2023-06

A Survey on Dimensionality Reduction Techniques for Time-series Data

ASHRAF, MOHSENA

Independent University, Bangladesh

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

A Survey on Dimensionality Reduction Techniques for Time-series Data

MOHSENA ASHRAF[1*], FARZANA ANOWAR[2*], JAHANGGIR H. SETU[3*], ATIQUL I. CHOWDHURY[4*], ESHTIAK AHMED[5,6#], ASHRAFUL ISLAM[6,7#] AND ABDULLAH AL-MAMUN[8#]

[1]Department of Computer Science, University of Colorado, Boulder, CO 80302 USA (e-mail: mohsena.ashraf@colorado.edu)
[2]Department of Computer Science, University of Regina, Regina, Saskatchewan, Canada (e-mail: anowar@uregina.ca)
[3]Department of Computer Science and Engineering, Daffodil International University, Dhaka 1216, Bangladesh (e-mail: jahanggir15-10533@diu.edu.bd)
[4]Department of Computer Science and Engineering, United International University, Dhaka 1212, Bangladesh (e-mail: achowdhury201036@mscse.uiu.ac.bd)
[5]Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland (e-mail: eshtiak.ahmed@tuni.fi)
[6]Center for Computational and Data Sciences (CCDS), Independent University, Bangladesh, Dhaka 1229, Bangladesh
[7]Department of Computer Science and Engineering, Independent University, Bangladesh, Dhaka 1229, Bangladesh (e-mail: ashraful@iub.edu.bd)
[8]School of Computer and Cyber Sciences, Augusta University, Augusta, GA 30912, USA (e-mail: aalmamun@augusta.edu)

Corresponding author: Ashraful Islam (e-mail: ashraful@iub.edu.bd)

[*, #] Authors contributed equally

ABSTRACT Data analysis in modern times involves working with large volumes of data, including time-series data. This type of data is characterized by its high dimensionality, enormous volume, and the presence of both noise and redundant features. However, the "curse of dimensionality" often causes issues for learning approaches, which can fail to capture the temporal dependencies present in time-series data. To address this problem, it is essential to reduce dimensionality while preserving the intrinsic properties of temporal dependencies. This will help to avoid lower learning and predictive performances. This study presents twelve different dimensionality reduction algorithms that are specifically suited for working with time-series data and fall into different categories, such as supervision, linearity, time and memory complexity, hyper-parameters, and drawbacks.

INDEX TERMS Time-series data, Dimensionality reduction, High-dimensional data, Machine learning

I. INTRODUCTION

MASSIVE amounts of information are the focus of modern data analysis. The increase in data volume is evident not just in the number of samples collected over time, but also in the number of attributes or features [1]. Nowadays, high dimensionality is a common feature of time series and large datasets. A dataset with a larger number of features generally contains more information, but as the number of features grows, there may be an increase in noise and redundancy. Additionally, big data can be challenging to analyze and visualize due to its higher dimensionality. The complexity of high-dimensional data not only increases computational demands and storage requirements, but it can also cause traditional learning approaches to fail in certain cases, leading to the "curse of dimensionality" [2]. The curse of dimensionality describes the problems that arise while categorizing, organizing, and analyzing high-dimensional data that do not exist in low-dimensional domains [3]. Working with high-dimensional data makes knowledge discovery and pattern recognition tasks more challenging due to the abundance of redundant and irrelevant features.

Reducing dimensionality is a viable approach to addressing this problem. By reducing the number of features in the data, this technique aims to improve data quality. Removing unimportant and irrelevant attributes through dimensionality reduction allows machine learning algorithms to run more accurately in less time as well as reducing computational complexity. However, it is imperative that the result of dimensionality reduction (lower feature space) still preserves important and significant information about the original representation. Higher dimensionality can be handled through different ways such as supervised, unsupervised, linear, and non-linear dimensionality reduction techniques.

The ‘curse of dimensionality’ is perceived as more crucial in the case of time-series data because of expanding usage of this sort of data in data mining. Time-series data are collections of chronological observations of temporal objects that can be constructed by collecting data from var-

ious applications at specific intervals [4], [5]. A time series analysis can assist in understanding the underlying process, the pattern of change over time, or the evaluation of the consequences of either a planned or unexpected intervention. Large volume, constant updates, and high dimensionality are the main properties of time-series data [6]. Time-series data is often collected over an extended period, which can result in high data dimensionality. As a result, it is often necessary to reduce the dimensionality of time-series data to facilitate analysis and processing [7]. When reducing time-series data, it is important to ensure that the primary characteristics and representation of the original data are maintained. This will help to improve the performance of the time-series data, as the reduced data must still accurately capture the essential features of the original data [8].

There is a dearth of literature discussing dimensionality reduction methods specifically for time-series data, and no thorough investigations have been conducted in this area. This survey aims to fill this gap by focusing solely on dimensionality reduction techniques for time-series data. The paper provides a review of twelve state-of-the-art techniques and presents a comprehensive investigation along with a brief comparison of these methods. The structure of the paper is as follows: Section 2 briefly introduces the selected dimensionality reduction methods for time-series data. In Sections 3 through 8, we discuss Autoencoder, Principal Component Analysis, Kernel Principal Component Analysis, Laplacian Eigenmap, Singular Value Decomposition, and Locally Linear Embedding, respectively. Next, Sections 9 through 14 cover Isometric Mapping, Maximum Variance Unfolding, Locality Preserving Projections, Diffusion Maps, Discrete Fourier Transform, and Discrete Wavelet Transform, respectively. In Section 15, we compare the discussed dimensionality reduction methods for time-series data. Finally, Section 16 concludes the paper.

II. SELECTION OF DIMENSIONALITY REDUCTION METHODS FOR TIME-SERIES DATA

In recent times, multiple dimensionality reduction techniques data have been proposed, however, none of them is effective for time-series data. For instance, (1) Piecewise Trend Approximation which only focuses on the local trend of the data [9], (2) Piecewise Vector Quantized Approximation which performs poorly for non-stationary datasets [10], (3) Piecewise Constant Approximation that estimates every time-series only with constant value segments [11], (4) Piecewise Aggregate Approximation where choosing the number of segments as a parameter is challenging and highly data dependent [12], (5) Sliced Inverse Regression which suffers from the presence of outliers [13], (6) Factor Analysis has become less popular as it is quite similar to Principal Component Analysis (PCA) [14], (7) Independent Component analysis where the order of the independent components is ambiguous [15]. Due to these issues, we did not consider these algorithms for time series data in our work.

In contrast, we review the following twelve state-of-the-art dimensionality reduction algorithms for analyzing time-series data. We select these twelve techniques for the following reasons:

1) **Autoencoder:** Autoencoder tends to provide better performance by using the latent features of the time-series data for learning, and can predict the unseen data efficiently [16].
2) **Principal Component Analysis (PCA):** PCA is one of the most widely used dimensionality reduction methods due to its simplicity [17]. By utilizing the linear transformation of the Eigen Decomposition, it contains more information, identifies outliers effectively, and reduces the time-series data dimension effectively [18]. This method can also be employed for non-stationary data.
3) **Kernel PCA (KPCA):** KPCA is a variant of PCA that is highly efficient for dealing with non-linear time-series data [19]. It also considers the combination of the predictive features intrinsically for the optimization of the dimension reduction for time-series data [20].
4) **Laplacian Eigenmap (LE):** LE efficiently handles non-linear time-series data [21]. Furthermore, it has the potential to produce the best classification result for time-series data. [22].
5) **Locally Linear Embedding (LLE):** LLE converts high-dimensional time-series data on a manifold into a lower-dimensional format while preserving the input data points' local properties. [23].
6) **Isometric Mapping (Isomap):** Isomap is a manifold-based method that adjusts the local structure of the input data to perform well on non-linear time-series data [24].
7) **Singular Value Decomposition (SVD):** SVD decomposes the input time-series data matrix into a simpler form, making the underlying data structure clearer to comprehend [25]. It also aids in the efficient visualization of time-series data and the speedy analysis of time-series data [26].
8) **Maximum Variance Unfolding (MVU):** MVU's trustworthiness and continuity evaluation appear to have enormous potential for the dimensionality reduction of time-series data [27]. The terminology 'trustworthiness and continuity' explains how well a dataset's local properties are retained after dimensionality reduction [27].
9) **Locality preserving projection (LPP)**: LPP aims to preserve the neighborhood local structure of time-series data as much as possible [28].
10) **Diffusion map (DM):** Short-circuiting in a time series data graph appears to be highly adaptable to diffusion distance [29]. This trait makes DM a reliable dimensionality reduction technique for time-series data. In addition to dimensionality reduction, DM provides effective time-series data visualization [30].
11) **Discrete Fourier Transform (DFT):** The fundamen-

tal idea behind DFT is that any complex signal can be expressed by the overlap of a limited number of waveforms, each denoted by a single complex number termed a Fourier coefficient [31]. The ability to minimize dimensionality is the most basic benefit of representing a time series in the frequency domain. It is a useful method for time-series data since it minimizes the quantity of the data without losing too much information.

12) **Discrete Wavelet Transform (DWT):** DWT can provide both the time and frequency information of the time-series data simultaneously, along with the advantage of dimension reduction of the input time-series data [32].

Table 1 shows the notations that are commonly used in this paper.

TABLE 1. Notations and Description

Notation	Description
A	input dataset
a_i	input data samples
a_j	input data samples
Y	output dataset
y_i	output data samples
y_j	output data samples
n	input dimensions
d	output dimensions
N	total samples
K	kernel function
K_m	kernel matrix
m	nearest neighbor
G	neighborhood graph
M	connectivity matrix

III. AUTOENCODER

Autoencoder is an unsupervised artificial neural network (ANN) that reduces the dimension of the input data from high to low by stacking up a layer of non-linear transformations [33]. It is also known as a replicator neural network (NN) as it replicates the input data to the output. Autoencoder is a widely used method for time-series data analysis which helps in the reduction of the input data dimension. For example, [16], [34]–[38] used Autoencoder based methods for time-series data analysis and dimensionality reduction.

An Autoencoder is composed of 3 different layers; 1) encoder layer, 2) code layer, and 3) decoder layer [39]. The encoder layer is a feedforward, fully connected NN that transforms the original input space to a compressed lower dimensional representation. This compressed output is the distorted version of the original input data. The mid layer of Autoencoder is the code layer, which imposes a bottleneck in the network and contains the reduced representation of the input. Then it passes the compressed input data to the decoder layer. The decoder is a feedforward network that attempts to reconstruct the original input data from the code layer. In simple words, the decoder layer tries to reverse the encoding process by recreating the higher dimensional space from the encoded low dimension.

Now, if the encoder function is ψ, and the decoder function is ϕ, then the conversion of the input data A is the following [40]:

$$\psi : A \rightarrow F \tag{1}$$

$$\phi : F \rightarrow A \tag{2}$$

where ψ maps the original data A to a latent space F and ϕ reconstructs the original data from F.

In this case, data reconstruction error is observed which is defined as the difference between the original input data and the reconstructed data [40]. Autoencoder tries to obtain the original data structure from the lower dimensional representation by minimizing the reconstruction error. The formula of the reconstruction error ($\mathcal{L}$) [40] is shown in Eq. 3.

$$\mathcal{L}(\mathbf{A}, \mathbf{A}') = A - A' \tag{3}$$

where A and A' are the original input data and the reconstructed input data respectively. The reconstruction error can also be referred to as 'loss' of the Autoencoder.

Though the Autoencoder is one of the most widely used dimensionality reduction algorithms, it only works well if the input features are correlated. If the input features are independent, it performs badly and produces lossy output as compared to the original input. Furthermore, because the Autoencoder does not need to learn from dense layers, it may instead use convolutional layers, which are superior at reducing the dimensionality of the image, video, and time-series data [41].

IV. PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is an unsupervised technique that is one of the widely used dimensionality reduction techniques for dealing with the high dimensionality for time-series data [42]. It is a linear transformation method and computationally less expensive [43]. As a result, applying PCA helps to run the MLAs faster while the structure of the original dataset is sustained [44]. Hence, many researchers believe PCA is appropriate for reducing the dimensionality of time-series datasets [45], [46], [47], [48].

PCA first determines the maximum variance of a dataset [49]. Then it calculates the covariance matrix which depicts the information of the correlation of the features using the formula given below [50]:

$$cov(f_1, f_2) = \frac{\sum(f_{1_i} - \overline{f_1})(f_{2_i} - \overline{f_2})}{N} \tag{4}$$

here, f_1 and f_2 are two random features of the input data set, $\overline{f_1}$ and $\overline{f_2}$ are the mean of f_1 and f_2 respectively, N is the number of total data points.

The next step is to find the Eigen values and Eigen vectors of the covariance matrix. The Eigenvectors represent the

information of the data direction which are known as the principal components (PCs) whereas the eigenvalues constitute the information of data variance in that direction. So, PCs are the new linear combinations of the primary features. During this transformation, the PCs are sorted in descending order and the top-most q number of PCs are selected as q features which contain the most important information from the dataset. Suppose, if the input data matrix A is of n-dimension, then PCA reduces it to a dimension of d where $d \leq n$.

The key factor of PCA lies in the projection of the dataset. With the re-orientation of the data from the original axes to the new axes represented by the PCs, the data are then transformed into the new subspace. From there, PCA tries to find out the best line on which the projected data points have the maximum variances [51].

PCA makes the feature selection process easier by scoring the features from the most important to less. As a result, depending on the application, the most significant set of features can be chosen from the dataset. It is a non-iterative process, which eliminates correlated features and reduces overfitting problems [44]. However, PCA does not perform well in the case of non-linear data [49]. Also, It is recommended to standardize the input dataset before applying PCA.

V. KERNEL PCA

PCA does not work well with nonlinear time-series data because it produces a non-optimal subspace [52]. Kernel PCA (KPCA) can manage the non-linearity of time-series data in these instances by generalizing the linear PCA approach into the non-linear case by using a 'kernel trick' [53]. In the new feature space, KPCA maps time-series data to a higher dimension, where data becomes linearly separable. Authors in [54], [55], [56] used KPCA for the reduction of dimensionality for time-series data.

Given a kernel function K, KPCA generates a new feature space for the non-linear data. The kernel function or kernel trick K works using the following equation [57]:

$$K(a_i, a_j) = \psi(a_i)^T . \psi(a_j) \tag{5}$$

where, a_i and a_j are two random data samples. The input data a_i is first mapped to a higher dimension $\psi(a_i)$ by KPCA. So, rather than using the input dataset A, it maps A to a high dimensional feature space $\psi(A)$. Then, in that higher-dimensional feature space, it calculates the linear PCA [53]. By incorporating the kernel trick, it indirectly manipulates the dot product of the samples a_i of A under ψ, and maps a_i to $\psi(a_i)$ which makes the method computationally cheaper [58]. Several kernel functions can be utilized to perform KPCA. Some of the most widely used kernel functions are Radial Basis Function (RBF), Linear Kernel Function, Polynomial Kernel Function, Sigmoid Kernel, etc. where they work using the following equation [59] [60]:

$$RBF : K(a_i, a_j) = exp\left(-\frac{(||a_i - a_j||^2)}{c}\right) \tag{6}$$

$$Linear\ Kernel : K(a_i, a_j) = a_i^T a_j + c \tag{7}$$

$$Polynomial\ Kernel : K(a_i, a_j) = (a_i, a_j)^p \tag{8}$$

$$Sigmoid\ Kernel : K(a_i, a_j) = tanh(\beta_0(a_i, a_j) + \beta_1) \tag{9}$$

here, a_i and a_j are two random data samples. c, β_0, and β_1 are constants that need to be priorly specified by the user [61]. p is the degree of the polynomial which also needs to be predefined. When the data is non-linearly separable, the RBF kernel is utilized [62]. The linear kernel is used for linearly separable data [63]. The polynomial kernel is used to represent the similarity between the data points in a feature space over the polynomials of the original variables [64]. And the sigmoid kernel is employed mostly for neural networks [65].

The selection of the kernel function in KPCA determines the form of mapping from the original variable space to the high dimensional feature space for dimensionality reduction. However, KPCA can lead to memory issues in the case of huge datasets [66].

VI. LAPLACIAN EIGENMAPS

Laplacian Eigenmap (LE) is a manifold-based unsupervised learning that works for non-linear time-series data [67]. This algorithm has a tendency to preserve the local geometrical properties of data points i.e. neighboring data samples in the initial data space remain neighbors in the reduced space. LE is widely used in the reduction of the dimension of time-series data [68], [21], [69].

LE first tries to join the neighboring points from a n-dimensional input vector A and constructs a neighborhood graph G. It uses the concept of the laplacian of the graph [70]. The neighboring points are identified using the following two methods [71]:

- ϵ-ball method: two points a_i and a_j of the input vector A are neighbors if $\| a_i - a_j \| \leq \epsilon$. ϵ is the radius of a ball which is centered at a_i and the point is determined as a linear combination of all other data points inside the ball [72]. This method considers the geometrical properties of the samples, however, it can lead to the construction of more than one connected graph [71]. Moreover, the selection of the parameter ϵ is difficult [73].
- KNN method: Two points a_i and a_j of the input vector A are neighbors if a_i is amongst the m nearest neighbors of a_j or vice versa. In contrast to the previous method, the selection of the parameter m is easier, though this method is not so geometrically instinctive [71].

LE uses two approaches: heat kernel approach and simple-minded approach [74] to provide weight to the connecting edges of any two points a_i and a_j in graph G. These are as follows [71]:

$$heat\ kernel\ approach : W_{ij} = \exp^{-\frac{\|a_i - a_j\|^2}{\tau}} \tag{10}$$

$$simple\ approach : W_{ij} = \begin{cases} 1, & \text{if } a_i \text{ and } a_j \text{ are connected} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where, τ is a parameter which can be any real number and W_{ij} is the weight between the two data points a_i and a_j.

Later, LE tries to find a lower-dimensional subspace where all the local properties of neighboring points are well sustained by minimizing the following cost function [75]:

$$\psi(Y) = \sum_{ij} \parallel y_i - y_j \parallel^2 .W_{ij} \tag{12}$$

Here, Y is the low dimensional representation of the input matrix A. y_i and y_j are the lower dimensional representations of a_i and a_j respectively. Then, LE calculates 'Laplacian matrix (L)' of G using the following formula [76]:

$$L = W - D \tag{13}$$

where D is a diagonal matrix which contains the row sums of the weight matrix W, so that $D_{ii} = \sum_j W_{ij}$. Now, using the laplacian matrix L and the diagonal matrix D, the cost function given in equation (12) can be redefined as an Eigen decomposition problem which is shown below:

$$\sum_{ij} \parallel y_i - y_j \parallel^2 .W_{ij} = 2.Y.L.Y^T \tag{14}$$

By optimizing the cost function, Laplacian Eigenmap tries to ensure that similar data points are mapped together in the low dimensional manifold by maintaining the local distances [77]. However, it can be considered as a limitation of LE that it preserves only the local properties of the neighboring data points [78].

VII. SINGULAR VALUE DECOMPOSITION

Singular Value Decomposition (SVD) is an unsupervised, linear method that is a widely used dimensionality reduction technique for time-series data [79]. SVD is used for the dimension reduction of time-series data by many researchers [26], [80], [81]. It decomposes a real or complex matrix A into three matrices and exhibits the useful characteristics of the primary matrix. This method is a data-driven generalization of the Fourier transform which is based on simple linear algebra [82]. With the help of SVD, the matrix rank can be generated and a lower rank estimation of the matrix decomposition can be acquired. The number of linearly independent rows or columns in a matrix is referred to as its rank [83].

The decomposition of a real matrix A, having a dimension of $(N \times n)$ by SVD is [84]:

$$A = U\Sigma V^T \tag{15}$$

Here, U and V are two unitary orthogonal matrices where for both matrices, the dot product of two columns of a matrix is equal to 0 i.e., $U^T.U = U.U^T = I$ or $V^T.V = V.V^T = I$ [49]. U has a dimension of $(N \times r)$ and V has a dimension of $(r \times n)$. U and V are known as the left singular vector and the right singular vector respectively and Σ is a diagonal matrix having a dimension of $r \times r$ where the non-diagonal elements are zero, and the diagonal elements are non-negative. These non-negative values in Σ are known as singular values. Hence, the matrix Σ can be represented as:

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & & & & \\ & . & & & \\ & & . & & \\ & & & . & \\ & & & & \sigma_r \end{bmatrix} \tag{16}$$

Here, r is the rank of the matrix A, where $r = min(N, n)$, and the singular values are ordered as $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r \geq 0$.

SVD then reconstructs the primary matrix A to a lower rank f by using the following equation [85]:

$$Y = U_f \Sigma_f V_f^T \tag{17}$$

here, Y is the lower dimensional output which only holds the information of the first f values. U_f, Σ_f and V_f^T are truncated versions of U, Σ and V^T and that's why such an SVD is called 'Truncated SVD' [86].

Despite many positives, SVD can be computationally expensive. It works fine on a user-defined matrix but does not work well in adaptive or experimental procedures [87]. However, it is a preferred dimensionality reduction algorithm for the prediction problem [49].

VIII. LOCALLY LINEAR EMBEDDING

Locally Linear Embedding (LLE) is an unsupervised, non-linear method that computes the low dimensional embedding of the input time-series data from a high dimensional space by preserving the geometric features of the original time-series data [88]. It recognizes the fundamental structure of the data and can generate highly non-linear embedding applied LLE for the dimensionality reduction for the time-series data [72], [89], [90].

If a well-sampled vector A consists of N real values, each having a dimension of n, LLE starts the dimensionality reduction by constructing a nearest neighbor graph. It computes nearest neighbors (m) using the euclidean distance or any other local metrics-based formula. The number m should be chosen carefully, otherwise, LLE will fail to illustrate the global geometry. Then, it reconstructs each data point from its neighbors and calculates the reconstruction error using the following formula [72]:

$$\xi(W) = \sum_i \mid \vec{a_i} - \sum_j W_{ij}\vec{a_j} \mid^2 \tag{18}$$

here, a_j contributes to the optimal weight W_{ij} while reconstructing the data point a_i. This reconstruction error is minimized while fulfilling two requirements [72]:

- $\vec{a_i}$ is reconstructed using its neighboring data only which implies to $W_{ij} = 0$ if $\vec{a_i}$ and $\vec{a_j}$ are non-neighbor samples.
- the sum of the rows of the weight matrix W is equal to 1 i.e., $\sum_j W_{ij} = 1$.

Lastly, LLE uses the obtained weights to embed the points in the low dimensional subspace by computing a low dimensional vector Y having d dimension (where $d \leq n$) with the weight that is defined earlier and it is done by minimizing the cost function as follows [91]:

$$\phi(Y) = \sum_i \mid \vec{y_i} - \sum_j W_{ij}\vec{y_j} \mid^2 \tag{19}$$

The key feature of LLE is that it is capable of handling non-linear data where other dimension reduction algorithms may fail [49]. It also uses the neighborhood concept of the data points where choosing the neighborhood parameter m plays a vital role [92]. In addition to that, LLE can handle sparse matrices effectively which results in less computational time and space. However, this algorithm is sensitive to noise and fails to perform well on noisy data [92].

IX. ISOMETRIC MAPPING

Isometric Mapping (ISOMAP) is an unsupervised non-linear time-series data dimensionality reduction technique that is considered as one of the earliest manifold learning approaches [93]. A manifold learning approach can be defined as an approach to non-linear dimensionality reduction. ISOMAP tries to retrieve the intrinsic geometry of the data points by maintaining the geodesic distances among the data in a lower-dimensional space. In the low dimensional space, the nearby data samples stay close and far away data samples stay distant from each other [94]. This technique is broadly used by several researchers for time-series data [95]–[97].

ISOMAP first constructs a neighborhood graph G by identifying the neighbors of an input data (a_i). For identifying the neighboring points, it uses the ϵ-ball method or KNN method like LE [98]. Here, the edges among the neighboring points of the graph are weighted using their euclidean distances. Then, it estimates the geodesic distance by calculating the shortest path between all the data points of the neighborhood graph G, which can be performed by Dijkstra's shortest path or Floyd's algorithm [99]. A pairwise geodesic distance matrix d_G is formulated using the geodesic distance between each of the data points.

At the last step of ISOMAP, it embeds d_G to a lower dimension with the help of the classical Multi-Dimensional Scaling (MDS), which is a non-linear and unsupervised feature extraction algorithm [100]. MDS transforms this matrix d_G to a kernel matrix K_m using the following formula [101]:

$$K_m = H.d_G.H \tag{20}$$

here, H is a centering matrix which can be defined as [102]:

$$H = I - \frac{1}{N}.(ee^T) \tag{21}$$

where, N is the total number of data samples, I is an identity matrix and e is a column vector containing all 1s. After determining the kernel matrix K_m, eigen decomposition of K_m is estimated as $K_m \rightarrow M.D.M^T$, where D is a diagonal matrix that contains the eigenvalues and M is a matrix containing the respective eigenvectors [101]. Finally, top d eigenvectors are selected as the d dimensions.

Generally, ISOMAP performs really well for data that lies on a well-sampled manifold [103]. However, it performs poorly in case of scattered data distribution, especially for far away data samples. Moreover, it can be computationally expensive as it is an extension of the classical MDS algorithm [104].

X. MAXIMUM VARIANCE UNFOLDING

Maximum Variance Unfolding (MVU), previously referred to as semi-definite embedding (SDE), has been proven to be an effective non-linear unsupervised dimensionality reduction approach [105]. It is considered as a variant of KPCA that aims to overcome KPCA's shortcomings in dealing with non-linear high dimensional time series data [106]. Some studies have employed MVU for time-series data [107]–[109].

As mentioned in Section V, KPCA allows PCA to be performed in higher dimensional space specified by the kernel function (K). But there is no clarity on how the optimal kernel function K should be determined. The goal of MVU is to determine the optimal kernel that fully unfolds the manifold of data. Using semidefinite programming (SDP), MVU finds the optimal kernel for manifold unfolding [110]. Semi-definite programming (SDP) is an extension of linear programming in which a semidefinite condition on matrix variables replaces the non-negativity condition [111]. MVU is an iterative method and the iterative solution of SDP in MVU is complex and time intensive [110]. However, the trustworthiness and continuity assessment of MVU is promising for the dimensionality reduction of time-series data. The term 'trustworthiness and continuity' refers to how effectively the structure of a dataset is maintained after dimensionality reduction [27].

Similar to ISOMAP, MVU first constructs a neighborhood graph on the high dimensional data [112]. MVU starts with defining a neighborhood graph G in which each datapoint (a_i) is linked to its closest neighbor (a_j). Following that, MVU seeks the optimal kernel matrix by maximizing the sum of the squared Euclidean distances between all the data points while keeping the distances constant inside the neighborhood graph [113]. Selecting the optimal kernel matrix is a convex optimization issue that SDP can solve. To put it another way, MVU solves the following convex optimization issue while seeking for the optimal kernel matrix [114].

$$\text{Maximize} \sum_{\text{ij}} \|y_i - y_j\|^2 \text{ subject to}$$

$$\|y_i - y_j\|^2 = \|a_i - a_j\|^2 \ , \ \forall (i,j) \in G \tag{22}$$

here, a denotes the input data samples. On the other hand, y denotes output data samples. By establishing the kernel matrix $K_{m_{ij}} = y_i.y_j$, MVU converts the optimization issue to an SDP. The optimization problem is a linear program with a constraint that the kernel matrix K_m consists of non-negative eigenvalues. Over the kernel matrix K_m, the SDP can be represented as [110]:

$$\text{Maximize trace}\,(K) \text{ subject to}$$

$$K_{m_{ii}} - 2K_{m_{ij}} + K_{m_{jj}} = \|\vec{a}_i - \vec{a}_j\|^2 \text{ for all } (i,j) \tag{23}$$

$$\sum_{\text{ij}} K_{m_{\text{ij}}} = 0 \tag{24}$$

$$K_m \succcurlyeq 0 \tag{25}$$

The matrix K_m must be positive semidefinite to satisfy the last constraint $K_m \succcurlyeq 0$. The kernel matrix used as input for KPCA is the solution K_m of the SDP. The low-dimensional data format Y is obtained via Eigen Decomposition of the optimum kernel matrix K_m where $\vec{y}_i \in Y$ satisfying $K_{m_{ij}} = \overrightarrow{y_i} \cdot \vec{y}_j$. The top d eigenvalues and eigenvectors of K_m can be used to achieve an d-dimensional representation. In summary, the algorithm consists of three steps: evaluating the m-nearest neighbors, estimating the kernel matrix, and determining the top eigenvectors and values to determine the reduced dimensions [110].

MVU has the advantage of being able to adapt to specific tasks. For instance, the SDP's distance-preserving conditions can be eased to deal with outliers or to produce more confrontational dimensionality reduction results [112]. The key drawback of MVU is the time taken to solve large problems in semidefinite programming [110]. Despite this shortcoming, MVU has been used to effectively localize sensors and analyze DNA microarray data.

XI. LOCALITY PRESERVING PROJECTIONS

Locality Preserving Projection (LPP) is a linear, unsupervised dimensionality reduction algorithm that aims to preserve the neighborhood structure of data as much as possible. It has similar locality-preserving properties to LE since it is built on the same principle as LE. With the help of LPP, the time series data can be projected into a lower dimensional area.

LPP looks for a low-dimensional projection that keeps the local structure intact of the high-dimensional data. It first creates a neighborhood graph for each data point, and then using the concept of Laplacian (divergence of the gradient of a function) of the graphs, it trains a transformation matrix that maps the data to a lower subspace where the original local structure is maintained [28]. The manifold structure is explicitly considered by the LPP by generating an adjacency graph that reflects the intrinsic data structures. A nearest-neighbor graph is used to produce the manifold structure, which preserves local structure [115]. However, because of its linearity, LPP is fast and ideal for real-life applications and widely employed for the dimensionality reduction of time-series data [116]. Furthermore, it shares the same data representation qualities as non-linear techniques such as LLE and LE [28].

Though LPP is a linear technique, however, is a linear approximation of the nonlinear LE [117]. The LPP algorithm searches for a transformation matrix P to project the high-dimensional input data (A) into a low-dimensional subspace (Y) in a way that y_i represents a_i, where $y_i = P^T a_i$ and $y_j = P^T a_j$. For reducing the dimensionality from A to Y, the objective function is minimized by LPP as follows:

$$\phi(Y) = \sum_{\text{ij}} (y_i - y_j)^2 S_{ij} \tag{26}$$

where S denotes a sparse symmetric matrix and evaluates the local structure of a_i. The objective function $\phi(Y)$ with S_{ij} suffers a substantial penalty if neighboring points a_i and a_j are mapped far away. The following is a way to define S [28]:

$$S_{ij} = \begin{cases} 1, & \text{if } a_i \text{ and } a_j \text{ are neigbors or vice-versa} \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

where, t is heat kernel parameter, a special case of Gaussian kernel. For the generalized eigenvector issue, the eigenvectors and eigenvalues equation is:

$$ALA^T v = \lambda ADA^T v \tag{28}$$

where, v is a column vector, D is a diagonal matrix, $D_{\text{ii}} = \sum_j S_{\text{ji}}$. $L = D - S$ is the Laplacian matrix.

When the number of dimensions is greater than the number of data points and the data are linearly independent, computational analysis of LPP and LE yields the same conclusion [118]. Nonetheless, LPP is less computationally expensive and more suitable for real-world applications. However, it does not perform efficiently if the dataset is comparatively smaller [119].

XII. DIFFUSION MAPS

Diffusion Maps (DM) is an unsupervised, non-linear dimensionality reduction technique [120]. DM can also be categorized as a spectral approach. The spectral approach constructs an Eigen Decomposition of an entire matrix to extract the covariance across dimensions while keeping the distances between data points intact [29], [121]. For dimensionality reduction of time-series data, DM has been broadly applied [122]–[124].

The Markov random walk found on the graph of the high dimensional data serves as DM's mathematical base [29]. A simple analogy of a Markov random walk is that the random walk on the integer number line which begins at 0, travels forward to +1 or backward to -1 with equal probability for each step. An estimate of each data point's nearest neighbor is produced by running the random walk for a number of time steps which is further used to define the diffusion distance [120]. The diffusion distances that occur in pairs are conserved in the low-dimensional dataset. The diffusion distance's main notion is that it is calculated by aggregating all pathways through the graph derived from high dimensional data. As a result, the diffusion distance seems to be more adaptable to short-circuiting compared to the geodesic distance that has been used in ISOMAP [29].

DM starts with creating a neighborhood graph of the high-dimensional data. For determining the values of the graph's edges the Gaussian kernel function is employed, resulting in a connectivity matrix M with the following elements [125]:

$$M_{ij} = exp(-\frac{\|a_i - a_j\|^2}{\sigma^2}) \tag{29}$$

here, a_i and a_j are two input data samples, and the variance of the Gaussian is denoted by σ. After that, the matrix M is normalized so that the summation of each row becomes 1. Therefore, a Markov matrix $MM^{(1)}$ with the following elements is created [126]:

$$MM_{ij}^{(1)} = \frac{M_{ij}}{\sum_j M_{ij}} \tag{30}$$

DM is derived from dynamical systems theory that results in a Markov matrix $MM^{(1)}$ using which the forward transition probability of a dynamic process can be specified. As a result, the matrix $MM^{(1)}$ denotes the likelihood of a transformation between two data points in a single time step. For calculating the diffusion distance, the random walk forward probabilities $MM_{ij}^{(t)}$ is employed. The diffusion distance is defined below [127]:

$$D^{(t)}(a_i,\ a_j) = \sqrt{\sum_i \frac{(MM_i^{(t)} - MM_j^{(t)})^2}{\psi(a_i)^{(0)}}} \tag{31}$$

In the above equation, more weight is given to the high-density regions of the graph by $\psi(a_i)^{(0)}$. It's defined as $\psi(a_i)^{(0)} = \frac{q_i}{\sum_j q_j}$, where the degree of the node node a_i is denoted by q_i which is a given by $q_i = \sum_j H_{ij}$. Here, data point pairs having high forward transition probability have a minimal diffusion distance between them, as shown by Equation (31). DM strives to maintain the diffusion distances and represents the data Y at the same time in a low-dimensional structure. Utilizing spectral theory on the random walk, the top d nontrivial eigenvectors get multiplied by the corresponding eigenvalues and thus, the following matrix is created:

$$MM^{(t)}v = \lambda v \tag{32}$$

The first eigenvector in $MM^{(t)}v$ is constant. Therefore, eigenvector v_1 is ignored and the following top d eigenvectors define the low-dimensional representation Y [29]:

$$Y = \{\lambda_2 v_2,\ \lambda_3 v_3,\ ...,\ \lambda_{d+1} v_{d+1}\} \tag{33}$$

DM delivers better visualizations and more precise understanding for a smaller amount of data [30]. However, memory consumption becomes intense during training.

XIII. DISCRETE FOURIER TRANSFORM

Discrete Fourier Transform (DFT) is essential in many scientific applications, including time-series and waveform analysis [128] [129]. The core idea behind DFT is that any complex signal may be described by overlapping a small number of waveforms, each of which is designated by a single complex value called the Fourier coefficient [31]. The DFT is a variant of the continuous Fourier transform that allows functions to be sampled at discrete intervals in space or time. The ability to minimize dimensionality is the most basic benefit of representing a time-series in the frequency domain (the analytic space in which mathematical functions or signals are communicated in terms of frequency rather than time is known as the frequency domain). A finite sequence of values is provided to observe in discrete time in time series analysis. Using the DFT, a time domain sequence will be mapped into a frequency domain sequence.

A signal with a length of n can be decomposed into n number of wave forms and then merged to reconstruct the initial signal. However, certain Fourier coefficients are quite short in amplitude and hence add almost nothing to the reconstructed signal. As a result, dimensionality reduction can be achieved by discarding these short-amplitude coefficients with little compromise of information.

A time domain sequence will be mapped into a frequency domain sequence using the DFT. The DFT of time series signal a of length n is computed to reduce its dimensionality to a reduced feature space of size N. A is the frequency domain form of signal a. Let $a = [a_n],\ n = 0,\ ...\ ,\ N-1$ be a vector representation of a signal and a defined to be a sequence A of N complex numbers A_k. The DFT can be calculated by the formula [128] [129]:

$$A_k = 1/\sqrt{N} \sum_{n=0}^{N-1} a_n exp(-i2\pi kn/N), [k = 0, 1, ..., N-1] \tag{34}$$

where i =$i = \sqrt{-1}$, is the imaginary unit. A_k is a complex number.

SVD, another dimensionality reduction approach based on spectral decomposition analyzes the entire data and rotates the axes to optimize variance along the first few dimensions. DFT, on the other hand, processes each data point independently [130]. As a result, DFT becomes a data-independent

transformation, which is critical if the dataset changes over time. One of the really crucial features of the DFT is that it may be used efficiently by using $O(NlogN)$ complexity instead of the $O(N^2)$ complexity [31]. DFT has some limitations also in terms of time series data. White noise is the worst-case indication for DFT, as it fails to compress information into the first few coefficients, resulting in a large number of error rates.

XIV. DISCRETE WAVELET TRANSFORM

Discrete Wavelet Transform (DWT) is a linear dimension reduction technique that holds many conducive properties that are useful in the context of analyzing the time-series data [131]. DWT can represent the time-series data in both frequency and time domain [132]. The basic concept behind the algorithm of DWT is to convert the time-series data into several coefficients, and these coefficients illustrate the information of how the data varies over specific time scales [133]. DWT also reduces the noise from the input data along with decreasing the input data size [134]. DWT has also been widely used for the dimensionality reduction of time-series data [135]–[137].

DWT uses a finite set of wavelets for the transformation where the wavelets are discretely sampled. A Wavelet means a rapidly decaying small wave-like oscillation that has zero mean. If a wavelet is represented as ω, then it can be written as follows [131]:

$$\int_{-\infty}^{\infty} \omega(t)dt = 0 \tag{35}$$

It is localized in time and is considered as the time scale for time-series data [131]. This wavelet must have finite energy which can be represented as [134],

$$\int_{-\infty}^{\infty} |\omega(t)|^2 dt < \infty \tag{36}$$

These wavelets are the mathematical functions that decompose the input time-series data into several components. DWT separates these components into different frequencies at different scales. After the decomposition of the data into frequencies, it multiplies a signal by a mathematical function or wavelet which decomposes the signal into multiple lower-resolution levels. It is known as multi-resolution analysis as this transformation can be done in multiple resolution levels for different frequencies [138]. However, a continuous wavelet transform (CWT) can be defined as below [139]:

$$CWT(f) = \frac{1}{\sqrt{\delta}} \int f(t) \,.\, \omega^* \left(\frac{t - \tau}{\delta} \right) . \, dt \tag{37}$$

Here, $CWT(f)$ is the continuous wavelet transform of the signal $f(t)$. $f(t)$ is a function of time t where τ is the timing parameter providing the location information of the signal. δ is the scaling parameter and ω is the basis function which can also be called the mother wavelet where $*$ stands for the complex conjugate.

DWT reduces the dimension of the original time-series data by keeping only a few of the wavelet coefficients which contain the most energy. The rest of the wavelets containing less energy are dropped. The energy of a wavelet can be obtained by calculating some statistical values from the wavelet coefficients, such as the sum of the square of the absolute values of the coefficients [140]. This energy regarding information can also be acquired by plotting the coefficient distribution values [131]. DWT is also known as a time-scale transformation technique as the decomposed signal component of DWT still stays in the time domain, and retains the original information of the time-series data in the form of energy within the transformed data [131], [141].

XV. COMPARISON SUMMARY OF THE DIMENSIONALITY REDUCTION METHODS FOR TIME-SERIES DATA

In Table 2, we present the key concepts of the dimensionality reduction algorithms discussed in this paper. Here, U stands for unsupervised, S stands for supervised, L stands for linear, NL stands for nonlinear, *NN* stands for the nearest neighbor, i stands for the number of iterations, w stands for the number of weights, n denotes the number of features in original high dimension, N is the number of samples or data points, and p stands for the ratio of the non-zero elements in a matrix.

For instance, Autoencoder is an unsupervised learning technique that can handle nonlinear data using nonlinear activation functions and it may generate lossy output. It has several hyper-parameters that need to be tuned for optimal performance, such as code layer size, loss function, number of layers, and number of nodes per layer and its goal is to compress and encode data.

PCA is an unsupervised, linear method that has the drawbacks of losing the meaning of the features after forming the linear combinations of the features and its goal is to maximize the variance. On the other hand, Kernel PCA is a nonlinear unsupervised learning technique that uses kernel tricks to extract principal components from the data. PCA works best for linear data, while Kernel PCA can handle nonlinear data by mapping the data into a higher-dimensional space and then applying PCA in that space. This non-linear mapping is determined by the choice of the kernel function, which can be Gaussian, polynomial, or sigmoid. PCA has only one hyper-parameter, which is the number of principal components to retain. This parameter determines the amount of variance retained in the reduced dimensionality data. In contrast, Kernel PCA has two hyper-parameters: the choice of kernel function and the kernel bandwidth or scale parameter. The choice of kernel function determines the type of non-linear mapping, while the kernel bandwidth controls the smoothness of the mapping.

LE is an unsupervised learning technique that learns to embed the input data into a low-dimensional space based on the underlying geometric structure of the data. LE can handle nonlinear data by using different distance metrics and kernel functions to construct the graph. The choice of distance

metric and kernel function determines the nonlinear relationships between the data points, which can be captured by the Laplacian eigenmaps. LE has several hyper-parameters that need to be tuned for optimal performance, such as the choice of a distance metric, kernel function, number of nearest neighbors, and regularization parameter. The choice of these hyper-parameters affects the graph construction and the low-dimensional embedding.

SVD is a linear technique used for both unsupervised and supervised learning tasks. SVD is a linear technique and works best for linear data. It may not be able to capture nonlinear relationships between the data points. This can be computationally expensive for large datasets. SVD has one hyper-parameter, which is the number of singular values and vectors to retain. This parameter determines the amount of variance retained in the reduced dimensionality data. A higher number of singular values and vectors result in a higher dimensional embedding, while a lower number results in a lower dimensional embedding.

LLE is an unsupervised learning technique that works by learning a low-dimensional representation of the input data based on the underlying manifold structure of the data. LLE is a nonlinear technique that can handle nonlinear relationships between data points. LLE can be computationally expensive for large datasets. LLE has several hyper-parameters that need to be tuned for optimal performance, such as the number of neighbors, the weight function, and the regularization parameter. The choice of these hyper-parameters affects the accuracy of the embedding and the runtime of the algorithm.

ISOMAP is an unsupervised learning technique and it works by preserving the geodesic distances between the data points in the low-dimensional space. It does this by modeling the data as points on a manifold and using the distances between the points on the manifold to construct the low-dimensional embedding. ISOMAP is a nonlinear technique that can handle nonlinear relationships between data points. It models the data as points on a manifold, which captures the underlying nonlinear structure of the data. The distances between the points on the manifold are used to construct the low-dimensional embedding, which preserves the nonlinear relationships in the data. ISOMAP has several hyper-parameters that need to be tuned for optimal performance, such as the number of neighbors, the dimension of the manifold, and the metric used to compute distances. The choice of these hyper-parameters affects the accuracy of the embedding and the runtime of the algorithm.

MVU and Diffusion Maps are unsupervised learning techniques and are nonlinear techniques that can capture complex nonlinear relationships between the data points. They use manifold learning to uncover the underlying structure of the data and represent it in a lower-dimensional space. The computational complexity of MVU and Diffusion Maps can be high, especially for large datasets. These methods involve constructing a graph or affinity matrix based on the similarity between the data points and performing eigenvalue decomposition or other optimization techniques to find the low-dimensional representation of the data. These have several hyper-parameters that need to be tuned for optimal performance, such as the number of neighbors used to construct the affinity matrix, the bandwidth used to measure similarity, and the number of eigenvectors used to represent the data in the low-dimensional space. The choice of these hyper-parameters affects the accuracy of the method and the level of noise reduction achieved.

LPP is an unsupervised learning technique and a linear technique in which the neighborhood graph structure of data is preserved. The computational complexity of LPP can be high, especially for large datasets. LPP involves constructing a graph or affinity matrix based on the similarity between the data points and performing eigenvalue decomposition or other optimization techniques to find the low-dimensional representation of the data. LPP has several hyper-parameters that need to be tuned for optimal performance, such as the number of neighbors used to construct the affinity matrix, the regularization parameter used to avoid overfitting, and the number of eigenvectors used to represent the data in the low-dimensional space. The choice of these hyper-parameters affects the accuracy of the method and the level of noise reduction achieved.

DFT and DWT are unsupervised learning techniques that transform the data from the time domain to the frequency domain based on the assumption that the data can be represented as a sum of sinusoidal functions. DFT and DWT are linear techniques that assume a linear relationship between the data points. They transform the data into the frequency domain, which allows for the analysis of periodic patterns in the data. However, they may not be able to capture nonlinear relationships between the data points. DFT and DWT have several hyper-parameters that need to be tuned for optimal performance, such as the window size, the sampling frequency, and the type of wavelet used (in the case of DWT). The choice of these hyper-parameters affects the accuracy of the transform and the level of noise reduction achieved. For example, the window size determines the size of the time window used to analyze the data, while the sampling frequency determines the resolution of the frequency domain.

XVI. CONCLUSIONS

To achieve more accurate results in machine learning using time-series datasets, it is common to include as many features as possible initially to detect important attributes. However, the model's performance can suffer when irrelevant features are added as the number of features grows. To address this issue, it is necessary to reduce the number of features while maintaining the model's performance. This paper presents a survey of twelve dimensionality reduction algorithms for time-series data, comparing their performance in terms of supervision, linearity, time and memory complexity, hyper-parameters, and drawbacks.

TABLE 2: Analysis of dimensionality reduction algorithms discussed in this research work.

Algorithm Name	Super-vision	Line-arity	Time Complexity	Memory Complexity	Hyper-parameters	Goals	Disadvantages
Auto-encoder	*U*	*NL*	*O(iNw)*	*O(w)*	code layer size, loss function, no. of layers, no. of nodes per layer	to compress and encode data	generates lossy output
PCA	*U*	*L*	$O(n^2N + N^3)$	$O(N^2)$	no. of components	maximizing the variance	features are meaningless after forming linear combinations
KPCA	*U*	*NL*	$O(N^3)$	$O(N^2)$	no. of components, kernel	linear separation of data	requires high training time
LE	*U*	*NL*	$O(nlog(m).$ $Nlog(N))$ $+O(nNm^3)$ $+O(dN^2)$	$O(pN^2)$	no. of components, NN	preservation of local geometrical properties	may generate disconnected neighborhood graph
SVD	*U*	*L*	$O(n^2N + N^3)$	$O(N^2)$	no. of components, number of iterations	to minimize the reconstruction error	computation-ally expensive
LLE	*U*	*NL*	$O(nlog(m).$ $Nlog(N))$ $+O(nNm^3)$ $+O(dN^2)$	$O(pN^2)$	no. of components, no. of iterations, NN	preserving local distances	sensitive to noise
ISOMAP	*U*	*NL*	$O(N^3)$	$O(N^2)$	no. of components, NN	to maintain pairwise geodesic distances	topologically unstable
MVU	*U*	*NL*	$O((nk)^3)$	$O(nk)^3$	no. of components, NN, kernel matrix	maximizing the sum of distances between all transformed data points	high computational complexity

Table 2 continued from previous page

Algorithm Name	Super-vision	Line-arity	Time Complexity	Memory Complexity	Hyper-parameters	Goals	Disadvantages
LPP	*U*	*L*	$O((n+k)m^2$ + $(n+d)n^2)$	$O((n+k)m^2$ + $(n+d)n^2)$	NN, weight of the edges	optimal preservation of the neighborhood structure of the data	sensitive to noise and outliers
Diffusion Map	*U*	*NL*	$O(n^3)$	$O(n^2)$	no. of components	preserving intrinsic geometry of the data	high computational complexity
DFT	*S*	*L*	*O(nlogn)*	*O(nlogn)*	no. of components	mapping the time domain sequence to the frequency domain/ preserving the euclidean distance between the data points	loss of time varying information
DWT	*S*	*L*	$O(N)$	$O(N)$	wavelets, filters containing cutoff frequencies	breaking down the time series data into meaningful signal components	greater complexity

REFERENCES

[1] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *Computational Intelligence and Bioinspired Systems: 8th International Work-Conference on Artificial Neural Networks, IWANN 2005, Vilanova i la Geltrú, Barcelona, Spain, June 8-10, 2005. Proceedings 8*. Springer, 2005, pp. 758–770.

[2] M. Li, H. Wang, L. Yang, Y. Liang, Z. Shang, and H. Wan, "Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction," *Expert Systems with Applications*, vol. 150, pp. 1–27, 2020.

[3] M. A. Bessa, R. Bostanabad, Z. Liu, A. Hu, D. W. Apley, C. Brinson, W. Chen, and W. K. Liu, "A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality," *Computer Methods in Applied Mechanics and Engineering*, vol. 320, pp. 633–667, 2017.

[4] T.-c. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.

[5] E. Lughofer and R. Nikzad-Langerodi, "Robust generalized fuzzy systems training from high-dimensional time-series data using local structure preserving pls," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 11, pp. 2930–2943, 2019.

[6] E. Keogh, *Indexing and Mining Time Series Data*. Boston, MA: Springer US, 2008, pp. 493–497. [Online]. Available: https://doi.org/10.1007/978-0-387-35973-1_598

[7] N. Sharma and K. Saroha, "Study of dimension reduction methodologies in data mining," in *International Conference on Computing, Communication & Automation*. IEEE, 2015, pp. 133–137.

[8] C. Cassisi, P. Montalto, M. Aliotta, A. Cannata, and A. Pulvirenti, "Similarity measures and dimensionality reduction techniques for time series data mining," *Advances in data mining knowledge discovery and applications'(InTech, Rijeka, Croatia, 2012,*, pp. 71–96, 2012.

[9] J. Dan, W. Shi, F. Dong, and K. Hirota, "Piecewise trend approximation: A ratio-based time series representation," *Abstract and Applied Analysis*, vol. 2013, pp. 1–7, 2013.

[10] Q. Wang and V. Megalooikonomou, "A dimensionality reduction technique for efficient time series similarity analysis," *Inf. Syst.*, vol. 33, no. 1, p. 115–132, Mar. 2008. [Online]. Available: https://doi.org/10.1016/j.is.2007.07.002

[11] I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, pp. 429–440, 2003.

[12] V. S. S. Fotso, E. M. Nguifo, and P. Vaslin, "Grasp heuristic for time series compression with piecewise aggregate approximation," *RAIRO-Operations Research*, vol. 53, no. 1, pp. 243–259, 2019.

[13] C. Becker and R. Fried, "Sliced inverse regression for high-dimensional time series," in *Exploratory Data Analysis in Empirical Research*, M. Schwaiger and O. Opitz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 3–11.

[14] W. W. Wei, *Factor analysis of multivariate time series*. John Wiley & Sons, Ltd, 2019, ch. 5, pp. 163–202. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119502951.ch5

[15] F. Aires, A. Chédin, and J.-P. Nadal, "Independent component analysis of multivariate time series: Application to the tropical sst variability," *Journal of Geophysical Research: Atmospheres*, vol. 105, no. D13, pp. 17 437–17 455, 2000. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2000JD900152

[16] N. Tavakoli, S. Siami-Namini, M. A. Khanghah, F. M. Soltani, and A. S. Namin, "Clustering time series data through autoencoder-based deep learning models," *arXiv preprint arXiv:2004.07296*, pp. 1–26, 2020.

[17] H. Li, "Asynchronism-based principal component analysis for time series data mining," *Expert Systems with Applications*, vol. 41, p. 2842–2850, 05 2014.

[18] D. Cao, Y. Tian, and D. Bai, "Time series clustering method based on principal component analysis," in *Proceedings 5th International Conference on Information Engineering for Mechanics and Materials (ICIMM 2015)*. Citeseer, 2015, pp. 888–895.

[19] M. Fauvel, J. Chanussot, and J. A. Benediktsson, "Kernel principal component analysis for feature reduction in hyperspectrale images analysis," in *Proceedings of the 7th Nordic Signal Processing Symposium - NORSIG 2006*, 2006, pp. 238–241.

[20] D. Wu, W. Su, and M. Carpuat, "A kernel pca method for superior word sense disambiguation," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, 2004, pp. 637–644.

[21] N. Pospelov, A. Tetereva, O. Martynova, and K. Anokhin, "The laplacian eigenmaps dimensionality reduction of fmri data for discovering stimulus-induced changes in the resting-state brain activity," *Neuroimage: Reports*, vol. 1, no. 3, pp. 1–13, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666956021000337

[22] A. Hayashi, Y. Mizuhara, and N. Suematsu, "Embedding time series data for classification," in *Machine Learning and Data Mining in Pattern Recognition*, P. Perner and A. Imiya, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 356–365.

[23] J. Lee, C. Archambeau, and M. Verleysen, "Locally linear embedding versus isotop," in *ESANN*, 2003, pp. 527–534.

[24] F. Liu, W. Zhang, and S. Gu, "Local linear laplacian eigenmaps," *Pattern Recogn. Lett.*, vol. 75, no. C, p. 30–35, May 2016. [Online]. Available: https://doi.org/10.1016/j.patrec.2016.03.003

[25] A. Khoshrou, A. B. Dorsman, and E. J. Pauwels, "Svd-based visualisation and approximation for time series data in smart energy systems," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 2017, pp. 1–6.

[26] I. Bowala and M. Fernando, "A novel model for Time-Series Data Clustering Based on piecewise SVD and BIRCH for Stock Data Analysis on Hadoop Platform," *Advances in Science, Technology and Engineering Systems Journal*, vol. 2, no. 3, pp. 855–864, 2017. [Online]. Available: http://astesj.com/v02/i03/p106/

[27] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski, "Information retrieval perspective to nonlinear dimensionality reduction for data visualization." *Journal of Machine Learning Research*, vol. 11, no. 2, pp. 451–490, 2010.

[28] X. He and P. Niyogi, "Locality preserving projections," in *Proceedings of the 16th International Conference on Neural Information Processing Systems*, ser. NIPS'03. Cambridge, MA, USA: MIT Press, 2003, p. 153–160.

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

[30] A. Juvonen, T. Sipola, and T. Hämäläinen, "Online anomaly detection using dimensionality reduction techniques for http log analysis," *Computer Networks*, vol. 91, pp. 46–56, 2015.

[31] H. Shatkay, "The fourier transform-a primer," *Brown University*, pp. 1–20, 1995.

[32] M. Sifuzzaman, M. Islam, and M. Ali, "Application of wavelet transform and its advantages compared to fourier transform," *J. Phys. Sci*, vol. 13, pp. 121–134, 01 2009.

[33] P. Baldi, "Autoencoders, unsupervised learning and deep architectures," in *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop*, ser. UTLW'11, vol. 27. JMLR.org, 2011, p. 37–49.

[34] N. Tavakoli, S. Siami-Namini, M. A. Khanghah, F. M. Soltani, and A. S. Namin, "Clustering time series data through autoencoder-based deep learning models," *CoRR*, vol. abs/2004.07296, 2020. [Online]. Available: https://arxiv.org/abs/2004.07296

[35] F. M. Bianchi, L. Livi, K. Ø. Mikalsen, M. Kampffmeyer, and R. Jenssen, "Learning representations for multivariate time series with missing data using temporal kernelized autoencoders," *CoRR*, vol. abs/1805.03473, pp. 1–16, 2018. [Online]. Available: http://arxiv.org/abs/1805.03473

[36] G. Richard, B. Grossin, G. Germaine, G. Hébrail, and A. de Moliner, "Autoencoder-based time series clustering with energy applications," *arXiv preprint arXiv:2002.03624*, pp. 1–6, 2020.

[37] H.-C. Shin, M. Orton, D. J. Collins, S. Doran, and M. O. Leach, "Autoencoder in time-series analysis for unsupervised tissues characterisation in a large unlabelled medical image dataset," in *2011 10th International Conference on Machine Learning and Applications and Workshops*, vol. 1, 2011, pp. 259–264.

[38] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 2725–2732. [Online]. Available: https://doi.org/10.24963/ijcai.2019/378

[39] G. Liu, H. Bao, and B. Han, "A stacked autoencoder-based deep neural network for achieving gearbox fault diagnosis," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–10, Jul. 2018. [Online]. Available: https://doi.org/10.1155/2018/5105709

[40] J. Jordan, "Introduction to autoencoders." https://www.jeremyjordan.me/autoencoders/, 2021, last accessed 21 December 2021.

[41] Sayantini, "Autoencoders tutorial," https://www.edureka.co/blog/autoencoders-tutorial/, 2021, last accessed 21 December 2021.

[42] N. Verbeeck, R. M. Caprioli, and R. Van de Plas, "Unsupervised machine learning for exploratory data analysis in imaging mass spectrometry," *Mass Spectrometry Reviews*, vol. 39, no. 3, pp. 245–291, 2020.

[43] L. J. Al-Omairi, J. Abawajy, M. U. Chowdhury, and T. Al-Quraishi, "An empirical analysis of graph-based linear dimensionality reduction techniques," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 5, pp. 1–13, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5990

[44] T. Howley, M. G. Madden, M.-L. O'Connell, and A. G. Ryder, "The effect of principal component analysis on machine learning accuracy with high dimensional spectral data," in *Applications and Innovations in Intelligent Systems XIII*. Springer London, pp. 209–222. [Online]. Available: https://doi.org/10.1007/1-84628-224-1_16

[45] F. Alshammri and J. Pan, "Moving dynamic principal component analysis for non-stationary multivariate time series," *Computational Statistics*, vol. 36, no. 3, pp. 1–41, Mar. 2021. [Online]. Available: https://doi.org/10.1007/s00180-021-01081-8

[46] P. Tanisaro and G. Heidemann, "Dimensionality reduction for visualization of time series and trajectories," in *Image Analysis*. Springer International Publishing, 2019, pp. 246–257. [Online]. Available: https://doi.org/10.1007/978-3-030-20205-7_21

[47] A. Widodo and I. Budi, "Model selection using dimensionality reduction of time series characteristics," in *International Symposium on Forecasting, Seoul, South Korea*, 2013, pp. 57–118.

[48] G. Gawde and J. Pawar, "Shape based time series reduction using pca," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2017, pp. 1–4.

[49] F. Anowar, S. Sadaoui, and B. Selim, "Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne)," *Computer Science Review*, vol. 40, pp. 1–13, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013721000186

[50] N. Sharma, "Principal component analysis," https://heartbeat.fritz.ai/understanding-the-mathematics-behind-principal-component-analysis-efd7c9ff0bb3, 2021, last accessed 21 December 2021.

[51] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, pp. 1–12, 2014.

[52] B. Ghojogh, M. N. Samad, S. A. Mashhadi, T. Kapoor, W. Ali, F. Karray, and M. Crowley, "Feature selection and feature extraction in pattern analysis: A literature review," *ArXiv*, vol. abs/1905.02845, pp. 1–14, 05 2019.

[53] L. Cao, K. Chua, W. Chong, H. Lee, and Q. Gu, "A comparison of pca, kpca and ica for dimensionality reduction in support vector machine," *Neurocomputing*, vol. 55, no. 1, pp. 321–336, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231203004338

[54] J. Mita, C. Babu, and G. Manivannan, "Performance analysis of dimensionality reduction using pca, kpca and lle for ecg signals," *IOP Conference Series: Materials Science and Engineering*, vol. 1084, pp. 1–8, 03 2021.

[55] G. Sidhu, N. Asgarian, R. Greiner, and M. Brown, "Kernel principal component analysis for dimensionality reduction in fmri-based diagnosis of adhd," *Frontiers in Systems Neuroscience*, vol. 6, pp. 1–16, 2012. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnsys.2012.00074

[56] K. Yang and C. Shahabi, "A pca-based kernel for kernel pca on multivariate time series," in *IEEE Intern. Conf. on Data Mining*, 2005, pp. 1–8.

[57] H. Hoffmann, "Kernel pca for novelty detection," *Pattern Recognition*, vol. 40, pp. 863–874, 03 2007.

[58] G. Baudat and F. Anouar, "Kernel-based methods and function approximation," *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, vol. 2, pp. 1244–1249 vol.2, 2001.

[59] Y. Zhang, "Enhanced statistical analysis of nonlinear processes using kpca, kica and svm," *Chemical Engineering Science*, vol. 64, no. 5, pp. 801–811, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0009250908005472

[60] Q. Wang, "Kernel principal component analysis and its applications in face recognition and active shape models," *CoRR*, vol. abs/1207.3538, pp. 1–9, 2012. [Online]. Available: http://arxiv.org/abs/1207.3538

[61] F. Zhang, S. Zong, and Z. Ling, "Fault diagnosis using kernel principal component analysis for hot strip mill," *The Journal of Engineering*, vol. 1, pp. 527–535, 02 2009.

[62] H. Li, L. Xiong, L. Ohno-Machado, and X. Jiang, "Privacy preserving RBF kernel support vector machine," *BioMed Research International*, vol. 2014, pp. 1–10, 2014. [Online]. Available: https://doi.org/10.1155/2014/827371

[63] H. M. Asraf, M. Nooritawati, and M. S. Rizam, "A comparative study in kernel-based support vector machine of oil palm leaves nutrient disease," *Procedia Engineering*, vol. 41, pp. 1353–1359, 2012. [Online]. Available: https://doi.org/10.1016/j.proeng.2012.07.321

[64] M. Tian and W. Wang, "Some sets of orthogonal polynomial kernel functions," *Applied Soft Computing*, vol. 61, pp. 742–756, Dec. 2017. [Online]. Available: https://doi.org/10.1016/j.asoc.2017.08.010

[65] Q. Liu, Y. Zhang, and Z. Hu, "Extracting decision rules from sigmoid kernel," in *Advanced Data Mining and Applications*. Springer Berlin Heidelberg, pp. 294–304. [Online]. Available: https://doi.org/10.1007/978-3-540-88192-6_28

[66] F. Zhao, I. Rekik, S.-W. Lee, J. Liu, J. Zhang, and D. Shen, "Two-phase incremental kernel pca for learning massive or online datasets," *Complexity*, vol. 2019, pp. 1–17, 2019.

[67] C. Örnek and E. Vural, "Nonlinear supervised dimensionality reduction via smooth regular embeddings," *Pattern Recognition*, vol. 87, pp. 55–66, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320318303522

[68] M. Lewandowski, J. M. del Rincon, D. Makris, and J.-C. Nebel, "Temporal extension of laplacian eigenmaps for unsupervised dimensionality reduction of time series," in *2010 20th International Conference on Pattern Recognition*. IEEE, Aug. 2010, pp. 161–164. [Online]. Available: https://doi.org/10.1109/icpr.2010.48

[69] A. Gramfort and M. Clerc, "Low dimensional representations of MEG/EEG data using laplacian eigenmaps," in *2007 Joint Meeting of the 6th International Symposium on Noninvasive Functional Source Imaging of the Brain and Heart and the International Conference on Functional Biomedical Imaging*. IEEE, oct 2007, pp. 169–172. [Online]. Available: https://doi.org/10.1109/nfsi-icfbi.2007.4387717

[70] J. X. Leon-Medina, M. Anaya, F. Pozo, and D. Tibaduiza, "Nonlinear feature extraction through manifold learning in an electronic tongue classification task," *Sensors*, vol. 20, no. 17, pp. 1–20, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/17/4834

[71] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, p. 1373–1396, Jun. 2003. [Online]. Available: https://doi.org/10.1162/089976603321780317

[72] L. Saul and S. Roweis, "An introduction to locally linear embedding," *Journal of Machine Learning Research*, vol. 7, p. 1–13, 01 2001.

[73] B. Shaw and T. Jebara, "Structure preserving embedding," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 937–944.

[74] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, ser. NIPS'01. Cambridge, MA, USA: MIT Press, 2001, p. 585–591.

[75] B. Li, Y.-R. Li, and X.-L. Zhang, "A survey on laplacian eigenmaps based manifold learning methods," *Neurocomputing*, vol. 335, pp. 336–351, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218312645

[76] L. Torres, K. S. Chan, and T. Eliassi-Rad, "Glee: Geometric laplacian eigenmap embedding," *Journal of Complex Networks*, vol. 8, no. 2, pp. 1–10, Mar 2020. [Online]. Available: http://dx.doi.org/10.1093/comnet/cnaa007

[77] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *CoRR*, vol. abs/1201.0490, pp. 2825–2830, 2012. [Online]. Available: http://arxiv.org/abs/1201.0490

[78] A. Fejjari, K. S. Ettabaa, and O. Korbaa, "Modified graph-based algorithm for efficient hyperspectral feature extraction," in *International Symposium on Computer and Information Sciences*. Springer, 2018, pp. 87–95.

[79] F. Fallucchi and F. M. Zanzotto, "Singular value decomposition for feature selection in taxonomy learning," in *Proceedings of the International Conference RANLP-2009*. Borovets, Bulgaria: Association for Computational Linguistics, Sep. 2009, pp. 82–87. [Online]. Available: https://www.aclweb.org/anthology/R09-1016

[80] W. Lin, J. Z. Huang, and T. McElroy, "Time series seasonal adjustment using regularized singular value decomposition," *Journal of Business & Economic Statistics*, vol. 38, no. 3, pp. 487–501, feb 2019. [Online]. Available: https://doi.org/10.1080/07350015.2018.1515081

[81] H. Li, C. Lin, X. Wan, and Z. xin Li, "Feature representation and similarity measure based on covariance sequence for multivariate time series," *IEEE Access*, vol. 7, pp. 67 018–67 026, 2019.

[82] V. Choqueuse, P. Granjon, A. Belouchrani, F. Auger, and M. Benbouzid, "Monitoring of three-phase signals based on singular-value decomposition," *IEEE Transactions on Smart Grid*, vol. PP, pp. 6156–6166, 02 2019.

[83] K. Pąk, "Basic properties of the rank of matrices over a field," *Formalized Mathematics*, vol. 15, pp. 199–211, 01 2007.

[84] L. J. Sciacca and R. J. Evans, "Multidimensional inverse problems in ultrasonic imaging," in *Multidimensional Systems: Signal Processing and Modeling Techniques*, ser. Control and Dynamic Systems, C. Leondes, Ed. Academic Press, 1995, vol. 69, pp. 1–48. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0090526705800374

[85] S. Brunton and J. Kutz, *Singular Value Decomposition (SVD)*. Cambridge University Press, 02 2019, pp. 3–46.

[86] M. Frank and J. M. Buhmann, "Selecting the rank of truncated SVD by maximum approximation capacity," in *2011 IEEE International Symposium on Information Theory Proceedings*. IEEE, Jul. 2011, pp. 1036–1040. [Online]. Available: https://doi.org/10.1109/isit.2011.6033687

[87] A. Dutta, J. Liang, and X. Li, "A fast and adaptive svd-free algorithm for general weighted low-rank recovery," *arXiv preprint arXiv:2101.00749*, pp. 1–27, 2021.

[88] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000. [Online]. Available: http://www.sciencemag.org/cgi/content/abstract/290/5500/2323

[89] A. Jog, A. J. Joshi, S. Chandran, and A. Madabhushi, "Classifying ayurvedic pulse signals via consensus locally linear embedding." in *Biosignals*, 2009, pp. 388–395.

[90] B. Yao, J. Su, L. Wu, and Y. Guan, "Modified local linear embedding algorithm for rolling element bearing fault diagnosis," *Applied Sciences*, vol. 7, pp. 1–16, 2017.

[91] L. Saul and S. Roweis, "Think globally, fit locally: Unsupervised learning of nonlinear manifolds," *J. Mach. Learn. Res.*, vol. 4, pp. 119–155, 08 2002.

[92] J. Chen and Y. Liu, "Locally linear embedding: a survey," *Artif. Intell. Rev.*, vol. 36, no. 1, pp. 29–48, 2011. [Online]. Available: https://doi.org/10.1007/s10462-010-9200-z

[93] Y. Huang and G. Kou, "A kernel entropy manifold learning approach for financial data analysis," *Decision Support Systems*, vol. 64, pp. 31–42, 2014.

[94] X. Geng, D.-C. Zhan, and Z.-H. Zhou, "Supervised nonlinear dimensionality reduction for visualization and classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 6, pp. 1098–1107, 2005.

[95] C. Orsenigo and C. Vercellis, "Dimensionality reduction via isomap with lock-step and elastic measures for time series gene expression classification," in *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, L. Vanneschi, W. S. Bush, and M. Giacobini, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 92–103.

[96] G. Tong and S. Li, "Construction and application research of isomap-RVM credit assessment model," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–7, 2015. [Online]. Available: https://doi.org/10.1155/2015/197258

[97] Y. Zhai, N. Wang, L. Zhang, L. Hao, and C. Hao, "Automatic crop classification in northeastern china by improved nonlinear dimensionality reduction for satellite image time series," *Remote Sensing*, vol. 12, no. 17, pp. 1–22, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/17/2726

[98] M. Elhenawy, M. Masoud, S. Glaser, and A. Rakotonirainy, "Topological stability: a new algorithm for selecting the nearest neighbors in nonlinear dimensionality reduction techniques," *CoRR*, vol. abs/1911.05312, pp. 1–9, 2019. [Online]. Available: http://arxiv.org/abs/1911.05312

[99] J. Tenenbaum, V. Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 01 2000.

[100] M. Yousaf, T. Rehman, and J. Li, "An extended isomap approach for nonlinear dimension reduction," *SN Computer Science*, vol. 1, pp. 1–10, 05 2020.

[101] S. France and J. Carroll, "Two-way multidimensional scaling: A review," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, pp. 644 – 661, 10 2011.

[102] C. K. I. Williams, "On a connection between kernel pca and metric multidimensional scaling," *Mach. Learn.*, vol. 46, no. 1, p. 11–19, 2002.

[103] C. Shao and H. Huang, "Improvement of data visualization based on isomap," in *MICAI*, 11 2005, pp. 534–543.

[104] S. Mahapatra and V. Chandola, "S-isomap++: Multi manifold learning from streaming data," *2017 IEEE International Conference on Big Data (Big Data)*, pp. 716–725, Dec 2017. [Online]. Available: http://dx.doi.org/10.1109/BigData.2017.8257987

[105] C. Wei, J. Chen, and Z. Song, "Developments of two supervised maximum variance unfolding algorithms for process classification," *Chemometrics and Intelligent Laboratory Systems*, vol. 159, pp. 31–44, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743916303409

[106] J. Wang, W. Ge, J. Zhou, H. Wu, and Q. Jin, "Fault isolation based on residual evaluation and contribution analysis," *Journal of the Franklin Institute*, vol. 354, no. 6, pp. 2591–2612, 2017.

[107] H. Lange, M. Hauhs, J. Schilcher, and G. Lischeid, "Investigating the lange bramke runoff dynamics: a nonlinear perspective," in *International Workshop on Status and Perspectives of Hydrology in Small Basins*, 2009, pp. 127–130.

[108] C. Bogner, "Characteristic patterns and processes from non-linear analysis of soil water and streamwater time series in the forested catchment lange bramke, harz mountains, germany," in *Geophysical Research Abstracts*, vol. 13, 2011, p. 10275.

[109] C. H. Lima, U. Lall, T. Jebara, and A. G. Barnston, "Statistical prediction of enso from subsurface sea temperature using a nonlinear dimensionality reduction," *Journal of Climate*, vol. 22, no. 17, pp. 4501–4519, 2009.

[110] K. Weinberger, B. Packer, and L. Saul, "Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization," in *International Workshop on Artificial Intelligence and Statistics*. PMLR, 2005, pp. 381–388.

[111] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine learning research*, vol. 5, no. Jan, pp. 27–72, 2004.

[112] K. Q. Weinberger, F. Sha, and L. K. Saul, "Learning a kernel matrix for nonlinear dimensionality reduction," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, pp. 1–8.

[113] K. Q. Weinberger and L. K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," *International journal of computer vision*, vol. 70, no. 1, pp. 77–90, 2006.

[114] Y.-J. Liu, T. Chen, and Y. Yao, "Nonlinear process monitoring and fault isolation using extended maximum variance unfolding," *Journal of process control*, vol. 24, no. 6, pp. 880–891, 2014.

[115] Z. Wang and B. He, "Locality perserving projections algorithm for hyperspectral image dimensionality reduction," in *2011 19th International Conference on Geoinformatics*. IEEE, 2011, pp. 1–4.

[116] X. Weng and J. Shen, "Classification of multivariate time series using locality preserving projections," *Knowledge-Based Systems*, vol. 21, no. 7, pp. 581–587, 2008.

[117] X. He, D. Cai, and W. Min, "Statistical and computational analysis of locality preserving projection," in *Proceedings of the 22nd international conference on machine learning*, 2005, pp. 281–288.

[118] K. Hu and J. Yuan, "Multivariate statistical process control based on multiway locality preserving projections," *Journal of Process Control*, vol. 18, no. 7-8, pp. 797–807, 2008.

[119] X. He, "Locality preserving projections," *Adv. Neural Inf. Process. Syst*, vol. 45, pp. 186–197, 01 2005.

[120] R. R. Coifman and M. J. Hirn, "Diffusion maps for changing data," *Applied and computational harmonic analysis*, vol. 36, no. 1, pp. 79–107, 2014.

[121] L. K. Saul, K. Q. Weinberger, F. Sha, J. Ham, and D. D. Lee, "Spectral methods for dimensionality reduction." *Semi-supervised learning*, vol. 3, pp. 1–18, 2006.

[122] W. Lian, R. Talmon, H. Zaveri, L. Carin, and R. Coifman, "Multivariate time-series analysis and diffusion maps," *Signal Processing*, vol. 116, pp. 13–28, 2015.

[123] V. Chinde, L. Cao, U. Vaidya, and S. Laflamme, "Spectral diffusion map approach for structural health monitoring of wind turbine blades," in *2015 American Control Conference (ACC)*. IEEE, 2015, pp. 5806–5811.
[124] V. Chinde, L. Cao, U. Vaidya, and S. Laflamme, "Damage detection on mesosurfaces using distributed sensor network and spectral diffusion maps," *Measurement Science and Technology*, vol. 27, no. 4, pp. 1–22, 2016.
[125] T. Sipola, T. Ristaniemi, and A. Averbuch, "Gear classification and fault detection using a diffusion map framework," *Pattern Recognition Letters*, vol. 53, pp. 53–61, 2015.
[126] B. Nadler, S. Lafon, R. Coifman, and I. Kevrekidis, "Diffusion maps, spectral clustering, and the reaction coordinates of dynamical systems. report, math. dept. yale, nov. 2004," *Journal of Applied and Computational Harmonic Analysis*, pp. 113–127.
[127] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis, "Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators," *arXiv preprint math/0506090*, pp. 1–8, 2005.
[128] T. Lu, Y.-F. Chen, B. Hechtman, T. Wang, and J. Anderson, "Large-scale discrete fourier transform on TPUs," *IEEE Access*, vol. 9, pp. 93 422–93 432, 2021.
[129] Y.-L. Wu, D. Agrawal, and A. El Abbadi, "A comparison of dft and dwt based similarity search in time-series databases," in *Proceedings of the ninth international conference on Information and knowledge management*, 2000, pp. 488–495.
[130] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *International conference on foundations of data organization and algorithms*. Springer, 1993, pp. 69–84.
[131] P. Chaovalit, A. Gangopadhyay, G. Karabatis, and Z. Chen, "Discrete wavelet transform-based time series analysis and mining," *ACM Computing Surveys (CSUR)*, vol. 43, no. 2, pp. 1–37, 2011.
[132] K. pong Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," in *Proceedings of 15th International Conference on Data Engineering (ICDE '99)*, 1999, pp. 126–133.
[133] D. B. Percival, "Analysis of geophysical time series using discrete wavelet transforms: An overview," *Nonlinear Time Series Analysis in the Geosciences*, vol. 112, pp. 61–79, 2008.
[134] V. Matz, M. Kreidl, and R. Smid, "Signal-to-noise ratio improvement based on the discrete wavelet transform in ultrasonic defectoscopy," *Acta Polytechnica*, vol. 44, no. 4, pp. 61–66, 08 2004.
[135] I. Liabotis, B. Theodoulidis, and M. Saraaee, "Improving similarity search in time series using wavelets," *International Journal of Data Warehousing and Mining*, vol. 2, no. 2, pp. 55–81, apr 2006. [Online]. Available: https://doi.org/10.4018/jdwm.2006040103
[136] Y. Qu, B. ling Adam, M. Thornquist, J. D. Potter, M. L. Thompson, Y. Yasui, J. Davis, P. F. Schellhammer, L. Cazares, M. Clements, G. L. Wright, and Z. Feng, "Data reduction using a discrete wavelet transform in discriminant analysis of very high dimensionality data," *Biometrics*, vol. 59, no. 1, pp. 143–151, mar 2003. [Online]. Available: https://doi.org/10.1111/1541-0420.00017
[137] Y. Wang, Y. Yan, and Q. Wang, "Wavelet-based feature extraction in fault diagnosis for biquad high-pass filter circuit," *Mathematical Problems in Engineering*, vol. 2016, pp. 1–13, 2016. [Online]. Available: https://doi.org/10.1155/2016/5682847
[138] H. Grubb and A. Walden, "Characterizing seismic time series using the discrete wavelet transform," *Geophysical Prospecting*, vol. 45, no. 2, pp. 183–205, 1997.
[139] D. Komorowski and S. Pietraszek, "The use of continuous wavelet transform based on the fast fourier transform in the analysis of multi-channel electrogastrography recordings," *Journal of medical systems*, vol. 40, no. 1, pp. 1–15, 2016.
[140] A. Osadchiy, A. Kamenev, V. Saharov, and S. Chernyi, "Signal processing algorithm based on discrete wavelet transform," *Designs*, vol. 5, no. 3, pp. 1–13, 2021. [Online]. Available: https://www.mdpi.com/2411-9660/5/3/41
[141] Y.-F. Sang, "A practical guide to discrete wavelet decomposition of hydrologic time series," *Water resources management*, vol. 26, no. 11, pp. 3345–3365, 2012.

MOHSENA ASHRAF is a Doctoral Researcher in the Brain, AI, and Child (BAIC) Lab at the University of Colorado Boulder, USA. She completed her B.Sc. in Computer Science and Engineering from Ahsanullah University of Science and Technology, Bangladesh in 2019. After finishing her B.Sc., she served as an Academic Lecturer at Ahsanullah University of Science and Technology, Bangladesh for three years. She also has research experience of more than five years, and her research area mostly focuses on Computer Vision, and Human-Computer Interaction (HCI).

FARZANA ANOWAR is currently working as the Senior Data Analytics Specialist at the Government of Saskatchewan, Canada. She received her Ph.D. and MSc in Computer Science (specialization in Data Science) from the University of Regina, Canada. Her research area mainly focuses on Machine Learning, Pattern Recognition, Data Mining, Deep Learning, Artificial Intelligence, Natural Language Processing, Health Informatics etc. She also worked at Ericsson Canada Inc. as Research Developer. Her research are mainly funded by the Government, MITACS and NSERC.

JAHANGGIR H. SETU has completed B.Sc. in Computer Science and Engineering from Daffodil International University, Bangladesh. His research interests include machine learning, data mining, IoT.

ATIQUL I. CHOWDHURY is employed with FinSource Limited as a Data Analyst. He graduated from Ahsanullah University of Science and Technology with a B.Sc. in CSE in 2019. In the field of data, he has over 3 years of expertise. He also has around 4 years of research experience. His areas of expertise include Data Mining, Image Processing, Machine Learning, Deep Learning etc.

ESHTIAK AHMED is a Doctoral Researcher at the Gamification Group of Tampere University, Finland. He completed his M.Sc. in Information Technology (Human Technology Interaction) in 2021 from Tampere University. He completed B.Sc. in Computer Science and Engineering (CSE) in 2016 from Ahsanullah University of Science and Technology. He has 1-year experience in working in the software industry and more than 2 years of experience in academia as a lecturer at Daffodil International University, Bangladesh. He has more than 6 years of experience in research, and he has published more than 20 peer-reviewed articles. He has also served as a reviewer at several renowned conferences. His research area of interest is Human-Computer Interaction (HCI), Human-Robot Interaction (HRI), Gamification, and Assistive Technologies.

ASHRAFUL ISLAM is currently working as an Assistant Professor of Computer Science and Engineering at Independent University, Bangladesh (IUB). Before moving to IUB, he received a Ph.D. and an M.S. in Computer Science from the University of Louisiana at Lafayette, USA in 2022 and 2020 respectively. His research interests include human–computer interaction, assistive technologies, applied machine learning, health informatics, and relational agent. He regularly acts as a PC Member and an international advisory board member for several conferences and a reviewer for top-tier venues and journals, including CHI, the Journal of Medical Internet Research, and the Journal of Neuroscience Methods.

ABDULLAH AL-MAMUN is an assistant professor in the Department of Computer and Cyber Sciences at Augusta University. His research interests encompass resource-efficient blockchain systems, distributed systems, and high-performance computing. He attained his Ph.D. from the University of Nevada, Reno. He obtained his M.Sc. with a double degree along with the Premio Di Merito award in computer science jointly from the University of Trento, Italy, and RWTH Aachen University, Germany. He was a recipient of the Summa Cum Laude upon completion of his B.Sc. from American International University Bangladesh. He worked at the Lawrence Berkeley National Lab.