2021 IEEE/ACIS 22nd International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD) | 978-1-6654-0403-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/SNPD51163.2021.9705000

Structure-Preserving Deep Autoencoder-based Dimensionality Reduction for Data Visualization

Ayushman Singh
Department of Computer Science and Engineering
Indian Institute of Information Technology Guwahati
Guwahati, India
singhayushman13@gmail.com

Kaustuv Nag
Department of Computer Science and Engineering
Indian Institute of Information Technology Guwahati
Guwahati, India
kaustuv.nag@gmail.com

***Abstract*—Here, we propose a structure-preserving deep autoencoder-based dimensionality reduction scheme for data visualization. For this, we introduce two regularizers for regularizing autoencoders. The proposed regularizers help the encoded feature space preserve the local and global structures present 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 method has two variants, depending on which regularizer it uses. The proposed approach, moreover, is unsupervised and has predictability. We use three synthetic datasets and one real-world dataset to illustrate the effectiveness of the proposed method. We also visually compare it with three state-of-the-art data visualization schemes and discuss several future research directions.**

***Index Terms*—Data visualization, deep autoencoders, dimensionality reduction, regularization.**

I. Introduction

Data visualization is one of the principal ways of exploratory data analysis. However, real-world data is usually more than three-dimensional, which we cannot visualize in a straightforward manner using traditional scatter plots. Researchers have proposed several ways to visualize such data. Some of these strategies map each sample to an icon, with visual features varying with data values. They include Chernoff faces [1], "stick figure" [2], and shape-coding [3]. The radial coordinate visualization (RadViz) [4] employs a high-dimensional geometric technique. Another frequently encountered techniques are pixel-based techniques [5]. They use pixels to represent data values with different attributes exhibited among different windows [5]. A detailed survey on such schemes is in [6]. However, while one can employ the above techniques to visualize data consisting of more than three dimensions, a common shortcoming is that the visualizations are hard to interpret. It limits the use of such techniques in real-world applications. Dimensionality reduction-based schemes for data visualization may reduce such limitations.

Let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a dataset having n samples with dimensionality D_1, such that, $\mathbf{x}_i \in \mathbb{R}^{D_1}; i = 1, 2, \ldots, n$. Dimensionality reduction schemes usually attempts to find a suitable mapping $\phi : \mathbb{R}^{D_1} \rightarrow \mathbb{R}^{D_2}$, $D_1 > D_2$, such that, $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$ be a D_2-dimensional representation of $\mathcal{X}$. Here, $\mathbf{y}_i = \phi(\mathbf{x}_i)$; $i = 1, 2, \ldots, n$; is the latent representation of $\mathbf{x}_i$; $i = 1, 2, \ldots, n$. For this, we typically minimize a loss function $\psi(\mathcal{X}, \mathcal{Y})$ to find the optimal mapping $\hat{\phi}(\cdot) = \text{argmin}_{\phi}\{\psi(\mathcal{X}, \mathcal{Y})\}$. When D_2 is either 2 or 3, we may visualize a given dataset using 2 or 3-dimensional scatter plots, respectively. Sometimes, we may choose D_2 to be 4, and then, we may represent the fourth dimension using a color code similar to as used in heat maps. Usually, dimensionality-reduction schemes find a suitable mapping by optimizing a judiciously chosen objective function. The construction of the objective function ensures that optimizing it would preserve some information/structures, which exist in the input high-dimensional space, in the latent reduced space. Typically there are two types of structures: global and local. To preserve a global structure, we need to keep the inter-cluster relationships. On the contrary, to conserve the local structures, we need to protect intra-cluster relationships.

There are many dimensionality reduction schemes that we use for data visualization. Linear methods for dimensionality reduction include principal component analysis (PCA) [7], linear discriminant analysis [8], canonical correlation analysis (CCA) [9], and locality preserving projections [10]. These methods, however, may only be effective if the data is lying on high-dimensional hyperplanes, as they may not model complex relationships [11]. They are generally not preferred as the real-world datasets may not lie on a hyperplane. Non-linear methods, such as, Isomap [12], multidimensional scaling (MDS) [13], Sammon mapping [14], locally linear embedding (LLE) [15], are more effective on learning and projecting non-linear relationships present in the data. t-distributed stochastic neighbour embedding (t-SNE) [16], and uniform manifold approximation and projection (UMAP) [17] are two current popular state-of-the-art non-linear dimensionality reduction schemes.

A shortcoming of most of the above methods is that they lack predictability. In other words, once the learning phase is over, these methods cannot make any prediction about unknown points which are not present in the training data. Some works, such as in [18], [19], have extended some of the above methods to achieve predictability. Another shortfall

*This work is financially supported by the Science and Engineering Research Board, Department of Science and Technology, Ministry of Science and Technology, Government of India under the grant number SRG/2020/002132.

of the above schemes is that they may not preserve both global and local structures. To address these issues, here we propose a structure-preserving deep autoencoder-based dimensionality reduction scheme for data visualization. For this, we introduce two regularizers. The proposed method has two variants depending on the regularizer it uses. It has the following abilities:

1) It enables visualization of the data in a 2 or 3-dimensional extracted feature space.
2) It preserves both local and global structures during the projection.
3) It has predictability. In other words, the proposed method can faithfully project unknown (test) data points.
4) The approach is unsupervised.

II. Related Works

A. Methods That Do Not Use Neural Networks

PCA [7] is one of the most frequently used dimensionality reduction techniques. The method aims to find linear transformations to retain the maximum possible amount of variance in the data. The lower-dimensional projections lie in a subspace with a coordinate system specified by a set of orthogonal vectors, called principal components. PCA has also been extended to kernel PCA [20] for situations when the high-dimensional data lies on a non-linear manifold. Kernel PCA computes the principal components in high-dimensional feature spaces. Here, we achieve the mapping from the input space to the high dimensional space using kernel transformation.

MDS [13] is another classical approach to dimensionality reduction. While mapping high-dimensional data to a lower-dimensional space, it attempts to preserve pairwise distances. Metric MDS, also known as classical MDS, is formulated as follows [21]:

$$E_{\texttt{MDS}} = \sqrt{\frac{\sum_{i<j} \left(\texttt{d}[\mathbf{x}_i,\mathbf{x}_j]-\texttt{d}[\mathbf{y}_i,\mathbf{y}_j]\right)^2}{\sum_{i<j} \left(\texttt{d}[\mathbf{x}_i,\mathbf{x}_j]\right)^2}}, \tag{1}$$

where n, $\mathbf{x}_i$, $\mathbf{x}_j$ are as defined earlier. Here, $\texttt{d}[\cdot,\cdot]$ is a distance measure. We usually choose Euclidean distance as $\texttt{d}[\cdot,\cdot]$. Sammon mapping [14] is a frequently adopted case of metric MDS. It minimizes the following error function, called Sammon's stress or Sammon's error:

$$E_{\texttt{Sammon}} = \frac{1}{\sum_{i<j} \texttt{d}[\mathbf{x}_i,\mathbf{x}_j]} \sum_{i<j}^{n} \frac{\left(\texttt{d}[\mathbf{x}_i,\mathbf{x}_j]-\texttt{d}[\mathbf{y}_i,\mathbf{y}_j]\right)^2}{\texttt{d}[\mathbf{x}_i,\mathbf{x}_j]}. \tag{2}$$

Isomap [12] is a nonlinear generalization of classical MDS. Here, we begin by computing the geodesic distances that represent the path between pairs of points along the surface of the manifolds. Isomap first constructs a k-nearest neighbour graph, where each edge has an associated weight. Here, the weight of an edge is the Euclidean distance between its corresponding vertices. Isomap then proceeds to compute the geodesic distance between points as the shortest path distance in the graph. For this, we usually use Floyd-Warshall [22] or Dijkstra's [23] algorithms. Isomap then applies classical MDS using these geodesic distances.

In [24], researchers have proposed a fuzzy rule-based system (FRBS). It is an unsupervised nonlinear manifold learning method. It utilizes the geodesic distance as in Isomap. Here, the authors use a Takagi Sugeno type [25] fuzzy rule-based system to optimize their objective function. The objective function was introduced in [26] and is similar to Sammon's stress function. The error function is as follows:

$$E_{\texttt{FRBS}} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{\left(\texttt{g}[\mathbf{x}_i,\mathbf{x}_j]-\texttt{d}[\mathbf{y}_i,\mathbf{y}_j]\right)^2}{\texttt{g}[\mathbf{x}_i,\mathbf{x}_j]}. \tag{3}$$

Here, $\texttt{d}[\cdot,\cdot]$ denotes the Euclidean distance and $\texttt{g}[\cdot,\cdot]$ denotes the geodesic distance.

Another dimensionality reduction scheme, called t-SNE [16], models the probability distribution of the neighbouring set of points for each point in the dataset. It considers a Gaussian distribution in the high-dimensional space and a t-distribution in the lower dimensional space. The heavy tails property of the t-distribution helps to evenly spread the points in the extracted feature space compared to the Gaussian distribution. t-SNE finds the lower-dimensional representation by minimizing the Kullback-Leibler divergence between the joint probability distributions from these two spaces.

A recent well-performing method is UMAP [17]. First, this method uses a combinatorial topological representation, called Čech complex, to construct a weighted graph in the high dimensional input space. The weights of the edges in this graph represent the closeness of a given point is to another. Then, it employs an optimization scheme to find a similar graph in a lower-dimensional space. For this, it projects data into a lower-dimensional space by a force-directed graph layout algorithm.

B. Neural Network-based Methods

Ivis [27] is a recent method for dimensionality reduction, where the authors make use of Siamese neural networks [28]. The neural network here is optimized using the following variant of the standard triplet loss function:

$$E_{\texttt{S}} = \sum_{\substack{\mathbf{y}_a\in\mathcal{Y}\\ \mathbf{y}_p\in\mathcal{Y}\\ \mathbf{y}_n\in\mathcal{Y}}} \left[\texttt{d}[\mathbf{y}_a,\mathbf{y}_p] - \min(\texttt{d}[\mathbf{y}_a,\mathbf{y}_n],\texttt{d}[\mathbf{y}_p,\mathbf{y}_n]) + m\right]_+, \tag{4}$$

where, $\mathbf{y}_a$, $\mathbf{y}_p$, $\mathbf{y}_n$ correspond to anchor, positive, and negative points, respectively. Here, $m \in \mathbb{R}^+$ is called the margin. Optimizing (4) leads to the anchor point being closer to the positive sample than to the negative sample by the margin m. The positive samples points are randomly selected from the k-nearest neighbours of the anchor point in the input space, while the negative sample can be any point not in the k-nearest neighbours of the anchor point.

In diffusion nets [11], a neural network encoder is trained to learn the diffusion map [29] embedding of the dataset. Diffusion maps [29] define a Markov random walk on the dataset graph. Diffusion distance, a measure of the proximity of data points, is obtained by performing the random walk for a fixed number of steps. The lower-dimensional embedding then tries

to retain these diffusion distances. Diffusion net subsequently trains a decoder network to reconstruct the original dataset from the encoder embedding. Finally, the encoder and decoder networks are stacked together to form an autoencoder (AE).

In generalized autoencoders (GAEs) [30], the authors modified the reconstruction loss, such that each point not only reconstructs itself but also reconstructs a set of instances. The reconstruction error involves a weight. The weight depends on the relation of each sample with the point that it rebuilds. This relational function is defined on the learned manifold, helping the GAE capture the intrinsic structure present in the data.

III. Proposed Method

Let, as described earlier, there be a dataset $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ with n samples, such that, $\mathbf{x}_i \in \mathbb{R}^{D_1}; i = 1, 2, \ldots, n$. Here, D_1 is the dimensionality of the data. We propose a structure-preserving deep autoencoder-based dimensionality reduction scheme for data visualization. This scheme embeds the data on D_2 dimensional space. For the convenience of data visualization, D_2 is either two or three. To achieve the embedding, we use a deep AE with an encoder $h(\cdot)$ and a decoder $f(\cdot)$. The encoder produces a lower-dimensional embedding of $\mathcal{X}$ preserving the intrinsic structure present in $\mathcal{X}$ in the embedded space. Therefore, there needs to be D_2 nodes in the output layer of the encoder and $h : \mathbb{R}^{D_1} \rightarrow \mathbb{R}^{D_2}$ holds. We also assume that the decoder reconstructs the data. Therefore, there need to be D_1 nodes in the output layer of the decoder and $f : \mathbb{R}^{D_2} \rightarrow \mathbb{R}^{D_1}$ holds. Moreover, after the training we expect $\mathbf{x}_i = f(h(\mathbf{x}_i)); i = 1, 2, \cdots, n$. However, if we use only a reconstruction loss, we cannot guarantee that the lower dimensional embedding would preserve the intrinsic structural relationships present in the high-dimensional dataset. To ensure the preservation of such structures, we use regularizers with the reconstruction loss. For this, we introduce two regularizers and use either of them. We keep simultaneous use of both the regularizers as the future scope of this work.

We first define the ratio of the distance between the ith training point and the jth training point in the embedding space to the distance between the ith training point and the jth training point in the input space, r_{ij}, as follows:

$$r_{ij} = {}^{\mathtt{d}_{\mathtt{E}}[h(\mathbf{x}_i), h(\mathbf{x}_j)]}\big/_{\mathtt{d}_{\mathtt{I}}[\mathbf{x}_i, \mathbf{x}_j]}. \tag{5}$$

Here, $\mathtt{d}_{\mathtt{I}}$ and $\mathtt{d}_{\mathtt{E}}$ are two distance measures used in the input space and in the embedded space, respectively. Here, we consider both of them to be Euclidean distances. Note that different distances measures, such as geodesic distances and Mahalanobis distance, may lead to significantly different outcomes.

To design the first regularizer, corresponding to the ith sample point $\mathbf{x}_i$, we define the following random variable:

$$\begin{aligned} X_i = \ \{ & \\ & \log(r_{i1}), \log(r_{i2}), \ldots, \log\left(r_{i(i-1)}\right), \\ & \log\left(r_{i(i+1)}\right), \ldots, \log(r_{in}) \\ \}; \ & i = 1, 2, \ldots, n. \end{aligned} \tag{6}$$

Now, we define the first regularizer R_1 as follows:

$$R_1 = (1/n) \sum_{i=1}^{n} \mathtt{var}[X_i], \tag{7}$$

where $\mathtt{var}[(\cdot)]$ denotes the variance of $(\cdot)$.

To construct the second regularizer, we define a single random variable corresponding to the entire dataset $\mathcal{X}$ as:

$$\begin{aligned} X = \ \{ & \\ & \log(r_{12}), \log(r_{13}), \ldots, \log(r_{in}), \\ & \log(r_{21}), \log(r_{23}), \log(r_{24}) \ldots, \log(r_{2n}), \\ & \ldots, \\ & \log(r_{i1}), \log(r_{i2}), \ldots, \log\left(r_{i(i-1)}\right), \\ & \log\left(r_{i(i+1)}\right), \ldots, \log(r_{in}), \\ & \ldots, \\ & \log(r_{n1}), \log(r_{n2}), \ldots, \log\left(r_{n(n-1)}\right) \\ \}. & \end{aligned} \tag{8}$$

We, now, define the second regularizer R_2 as follows:

$$R_2 = \mathtt{var}[X]. \tag{9}$$

To reduce the reconstruction error, an AE usually minimizes mean squared error (MSE) as follows:

$$E_{\mathtt{MSE}} = (1/n) \sum_{i=1}^{n} \left(\mathbf{x}_i - f(h(\mathbf{x}_i))\right)^2. \tag{10}$$

We use either of R_1 and R_2 along with $E_{\mathtt{MSE}}$ to train the AE. Therefore, the system learns either of the two loss functions:

$$L_1 = E_{\mathtt{MSE}} + \lambda_1 R_1, \ L_2 = E_{\mathtt{MSE}} + \lambda_2 R_2, \tag{11}$$

where $\lambda_1 \in \mathbb{R}^+$ and $\lambda_2 \in \mathbb{R}^+$ are coefficients. The proposed method has two variants: AE-R_1 and AE-R_2, which use L_1 and L_2, respectively, as the loss function during training.

Next, we justify how learning the AE with L_1 helps in preserving both the local and global structure present in the data. When the learning ends successfully, L_1 should have a small positive value. It means, at this point, R_1 has a small value. In other words, $\mathtt{var}[X_i]; \ i = 1, 2, \cdots, n;$ has become small. Therefore, $\mathtt{d}_{\mathtt{E}}[h(\mathbf{x}_i), h(\mathbf{x}_j)] \approx c_i \times \mathtt{d}_{\mathtt{I}}[\mathbf{x}_i, \mathbf{x}_j]; \ i = 1, 2, \ldots, n;$ holds, where $c_i \in \mathbb{R}^+; \ i = 1, 2, \ldots, n;$ is a constant. Consequently, in the extracted feature space, the distance of the ith training point from each other sample has been scaled linearly by a factor of c_i.

We provide a similar argument for L_2. When the learning ends successfully, L_2 should have a small positive value. It means, at this point, R_2 has a small value, and hence, $\mathtt{var}[X]$ has become small. Consequently, $\mathtt{d}_{\mathtt{E}}[h(\mathbf{x}_i), h(\mathbf{x}_j)] \approx c \times \mathtt{d}_{\mathtt{I}}[\mathbf{x}_i, \mathbf{x}_j]$ holds, where $c \in \mathbb{R}^+$ is a constant. In the extracted feature space, therefore, the distance between each pair of points has been scaled linearly by a factor of c.

IV. Experiments, Results, and Discussions

A. Experimental Protocols

We use three artificial datasets: C-Curve, S-Curve, and Swiss Roll. They are shown in Figures 1a, 2a, and 3a, respectively. Each of them has 2000 samples and dimensionality of 3.

We also use a real-world dataset: Iris. The input feature values are linearly scaled in $[0.05, 0.95]$, with the minimum and the maximum values in the dataset. Throughout this paper, in each figure corresponding to artificial datasets, we show the training points using a varying degree of red dots and test points with a varying degree of blue filled squares. Similar colours of points indicate that they are in the neighbourhood of each other in the original feature space.

To realize the encoder $h(\cdot)$ and $f(\cdot)$, we use two deep multilayer perceptrons with the architectures $\mathbf{a}_h = (D_1, 100, 100, 100, D_2)^T$ and $\mathbf{a}_f = (D_2, 100, 100, 100, D_1)^T$, respectively. Here, the ith element in the tuple indicates the number of nodes in that layer. We consider $D_2 = D_1 - 1$. We use linear, leaky-ReLU, and sigmoid activation functions in the input layer, the hidden layers, and the output layer, respectively. Before each non-linear activation function, we use batch normalization in each layer. We train for 1000 epochs, a batch size of 32, $\lambda_1 = 1$, and $\lambda_2 = 1$. We have implemented the proposed method in Python using `PyTorch` (version 1.2). We use the `Adam` optimizer. We use 0.0005, 0.001, 0.0001, 0.001 respectively, as the learning rates for the C-Curve, S-Curve, Swiss Roll, and Iris datasets. To make the experiments reproducible, at the beginning of the experiments corresponding to each dataset, we set the seeds of random number generators of Python, `NumPy`, and `PyTorch` to 0, 1, and 2, respectively. 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 visually compare the proposed method with Isomap, LLE, and Sammon mapping on S-Curve data. We present the results obtained with them in Figures 4a, 4b, and 4c, respectively. For Sammon mapping we use the `mdscale` function in MATLAB with the `Criterion` parameter set to `sammon`. We use the implementations in Matlab Toolbox for Dimensionality Reduction (drtoolbox) by Laurens van der Maaten for Isomap and LLE. For Isomap and LLE, the neighbourhood size parameter is set as 1% of the training data as in [24].

B. Results and Discussions

In Figures 1b, 2b, and 3b, we visualize C-Curve, S-Curve, and Swiss Roll datasets, respectively, with an AE. In Figures 1c, 2c, and 3c, we visualize C-Curve, S-Surve, and Swiss Roll datasets, respectively, with AE$-R_1$ that simultaneously minimizes reconstruction loss and R_1 during training. In Figures 1d, 2d, and 3d, we visualize C-Curve, S-Curve, and Swiss Roll datasets, respectively, with AE$-R_2$ that simultaneously minimizes reconstruction loss and R_2 during training. From Figures 1b, 2b, and 3b, we observe that AE fails to preserve global structures as the visual structure of the manifold changes. From Figures 1b and 3b, we also observe that AE fails to protect the local structure. It is so because samples with heterogeneous colours appear close to each other. We also notice that the predictability of AE is not satisfactory, specifically for Swiss Roll.

From Figures 1c, 2c, 3c, 1d, 2d, and 3d, we observe that when the AE is regularized with R_1 or with R_2, in most of the cases, the regularized AEs preserve both the global and local structures. We note that the predictability of the AE enhances when we use these regularizers. Specifically, for Swiss Roll, the improvement is visually significant.

We now visually compare the proposed method on S-Curve with Isomap, LLE, and Sammon mapping by observing Figures 2c, 2d, 4a, 4b, 4c. Unlike the proposed method, Isomap flattens the manifold so that the extracted samples lie on a plane. In the case of LLE, the extracted samples lie on a 2-dimensional curved surface, which does not look like an 'S'. Similar to the proposed approach, Sammon mapping preserves the 'S' shape of the data. However, as mentioned earlier, Sammon mapping lacks predictability.

In Figures 5a, 5b, and 5c, we, respectively, use AE, AE-R_1, and AE-R_2 to visualize the Iris dataset in a three-dimensional space. From this experiment, we visually confirm that the proposed method retains the global (class) structure. This experiment also confirms that these regularizers help in preserving local (intra-class) structure.

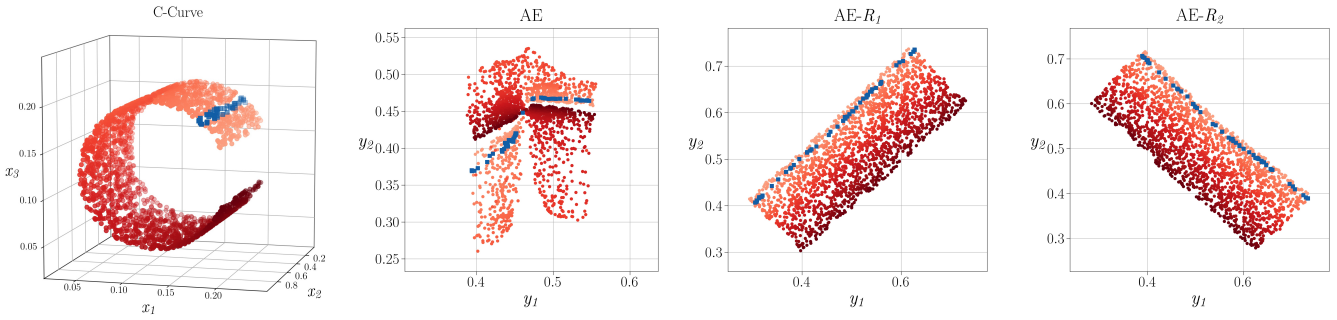
V. Conclusions, Limitations, and Future Scopes

In this work, we propose a data visualization scheme based on dimensionality reduction. The proposed approach uses a structure-preserving deep regularized AE to project the data onto a 2 or 3 dimensional latent space. We illustrate the effectiveness of the proposed method using three artificial and one real-world datasets.

Several questions remain unanswered. For instance, how does the performance change with a change in the choices of $\mathtt{d}_I[\cdot,\cdot]$ and $\mathtt{d}_E[\cdot,\cdot]$? How does the performance change, if we consider $\mathtt{d}_I[\cdot,\cdot]$ to be geodesic distance and $\mathtt{d}_E[\cdot,\cdot]$ to be Euclidean distance? How do the performances change with changes in λ_1, λ_2, and the normalization scheme? What happens if we use both the regularizers simultaneously? 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 existing methods. In the future, we plan to address these issues. The reason behind such limitations is that this paper shows preliminary outcomes of our work as an early showcase. We are on the way to performing deeper investigations in the same direction.

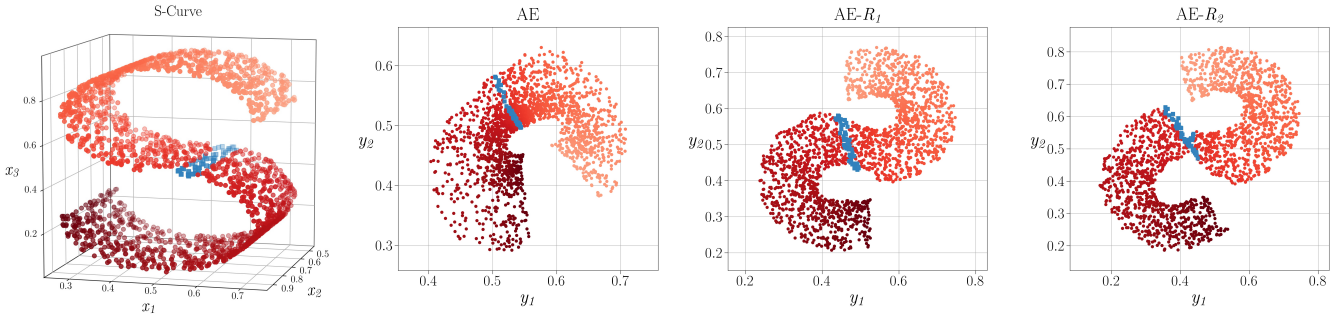
We plan to use the proposed method to achieve the following, which we do not investigate in this work as we consider them in the future scopes of this work:

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) The proposed scheme may reject samples if they are not from the manifold where the training data belong.



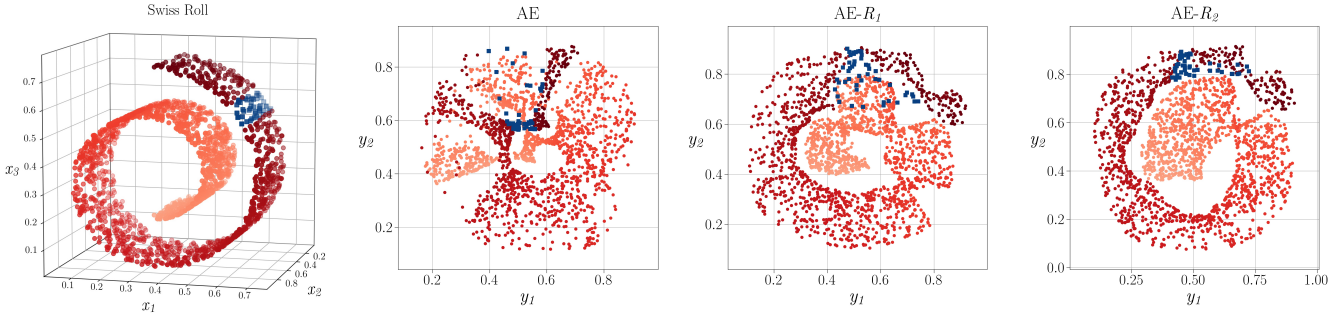
(a) Original data. (b) Visualization with AE. (c) Visualization with AE$-R_1$. (d) Visualization with AE$-R_2$.

Fig. 1. Results with C-Curve. Training points and test points are shown with varying degrees of red dots and blue filled squares, respectively.



(a) Original data. (b) Visualization with AE. (c) Visualization with AE$-R_1$. (d) Visualization with AE$-R_2$.

Fig. 2. Results with S-Curve. Training points and test points are shown with varying degrees of red dots and blue filled squares, respectively.



(a) Original data (b) Visualization with AE. (c) Visualization with AE-R_1. (d) Visualization with AE-R_2.

Fig. 3. Results with Swiss Roll. Training points and test points are shown with varying degrees of red dots and blue filled squares, respectively.

3) Given a point in the extracted feature space, the proposed approach may predict its pre-image in the original feature space.
4) It may enable us to perform calculations in the embedding space and then pull them back to the data space [11]. For instance, we may find interpolation of points in the embedding space or may compute centroids in a clustering application [11].

REFERENCES

[1] H. Chernoff, "The use of faces to represent points in k-dimensional space graphically," *Journal of the American Statistical Association*, vol. 68, no. 342, pp. 361–368, 1973.

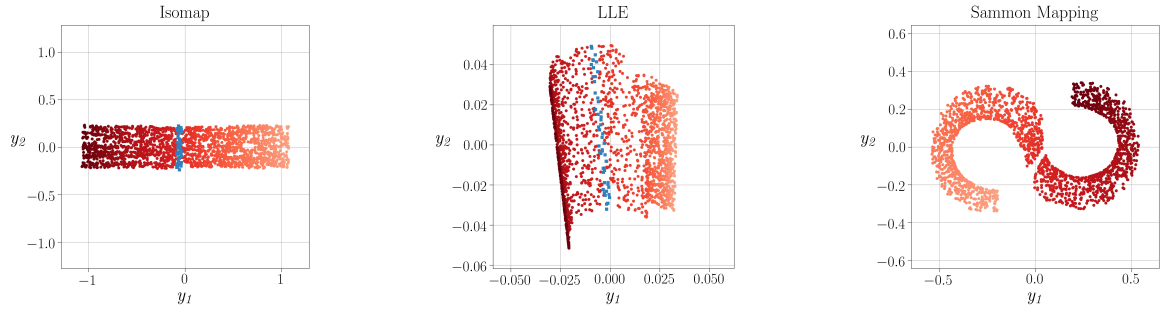
[2] R. M. Pickett and G. G. Grinstein, "Iconographic displays for visualizing multidimensional data," in *Proceedings of the 1988 IEEE Conference on Systems, Man, and Cybernetics*, vol. 514, 1988, p. 519.

[3] J. Beddow, "Shape coding of multidimensional data on a microcomputer display," in *Proceedings of the First IEEE Conference on Visualization: Visualization '90*. IEEE, 1990, pp. 238–246.

[4] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley, "DNA visual and analytic data mining," in *Proceedings. Visualization '97 (Cat. No. 97CB36155)*. IEEE, 1997, pp. 437–441.

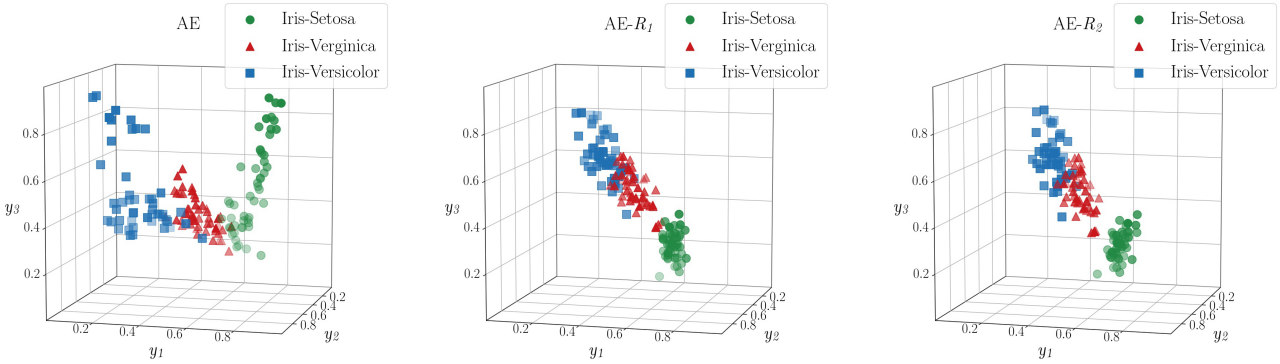
[5] D. A. Keim, "Designing pixel-oriented visualization techniques: theory and applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 59–78, 2000.

[6] M. F. De Oliveira and H. Levkowitz, "From visual data exploration to visual data mining: a survey," *IEEE Transactions on Visualization and*



(a) Isomap. (b) LLE. (c) Sammon mapping.

Fig. 4. Visualization of S-Curve data with (a) Isomap, (b) LLE, and (c) Sammon mapping.



(a) Visualization with AE. (b) Visualization with AE-R_1. (c) Visualization with AE-R_2.

Fig. 5. Visualization of Iris dataset using (a) AR, (b) AE$-R_1$, and (c) AE$-R_2$.

Computer Graphics, vol. 9, no. 3, pp. 378–394, 2003.

[7] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology*, vol. 24, no. 6, p. 417, 1933.

[8] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[9] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, no. 3/4, pp. 321–377, 1936.

[10] X. He and P. Niyogi, "Locality preserving projections," *Advances in Neural Information Processing Systems*, vol. 16, no. 16, pp. 153–160, 2004.

[11] G. Mishne, U. Shaham, A. Cloninger, and I. Cohen, "Diffusion nets," *Applied and Computational Harmonic Analysis*, vol. 47, no. 2, pp. 259–285, 2019.

[12] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[13] J. B. Kruskal, *Multidimensional scaling*. Sage, 1978, no. 11.

[14] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. 100, no. 5, pp. 401–409, 1969.

[15] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[16] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE." *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.

[17] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2020.

[18] L. K. Saul and S. T. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *Departmental Papers (CIS)*, p. 12, 2003.

[19] Y. Bengio, J.-f. Paiement, P. Vincent, O. Delalleau, N. Roux, and M. Ouimet, "Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering," *Advances in Neural Information Processing Systems*, vol. 16, pp. 177–184, 2003.

[20] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *International Conference on Artificial Neural Networks*. Springer, 1997, pp. 583–588.

[21] D. J. Bartholomew, F. Steele, and I. Moustaki, *Analysis of multivariate social science data*. CRC Press, 2008.

[22] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[23] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[24] S. Das and N. R. Pal, "Nonlinear dimensionality reduction for data visualization: An unsupervised fuzzy rule-based approach," *IEEE Transactions on Fuzzy Systems*, 2021.

[25] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 1, pp. 116–132, 1985.

[26] L. Yang, "Sammon's nonlinear mapping using geodesic distances," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2. IEEE, 2004, pp. 303–306.

[27] B. Szubert, J. E. Cole, C. Monaco, and I. Drozdov, "Structure-preserving visualisation of high dimensional single-cell datasets," *Scientific Reports*, vol. 9, no. 1, pp. 1–10, 2019.

[28] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2. Lille, 2015.

[29] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[30] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: a neural network framework for dimensionality reduction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 490–497.