
2. using a contiguous portion of the manifold as the test set and rest as the training set;
3. leave one out validation.

For the first experiment, we use the data sets: Swiss Roll, Frey face. We used 75% of randomly selected instances of the Swiss Roll data set as the test set and the remaining instances as the train set. For Frey face, we used 15 randomly selected instances from the data set as the test set and the rest of the instances as train set. Throughout this paper, in figures corresponding to artificial data sets, we depict the training points using a varying degree of red circles and the test points using a varying degree of blue squares. Similar colours of points indicate that they are in the neighbourhood of each other in the original feature space. For the second experiment, we use only the Swiss roll data set. A contiguous portion of the data set consisting of 50 points has been removed and the remaining 1950 points serve as the training data set. For the third experiment, COIL data set has been used. Each of the 72 instances is considered as the test instance and the remaining 71 instances are used as the training set.

For experiment 1 of the predicatability tests, from fig. 4.6b, we can observe that the proposed method was able to correctly project the test points to their expected locations. Figs. S-25, S-26, 4.7, S-27, S-28 show the projections of the training and test sets by Isomap, LLE, AE, second-order SNPPE, and third-order SNPPE, respectively. Except AE, all the other methods were able to preserve the geometry of data set and appropriately predict the test points as well. Figure S-29 illustrates the projected output by the proposed method for the Frey face data set. In figs. S-31, S-32, we can see that the test instances are projected to the regions where the training instances have the same/similar facial expression and orientation. Thus, for high-dimensional image data sets also, the proposed method, as well as LLE and Isomap, are successful in prediction. The autoencoder without regularization-based lower dimensional projection, shown in fig. S-30 has also clustered different facial expressions and different face orientations in different regions. However, AE is unable to preserve the geometry properly because in some cases, it placed faces with opposite orientations (left and right) closer to each other compared to the faces with the same orientations. A similar placing was observed for opposite expressions (e.g., happy and annoyed) and neutral expressions. But the figs. S-33 and S-34 reveal that SNPPE-based methods neither preserve the geometry of the higher dimensional data nor predict the positions of the test points in a desirable manner.

For the second experiment, fig. 4.9a displays the output of the proposed method corresponding to the training set. Fig. 4.9b shows the test set outputs of the proposed method along with the training set. We represent the results in the same manner for the other comparing methods as well. Figs. S-35a, S-36a, 4.10a, S-37a and S-38a show the projections of the training data for ISOMAP, LLE, AE, second-order SNPPE, and third-order SNPPE, respectively. Similarly, figs. S-35b, S-36b, 4.10b, S-37b and S-38b show the projections of the test set along with the training set for Isomap, LLE, AE, second-order SNPPE, and third-order SNPPE, respectively. These results reveal that the proposed method, as well as the comparing methods except AE, have appropriately interpolated the test points.

For the third experiment, the results of all 72 tests are shown figs. 4.11, S-39, S-40, S-41, S-42 and S-43. The training points are demarcated in green and the test point in red. Instead of showing the images of the object, the consecutive viewing angles of the images are represented by consecutive numbers. In all of the cases, it is found that the test object is placed in the desired position, i.e., in between points corresponding to input images captured in plus-minus five degrees of the viewing angle of the given test point.



(a) Original Data

(b) Visualization with proposed method

(c) Visualization with *AE*

(d) Visualization with diffusion net

(e) Visualization with Isomap

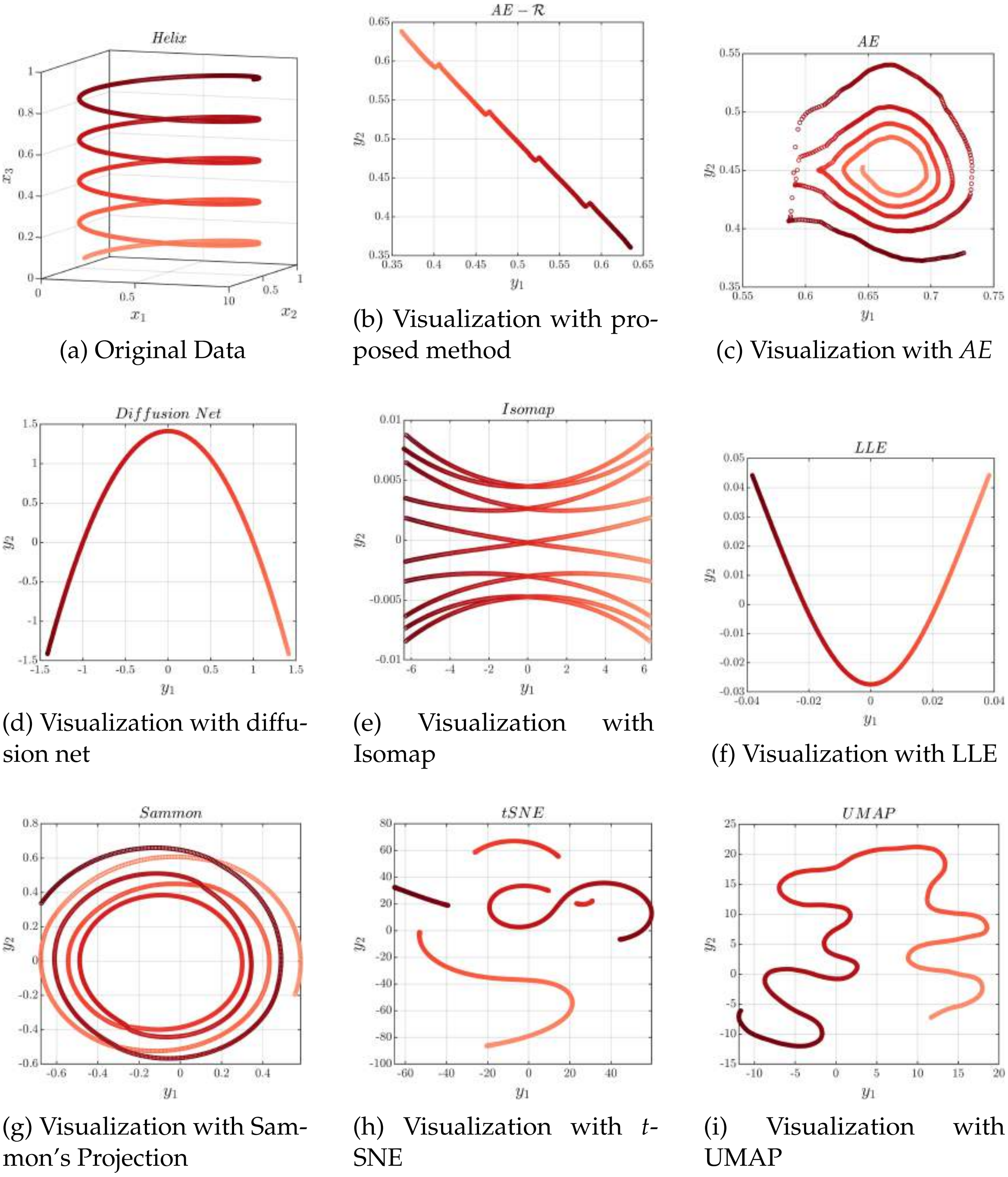
(f) Visualization with LLE

(g) Visualization with Sammon's Projection

(h) Visualization with *t*-SNE

(i) Visualization with UMAP

Fig. 4.1 Swiss Roll: Original data and visualization with various methods



(a) Original Data

(b) Visualization with proposed method

(c) Visualization with *AE*

(d) Visualization with diffusion net

(e) Visualization with Isomap

(f) Visualization with LLE

(g) Visualization with Sammon's Projection

(h) Visualization with *t*-SNE

(i) Visualization with UMAP

Fig. 4.2 Helix: Original data and visualization with various methods.

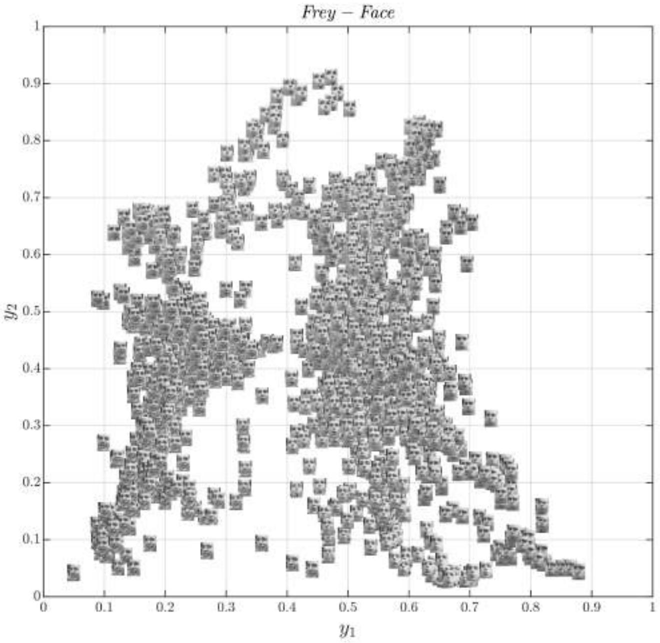


Fig. 4.3 Visualization of Frey Face with proposed method

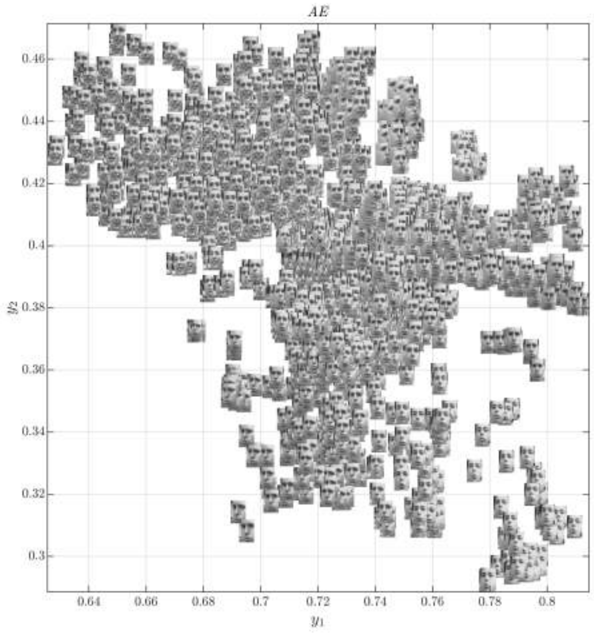
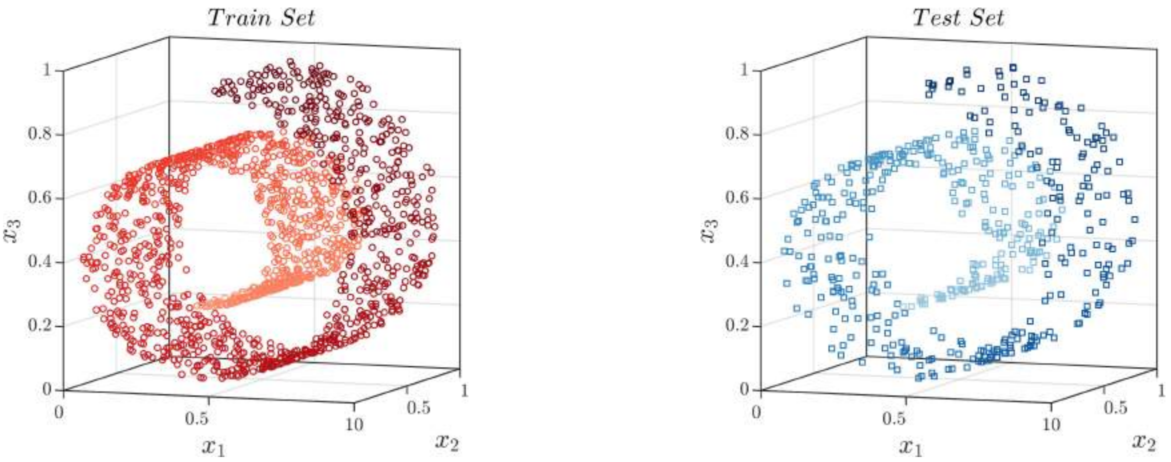


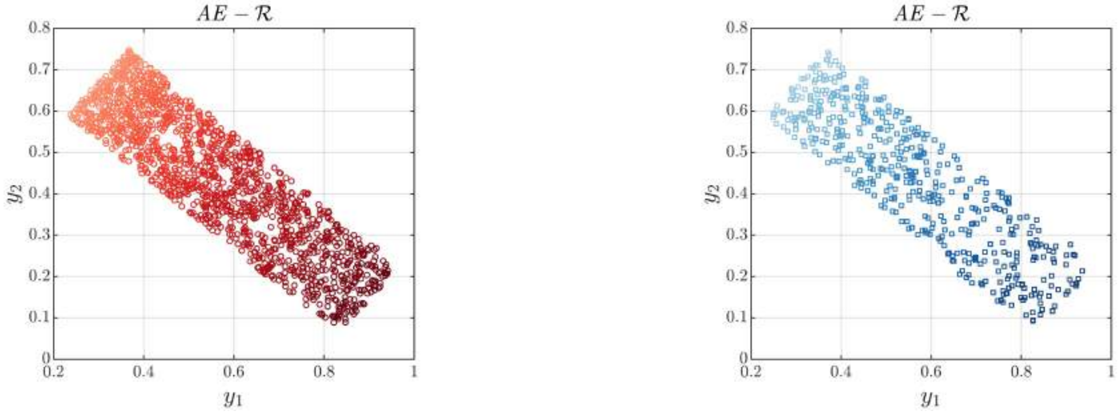
Fig. 4.4 Visualization of Frey Face with *AE*



(a) Train Set

(b) Test Set

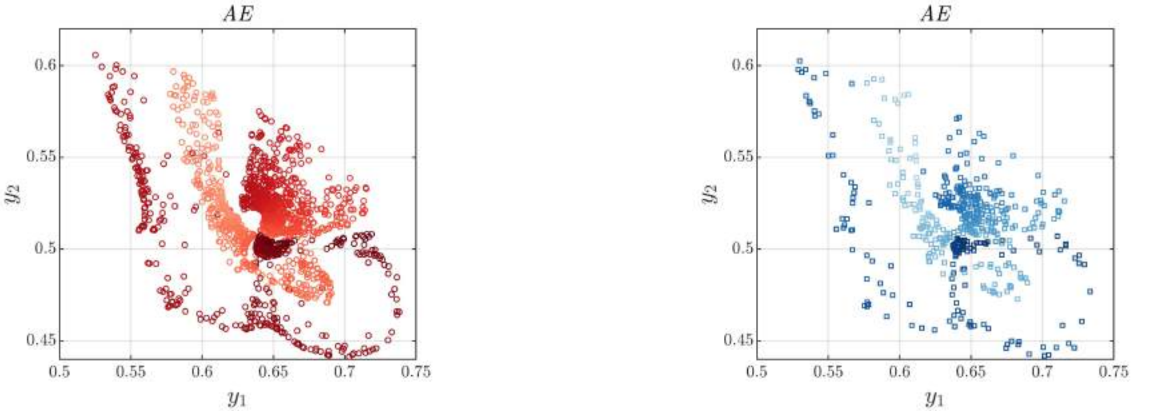
Fig. 4.5 For experiment 1 on the validation of predictability with the Swiss Roll data.



(a) Visualization with proposed method for train set

(b) Visualization with proposed method for test set

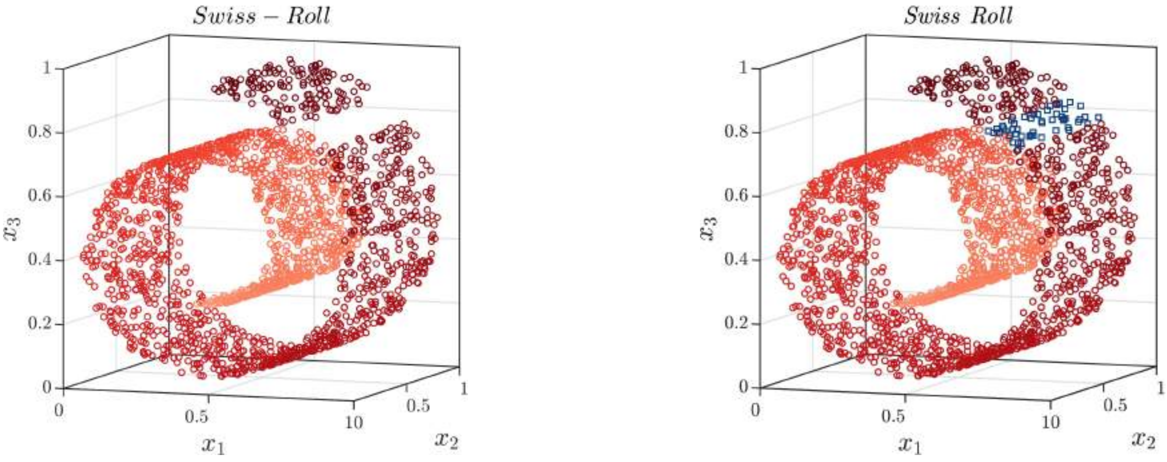
Fig. 4.6 For experiment 1 on the validation of predictability with the Swiss Roll data.



(a) Visualization with *AE* for train set

(b) Visualization with *AE* for test set

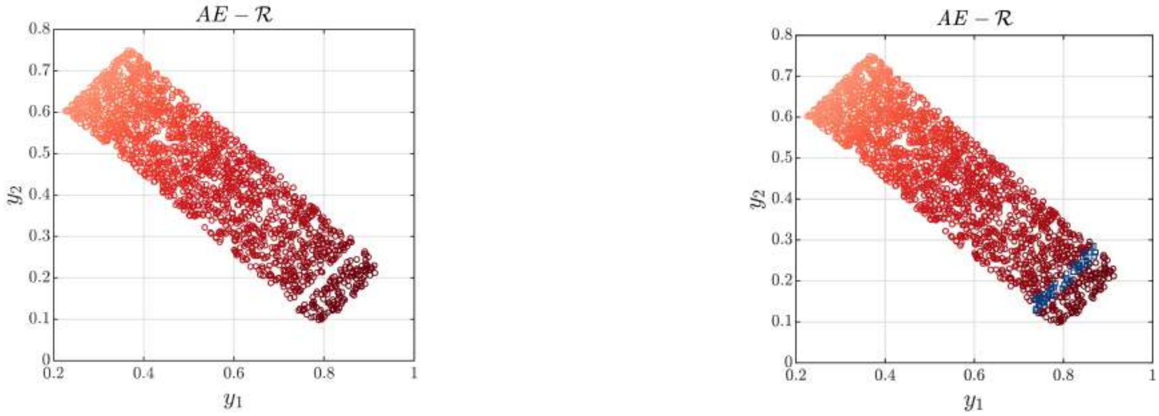
Fig. 4.7 For experiment 1 on the validation of predictability with the Swiss Roll data.



(a) Train Set

(b) Train Set and Test Set

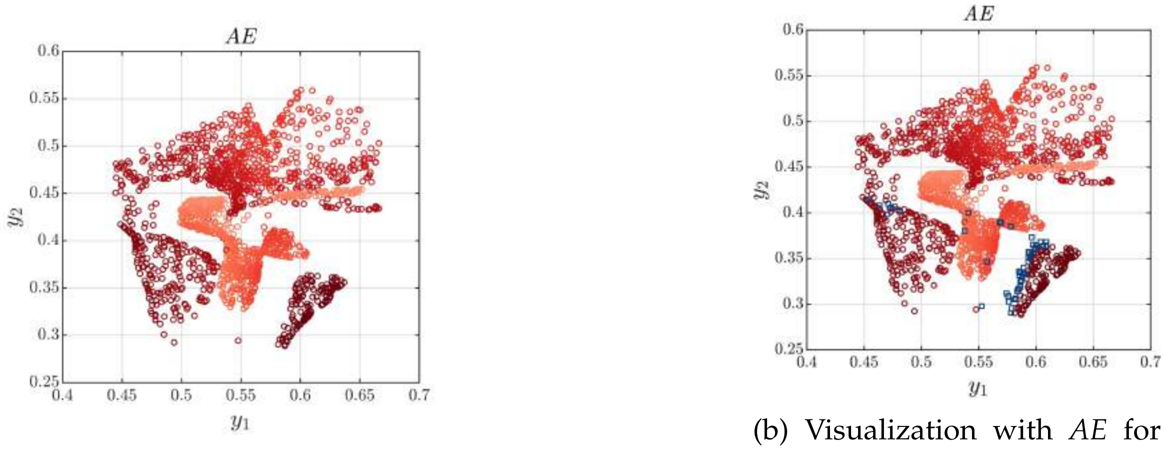
Fig. 4.8 For experiment 2 on the validation of predictability with the Swiss Roll data.



(a) Visualization with proposed method for train set

(b) Visualization with proposed method for train set and test set

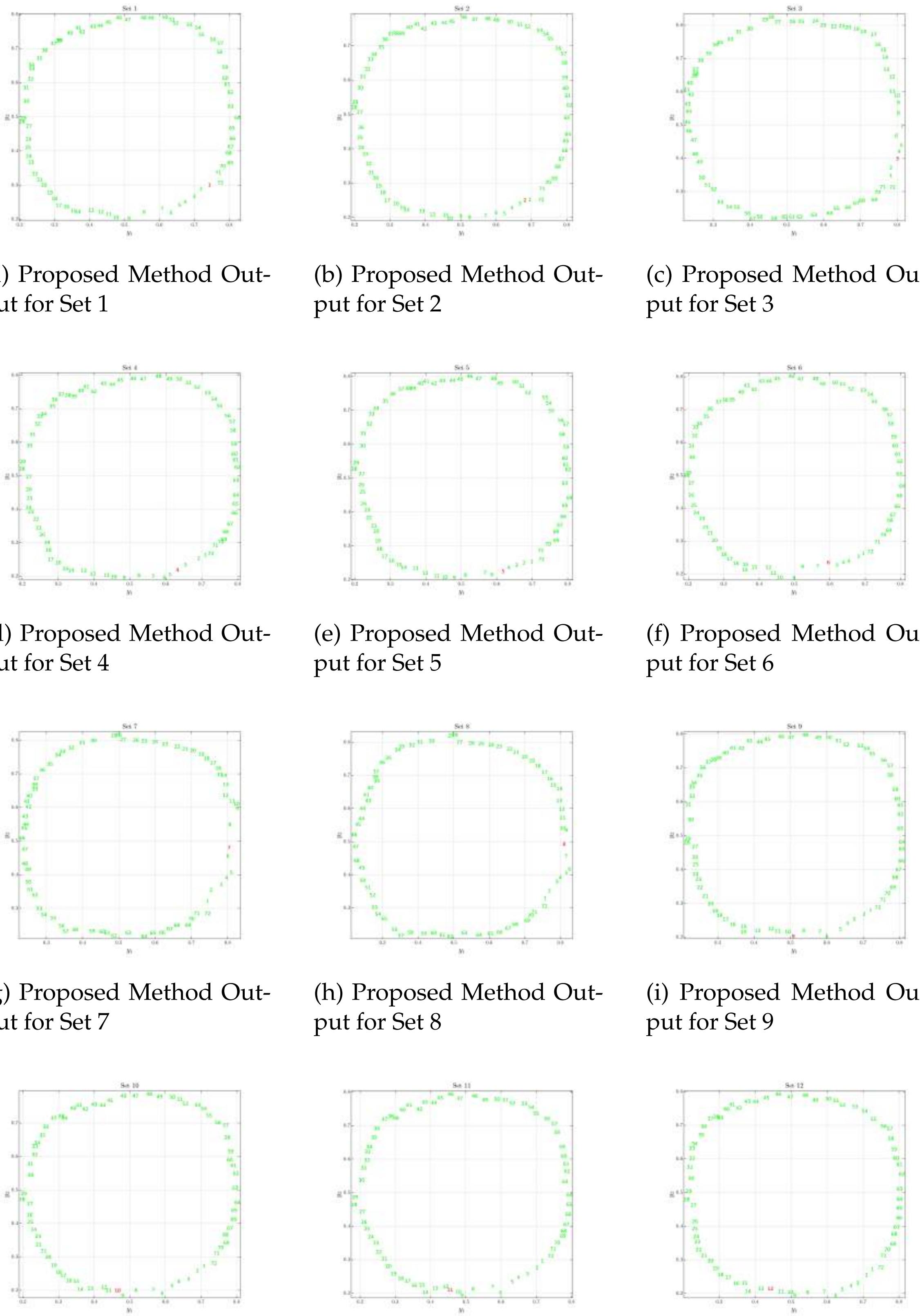
Fig. 4.9 For experiment 2 on the validation of predictability with the Swiss Roll data.



(a) Visualization with *AE* for train set

(b) Visualization with *AE* for train set and test set

Fig. 4.10 For experiment 2 on the validation of predictability with the Swiss Roll data.



(a) Proposed Method Output for Set 1

(b) Proposed Method Output for Set 2

(c) Proposed Method Output for Set 3

(d) Proposed Method Output for Set 4

(e) Proposed Method Output for Set 5

(f) Proposed Method Output for Set 6

(g) Proposed Method Output for Set 7

(h) Proposed Method Output for Set 8

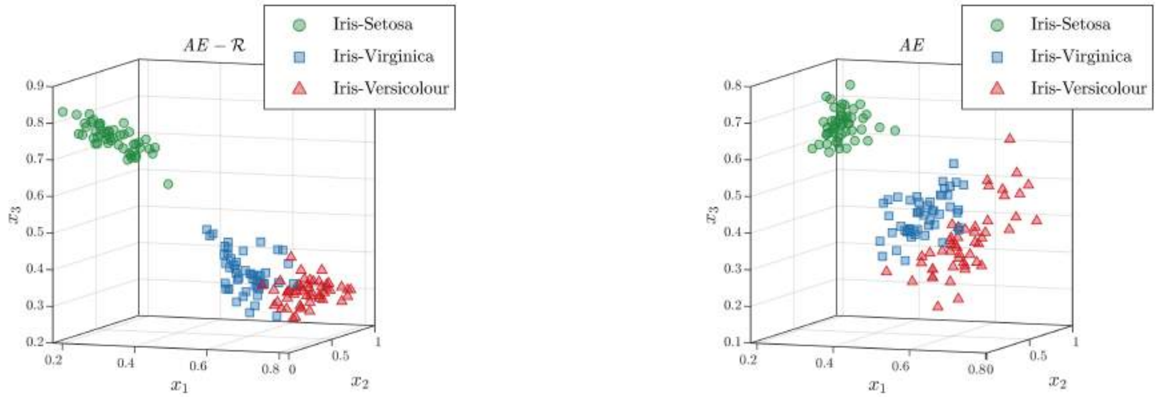
(i) Proposed Method Output for Set 9

(j) Proposed Method Output for Set 10

(k) Proposed Method Output for Set 11

(l) Proposed Method Output for Set 12

Fig. 4.11 For experiment 3 on validation of predictability with leave one out sets composed of the first object of the COIL data set



(a) Visualization of Iris data set with proposed method

(b) Visualization of Iris data set with *AE*

Fig. 4.12 Visualization of Iris data set

Chapter 5

Conclusions, Limitations, and Future Scope

In this work, we propose a neural network-based dimensionality reduction scheme for data visualization. The proposed method uses a structure-preserving regularizer to preserve the local and global geometry of the datasets. We illustrate the effectiveness of the proposed method using four artificial and four real-world datasets. We visually compare with seven other methods. Further, we have conducted three experiments to test the predictability of our method using both artificial and real-world datasets.

Several questions remain unanswered. How does the performance change with changes in the regularization coefficient λ and the normalization scheme? We need to perform more experiments to investigate further. We also need to find some ways to assess the performance of the proposed method numerically. It will enable us to numerically compare our method with the existing methods. In the future, we plan to address these issues.

We plan to use the proposed method to achieve the following, which we do not investigate in this work as we consider them in the future scopes of this work:

1. We plan to use the proposed method for feature extraction to achieve tasks other than data visualization. They include clustering and classification performed in the extracted feature space.

2. Given a point in the extracted feature space, the proposed approach may predict its pre-image in the original feature space.

3. It may enable us to perform calculations in the embedding space and then pull them back to the data space [10]. For instance, we may find interpolation

of points in the embedding space or may compute centroids in a clustering application [10].

References

[1] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271.

[2] Ferreira de Oliveira, M. and Levkowitz, H. (2003). From visual data exploration to visual data mining: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):378–394.

[3] FISHER, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188.

[4] Floyd, R. W. (1962). Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345.

[5] G.Mishne (2017). Diffusionnet - a geometric autoencoder.

[6] He, X. and Niyogi, P. (2003). Locality preserving projections. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, page 153–160, Cambridge, MA, USA. MIT Press.

[7] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417.

[8] Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

[9] McInnes, L., Healy, J., Saul, N., and Großberger, L. (2018). Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861.

[10] Mishne, G., Shaham, U., Cloninger, A., and Cohen, I. (2019). Diffusion nets. *Applied and Computational Harmonic Analysis*, 47(2):259–285.

[11] Nene, S. A., Nayar, S. K., and Murase, H. (1996). Columbia university image library (coil-20).

[12] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(null):2825–2830.

[13] Qiao, H., Zhang, P., Wang, D., and Zhang, B. (2013). An explicit nonlinear mapping for manifold learning. *IEEE Transactions on Cybernetics*, 43(1):51–63.

[14] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

[15] S.T.Roweis (2020). Data for matlab hackers, frey face.

[16] Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

[17] Torgerson, W. S. (1952). Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419.

[18] van der Maaten, L. (2007). Matlab toolbox for dimensionality reduction. *Proceedings of the Belgian-Dutch Artificial Intelligence Conference*, 2007.

[19] Wang, W., Huang, Y., Wang, Y., and Wang, L. (2014). Generalized autoencoder: A neural network framework for dimensionality reduction. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 496–503.