Graph Regularized Autoencoder and its Application in Unsupervised Anomaly Detection

Imtiaz Ahmed, Travis Galoppo, Xia Hu, *Member, IEEE*, and Yu Ding, *Senior Member, IEEE*

Abstract—Dimensionality reduction is a crucial first step for many unsupervised learning tasks including anomaly detection and clustering. Autoencoder is a popular mechanism to accomplish dimensionality reduction. In order to make dimensionality reduction effective for high-dimensional data embedding nonlinear low-dimensional manifold, it is understood that some sort of geodesic distance metric should be used to discriminate the data samples. Inspired by the success of geodesic distance approximators such as ISOMAP, we propose to use a minimum spanning tree (MST), a graph-based algorithm, to approximate the local neighborhood structure and generate structure-preserving distances among data points. We use this MST-based distance metric to replace the euclidean distance metric in the embedding function of autoencoders and develop a new graph regularized autoencoder, which outperforms a wide range of alternative methods over 20 benchmark anomaly detection datasets. We further incorporate the MST regularizer into two generative adversarial networks and find that using the MST regularizer improves the performance of anomaly detection substantially for both generative adversarial networks. We also test our MST regularized autoencoder on two datasets in a clustering application and witness its superior performance as well.

Index Terms—Autoencoder, clustering, minimum spanning tree, nonlinear embedding, unsupervised learning

1 Introduction

Autoencoder [1], [2], [3] is a widely used tool in many unsupervised learning tasks such as clustering and anomaly detection [4], [5]. It is an efficient dimensionality reduction mechanism, converting a data matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$, of which columns and rows represent the attributes and observations respectively to a dimension-reduced output $\mathbf{Z} \in \mathbb{R}^{m \times p}$, such that $p < d$, but preferably $p \ll d$. Autoencoders frame the unsupervised problem of dimensionality reduction through the use of a pair of encoder and decoder—while the encoder reduces $\mathbf{X}$ to $\mathbf{Z}$, the decoder reconstructs $\mathbf{Z}$ to an $\mathbf{X}' \in \mathbb{R}^{m \times d}$, which is of the same dimension as $\mathbf{X}$. The goal of finding the optimal low-dimensional representation $\mathbf{Z}$ is to be accomplished by designing the encoder/decoder pair to minimize the reconstruction error between $\mathbf{X}$ and $\mathbf{X}'$. Please see the illustration in Fig. 1.

By producing the low-dimensional $\mathbf{Z}$, an autoencoder does not automatically perform clustering or anomaly detection. Yet, people argue that once a good low-dimensional representation of the originally high-dimensional data is obtained, the subsequent tasks become easier and manageable. For this reason, autoencoders are considered a key technique to address one of the challenges in unsupervised learning, especially when dimensionality reduction is inevitably necessary for achieving good performances. While the general idea of autoencoders can be materialized by various choices of encoder and decoder, the practical ones in use are almost invariably artificial neural networks (ANN), as depicted in Fig. 1.

Autoencoder can handle complex data, thanks to the current advancement in deep neural networks. With deep neural networks serving as an encoder/decoder, one can reach to the latent space through multiple steps of nonlinear transformation, which can help unfold data with complex intrinsic structure and greatly facilitate the subsequent detection objective.

It is not surprising that the loss function characterizing the reconstruction error between $\mathbf{X}$ and $\mathbf{X}'$ plays a crucial role in a successful autoencoder design. The choice of loss function often depends on the ultimate learning tasks, e.g., binary classification, multi-class classification, or regression, and the typical choices include *cross-entropy*, *Kullback-Leibler divergence*, *mean squared error*, and *mean absolute error*. These loss functions, however, do not have any provision to preserve the neighborhood structure in the reduced representation. But preserving the neighborhood structure while reducing data dimension is demonstrably critical and in fact a key requirement studied in the nonlinear embedding research [6], [7], [8], [9].

Dimensionality reduction problems are closely related to the manifold approximation or intrinsic structure recovery problem. According to the manifold hypothesis [10], [11], high dimensional data tend to lie on a low-dimensional manifold embedded in the high-dimensional space. An autoencoder, if properly designed, can help us to find the proper low-dimensional manifold embedding. In doing so, autoencoders need an additional embedding component in

- *Imtiaz Ahmed and Yu Ding are with the Department of Industrial & Systems Engineering, Texas A&M University, College Station, TX 77843 USA. E-mail: {imtiazavi, yuding}@tamu.edu.*
- *Travis Galoppo is with the BAE Systems, Inc., Charlotte, NC 28277 USA. E-mail: travis.galoppo@baesystems.com.*
- *Xia Hu is with the Department of Computer Science & Engineering, Texas A&M University, College Station, TX 77843 USA. E-mail: hu@cse.tamu.edu.*

Manuscript received 3 Mar. 2020; revised 15 Jan. 2021; accepted 10 Mar. 2021. Date of publication 17 Mar. 2021; date of current version 1 July 2022.
(Corresponding author: Yu Ding.)
Recommended for acceptance by D. Lowd.
Digital Object Identifier no. 10.1109/TPAMI.2021.3066111

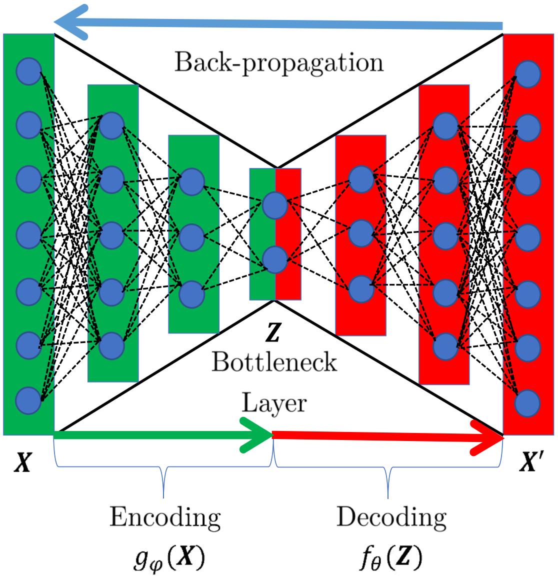


Fig. 1. Autoencoder framework using feed-forward neural networks. Blue circles represent nodes/neurons, dotted lines represent neuron connections, and the subscripts, ϕ and θ, represent the hyperparameters used in the networks.

the loss function to ensure that data points maintain the structural similarity in the low dimensional space as they are in the original, high-dimensional space.

The first question to address is how to measure the similarity between data points given the possibility of an embedded manifold structure. We cannot use traditional distance metrics such as the euclidean distance, as according to [7], they cannot accurately measure the relative closeness of data points sampled from the manifold. For details, please refer to the illustrative example in Fig. 3 of [7]. The consensus is that one would need to use some sort of geodesic distance metrics in the presence of nonlinear manifold structure. Though there are already a few efforts made to incorporate the nonlinear embedding methods in an autoencoder ([12], [13], [14], [15], [16], [17]), none of them have the provision to approximate and preserve the geodesic similarities between data points: [12] incorporated multidimensional scaling (MDS) [18] or a Laplacian eigenmap (LE) [8] in a single layer autoencoder; [13] added an additional group sparsity constraint, effective for clustering, to embedding regularization; [14] and [16] explored the locally linear embedding (LLE) [6] as an embedding regularization; [15] proposed a multilayer Laplacian autoencoding with supervised fine-tuning. The concept of local embedding or neighborhood similarity was also used for deep subspace clustering [17]. Geometric or neighborhood regularization has other benefits. When used in the deep learning models, they tend to improve the robustness of the models [19], [20]. Specifically, the deep k-NN proposed in [20] produces superior performance under adversarial attacks when compared with the traditional approaches.

The closest to what we are about to propose is a graph regularized autoencoder (GAE), which was originally developed for image classification and clustering [21]. GAE borrows the idea from the graph regularized non-negative matrix factorization (GNMF) [9] but applies the same graph regularizer (as used in GNMF) in an autoencoder. Like GNMF, GAE uses a graph Laplacian embedding regularizer, composed of a k-nearest neighbors (k-NN) graph along with the heat kernel similarity.

The majority of the embedding approaches discussed above have one major limitation in common, which is the use of euclidean distance based similarity (under MDS and LLE framework) and thereby imposes no provision to capture the intrinsic/geodesic distance among data points. The rest of them, e.g., GAE, utilizes a Gaussian kernel (also known as a heat kernel)-based weighted similarity scheme, which is also demonstrated as less effective in the presence of a nonlinear manifold [22], [23].

In this article, we propose to use a minimum spanning tree (MST) [24], a graph-based approach, to approximate the geodesic distance among data points. According to [25], the use of MST can track the manifold structure starting from a random point without any prior knowledge of local neighborhood and could potentially provide a better distance/similiary measure. With this potential, a further problem that needs to be addressed is how to preserve the MST approximated intrinsic structure in the latent space created by the autoencoder. Here, we see an opportunity to provide an integrated solution. We devise a graph regularizer, based on MST, and plant it inside the autoencoder framework as an extra loss-function component (in addition to the original reconstruction loss). We provide two alternative formulations for the graph regularizer, both of which guide the autoencoder to preserve the original data structure when generating the latent features. These latent features in turn help detect the anomalies or rare events or inform how to cluster a dataset.

We apply the resulting graph regularized autoencoder to 20 benchmark datasets to demonstrate its merit in terms of enhanced capability and robustness in anomaly detection. In order to highlight the efficacy of using MST, we compare our graph regularized autoencoder with a wide range of alternatives, including GAE, various kinds of autoencoders with or without a regularizer, as well as six anomaly detection baselines of different strength. To further demonstrate the benefit of our graph regularizer, we incorporate it into two generative adversarial network (GAN)-based anomaly detection methods [26], [27]. We find that adding the MST-based graph regularizer significantly improves the detection capability of the existing GAN-based methods. To demonstrate that our proposed approach can be useful to applications other than anomaly detection, we present a comparative study on a clustering application for which GAE was originally developed. Using the same datasets as used in [21], our proposed MST-based graph regularized autoencoder is able to outperform GAE, GNMF, and three other autoencoders initially tested in [21].

The rest of the paper unfolds as follows. Section 2 discusses the basic concepts and working mechanism of autoencoders. Section 3 describes the formulation and design of the proposed autoencoder. Section 4 compares the proposed MST-regularized autoencoder with other alternatives on the benchmark datasets for anomaly detection. The section also demonstrates the performance enhancement when the proposed graph regularizer is added to the GAN-based anomaly detection and when the proposed autoencoder is applied to clustering. Finally, we conclude the paper in Section 5.

2 Autoencoder Framework

An autoencoder framework consists of three basic components: an encoder, a decoder, and the loss function. We provide a summary here explaining how autoencoder works.

2.1 Basic Setup

An encoder is typically a feed-forward neural network, practically almost always of multiple layers. The encoding mechanism, g_ϕ, is summarized as follows:

$$\mathbf{Z} = g_\phi(\mathbf{X}) = h(\mathbf{W}_{enc}\mathbf{X} + \mathbf{b}_{enc}), \tag{1}$$

where $\phi = (\mathbf{W}_{enc}, \mathbf{b}_{enc})$ contains the parameters and $h(\cdot)$ is the activation function.

Decoder is essentially another neural network, reconstructing the original data from $\mathbf{Z}$. The decoding mechanism, f_θ, is summarized in (2)

$$\mathbf{X}' = f_\theta(\mathbf{Z}) = h(\mathbf{W}_{dec}\mathbf{Z} + \mathbf{b}_{dec}), \tag{2}$$

where $\theta = (\mathbf{W}_{dec}, \mathbf{b}_{dec})$ are the parameters associated with the decoder.

To achieve an effective representation in the latent space, the autoencoder design is to minimize a loss function, $L(\cdot)$, that quantifies the reconstruction error between $\mathbf{X}$ and $\mathbf{X}'$. The most widely used is the squared error loss, as in (3)

$$\min_{\phi,\theta} L(\mathbf{X}, \mathbf{X}') = \|\mathbf{X} - \mathbf{X}'\|_F^2. \tag{3}$$

The autoencoder concept can be materialized using many different types of neural networks such as the feedforward neural networks, convolutional neural networks (CNN), recurrent neural networks (RNN), and most recently, GAN. In this paper, we limit ourselves mostly in the regime of feedforward neural network.

2.2 Embedding Loss Function

To unearth meaningful, effective latent representations in the presence of nonlinear manifold, autoencoders need to have an additional loss component to take care of the embedding problem. The plain autoencoder and the nonlinear embedding approaches can be combined; for that, researchers have used a joint loss framework as follows [12]:

$$\min_{\phi,\theta} L(\mathbf{X}, \mathbf{X}') + \sum_{1 \le i < j \le m} G(\mathbf{z}_i, \mathbf{z}_j, \mathbf{D}_{ij}), \tag{4}$$

where the first loss component, $L(\cdot)$, reflects the autoencoder's reconstruction loss, and the second component, $G(\cdot)$, captures the embedding loss. In the embedding loss function, $\mathbf{z}_i, \mathbf{z}_j$ are the hidden representation of any two points and $\mathbf{D}_{ij}$ summarizes the euclidean distance between the same two points in the original space. The embedding loss function aims at minimizing the differences between an original pairwise distance and the corresponding pairwise distance in the hidden space. There are two popular nonlinear embedding functions used:

(1) *Multidimensional Scaling (MDS)*

$$G(\mathbf{Z}) = \sum_{i<j} (\mathbf{D}_{ij} - \|\mathbf{z}_i - \mathbf{z}_j\|_2)^2. \tag{5}$$

In this approach, the main objective is to preserve the inter-point distances in the hidden space [18]. In other words, MDS tries to minimize the difference between the pairwise euclidean distances, $\mathbf{D}_{ij}$, in the original space and their counterpart in the hidden space. In practice, the minimum of this $G(\cdot)$ function is given by the eigen-decomposition of the Gram matrix of the high dimensional data in $\mathbf{X}$ after double centering it.

(2) *Laplacian Eigenmap (LE)*

In a Laplacian eigenmap, the local properties are generated based on the pairwise similarities among data points [8]. The low dimensional representation is calculated in such a way so that data points closer in the original space should maintain the relative closeness compared to other pairs of data points in the hidden space. It means that if two points are highly similar, then the reward of minimizing the distance between them will be higher compared to another pair of points whose extent of similarity is comparatively lower. In practice, the Gaussian kernel (heat kernel) is one of the most widely used form of similarity measure. The LE uses the following formulation as its embedding function:

$$\min G(\mathbf{Z}) = \frac{1}{2}\sum_{i<j} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 \mathbf{W}_{ij} = \mathbf{Z}^T\mathbf{L}\mathbf{Z}, \tag{6}$$

where $\mathbf{W}_{ij}$ is the adjacency matrix of the graph, or known as the graph similarity matrix, used as the weight to the distances in the hidden space. The second expression is a result by invoking the spectral graph theory, where $\mathbf{L}$ is the graph Laplacian matrix, obtained by $\mathbf{L} = \mathbf{S} - \mathbf{W}$ and $\mathbf{S}$ is a diagonal matrix, also known as the degree matrix [9].

3 Graph Regularized Autoencoder

In this section, we propose a graph regularizer and show how it can be incorporated in an autoencoder. Because the graph regularizer is based on MST, we start off with a brief discussion of the MST's role in manifold approximation.

3.1 MST for Manifold Approximation

Suppose that one has a connected edge weighted undirected graph $G = (V, E)$, where V denotes the collection of vertices and E represents the collection of edges with a real valued weight e_{ij} assigned to each of them, where i, j represent a pair of vertices from V. A minimum spanning tree is a subset of the edges in E of graph G that connects all the vertices in V, without any cycles and with the minimum possible total edge weight. In other words, it is a spanning tree whose sum of edge weights is as small as possible. Here, $\mathbf{D}_{ij}$ is used to represent the distance between vertex i and vertex j connected by an edge. The weight, however, does not always mean physical distances. For example, the

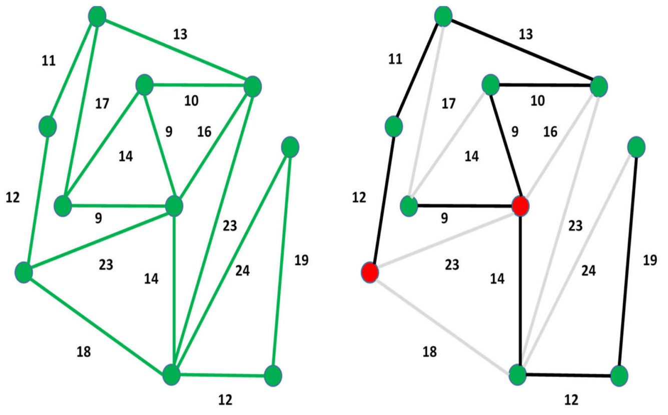


Fig. 2. Formation of an MST. The left panel is the initial graph. In the right panel, the dark black edges form the MST. The total MST weight is 109.

weight could represent the amount of flow between a pair of vertices or sometimes the cost of constructing this edge.

Consider a simple example in Fig. 2, left panel, where there are 10 vertices and 16 edges. Each of the edges has a unique edge length, which is represented by a numeric value. If we want to connect all the nodes using a subset of the given edges without forming a cycle, there could be many such combinations, but the one(s) having the minimum total edge length is the MST. MST may not be unique, but for this example, it is unique and shown in the right panel of Fig. 2. The edges in black color represent the selected nine edges from the 16 total, forming the MST.

We see that MST compresses the original graph by reducing the total energy, and thereby, provide a new measure of distance between the vertices. For example, the new distance between the two colored vertices is $12 + 11 + 13 + 10 + 9 = 55$ (right panel), while their original distance in the left panel is 23. We store in $\mathbf{M}_{ij}$ the new pairwise distance of vertices in the MST for future use.

These MST-based distances approximate the geodesic distances among vertices [25], which works better than the euclidean distance in the presence of nonlinear manifold structure [7]. MST can be applied to any dataset after the data points are represented by a graph object. We can do so by considering each observation as a vertex and the pairwise euclidean distances among the vertices as the edge weights. There are three major algorithms [24], [28], [29] that can construct a MST for a given graph. The computational cost of constructing a MST is $O(n\log m)$, where n is the number of edges in the graph and m is the number of vertices. Here, our n will always be $\binom{m}{2}$ (one edge for every pair of vertices, i.e., a fully connected graph) with $O(m^2)$ time complexity, which anyway matches the complexity of computing the pairwise euclidean distances. This means that an MST can be constructed efficiently even for a large dataset.

3.2 Proposed Graph Regularized Autoencoder

To design the graph regularizer, we decide to stick with the two embedding loss functions introduced in Section 2.2 but our proposal is to incorporate the MST distance in place of euclidean distance. For the MDS framework, we replace the pairwise euclidean distances, $\mathbf{D}_{ij}$ in (5) with the MST-based distances $\mathbf{M}_{ij}$. For the LE framework, unlike the traditional LE embedding, we define our similarity measure, $\mathbf{W}_{ij}$ through the inverse of MST-based distances $\mathbf{M}_{ij}$. Specifically

$$W_{ij} = \begin{cases} \frac{1}{M_{ij}}, & \text{if } M_{ij} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

The proposed graph regularized autoencoder is expressed as a minimization of the joint loss function of the reconstruction error and the MST-based embedding function, as in (8) below:

$$\min_{\phi,\theta} L(\mathbf{X}, \mathbf{X}') + G(\mathbf{Z}),$$

where

$$G(\mathbf{Z}) = \begin{cases} \sum_{i<j} (\mathbf{M}_{ij} - \|\mathbf{z}_i - \mathbf{z}_j\|_2)^2 \\ \qquad\qquad or \\ \frac{1}{2}\sum_{i<j} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 \mathbf{W}_{ij} = \mathbf{Z}^T\mathbf{L}\mathbf{Z}, \end{cases} \tag{8}$$

$$\begin{aligned} & L(\cdot,\cdot) \text{ is the same as in (3)}, \\ & \mathbf{Z} = g_\phi(\mathbf{X}) = h(\mathbf{W}_{enc}\mathbf{X} + \mathbf{b}_{enc}), \\ & \mathbf{X}' = f_\theta(\mathbf{Z}) = h(\mathbf{W}_{dec}\mathbf{Z} + \mathbf{b}_{dec}), \\ & \phi = (\mathbf{W}_{enc}, \mathbf{b}_{enc}), \quad \theta = (\mathbf{W}_{dec}, \mathbf{b}_{dec}). \end{aligned}$$

We guide the autoencoder reconstruction mechanism using the proposed graph regularizer, so that it maintains the manifold structure in the low-dimensional space. We refer to this proposed autoencoder as the *MST-based graph regularized autoencoder* or briefly as *MST-regularized autoencoder*. We want to note that this is a general framework, as one can involve in this framework different types of autoencoding mechanism and choose various kinds of neural network architectures.

We would like to add a note, explaining the difference between the MST-regularizer and GAE. GAE's loss function can be expressed as

$$\text{GAE} : \min_{\phi,\theta} L(\mathbf{X}, \mathbf{X}') + \lambda \cdot \frac{1}{2} \sum_{i<j} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 \mathbf{V}_{ij}, \tag{9}$$

where $\mathbf{V}_{ij}$ denotes the graph similarity matrix used in GAE and λ is a regularization parameter.

The reason that we said GAE is the closest to our work is because GAE only differs from the LE version of our MST-based autoencoder in terms of the graph similarity matrix. In GAE, $\mathbf{V}$ is constructed by forming a k-NN graph first and then using a heat kernel-based weighting scheme to generate the final weights, which is essentially the same graph similarity matrix as used in GNMF [9]. Our graph similarity matrix $\mathbf{W}$, as explained above, is constructed using the MST-based distance matrix, $\mathbf{M}$. This new graph similarity matrix is used in our other work with non-negative matrix factorization [30], and here it is used the first time with the autoencoder framework. This is in fact the *key difference* and makes all the differences in performance outcomes, to be shown later in Section 4.

Like GNMF, GAE uses a regularization parameter λ to control the weight of its graph regularizer. Similar to GNMF, GAE is sensitive to the choice of λ. As an added benefit, our MST-based graph regularizer approach does not require λ and thus avoids the sensitivity caused by it.

The MDS version of our approach is obviously different from GAE in terms of both the embedding loss function $G(\cdot)$ and the similarity matrix used.

3.3 Anomaly Detection

As mentioned earlier, an autoencoder does not necessarily produce an outcome for anomaly detection right away. To flag data points, one assigns the data points an anomaly score to signify how much it is different from normal observations.

To generate anomaly scores, we can follow two routes. The first option is to use the reconstruction error associated with a data point, defined below:

$$O_i = \|\mathbf{x}_i - \mathbf{x}_i{}'\|_2^2. \tag{10}$$

We repeat this process for all observations and rank the scores, $\{O_i, i = 1, \ldots, m\}$, in descending order, where a higher value suggests more likely to be anomalous.

The second option is to extract the low-dimensional representation and then feed it to some existing anomaly detection approaches to detect the anomalies. Using this option, we in this work make use of two existing anomaly detection approaches. One is the local minimum spanning tree (LoMST) [31] and another is the connectivity outlier factor (COF) [32].

In LoMST, a local MST is formed for each observation and its k-nearest neighbors. A LoMST score is calculated for each node, which is the total edge length of the local MST associated with the node. The anomaly score for a node is then calculated as the difference between this node's LoMST score and the average of its k-nearest neighboring nodes' LoMST scores. For this method, a local neighborhood size, k, needs to be specified *a priori*.

COF introduces a new distance measure known as the *average chaining distance* to reflect the isolation of a data point from other points. Chaining is defined as a way to connect the nearest neighbors of an observation by calculating the shortest path incrementally starting from the observation itself without producing a cycle. The length of this chain is known as the chaining distance. The corresponding anomaly scores are calculated by comparing the individual average chaining distance to its neighbors' chaining distances. For this method, again, the neighborhood size, k, needs to be specified *a priori*.

Regardless of which option we choose to produce the anomaly scores, we have to select a cut-off value to flag a data point as an anomaly. For this purpose, we choose the simplest approach, which is to flag the top N scores as anomalies, with the value of N pre-determined. Unsupervised anomaly detection methods are typically used as a first-step screening tool, flagging potential anomalies to be further analyzed and authenticated by more complex and expensive procedure. The choice of N is usually a trade-off between the goal of covering all possible anomalies and the desire to make the authentication of anomalies manageable, i.e., make the more expensive or time consuming subsequent steps practical and feasible.

3.4 Design of MST-Regularized Autoencoder

To reach an efficient design for our graph regularized framework, we have to settle on some important parameters and components essential for an autoencoder.

3.4.1 Embedding Layer Dimension

For an autoencoder, the most important parameter is arguably the number of nodes in the embedding or bottleneck layer, $\mathbf{Z}$. Note that if an autoencoder has multiple hidden layers then the last hidden layer is known as the embedding or bottleneck layer. The intrinsic dimension of this embedding layer is also a critical parameter from the manifold approximation perspective. If the dimension is chosen too small, useful features might be collapsed onto each other and become then entangled, while if the dimension is too large, the projections might become noisy and unstable [33].

Unfortunately after many years of research in this area, there is still no consensus of how to choose this embedding layer dimension. After much investigation, we choose to follow the procedure established in [34]. The main steps for the intrinsic dimension estimation method are as follows:

(1) For each data point, i, calculate the ratio of the distance of the nearest and second nearest neighbors from this point, $\mu_i = \frac{r_2}{r_1}$.
(2) Calculate the empirical distribution of μ_i by sorting them in ascending order, $F_{emp}(\mu_i) = \frac{i}{m}$.
(3) Fit a straight line through the origin and the points $(\log \mu_i, -\log(1 - F_{emp}(\mu_i)))$.
(4) Estimate the slope of this line. Round the estimated slope value to the nearest integer and use it as the intrinsic dimension, p

There are a couple of advantages of this method leading to our choice. The method uses minimal neighborhood information to find out the intrinsic dimension, and because of that, it runs rather efficiently as compared to other approaches. Moreover, we find that the method also saves us from adverse impact of dataset inhomogeneity in the estimation process. The estimated intrinsic dimensions of the datasets used in the performance study are listed in the last column of Table 3. For multilayer networks, once the dimension of the embedding layer is chosen, we decide on the dimension of the other hidden layers in a way such that the sizes of the layers gradually decrease from the input layer size to the embedding layer size.

3.4.2 Additional Components and Hyperparameters

Apart from the three main components discussed in Section 2.1, we use two auxiliary components in our graph regularized autoencoder. The auxiliary components are not specific to an autoencoder but play important roles in a neural network's training process. The first one is *batch normalization* [35], which is to normalize the data before passing to the autoencoding process. Once the input features are normalized to be on the same scale, the weights associated with them would also be on the same scale. Doing so helps avoid an uneven distribution of weights during the training process and prevent the learning algorithm from spending too much time oscillating in the plateau while looking for a global minimum.

TABLE 1
Strategy Adopted Regarding Parameters and Components of the Graph Regularized Framework

Parameters/Components	Setting adopted
Number of hidden layers	4
Activation function	Sigmoid
Dropout probability	0.5
Initialization strategy	Xavier
Optimization strategy	Gradient Descent
Number of epochs	500

The second component is known as *dropout* [36]. Dropout is a regularization method that helps in reducing the chance of overfitting. When applied to a layer, it means some nodes of that layer will be randomly dropped off, along with all of their incoming and outgoing connections. If dropout is applied, then the layers will look like consisting of different number of nodes and connectivity to the prior layer. Dropout, since what it does is to make the nodes on a layer have a random probability of being ignored, ensures that the resulting neural network does not rely on any particular input node. Generally, the dropout probability could be set as 0.5 [36], which is proved to be optimal for varieties of network architectures and tasks.

As autoencoders follow neural network architectures, one also needs to choose the values of a few standard hyperparameters. They include the number of layers, the choice of activation function, how to initialize the weight and bias values, optimization strategy and the number of epochs during the optimization etc. We summarize in Table 1 our choices regarding parameters and components of the graph regularized autoencoder. In Section 4.3, we discuss the robustness of our method in the presence of varying hyperparameters.

3.4.3 Denoising Graph Regularized Autoencoder

Basic autoencoder technology is sometimes criticized by the argument that the reconstructed output can be just a copy of the input provided. To prevent such risk, a variant of autoencoder is introduced, known as *Denoising Autoencoder* [37]. It takes partially corrupted input and reconstruct the original input with the autoencoder starting from this corrupted input. In this way, the autoencoder cannot simply memorize the training data and copy the input to its output. The steps of the denoising version of autoencoder is outlined below:

(1) The initial input $\mathbf{X}$ is corrupted into $\widetilde{\mathbf{X}}$ through stochastic mapping, $\widetilde{\mathbf{X}} = q_d(\widetilde{\mathbf{X}}|\mathbf{X})$. The corruption process is to add either Gaussian noises, salt and pepper noises, or the like.

(2) The corrupted input $\widetilde{\mathbf{X}}$ is then mapped to a hidden representation with the same process of the standard autoencoder, $\mathbf{Z} = g_\phi(\widetilde{\mathbf{X}}) = h(\mathbf{W}_{enc}\widetilde{\mathbf{X}} + \mathbf{b}_{enc})$.

(3) From the hidden representation the decoder reconstructs $\mathbf{Y} = f_\theta(\mathbf{Z})$

(4) The loss function then becomes $L = \|\mathbf{X} - \mathbf{Y}\|_F^2$

During our experimentation, we find that the traditional practice of incorporating noise (e.g., adding Gaussian noise to data) does not provide any noticeable improvement to our original MST regularizer model. So, we devise a new way of incorporating noise in the input data which actually helps us in achieving better latent space representation and consequently improves the detection of anomalies. In this approach, the noisy version of each data point is constructed as the average of its first 5 nearest neighbors as in (11). This idea of reconstruction from neighbors is actually on par with the concept of LLE [6], a popular non linear embedding approach. The choice of 5 neighbors is arbitrary here, which can be replaced with any meaningful value

$$\widetilde{\mathbf{x}_\mathbf{i}} = \frac{1}{5}\sum_{k=1,k\neq i}^{k=5} \mathbf{x}_k, \tag{11}$$

where $\mathbf{x}_k$ is the kth nearest neighbors of point $\mathbf{x}_i$. If an observation is different from its neighbors, the reconstruction loss would be high; the thought process aligns with the very definition of anomaly.

3.5 Conceptual Difference With Other Autoencoders

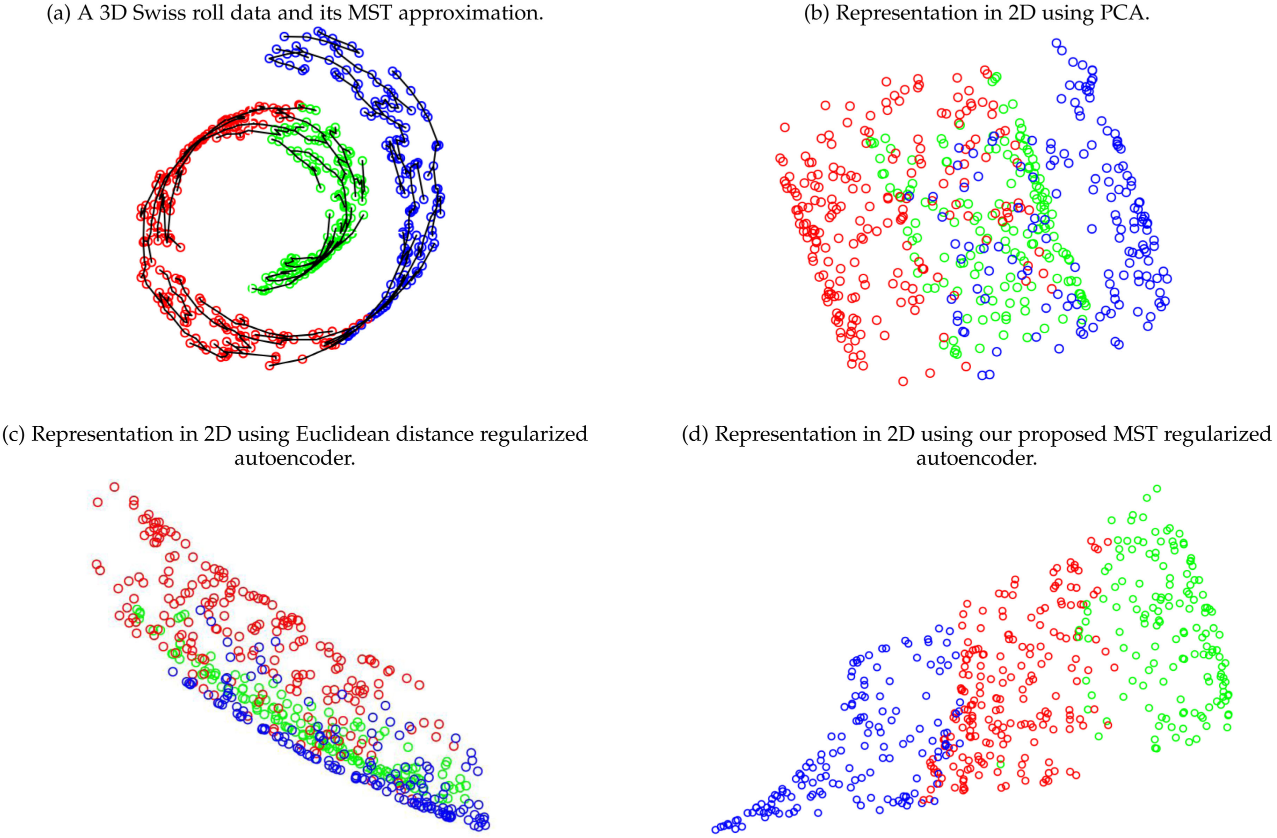
To highlight how the MST-regularized autoencoder differs from other autoencoders, we summarize several attributes associated with these autoencorders in Table 2. The main difference is the different similarity metrics incorporated in the embedding function. Table 2 includes two of the three groups of alternative methods that we will compare with in the next section of performance analysis. The first group is the methods listed in rows 1–8, consisting of methods in three subcategories: with no embedding loss (using (3)), with an embedding loss based on euclidean similarity (using (4)–(6)), and with an embedding loss based on a k-NN graph coupled with a heat-kernel based similarity (using (9)). The second group is the methods listed in rows 9–10 which are the GAN-based autoencoders. By comparing with this first group, we mean to demonstrate the impact on anomaly detection of using the MST-based graph regularizer in the embedding function. The third group in performance analysis is a set of anomaly detection baselines that do not use the autoencoder or GAN framework and they are thus not listed here.

Let us visualize, by using a toy example in Fig. 3, the impact of MST and the effectiveness of low-dimensional projection obtained by the MST-regularized autoencoder. Fig. 3a represents the well-known Swiss swirl data (a type of nonlinear manifold structure) and the MST approximation of this structure (the black edges). Note that before applying the MST, an initial graph is constructed by converting each data point into a vertex and the pairwise euclidean distance into edge weights between the vertices. The resulting MST is a sparse graph. We extract the similarities between data points from this new graph. These similarities are then used in the autoencoder. In this example, the data points are color coded to visualize their relative positions. We obtain a 2D representation of the Swiss swirl by three different methods: the principal component analysis (PCA), as in Fig. 3b; an autoencoder using euclidean

TABLE 2
Summary of Autoencoder Models

Criteria Papers	Reconstruction loss	Multilayer network	Embedding regularizer (Euclidean similarity)	Embedding regularizer (Heat kernel similarity)	Embedding regularizer (Geodesic similarity)	GAN version
Hinton et al. [2]	✓	✓				
Yu et al. [12]	✓		✓	✓		
Huang et al. [13]	✓	✓	✓			
Lu et al. [14]	✓	✓	✓			
Jia et al. [15]	✓	✓		✓		
Wei et al. [16]	✓		✓			
Liao et al. [21] (GAE)	✓	✓		✓		
Ji et al. [17]	✓	✓	✓			
Schlegl et al. [26]	✓	✓				✓
Zenati et al. [27]	✓	✓				✓
MST regularized autoencoder	✓	✓			✓	✓

Here (✓) indicates the model includes the criteria.


(a) A 3D Swiss roll data and its MST approximation.

(b) Representation in 2D using PCA.

(c) Representation in 2D using Euclidean distance regularized autoencoder.

(d) Representation in 2D using our proposed MST regularized autoencoder.

Fig. 3. MST regularized autoencoder can maintain the structural similarity in low-dimensional representation.

distance embedding function, as in Fig. 3c, and our MST-regularized autoencoder, as in Fig. 3d. The proposed approach is able to maintain the structural similarity presented in the high-dimensional space when the data is projected to the lower dimension, while the other two approaches did not.

4 Performance Analysis of Graph Regularized Autoencoder

This section is devoted to evaluating the performance of the MST-regularized autoencoder. In Section 4.1, we summarize the benchmark datasets used for performance evaluation. In Section 4.2, we compare the anomaly detection performance with several other autoencoders including GAE. In Section 4.3, we analyze the effects of changing some of the hyperparameters in the autoencoder. In Section 4.4, we provide additional performance data of the MST-regularized autoencoder against a variety of anomaly detection baselines. In Section 4.5, we incorporate the MST-based graph regularizer into two GAN-based anomaly detection approaches. In Section 4.6, we apply the MST-regularized autoencoder to two clustering datasets to show its application other than anomaly detection.

4.1 Benchmark Datasets

In this study, we use 20 benchmark anomaly detection datasets from [38] for the purpose of performance comparison. Table 3 summarizes the basic characteristics of these 20

TABLE 3
Anomaly Detection Benchmark Datasets

Dataset	Number of observations (m)	Number of anomalies ($\|O\|$)	Number of attributes (d)	Intrinsic dimension (p)
Annthyroid	7,200	347	21	4
Arrhythmia	450	12	259	17
Cardiotocography	2,126	86	21	3
HeartDisease	270	7	13	4
Page Blocks	5,473	99	10	3
Parkinson	195	5	22	4
Pima	768	26	8	6
SpamBase	4,601	280	57	3
Stamps	340	16	9	4
WBC	454	10	9	6
Waveform	3,443	100	21	17
WPBC	198	47	33	9
WDBC	367	10	30	9
ALOI	50,000	1,508	27	4
KDD	60,632	200	41	2
Shuttle	1,013	13	9	1
Ionosphere	351	126	32	8
Glass	214	9	7	4
Pen digits	9,868	20	16	6
Lymphography	148	6	19	4

datasets. For all these benchmark datasets, we know which observations are anomalies. We therefore use the actual number of anomalies as our choice of N and treat it as the same cut-off for all methods while generating the F1-scores [39]. The last column of Table 3, as mentioned earlier, is the intrinsic dimension of the dataset estimated by using the method in Section 3.4.1.

4.2 Performance Comparison

We summarize the anomaly detection performance of six autoencoder variants in Tables 4 and 5. The first two autoencoders use the euclidean distance based regularizer as in (5) and (6), respectively. The third and fourth ones are the two versions of the proposed MST-regularized autoencoder. The fifth one is GAE, and the sixth one is the plain autoencoder with no regularizer. To better reflect the method's comparative edge, we break down the comparison into four major categories in Table 4, namely *Better*, *Equal*, *Close* and *Worse*, as explained in the table. In this comparison, we use the reconstruction loss generated from the autoencoder to mark the anomalies (the first option mentioned in Section 3.3).

From Table 4, we see that MST-regularized methods outperform the other variants of autoencoder comprehensively. Individually, MST regularizer (MDS) produces 12 best detection results and is close to (within 10 percent) the best results for another 5 cases, whereas MST regularizer (LE) produces 17 best detection results and is close to the best results in the remaining 3 cases. Between the two MST formulations, MDS and LE, LE seems to produce better results by generating the uniquely best detection in 8 cases compared to 3 cases done by MDS. GAE and the two euclidean distance based regularizers are shown to be inferior than the MST regularizer, with GAE not much different from the euclidean distance based regularizers. The autoencoder with no regularizer performs clearly the worst. The autoencoder with no regularizer never scores any best detection but rather often lies far behind the best performer.

Table 5 presents the F1-scores produced by each of the competing methods. Table 4 is summarized based on the information in Table 5. To verify the statistical significance between the MST regularizer and its competitors, we apply the Friedman test, a non-parametric testing method [40], to the detection outcomes in Table 5. The Friedman test yields a very small p-value (5.05×10^{-13}), showing a sufficient significance to reject the null hypothesis and confirming that the approaches in Table 5 are significantly different from each other.

We convert the numeric performance in Table 5 into ranks (lower the better and 1 being the best) and conduct a post-hoc analysis of the competing approaches. The pairwise comparisons among all the autoencoder variants are presented in Table 6. The p-values are calculated using the Conover post-hoc test [41]. We also employed the Bonferroni correction [42] to adjust the p-values for multiple comparisons at the significance level of 0.05.

Apparently, the results in Table 6 place the autoencoders into three groups: the first group, which has the best performance, is the two MST-regularized autoencoders (our proposal); the second group includes GAE and the two euclidean distance-based regularizers; and the third group is the autoencoder with no regularizer. The performance between the groups is significantly different, while the difference within the second group is not significant and the difference within the first group, i.e., that between the two versions of the MST-based regularizer, is marginally significant. We graphically highlight these results in Fig. 4.

In Section 3.3, we mention a second option for flagging anomalies, which is to take the dimension-reduced output, **Z**, and feed it to an existing anomaly detection method. We implement the second option also, which pairs the LE version of the MST-regularized autoencoder for dimensionality reduction with either LoMST or COF for subsequent anomaly detection.

Table 7 presents the F1-score produced by the LoMST and COF approaches. We compare them with the detection resulting from using the reconstruction loss. We see that using another anomaly detection method, instead of the reconstruction error, sometimes help with the detection, but oftentimes does not. When it helps or when it does not is still ongoing research. But it appears to us that using the reconstruction error for flagging anomalies is a reasonable choice for autoencoders and is in fact what is used in all our comparison and performance studies.

Lastly, we explore the benefit of the denoising action. Table 8 presents the comparison of our MST-regularized

TABLE 4
Performance Comparison of Autoencoder Variants in Terms of Anomaly Detection

Autoencoders Result (number of datasets)	Euclidean regularizer (MDS)	Euclidean regularizer (LE)	MST regularizer (MDS)	MST regularizer (LE)	GAE	No regularizer
Better (uniquely best result)	0	0	3	8	0	0
Equal (equal to the existing best result)	2	3	9	9	4	0
Close (within 10% of the best result)	12	11	5	3	9	8
Worse (not within 10% of the best result)	6	6	3	0	7	12

TABLE 5
F1-Score Produced by Autoencoder Variants

Dataset	Euclidean regularizer (MDS)	Euclidean regularizer (LE)	MST regularizer (MDS)	MST regularizer (LE)	GAE	No regularizer
WBC	0.70	0.70	**0.80**	**0.80**	**0.80**	0.60
Heart	0.43	0.43	**0.57**	**0.57**	0.43	0.29
Cardiotocography	0.34	0.35	0.35	**0.37**	0.34	0.31
SPAMBASE	0.15	0.15	**0.18**	**0.18**	0.17	0.10
PIMA	0.12	**0.15**	0.08	**0.15**	0.04	0.08
WDBC	**0.60**	**0.60**	**0.60**	**0.60**	**0.60**	0.30
Glass	0.00	0.00	**0.11**	**0.11**	0.00	0.00
Shuttle	0.00	0.00	**0.06**	**0.06**	0.00	0.00
Stamps	0.23	0.38	0.23	**0.69**	0.23	0.08
Ionosphere	0.54	0.52	**0.61**	0.59	0.59	0.50
WPBC	0.19	0.23	0.17	**0.26**	0.13	0.15
KDD	0.41	0.44	0.46	**0.47**	0.39	0.22
Lymphography	0.50	0.67	0.50	**0.83**	0.33	0.17
Arrhythmia	0.17	0.17	**0.25**	**0.25**	**0.25**	0.17
Pendigits	0.00	0.00	**0.05**	**0.05**	0.00	0.00
Parkinsons	0.20	0.40	0.40	**0.60**	0.20	0.00
ALOI	0.25	0.23	**0.27**	0.26	0.21	0.09
Annthyroid	0.03	0.03	0.03	**0.04**	0.03	0.02
Waveform	0.08	0.07	**0.13**	0.10	0.08	0.03
PBLOCK	**0.19**	**0.19**	**0.19**	**0.19**	**0.19**	0.12

Bold entries represent the best detection performance in a respective dataset.

TABLE 6
The p-Values of Pairwise Comparisons Between the Competing Approaches

Approaches	MST regularizer (LE)	MST regularizer (MDS)	Euclidean regularizer (MDS)	Euclidean regularizer (LE)	GAE
MST regularizer (MDS)	0.02	-	-	-	-
Euclidean regularizer (MDS)	2.61×10^{-18}	2.67×10^{-11}	-	-	-
Euclidean regularizer (LE)	8.28×10^{-15}	6.64×10^{-08}	1	-	-
GAE	2.00×10^{-20}	1.87×10^{-13}	1	0.14	-
No regularizer	6.96×10^{-35}	4.08×10^{-29}	1.21×10^{-12}	3.67×10^{-16}	1.68×10^{-10}

autoencoder under LE formulation with regular and the new denoising option. We can see that the regular denoising option does not help much when compared to the detections obtained without applying any sort of denoising (see Column 5 in Table 5). The new denoising option can improve, although marginally, the detection performance in half of the cases, as compared to the regular denoising process.

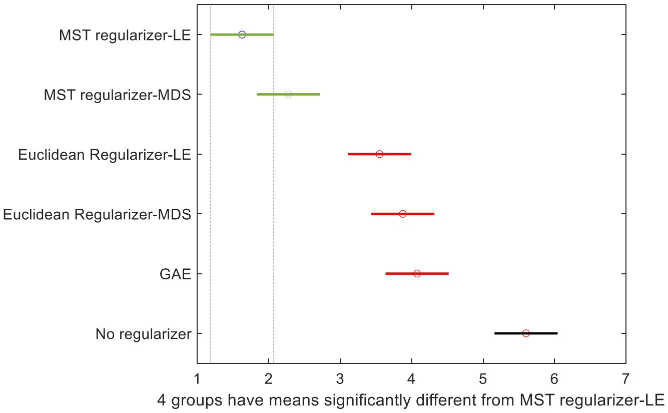


Fig. 4. Post hoc analysis on the ranking data obtained by the Friedman test.

TABLE 7
Comparison Between Reconstruction Loss Based Detection and Those Using an Existing Anomaly Detection Methods After the MST-Regularizer (LE)

Dataset	LoMST	COF	Reconstruction loss
WBC	0.5	0.5	**0.8**
Heart	0.29	0.14	**0.57**
Cardiotocography	0.2	0.18	**0.37**
SPAMBASE	**0.18**	0.14	**0.18**
PIMA	0.038	0.076	**0.15**
WDBC	**0.7**	0.6	0.6
Glass	**0.33**	0.11	0.11
Shuttle	**0.15**	**0.15**	0.06
Stamps	0.38	0.31	**0.69**
Ionosphere	**0.71**	0.69	0.59
WPBC	**0.26**	0.21	**0.26**
KDD	0.02	0.01	**0.47**
Lymphography	0.5	0.5	**0.83**
Arrhythmia	**0.33**	**0.33**	0.25
Pendigits	0	0	**0.05**
Parkinsons	0.4	0.2	**0.6**
ALOI	0.09	0.07	**0.26**
Annthyroid	0.05	**0.09**	0.04
Waveform	0.11	**0.12**	0.1
PBLOCK	**0.25**	**0.25**	0.19

Bold entries represent the best detection performance in a respective dataset.

4.3 Influence of the Hyperparameters

This subsection studies the robustness of our method in the presence of hyperparameter changes, including the number of layers, the dropout rate, and the noise parameter.

4.3.1 Changing the Number of Layers

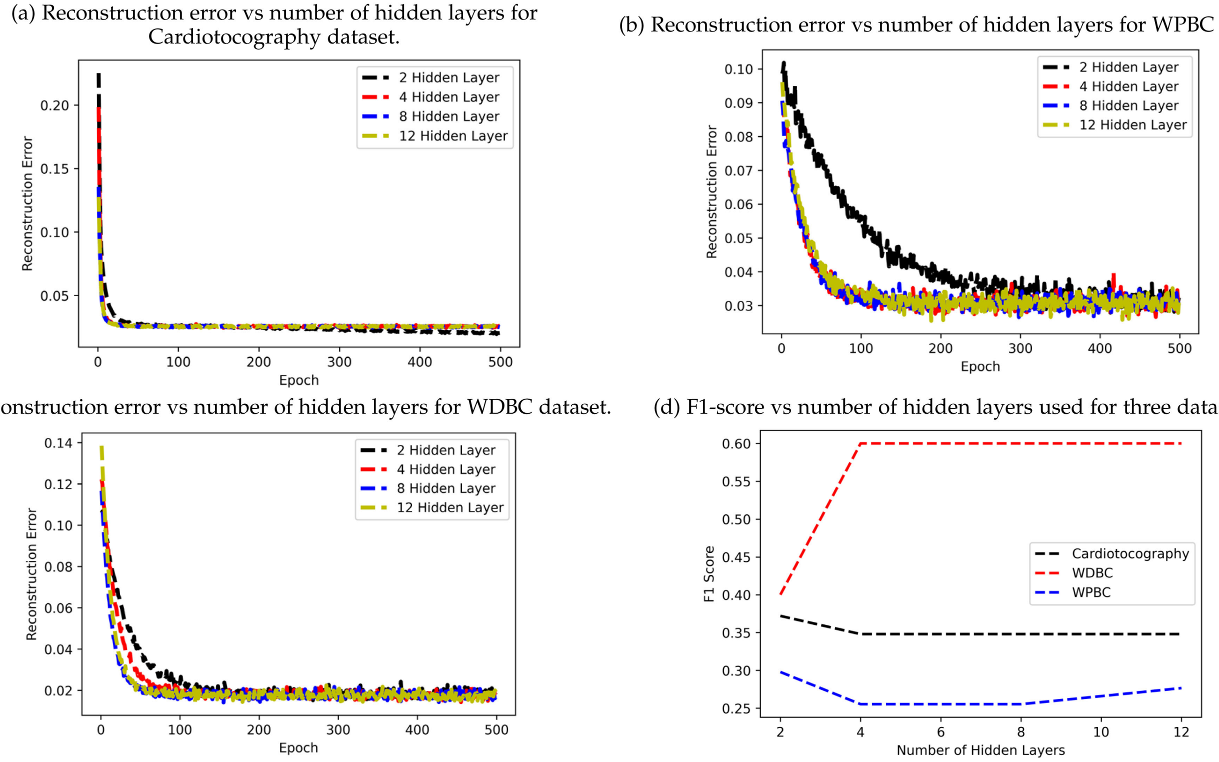
We test the effect of the number of layers by changing it from 2 to 12. Figs. 5a, 5b, and 5c presents, for three datasets, the change of the reconstruction error as the optimization is progressing. We omit plotting for other datasets to save space, as the message is the same (the same reason applies to the latter subsections where only a few sample datasets are used). When given long enough time, regardless of the number of layers used, the reconstruction error converges to more or less the same level. We do observe, however, for some datasets, like WPBC and WDBC, using 2 hidden layers shows a slower convergence. The 2-layer network takes more epochs to catch up with other settings. This is certainly a disadvantage for using 2 hidden layers. Fig. 5d presents the F1-scores versus the number of layers for the same three datasets. Once there are four or more hidden layers, the performance of the proposed autoencoder does not appear to fluctuate much. Between 2 and 4 layers, using 4 layers appears to be a safer choice. This is the reason that we used 4 hidden layers in the MST-regularized autoencoder in the previous subsection while generating Tables 4, 5, 6, 7, and 8. In fact, we set 4 hidden layers as default unless otherwise specified.

TABLE 8
Change in F1-Score Using the New Denoising Approach Under the LE Formulation

Dataset	With regular denoising	With new denoising
WBC	0.80	0.80
Heart	0.57	0.57
Cardiotocography	0.37	0.37
SPAMBASE	0.19	0.20
PIMA	0.15	0.19
WDBC	0.60	0.70
Glass	0.11	0.11
Shuttle	0.00	0.06
Stamps	0.54	0.69
Ionosphere	0.60	0.63
WPBC	0.30	0.30
KDD	0.47	0.47
Lymphography	0.83	0.83
Arrhythmia	0.25	0.33
Pendigits	0.05	0.05
Parkinsons	0.60	0.80
ALOI	0.26	0.26
Annthyroid	0.05	0.05
Waveform	0.09	0.15
PBLOCK	0.19	0.23

4.3.2 Changing the Dropout Rate

We test on the effect of different dropout rates. Figs. 6a, 6b, and 6c show the effects of dropout rate on the reconstruction error for three sample datasets (some of the datasets differ from the ones used in the number of layers test). In our analysis across datasets, we find that using a very low dropout rate, e.g., 0.1, is not an effective choice but a

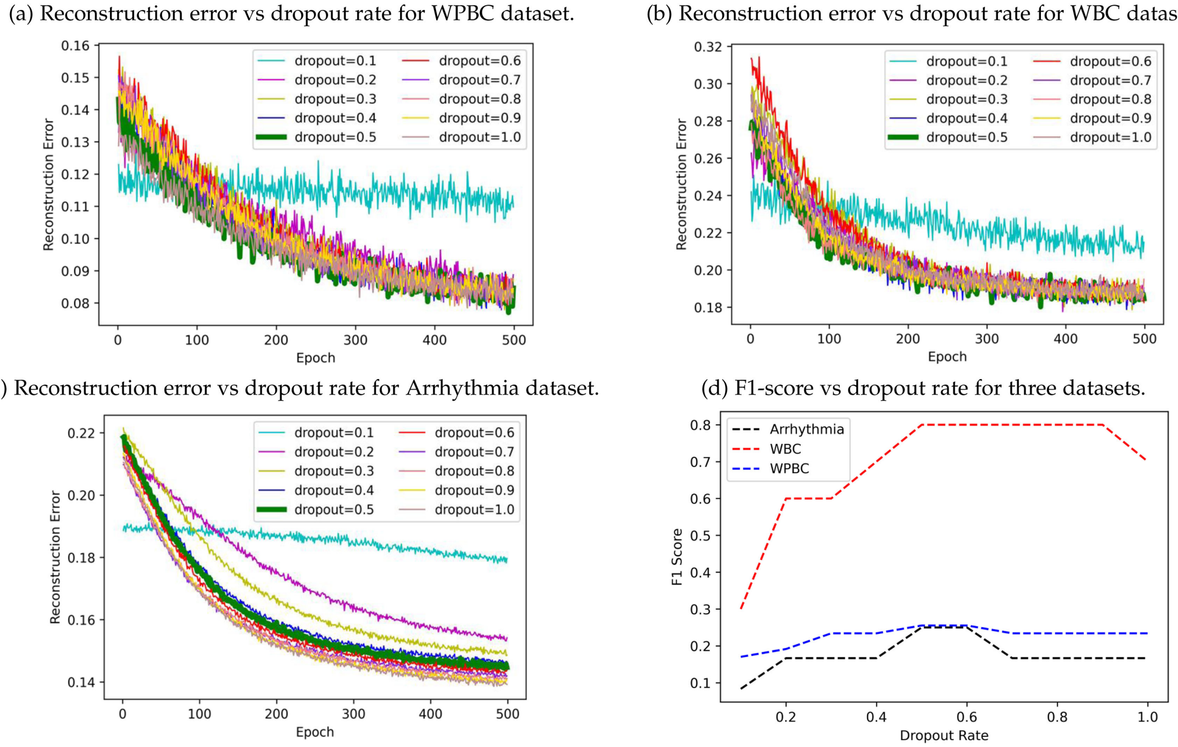

(a) Reconstruction error vs number of hidden layers for Cardiotocography dataset.

(b) Reconstruction error vs number of hidden layers for WPBC dataset.

(c) Reconstruction error vs number of hidden layers for WDBC dataset.

(d) F1-score vs number of hidden layers used for three datasets.

Fig. 5. Effect of changing the number of hidden layers on the MST-regularized autoencoder.


(a) Reconstruction error vs dropout rate for WPBC dataset.

(b) Reconstruction error vs dropout rate for WBC dataset.

(c) Reconstruction error vs dropout rate for Arrhythmia dataset.

(d) F1-score vs dropout rate for three datasets.

Fig. 6. Effect of changing the dropout rate on the MST-regularized autoencoder. In (a)-(c), the bold dark-green line corresponds to the dropout rate of 0.5.

dropout rate of 0.5 or greater generally leads to comparable performance. Fig. 6d plots the F1-scores for the same three datasets versus the dropout rate. We again notice that using a dropout rate around 0.5 is a safe choice. We thereby set our default choice for this parameter as 0.5.

4.3.3 Changing the Noise Parameter in Denoising

The noise parameter is the standard deviation of the Gaussian noise. We change the value of the noise standard deviation from 0.1 to 0.8 and summarize its effect in Fig. 7 for 8 datasets. We see that using a high values of noise standard

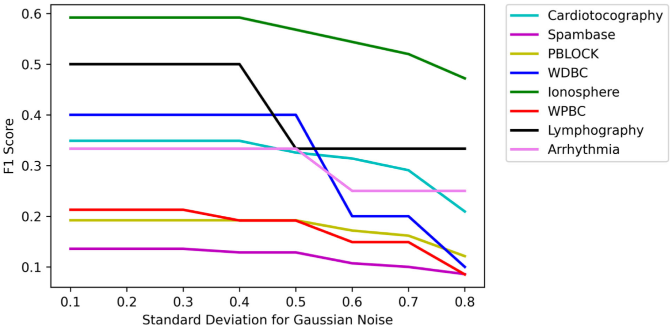


Fig. 7. F1-score versus noise standard deviation for 8 benchmark datasets.

deviation (≥ 0.6) does not help with the detection mission. Generally speaking, using a smaller noise is safer. The middle-to-low value, like 0.3 or 0.4, appears to produce the best result. For our model, we choose 0.3 as the default standard deviation of the noise.

4.4 Comparison With State of the Art Anomaly Detection Baselines

We also evaluate our MST-regularized autoencoder with a variety of the state of the art (SOTA) anomaly detection baselines. These SOTA baselines are chosen from different families of anomaly detection methods, including nearest neighbors-based approaches, classification-based approaches, deep learning-based approaches, isolation-based approaches and subspace approaches. The specific approaches that are included from these families are the local outlier factor (LOF) [43], one-class support vector machine (OC-SVM) [44], deep autoencoding Gaussian mixture model (DAGMM) [45], deep structured energy based models (DSEBM) [46], isolation forest (IF) [47], and subspace outlier detection (SOD) [48]. All of these approaches were originally proposed for anomaly detection. Rather than running for all 20 benchmark datasets again, we choose three representative datasets for this comparison, which are KDD (with the most observations), ALOI (with the most anomalies), and Arrhythmia (with the most features). The comparison results are highlighted in Table 9. We see that the LE version of the MST-regularized autoencoder achieves the best performance in all three cases (although tied for the third dataset).

4.5 Impacts of MST-Based Graph Regularizer on GAN-Based Anomaly Detection

GAN has been called one of the most interesting ideas proposed in the last 10 years [49]. We would like to see how our MST-regularizer impacts GAN-based anomaly detection once it is incorporated into its loss function.

TABLE 9
F1-Scores Produced by the Competing Approaches

Dataset	MST regularizer	LOF	OC-SVM	DAGMM	DSEBM	IF	SOD
KDD	**0.47**	0.02	0.42	0.41	0.45	0.42	0.14
ALOI	**0.26**	0.21	0.07	0.03	0.13	0.03	0.18
Arrhythmia	**0.25**	0.17	**0.25**	0.17	**0.25**	0.17	**0.25**

Bold entries represent the best detection performance in a respective dataset.

We consider two GAN-based anomaly detection approaches. The first one is known as the AnoGAN [26]. AnoGAN involves training a deep convolutional GAN, and, at inference, using the trained GAN to recover a latent representation for each test data point. The anomaly score is measured by taking the sum of reconstruction loss and discrimination loss. The reconstruction loss (also called the residual loss) measures how well the resulting GAN is able to reconstruct the data from the representative latent points using the trained generator, whereas the discrimination loss measures the performance of the discriminator of the GAN, which is to separate the real data from the fake sample generated by the generator of the GAN. The discrimination loss ensures that the generated data point from the latent space lie on the data manifold. To test the impact of our MST-based graph regularizer, we incorporate the MST-based regularizer as an additional loss component in the discriminator. If a test point is an anomaly, both of reconstruction loss and discrimination loss would be high.

The second approach that we consider, known as the adversarially learned anomaly detection [27, ALAD]. It is claimed to be an improvement over AnoGAN. In contrast to AnoGAN, ALAD uses bi-directional GANs, where an encoder network is used to map data samples to latent variables. This design enables ALAD to avoid the computationally expensive inference procedure required by AnoGAN as the latent space coordinates can now be generated by using a single feed-forward pass through the encoder network. ALAD also incorporates other ideas to stabilize the GAN training procedure. We add our MST regularizer as an additional loss measurement during the encoder training process to test on the impact of such modification.

Table 10 presents the performance of these two GAN-based anomaly detection approaches with and without the MST regularizer. The dataset used here is KDD. We use the parameters and other choices as suggested by ALAD's authors [27]. We use their code shared at `GitHub`, modify it to incorporate the MST regularizer, and generate the evaluation scores, which include precision, recall, and F1-score. It is evident that the MST regularizer improves the detection performance for both approaches.

4.6 Performance on Clustering Task

GAE [21] was originally developed for clustering tasks. Having included GAE for anomaly detection, we wonder what if we apply the proposed MST-regularized autoencoder to clustering. The good performance of MST-regularized autoencoder in anomaly detection leads us to think that MST indeed provides a competitive similarity measure and could help with clustering as well.

To test on this thought, we decide to do an analysis of our MST-regularized autoencoder on the clustering application used in the GAE paper [21]. We used the same two datasets (COIL-20 and MNIST) and the same two performance metrics (the normalized mutual information or NMI and the clustering accuracy or ACC) as used in [21]. The GAE paper [21] tested on the following clustering approaches: GAE itself, GNMF [9], sparse autoencoder (SAE) [50],

TABLE 10
Performance of GAN-Based Anomaly Detection With and Without the MST Regularizer

Method	Precision	Recall	F1-score
AnoGAN	0.75	0.76	0.75
(1000 iterations, with regularizer)			
AnoGAN	0.44	0.45	0.44
(1000 iterations, without regularizer)			
AnoGAN	0.35	0.36	0.35
(100 iterations, with regularizer)			
AnoGAN	0.14	0.14	0.14
(100 iterations, without regularizer)			
ALAD	0.19	0.47	0.27
(100 iterations, with regularizer)			
ALAD	0.03	0.06	0.04
(100 iterations, without regularizer)			
ALAD	0.36	0.36	0.36
(500 iterations, with regularizer)			
ALAD	0.19	0.45	0.26
(500 iterations, without regularizer)			

TABLE 11
Clustering Performance of the Competing Methods on the COIL-20 Dataset

Methods	NMI (%)	ACC (%)
MST regularizer (LE)	**85.24±2.02**	**74.10±4.27**
GAE	81.12±1.58	69.73±3.81
GNMF	84.59±1.79	71.58±4.04
CAE	76.58±0.71	66.81±3.45
SAE	76.15±1.59	65.69±2.71
RAE	75.31±1.04	64.69±3.09

The values in the table are in the form of "mean±standard deviation." Bold entries represent the best clustering performance.

TABLE 12
Clustering Performance of the Competing Methods on the MNIST Dataset

Methods	NMI (%)	ACC (%)
MST regularizer (LE)	**58.41±2.18**	**55.12±1.77**
GAE	50.12±2.27	51.33±1.35
GNMF	53.64±2.02	50.04±3.04
CAE	49.44±2.77	54.58±3.30
SAE	42.97±2.07	52.86±3.36
RAE	40.35±1.03	49.64±1.34

The values in the table are in the form of "mean±standard deviation." Bold entries represent the best clustering performance.

contractive autoencoder (CAE) [51], and regular autoencoder (RAE) without any additional regularization. To generate the clustering assignments from the encoded/latent representations, a k-means clustering algorithm is used for all autoencoder varieties.

Our focus is to compare the MST-regularized autoencoder with GAE and GNMF, because GAE and GNMF were shown in [21] as most competitive. To make sure that we have a fair comparison, we re-run GAE and GNMF, together with our MST-regularized autoencoder, on the same datasets on our own computer. The GAE code was provided by its authors and the code of GNMF was released by its authors for public use. Because of this re-running, the results reported here do not match exactly those reported in [21]. We did not re-run SAE, CAE, or RAE, but adopted their performance values directly from [21]. Based on [21], SAE, CAE and RAE are not nearly as competitive as the other approaches, so that some fluctuations in performance, resulting of re-running, will unlikely change the order of the performance. The clustering performance comparisons are summarized in Tables 11 and 12. In both cases, our approach comes superior to GAE, GNMF, and all other autoencoder varieties.

5 Summary

To obtain a useful representation of high dimensional data, we need to preserve the intrinsic structure of the data during the process of dimensionality reduction. In this paper, we propose a new MST-based graph approach to approximate the manifold structure embedded in the data. We design two separate frameworks for incorporating this MST-based distance metric as an additional regularizer to an autoencoder. The proposed MST-based graph regularized autoencoder helps process complex data in high dimensions and obtain an effective low-dimensional latent space representation. We argue that adding this MST-based graph regularizer enhances an autoencoder's performance in the application of anomaly detection.

To support our claim, we present a detailed performance comparison study using 20 benchmark anomaly detection datasets, where the MST-based graph regularized autoencoder clearly outperforms a wide variety of alternatives, including several autoencoder variants and a set of six non-autoencoder anomaly detection baselines, chosen from different families of anomaly detection approaches. To further demonstrate the positive impact that can be made by this MST-based graph regularizer, we incorporate it into two GAN-based anomaly detection approaches and compare the detection results before and after adding the regularizer. As we anticipated, GAN with the MST-based graph regularizer performs substantially better than their no-regularizer counterparts. Our application of the MST-regularized autoencoder to clustering turns out to be successful too. The sets of empirical evidence appears to strongly support the merit of the MST-based graph regularizer.

We investigate a number of issues related to the design of an autoencoder with the graph regularizer. One issue that still eludes us is how to choose the optimal number of layers for a much broader range in the encoder/decoder network design. This issue may be resolved by adopting some of the most recent methods in the AutoML research [52].

ACKNOWLEDGMENTS

The work of Imtiaz Ahmed and Yu Ding was supported in part by the grants from US National Science Foundation under grant no. IIS-1849085, and ABB project contract no. M1801386. The work of Xia Ben Hu was supported in part by the US National Science Foundation under Grants IIS-1750074 and IIS-1718840.

REFERENCES

[1] M. A. Kramer, “Nonlinear principal component analysis using autoassociative neural networks,” *AIChE J.*, vol. 37, no. 2, pp. 233–243, 1991.

[2] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[4] C. Zhou and R. C. Paffenroth, “Anomaly detection with robust deep autoencoders,” in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 665–674.

[5] M. Schreyer, T. Sattarov, D. Borth, A. Dengel, and B. Reimer, “Detection of anomalies in large scale accounting data using deep autoencoder networks,” 2017, *arXiv: 1709.05254*.

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

[7] J. B. Tenenbaum, V. De Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *Science*, vol. 290, pp. 2319–2323, 2000.

[8] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2002, pp. 585–591.

[9] D. Cai, X. He, J. Han, and T. S. Huang, “Graph regularized nonnegative matrix factorization for data representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.

[10] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.

[11] C. Fefferman, S. Mitter, and H. Narayanan, “Testing the manifold hypothesis,” *J. Amer. Math. Soc.*, vol. 29, no. 4, pp. 983–1049, 2016.

[12] W. Yu, G. Zeng, P. Luo, F. Zhuang, Q. He, and Z. Shi, “Embedding with autoencoder regularization,” in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2013, pp. 208–223.

[13] P. Huang, Y. Huang, W. Wang, and L. Wang, “Deep embedding network for clustering,” in *Proc. 22nd Int. Conf. Pattern Recognit.*, 2014, pp. 1532–1537.

[14] S. Lu, H. Liu, and C. Li, “Manifold regularized stacked autoencoder for feature learning,” in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2015, pp. 2950–2955.

[15] K. Jia, L. Sun, S. Gao, Z. Song, and B. E. Shi, “Laplacian autoencoders: An explicit learning of nonlinear data manifold,” *Neurocomputing*, vol. 160, pp. 250–260, 2015.

[16] C. Wei, S. Luo, X. Ma, H. Ren, J. Zhang, and L. Pan, “Locally embedding autoencoders: A semi-supervised manifold learning approach of document representation,” *PLoS One*, vol. 11, no. 1, 2016, Art. no. e0146672.

[17] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, “Deep subspace clustering networks,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 24–33.

[18] J. B. Kruskal and M. Wish, *Multidimensional Scaling*. Newbury Park, CA, USA: Sage, 1978.

[19] C. Sitawarin and D. Wagner, “On the robustness of deep k-nearest neighbors,” in *Proc. IEEE Security Privacy Workshops*, 2019, pp. 1–7.

[20] N. Papernot and P. McDaniel, “Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning,” 2018, *arXiv: 1803.04765*.

[21] Y. Liao, Y. Wang, and Y. Liu, “Graph regularized auto-encoders for image representation,” *IEEE Trans. Image Process.*, vol. 26, no. 6, pp. 2839–2852, Jun. 2017.

[22] M. T. Harandi, C. Sanderson, A. Wiliem, and B. C. Lovell, “Kernel analysis over Riemannian manifolds for visual recognition of actions, pedestrians and textures,” in *Proc. IEEE Workshop Appl. Comput. Vis.*, 2012, pp. 433–439.

[23] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi, “Kernel methods on the Riemannian manifold of symmetric positive definite matrices,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 73–80.

[24] R. C. Prim, “Shortest connection networks and some generalizations,” *Bell Labs Tech. J.*, vol. 36, no. 6, pp. 1389–1401, 1957.

[25] J. A. Costa and A. O. Hero, “Geodesic entropic graphs for dimension and entropyestimation in manifold learning,” *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2210–2221, Aug. 2004.

[26] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, “Unsupervised anomaly detection with generative adversarial networks to guide marker discovery,” in *Proc. Int. Conf. Inf. Process. Med. Imag.*, 2017, pp. 146–157.

[27] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar, “Adversarially learned anomaly detection,” in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 727–736.

[28] J. B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem,” *Proc. Amer. Math. Soc.*, vol. 7, no. 1, pp. 48–50, 1956.

[29] J. Nešetřil, E. Milková, and H. Nešetřilová, “Otakar Boruvka on minimum spanning tree problem translation of both the 1926 papers, comments, history,” *Discrete Math.*, vol. 233, no. 1–3, pp. 3–36, 2001.

[30] I. Ahmed, X. B. Hu, M. P. Acharya, and Y. Ding, “Neighborhood structure assisted non-negative matrix factorization and its application in unsupervised point-wise anomaly detection,” *J. Mach. Learn. Res.*, vol. 22, no. 34, pp. 1–32, 2021.

[31] I. Ahmed, A. Dagnino, and Y. Ding, “Unsupervised anomaly detection based on minimum spanning tree approximated distance measures and its application to hydropower turbines,” *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 654–667, Apr. 2019.

[32] J. Tang, Z. Chen, A. W. Fu, and D. W. Cheung, “Enhancing effectiveness of outlier detections for low density patterns,” in *Proc. 6th Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2002, pp. 535–548.

[33] E. Levina and P. J. Bickel, “Maximum likelihood estimation of intrinsic dimension,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2005, pp. 777–784.

[34] E. Facco, M. d’Errico, A. Rodriguez, and A. Laio, “Estimating the intrinsic dimension of datasets by a minimal neighborhood information,” *Sci. Rep.*, vol. 7, no. 1, 2017, Art. no. 12140.

[35] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[37] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

[38] G. O. Campos *et al.*, “On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study,” *Data Mining Knowl. Discov.*, vol. 30, no. 4, pp. 891–927, 2016.

[39] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Waltham, MA, USA: Butterworth-Heinemann, 1979.

[40] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.

[41] W. J. Conover, *Practical Nonparametric Statistics*. New York City, NY, USA: Wiley, 1999.

[42] J. M. Bland and D. G. Altman, “Multiple significance tests: The Bonferroni method,” *Brit. Med. J.*, vol. 310, 1995, Art. no. 170.

[43] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: Identifying density-based local outliers,” in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2000, pp. 93–104.

[44] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.

[45] B. Zong *et al.* “Deep autoencoding Gaussian mixture model for unsupervised anomaly detection,” in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–19.

[46] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, “Deep structured energy based models for anomaly detection,” in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1100–1109.

[47] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.

[48] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Outlier detection in axis-parallel subspaces of high dimensional data," in *Proc. 13th Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2009, pp. 831–838.

[49] I. Goodfellow *et al*., "Generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst*., 2014, pp. 2672–2680.

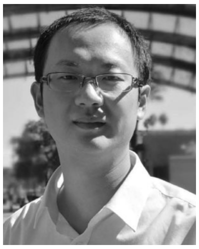
[50] M. Ranzato, C. Poultney, S. Chopra, and Y. L. Cun, "Efficient learning of sparse representations with an energy-based model," in *Proc. Int. Conf. Neural Inf. Process. Syst*., 2007, pp. 1137–1144.

[51] S. Rifai *et al*., "Higher order contractive auto-encoder," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2011, pp. 645–660.

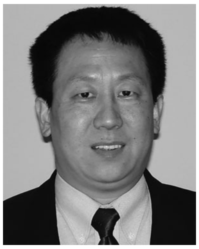
[52] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 847–855.

Imtiaz Ahmed received the BSc and MSc degrees in industrial and production engineering from the Bangladesh University of Engineering & Technology, Dhaka, Bangladesh, and the PhD degree in industrial engineering from Texas A&M University, College Station, Texas. He is currently a postdoctoral researcher at the Industrial and Systems Engineering Department, Texas A&M University. His research interests include data analytics, machine learning, and quality control.

Travis Galoppo received the BS degree in computer and information sciences from Niagara University, New York, in 1998, and the MS degree in computer science from Columbia University, New York City, New York, in 2009. He is currently a senior principal scientist at BAE Systems, Inc., North Carolina. His research interests include machine learning, computational statistics, and high performance computing.



Xia Hu (Member, IEEE) received the BS and MS degrees in computer science from Beihang University, Beijing, China, and the PhD degree in computer science and engineering from Arizona State University, Tempe, Arizona. He is currently an associate professor at the Department of Computer Science and Engineering, Texas A&M University, College Station, Texas. He has published nearly 100 papers in several major academic venues, including WWW, SIGIR, KDD, ICDM, SDM, WSDM, IJCAI, AAAI, CIKM, and ICWSM. His developed automated machine learning (AutoML) package, AutoKeras, has received more than 6,000 stars on GitHub and has become the most rated open-source AutoML system. For more information, please visit http://faculty.cs.tamu.edu/xiahu/



Yu Ding (Senior Member, IEEE) received the BS degree from the University of Science and Technology of China, Hefei, China, in 1993, the MS degree from Tsinghua University, Beijing, China, in 1996, the MS degree from Penn State University, Pennsylvania, in 1998, and the PhD degree in mechanical engineering from the University of Michigan, Ann Arbor, Michigan, in 2001. He is currently the Mike and Sugar Barnes professor of industrial and systems engineering and a professor of electrical and computer engineering with Texas A&M University. His research interests include data and quality science. He is the editor-in-chief of the *IISE Transactions* for the term of 2021–2024. He is a fellow of IIE, a fellow of ASME, and a member of INFORMS.