Efficient Anomaly Detection for High-Dimensional Sensing Data with One-Class Support Vector Machine

Yan Qiao[†‡§], *Member, IEEE*, Kui Wu[§], *Senior Member, IEEE*, Peng Jin[¶]
[†]School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, Anhui 230009 China
[‡]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876 China
[§]Department of Computer Science, University of Victoria, Victoria, BC V8P5C2, Canada
[¶]School of Information and Computer, Anhui Agriculture University, Hefei, Anhui 230036 China

***Abstract*—This paper addresses the problem of anomaly detection for high-dimensional sensing data. The one-class support vector machine (OCSVM) is one of the most popular unsupervised methods for anomaly detection. When data are high dimensional and large scale, however, the efficiency of OCSVM-based methods in anomaly detection suffers. Although dimensionality-reduction tools, such as deep belief networks, can be applied to compress the high-dimensional data to alleviate the problem, the accuracy and timely detection are still hard to improve due to the inherent features of OCSVM. In this paper, we propose a new form of OCSVM model based on the structure of the compressed data and the characteristics of OCSVM. Based on the new model, we design both optimal and approximate methods for model training and testing. We evaluate the performance of our methods with extensive experiments on four real-world datasets. The experimental results demonstrate that our new methods, both optimal and approximate ones, not only significantly outperform the state-of-the-art in accuracy and efficiency, but also achieve the good performance without the need of manual parameter tuning. In addition, our approximate training and testing mechanism can reduce the computing time by three orders of magnitude with a negligible loss in accuracy.**

***Index Terms*—Internet of things; anomaly detection; high-dimensional data; deep belief network; one-class support vector machines**

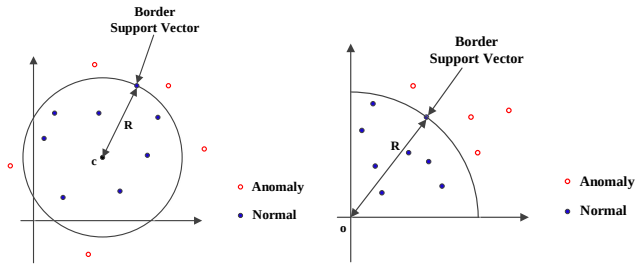
I. Introduction

As Internet of Things (IoT) and artificial intelligence (AI) technologies become widespread, many objects, as well as physical environments, are under a full range of monitoring and offer smart services to make our life better [1]. In most IoT platforms, large volumes of sensing data are collected and transported to servers, which are responsible for processing and analysing those data for specific tasks. One such task is anomaly detection. *Anomalies* in this paper are defined as data whose pattern does not conform to that exhibited by the majority of the data set. Anomalies, being good or bad depending on particular context, indicate interesting events for which we should pay special attention and take corresponding actions. Clearly, a late detection of anomalies may not be useful, and hence it is an essential requirement for nearly real-time anomaly detection.

Anomaly detection for sensing data in real-time has inherent challenges due to the following two reasons. First, data collected by sensors are becoming increasingly high-dimensional and large-scale nowadays. The high-dimensional data not only challenge the anomaly detection techniques in efficiency but also make the anomalous inconspicuous [2]. Thus, anomaly detection should be scalable and efficient, and sensitive to anomalous in high-dimensional space. Second, since sensing data usually arrive in the streaming manner (e.g. environmental data, behavioral data, and visual data), it is prohibitive and sometimes difficult to label the data, especially when the data have high dimensions. Therefore, anomaly detection should process the data in an unsupervised way. To summarize, good real-time anomaly detection methods should be *efficient*, *accurate*, *scalable*, and *unsupervised*.

Many anomaly-detection methods for IoT have been proposed in the past several years [3]. Among them, *classification-based* methods have attracted much more research attention in recent years due to their good performance in anomaly detection for large-scale data. The focus of this paper is thus on this category of methods. In a nutshell, classification-based anomaly-detection methods first train a classification model using historical data (i.e., training data) and test new data instances by the model to identify their classes (i.e., normal or anomaly).

In the category of classification-based method, sphere-based one-class support vector machine (SOCSVM) [4] is one of the most popular shallow model for anomaly detection. It is unsupervised, since it uses unsupervised learning to identify sparsely populated data areas and treats the data points that fall into such areas as anomalies. To be more specific, it uses a hypersphere to separate normal data and anomalies, as shown in Fig. 1(a). With training data, a SOCSVM model is built, and this model is then used to check whether or not a new incoming data is anomaly. This framework is shown in Fig. 2(a). Nevertheless, the model training phase requires to find the minimum radius hypersphere and compute the distances among instances in high-dimensional space. Since finding the minimum radius hypersphere involves quadratic programming, SOCSVM is



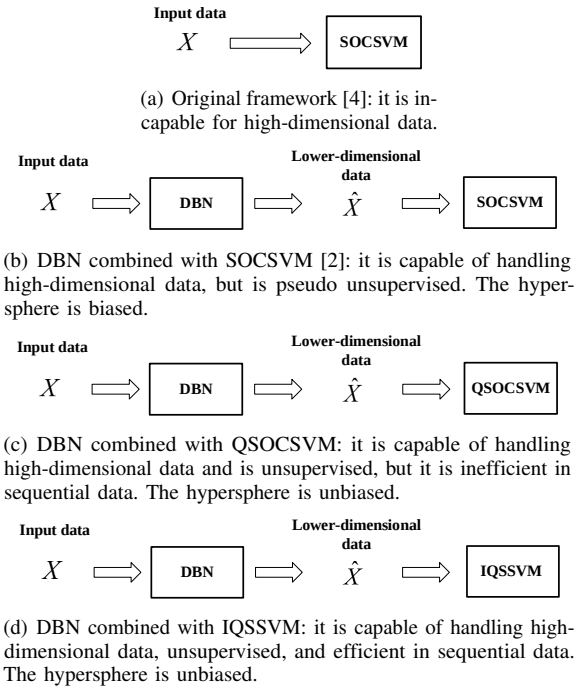
(a) SOCSVM (b) QSOCSVM

Fig. 1. The geometry of SOCSVM vs the geometry of QSOCSVM.

memory and time intensive. Therefore, **the framework in Fig. 2(a) is only suitable for relatively small scale and low-dimensional input data**.

To address the above problem, deep learning models have been adopted to improve the efficiency of SOCSVM. Ruff et al. [5] proposed a variant of SOCSVM named *deep support vector data description* (Deep SVDD) that extracts the common factors of variation of the data distribution by training a deep neural network. Although Deep SVDD improves the efficiency of data mapping in SOCSVM, it also needs to solve a quadratic programming problem. Andrews et al. [6] used autoencoders (AEs) to learn the features of the input data and fed the learned features to the SOCSVM. Erfani et al. [2] used a deep belief network (DBN) as a dimensionality-reduction tool to pre-process the input data, and fed the SOCSVM with the compressed data. The experimental results in [2] show that the hybrid model can achieve the same accuracy of autoencoders, but is 3 times faster in training time and 1000 times faster in testing time. This is because DBN not only reduces the dimension of input data to increase the processing speed of SOCSVM, but also amplifies the discrepancy between normal data and anomalies to improve the detection rate. The framework is shown in Fig. 2(b).

Nevertheless, the framework has two inherent problems caused by the ill marriage of DBN and SOCSVM. *First*, SOCSVM can only work well when the anomalies *surround* the normal data with roughly equal distance (as in Fig. 1(a)). When the anomalous data distribute on one side of the normal ones, SOCSVM would lead to a biased hypersphere. Unfortunately, the output data compressed by DBN nearly always result in anomalies, if any, that are one-sidedly distributed (see Sec. III-B3 for details). Thus, SOCSVM may not achieve its optimal performance when combined with DBN. *Second*, since the performance of SOCSVM depends heavily on the predefined kernel parameters, it is important to find the optimal parameters for a given set of training data. As shown later in the paper (Fig. 6 in particular), the optimal parameters may vary largely in the presence of one-sidedly distributed anomalies. Even worse, the optimization requires the help of the ground truth of the training instances, or otherwise it is hard to determine the boundary of the sphere. In other words, we need labelled data, which completely defeats the original purpose of using SOCSVM. Therefore, **the framework in Fig. 2(b) is greatly limited when applied in anomaly detection of high-dimensional sensing data.**



(a) Original framework [4]: it is incapable for high-dimensional data.

(b) DBN combined with SOCSVM [2]: it is capable of handling high-dimensional data, but is pseudo unsupervised. The hypersphere is biased.

(c) DBN combined with QSOCSVM: it is capable of handling high-dimensional data and is unsupervised, but it is inefficient in sequential data. The hypersphere is unbiased.

(d) DBN combined with IQSSVM: it is capable of handling high-dimensional data, unsupervised, and efficient in sequential data. The hypersphere is unbiased.

Fig. 2. Different frameworks for anomaly detection.

It seems natural to "improve" the framework in Fig. 2(b) by replacing SOCSVM with quarter-sphere-based one-class SVM (QSOCSVM) [7], which was proposed exactly to fix the problems caused by asymmetric distribution of anomalies [7]. The geometry of QSOCSVM is shown in Fig. 1(b). More importantly, QSOCSVM reduces the quadratic programming in SOCSVM to a linear programming problem, and has sharp sensitivity to one-sidedly distributed anomalies. Looking deeper, however, we find that QSOCSVM must transform the kernel function for the input data to obtain a unique solution for the quarter-sphere boundary (technical details disclosed in Sec. III-B2). It is not possible to directly calculate the *relative* location of the testing data to the quarter-sphere boundary through current kernel functions. Instead, calculating the *relative* location of the testing data to the original boundary requires to re-transform the kernel function to integrate the testing data, which means the model needs to be retrained. Therefore, QSOCSVM becomes extremely inefficient in the scenarios where new sensing data arrive sequentially. **The framework in Fig. 2(c) is not an effective improvement for Fig. 2(b) in anomaly detection of high-dimensional sensing data.**

To tackle all the above challenges, we design an improved QSOCSVM (named IQSSVM for short) and combine it with DBN. The new framework for anomaly detection is shown in Fig. 2(d). Based on the framework, we propose two anomaly detection approaches, which can (1) process high-dimensional data in an unsupervised manner, (2) detect anomalies accurately and efficiently, and (3) work for sequential data arrivals. In summary, this paper makes the following contributions:

- We design a new DBN-OCSVM-based model for anomaly detection with the ability to process high-dimensional data accurately in an unsupervised manner. The new model

not only reduces the quadratic programming problem in SOCSVM to a sorting problem, but also, to a certain extent, eliminates the dependence on kernel parameters. Furthermore, compared to the existing OCSVM-based model, the new model aligns the sphere much better when the anomalies are one-sidedly distributed and thus significantly improves both accuracy and efficiency.

- Based on our new model, we design both optimal and approximate methods for model training and testing. Until now, all existing OCSVM-based methods try to determine a hyperplane (or hypersphere) in the training stage, and classify the testing data by the relative location to the plane (or sphere). Such training/testing frame not only requires high time complexity (e.g., in the frameworks of Figs. 2(a) and 2(b)), but also cannot be applied to the models whose training data has been transformed (e.g., in the framework of Fig. 2(c)). We design a new form of training/testing method, which finds a *border support vector* (BSV) in the training set and then compares the testing data *directly* with the BSV by the properties proposed in Sec. IV. We further design approximate training and testing mechanisms which can dramatically reduce the complexity of finding the BSV while ensuring the accuracy.
- We confirm the benefits of our proposed methods in comparison with the state-of-the-art anomaly detection approaches for high-dimensional data with extensive trace-driven simulations. The experimental results on four real-world datasets show that high-dimensional anomalies can be detected accurately and efficiently with our methods in an unsupervised manner. Compared to existing solutions, our methods are resilient to small changes in kernel parameters and can improve the accuracy of the state-of-the-art by 12%. In addition, our approximate method reduces the computing time by three orders of magnitude and achieves nearly identical accuracy of the optimal method.

The remainder of this paper is organized as follows. First, the related work is reviewed in Sec. II. The background models, as well as the models' characteristics, are presented and analyzed in Sec. III. In Sec. IV we prove lemmas and propositions, which help develop a new form of OCSVM model. The anomaly detection methods based on the new model are presented in Sec. V. We evaluate our methods with four real-world datasets in Sec. VI. Finally, we conclude the paper in Sec. VII.

II. Related Work

Many anomaly-detection methods for IoT have been proposed in the past several years, which can be roughly divided into four categories [3]: clustering-based methods [8], [9], neighbor-based methods [10], [11], statistical-based methods [12], [13], [14], and classification-based methods [4], [15]. As this paper focuses on the methods based on classification, we only review the related work on this category of methods.

The *classification-based* methods use historical data (i.e., training data) to train the classification model and test the new collections (i.e., testing data) by the trained model. These models can be roughly divided into three categories: shallow-learning models, deep-learning models, and the hybrid models. One-class SVM (OCSVM) [4] is one of the most common shallow-learning models for anomaly detection. Huan et al [16] proposed an anomaly detection algorithm using model selection-based OCSVM. The method can select an optimal decision model for the support vector data description to avoid both under-fitting and over-fitting. Deng et al. [17] proposed a one-class support Tucker machine based on tensor Tucker decomposition to detect the anomalies. Since general OCSVM-based methods must solve a quadratic programming problem, they have relatively high computational complexities and cannot be applied directly to high-dimensional data. Rajasegarar et al. [18] proposed an anomaly detection method based on quarter-sphere one-class SVM (QSOCSVM) that converts the quadratic programming problem to a linear programming problem. However, QSOCSVM needs to retrain the model once a new testing data is coming. Therefore, it is quite inefficient for the streaming data.

Deep-learning-based methods utilize deep models, such as autoencoders (AEs) and deep neural networks (DNNs), to automatically learn the latent features of the input data. Aboelwafa et al. [19] utilized AEs to detect false data injection (FDI) attack in industrial IoT. Chalapathy et al. [20] developed a variant of DNN, named one class neural network (OC-NN), that combines DNN with one-class objective. Ruff et al. [5] proposed a variant of OCSVM that uses DNN as a kernel function to map the input data into a hypersphere. Although deep-learning-based methods can, to some extent, improve the efficiency of the traditional shallow-based methods, they still need a relative long time for model training and parameter tuning when the input data have high dimension [15].

The hybrid models use deep networks as feature extractors or dimensionality-reduction tools, and then input the learned features into traditional shallow models such as OCSVM to detect anomalies. Andrews et al. [6] proposed an anomaly detection method by combining sparse autoencoders (SAEs) with the sphere-based OCSVM (SOCSVM). Erfani et al. [2] designed a high-dimensional anomaly detection method by combining DBN with the SOCSVM. Experiments in [2] compared the performance of DBN and AEs, and showed that DBN performs much more efficiently than AEs for high-dimensional anomaly detection. However, as we have discussed before in the introduction, although DBN can effectively reduce the size of the input data as well as amplify the discrepancy between normal data and anomalies, the hypersphere of SOCSVM may suffer a non-negligible bias when it is directly fed with the reduced data.

In the following sections, we aim to design a new form of DBN-OCSVM-based hybird model and propose a novel anomaly detection approach based on the new model, which can efficiently process the high-dimensional sensing data and accurately detect the anomalies in real time.

III. Background Models and Their Properties

In this section, we introduce three models that are related to our solution and analyze their properties on anomaly detection

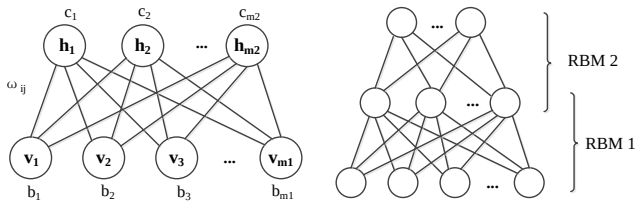
TABLE I
DEFINED SYMBOLS

Symbols	Meanings
v	The visible inputs of RBM
h	The the hidden units in RBM
w, b, c	The parameters in RBM model
$\mathbf{X}$	The training set
$\hat{\mathbf{X}}$	The lower-dimensional training set compressed by DBN
x_i	The i-th vector in $\mathbf{X}$
$\hat{x}_i$	The i-th vector in $\hat{\mathbf{X}}$
n	The number of vectors in $\mathbf{X}$
m	The number of dimensions in $\mathbf{X}$
r	The number of dimensions in $\hat{\mathbf{X}}$
R	The radius of SOCSVM or QSOCSVM
c	The center of SOCSVM
ν	The fraction of anomaly points in the training set
ξ_i	The slack variable for x_i
α_i	The Lagrange multiplier for x_i
x_s	An border support vector
$\Phi(\centerdot)$	A mapping function in SVM that maps a vector to a higher dimensional space
$\Phi'(\centerdot)$	The centered mapping function
$k(x, y)$	The RBF kernel function equaling $\Phi(x)\Phi(y)$
$k_c(x, y)$	The centered kernel function
σ	The parameter in RBF kernel
d_i	Equals $k_c(x_i, x_i)$
κ_i	Equals $\sum_{l=1}^{n} k(x_i, x_l)$
κ'_i	Equals $k(x_i, x_0)$
$\tilde{\kappa'}_i$	Equals $k(x_i, \bar{x})$
x_0	A vector in the input space which satisfies $\Phi(x_0) = \frac{1}{n}\sum_{i=1}^{n} \Phi(x_i)$
$\bar{x}$	The mean of the training vectors, i.e., $\frac{1}{n}\sum_{i=1}^{n} x_i$
$f(x)$	The optimal decision function for IQSSVM
$f'(x)$	The approximate decision function for IQSSVM

with our experimental data. All related symbols in this paper are listed in Tab. I.

A. Deep Belief Network (DBN)

1) Model: A deep belief network (DBN) is a probabilistic generative model proposed recently as a classifier and dimensionality-reduction tool [21]. A restricted Boltzmann machine (RBM) (shown in Fig. 3(a)) is a bipartite graph $G(v, h)$, where v represents visible inputs and h denotes the hidden units. A DBN (shown in Fig. 3(b)) is a multi-layer RBM that maps the input vector from an input space of dimension m to an output space of dimension r [22]. It can be used as a dimensionality-reduction tool when $r < m$. Parameters $\{w, b, c\}$ in the multi-layer RBM can be trained one layer at a time by a greedy layer-by-layer technique, and the output



(a) RBM (b) DBN

Fig. 3. Model architecture of RBM and DBN.

of the prior RBM can be considered as the input to the next RBM.

In an RBM network, an energy function of the visible and hidden units is defined as

$$E(v, h) = -\sum_i b_i v_i - \sum_j c_j h_j - \sum_i \sum_j w_{ij} v_i h_j. \quad (1)$$

The joint probability of the visible and hidden variables can be formulated through Eqn. (1) by

$$p(v, h) = \frac{e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}}. \quad (2)$$

The probability distribution for the visible vector v can be computed by summing over all possible hidden vectors,

$$p(v) = \frac{\sum_h e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}}. \quad (3)$$

Given a group of training vectors v, parameters (w, b, c) in the model can be learned by maximizing the probability $p(v)$. The derivative of the log probability of a training vector with respect to a weight can be computed by

$$\frac{\partial \log p(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}. \quad (4)$$

Here, the angle brackets indicate the expectations over the corresponding distributions specified by the subscript.

Because there are no direct connections between visible units in an RBM, the conditional probability of the hidden unit can be calculated by

$$p(h|v) = \prod_j p(h_j|v) = \prod_j sigm(c_j + \sum_i v_i w_{ij}). \quad (5)$$

Here, $sigm(x)$ is the logistic sigmoid function $\frac{1}{1+e^{-x}}$.

Similarly, because there are no direct connections between hidden units, the conditional probability of the visible unit is

$$p(v|h) = \prod_i p(v_i|h) = \prod_i sigm(b_j + \sum_j h_j w_{ij}). \quad (6)$$

It is quite easy to obtain an unbiased sample of $\langle v_i h_j \rangle_{data}$ based on Eqn. (5) and Eqn. (6). However, obtaining an unbiased sample of $\langle v_i h_j \rangle_{model}$ is intractable. Hinton [23] proposed the contrastive divergence (CD) method to estimate the expectation with k iterations of Gibbs sampling, and replaced $\langle v_i h_j \rangle_{model}$ with $\langle v_i h_j \rangle_{recon}$, where $\langle v_i h_j \rangle_{recon}$ denotes the expectation over the sample distribution, to reduce the computational complexity.

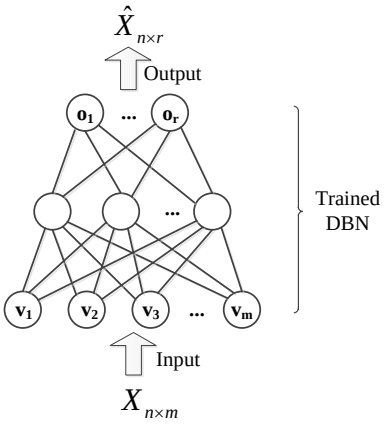


Fig. 4. Using DBN as a dimensionality reduction tool.

After training an RBM, another RBM can be stacked on its top. Furthermore, the hidden layer of the first RBM is used as the input layer of the second RBM.

As a dimensionality-reduction tool, DBN processes the input data from bottom to top in an unsupervised manner (as shown in Fig. 4). The numbers of units in the bottom and top layers denote the input and output dimensions of a group of data, respectively.

2) Properties of Dimensionality Reduction with DBN: We found that features extracted by DBN in our experiments always present a consistent trend: after dimension reduction, the features of almost all anomaly points tend to distribute on one side of normal records. We tested four different kinds of real-world data sets from the UCI Machine Learning Repository [24], including GAS (Gas sensor array drift) with 128 dimensions, HAR (Human Activity Recognition recognition using smartphones) with 561 dimensions, DSA (Daily and Sports Activities) with 315 dimensions, and FCT (Forest Cover Type) with 54 dimensions. Records in GAS and FCT are environmental data presenting periodic variation with time. Records in HAR and DSA are behavioral data. We randomly mixed 5% stochastic anomalies with those data sets; the details can be found in Sec. VI-A. For ease of visualization, we compress the four sets of data to 2 dimensions, but note that the similar trend can be observed should the datasets be compressed to $r\,(r > 2)$ dimensions.

We can see that the features extracted by DBN have clear divisions between anomalies and normal points in Fig. 5(a) and Fig. 5(b). Because the environmental data records in the two datasets have periodic pattern, and anomalies occur randomly without any pattern. In Fig. 5(c) and Fig. 5(d), some anomalies mix with the normal data since the behavioral data recorded in these two datasets do not present significant regular pattern. However, for all of the four datasets, the vast majority of anomalous points distribute on one side of normal points. We changed the structure of DBN (the number of hidden layers as well as the number of units on each layers), but such a distribution remained the same. Therefore, it is necessary to utilize the characteristic of such a distribution and design a classification model to identify the anomalies following such a distribution.

B. One-Class Support Vector Machine

One-class SVM (OCSVM) is an extension of the general SVM from the supervised training paradigm to unsupervised one. It finds the optimal hyperplane (or hypersphere) to separate alien data from the majority of the input data. OCSVM has been widely used for anomaly detection. Here we compare two well-known forms of OCSVM, sphere-based one-class SVM (SOCSVM) and quarter-sphere-based one-class SVM (QSOCSVM), and show that QSOCSVM is more suitable for anomaly detection when the dimensionality of input data has been reduced by DBN.

1) Sphere-based One-Class SVM: SOCSVM, proposed by Tax et al. [4], models the tightest hypersphere that can encompass the majority of the data points, allowing a few points outside the sphere.

The fitting problem can be formulated as follows:

$$\begin{aligned} \min_{R,\xi,c} \quad & R^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \\ s.t. \quad & ||\Phi(x_i) - c||^2 \le R^2 + \xi_i, \\ & \xi_i \ge 0, i = 1, 2, \cdots, n. \end{aligned} \tag{7}$$

Here, $\Phi(\centerdot)$ is a non-linear function that maps vectors to a higher dimensional space (i.e., the so-called feature space in SVM). ξ_i is a non-negative slack variable for x_i that allows for some points to lie outside the hypersphere, and ν is the fraction of anomaly points predefined by users to make the tradeoff between the size of the hypersphere and the fraction of points outside the sphere. c and R are the center and radius of the hypersphere, respectively. We can define the anomaly for SOCSVM as follows.

Definition: If the distance between the data point x_i and the center of the hypersphere is larger than the radius R, x_i is an anomaly.

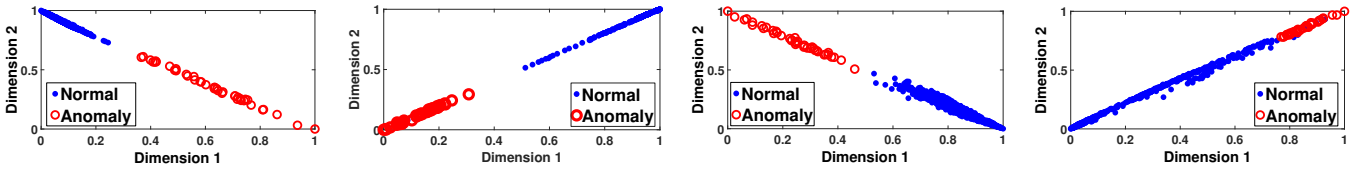
Practically, Eqn. (7) can be solved through its dual problem Eqn. (8), by introducing the Lagrange multipliers $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$.

$$\begin{aligned} \min_{\alpha} \quad & \sum_{ij}\alpha_i\alpha_j k(x_i, x_j) - \sum_{i}\alpha_i k(x_i, x_i) \\ s.t. \quad & \sum_{i}\alpha_i = 1, \\ & 0 \le \alpha_i \le \frac{1}{\nu n}, i = 1, 2, \cdots, n. \end{aligned} \tag{8}$$

$k(x_i, x_j)$ is the kernel function between x_i and x_j, equaling the inner products of $\Phi(x_i)$ and $\Phi(x_j)$ in the feature space. The *radial basis function* (RBF) kernel is one of the most popular kernels that can work well in most SVM models. The RBF kernel on two samples x_i and x_j can be defined as

$$k(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \tag{9}$$

Here, $||x_i - x_j||$ is the Euclidean distance between x_i and x_j, σ is the parameter that controls the scale of mapping in the feature space. In most cases, the performance of classification depends heavily on the assignment of σ. Thus, it is important for RBF-based models to search for the optimal value of σ in order to ensure their classification accuracy. Unless otherwise



(a) FCT (b) GAS (c) DSA (d) HAR

Fig. 5. Distributions of features extracted by DBN.

noted, the symbol $k(x, y)$ by default denotes the RBF kernel function of vectors x and y in the rest of the paper.

After obtaining α by solving Eqn. (8), we can classify the training data into three classes: i) $\alpha_i = 0$ indicates that the corresponding point x_i lies inside the hypersphere, i.e., x_i is a normal record; ii) $0 < \alpha_i < \frac{1}{\nu n}$ denotes that x_i lies on the border of the hypersphere, i.e., x_i is a *border support vector* (BSV); iii) $\alpha_i = \frac{1}{\nu n}$ means that x_i is outside the hypersphere, i.e., x_i is an anomaly. The border support vectors and anomalous points comprise the set of *support vectors* (SVs) in the model.

The decision function to judge a new record x can be computed as

$$f(x) = sgn(R^2 - ||\Phi(x) - c||^2). \tag{10}$$

$f(x) = -1$ if x_i is an anomaly, otherwise, x_i is normal. c in Eqn. (10) is the center of the hypersphere, which can be computed by $c = \sum_i \alpha_i x_i$. R is the radius of the hypersphere, i.e., the distance between BSV x_s and the center c. It can be computed as

$$\begin{aligned} R^2 =& ||\Phi(x_s) - c||^2 \\ =& k(x_s, x_s) - 2\sum_i \alpha_i k(x_s, x_i) + \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j). \end{aligned} \tag{11}$$

Similarly, $||\Phi(x) - c||^2$ can also be computed through the same formulation as Eqn. (11) by replacing x_s with a new record x.

2) Quarter-sphere-based One-Class SVM: SOCSVM works well only when anomalies surround normal points in roughly equal distance (as shown in Fig. 1(a)). Laskov et al. [7] extended the idea of SOCSVM to handle the scenarios where the anomalies distribute at one side of normal data, by fixing the center of the sphere at the origin, i.e., $c = 0$ (as shown in Fig. 1(b). The original fitting problem in Eqn. (7) is transformed to the following:

$$\begin{aligned} \min_{R,\xi,c} \quad & R^2 + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i \\ s.t. \quad & ||\Phi(x_i)||^2 \leq R^2 + \xi_i, \\ & \xi_i \geq 0, i = 1, 2, \cdots, n. \end{aligned} \tag{12}$$

We can define the anomaly for QSOCSVM as follows.

Definition: If the distance between the data point x_i and the origin of the quarter sphere is larger than the radius R, x_i is an anomaly.

The optimization problem can also be converted into its dual form

$$\begin{aligned} \min_{\alpha} \quad & -\sum_i \alpha_i k(x_i, x_i) \\ s.t. \quad & \sum_i \alpha_i = 1, \\ & 0 \leq \alpha_i \leq \frac{1}{\nu n}, i = 1, 2, \cdots, n. \end{aligned} \tag{13}$$

In Eqn. (13), the quadratic programming problem in Eqn. (8) has been replaced by a linear programming problem, which significantly reduces the computational complexity. However, solving Eqn. (13) may encounter the problem that when using a distance-based kernel, such as the RBF kernel [25], the inner products of the vector with itself (i.e., $k(x_i, x_i)$) are always equal to 1 for all data vectors. Thus, there could be no meaningful solution to Eqn. (13). Fortunately, this problem can be solved by centering the training points $\Phi(x_i)$ in the feature space [7]. In other words, we translate the mapped points in the local coordinate system anchored at the center of all mapped data by subtracting the mean from all mapped values. Therefore, the new coordinate for x_i in the feature space can be computed by

$$\Phi'(x_i) = \Phi(x_i) - \frac{1}{n}\sum_{i=1}^{n} \Phi(x_i). \tag{14}$$

The centered kernel matrix, denoted by k_c, among the mapped points can be easily expressed by the original kernel matrix. Let $\mathbf{k}$ denote the kernel matrix $\mathbf{k}(x, x) = [k(x_i, x_j)]_{n\times n}$, where $x = \{x_1, x_2, \ldots, x_n\}$ is the set of all instances. The centered kernel matrix $\mathbf{k}_c = [k_c(x_i, x_j)]_{n\times n}$ can be computed by

$$\mathbf{k}_c = \mathbf{k} - \mathbf{1}_n\mathbf{k} - \mathbf{k}\mathbf{1}_n + \mathbf{1}_n\mathbf{k}\mathbf{1}_n, \tag{15}$$

where $\mathbf{1}_n$ is an $n \times n$ matrix with all values equal to $\frac{1}{n}$. After centering the kernel matrix in the feature space, $k(x_i, x_i)$ in Eqn. (13) will be replaced by $k_c(x_i, x_i)$ which are no longer the same for different x_i. The linear programming problem can be solved easily.

3) Comparisons: Both SOCSVM and QSOCSVM are capable of detecting anomalies in an unsupervised manner. Comparing SOCSVM and QSOCSVM, we have the following conclusions:

- (a) QSOCSVM works much better than SOCSVM when the anomalies are distributed on one side of the normal points.
- (b) The training complexity of SOCSVM is higher than QSOCSVM.

- (c) SOCSVM is more efficient than QSOCSVM when testing steaming data. In the testing phase, QSOCSVM needs to re-transform the kernel function to integrate the testing data.

To explain (a), Fig. 6 plots the isograms of FCT records (in the space compressed with DBN), under both QSOCSVM and SOCSVM models. The points in the isogram contour have $+1$ state (i.e., identified as normal), and points outside the contour have -1 state (i.e., identified as anomaly). Although we vary σ in the RBF kernel, there are always some anomalies (marked in red) that are marked mistakenly as normal points by SOCSVM. In contrast, anomalies can be separated accurately by QSOCSVM no matter what the value σ is.

This is because SOCSVM computes the center of the sphere by $c = \sum_i \alpha_i x_i$, and only BSVs as well as anomalies have non-zero α. Therefore, the center is determined based on BSVs and anomalies, and there is a real possibility that one-sidedly distributed anomalies may lead to a seriously biased center (as shown in Fig. 6(a), 6(b), and 6(c)).

QSOCSVM fixes the origin at the center of all mapped data in the feature space and models the quarter sphere to encompass the majority of the data near the origin. In other words, the closer a mapped point lies to the center, the less likely that the point is an anomaly. Because the center of the quarter sphere is determined by *all* mapped data, and the anomalies can only be minority, one-sidedly distributed anomalies will not cause much bias toward the center. Furthermore, as the kernel function has been centered by Eqn. (15), the impact of σ on the performance QSOCSVM is minor.

To explain (b), in the training phase, SOCSVM needs to solve a quadratic programming problem (Eqn. (8)), whereas QSOCSVM only requires to solve a linear programming problem (Eqn. (13)).

To explain (c), in the testing phase, SOCSVM can use the kernel function to directly calculate the distance between the new testing data and the center c (with Eqn. (11)). Nevertheless, since QSOCSVM determines the sphere center with the average of all involving points, it needs to re-calculate the sphere center in order to check the relative distance between the test data and the sphere center, i.e., it needs to use Eqn. (15), together with all training data and the incoming test data point[1], to determine whether or not the test data point is normal [7]. Our experimental results in Tab. IV also show that QSOCSVM is quite inefficient to deal with sequential data.

The main differences between SOCSVM and QSOCSVM are listed in Tab. II.

TABLE II
THE MAIN DIFFERENCES AMONG THE THREE SVM MODELS

Models	SOCSVM	QSOCSVM	IQSSVM
Aligning shape	Whole sphere	Quarter sphere	Quarter sphere
Well-performed scenario	Symmetrically -distributed anomalies	Asymmetrically -distributed anomalies	Asymmetrically -distributed anomalies
Training complexity	Quadratic programming	Linear programming	Sorting
Training method	Learn R and c by Eqn. (8)	Solve Eqn. (13) and identify anomalies by α_i	Find a BSV by our optimal or approximate methods
Testing method	Identify anomalies by Eqn. (10)		Identify anomalies by Eqn. (18) (optimal) or by Eqn. (20) (approximate)
Efficient in testing sequential data	Yes	No	Yes

IV. IMPROVEMENT ON QUARTER-SPHERE-BASED SVM FOR REAL-TIME ANOMALY DETECTION

In this section, we develop an improved QSOCSVM model (named IQSSVM for short), and design two decision functions (optimal function and approximate function) to test new incoming data in real time. In IQSSVM, the training and testing methods are different (as shown in Tab. II) from the traditional SVM-based models which learn the parameters for hyperplane (or hypersphere) in the training stage and classify the testing data by determining its relative location to the hyperplane (or hypersphere). In our model, we first quickly find one border support vector (BSV) in the training stage, and determine the status of the testing data by comparing it with the BSV using the properties proposed in this section.

A. Find the Border Support Vector (BSV)

To find the BSV efficiently, we first introduce Proposition 1 from [26] and then propose Proposition 2 to find the BSV using a sorting method.

Let d_i denote the inner product $\langle \Phi'(x_i), \Phi'(x_i) \rangle$ (which can also be represented by the centered kernel function $k_c(x_i, x_i)$). For all input vectors, we sort their d_i by *descending* order and obtain a sequence $\{d_{i'} | 1 \le i' \le n\}$. Replacing $k_c(x_i, x_i)$ in Eqn. (13) by the ordered sequence $\{d_{i'} | 1 \le i' \le n\}$, the problem can be transformed to

$$\begin{aligned} \min_{\alpha} \quad & -\sum_i \alpha_i d_{i'} \\ s.t. \quad & \sum_i \alpha_i = 1, \\ & 0 \le \alpha_i \le \frac{1}{\nu n}, i = 1, 2, \cdots, n. \end{aligned} \tag{16}$$

Proposition 1: The square of radius R for the quarter sphere obtained by solving Eqn. (16) equals the $(j'+1)$-th inner product $d_{j'+1}$, where $j' = \lfloor \nu n \rfloor$. It can be formulated by

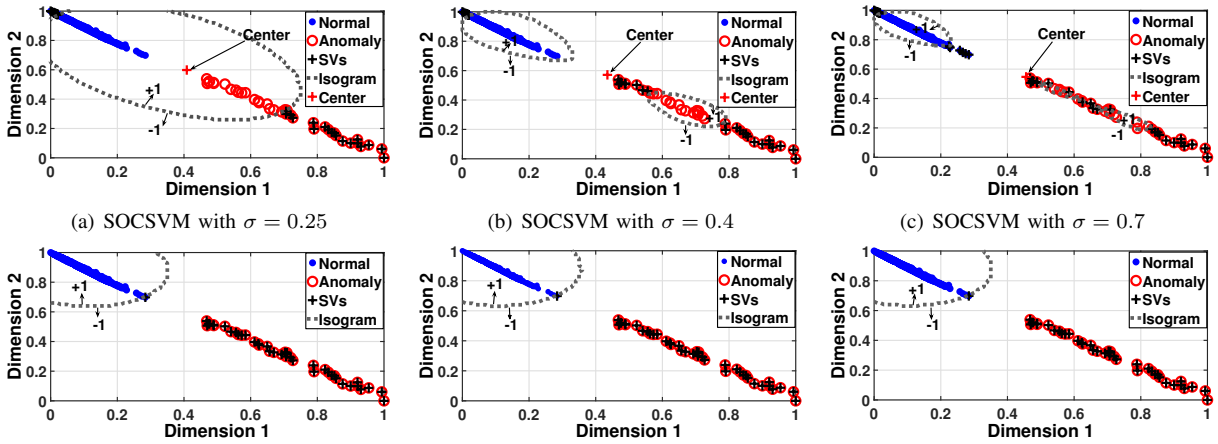
$$R^2 = d_{j'+1}, \quad j' = \lfloor \nu n \rfloor \tag{17}$$

Proof: Refer to [26].

Lemma 1: The $(j'+1)$-th training record $x_{j'+1}$ in $k_c(x_{j'+1}, x_{j'+1})$, where $j' = \lfloor \nu n \rfloor$, is a BSV.

Proof: See the Appendix.

[1]Note that the matrices in Eqn. (15) become $(n+1) \times (n+1)$ assuming that the training data have n points.



(a) SOCSVM with $\sigma = 0.25$ (b) SOCSVM with $\sigma = 0.4$ (c) SOCSVM with $\sigma = 0.7$

(d) QSOCSVM with $\sigma = 0.25$ (e) QSOCSVM with $\sigma = 0.4$ (f) QSOCSVM with $\sigma = 0.7$

Fig. 6. Comparison of two SVM-based methods on the data with one-sidedly distributed anomalies.

Next, we show that the BSV can be found in a more efficient way.

Let κ_i denote the summation of the original kernels $\sum_{l=1}^{n} k(x_i, x_l)$. Sorting the summations of all the training data in *ascending* order, we can obtain an ordered sequence $\{\kappa_{i'} | 1 \leq i' \leq n\}$.

Proposition 2: The $(j'+1)$-th training record $x_{j'+1}$, $j' = \lfloor \nu n \rfloor$, in the ordered sequence $\{\kappa_{i'} | 1 \leq i' \leq n\}$ is a BSV.

Proof: See the Appendix.

B. Decision Function for IQSSVM

Proposition 3: The new incoming data x is an anomaly if and only if $\sum_{i=1}^{n} k(x, x_i) < \kappa_{j'+1}$, where $j' = \lfloor \nu n \rfloor$.

Proof: See the Appendix.

According to Proposition 3, for any testing vector x, we can design an optimal decision function $f(x)$ as follows:

$$f(x) = sgn(\sum_{i=1}^{n} k(x, x_i) - \kappa_{j'+1}). \tag{18}$$

$f(x) = -1$ indicates that x is an anomaly; otherwise, x is normal. Hence, our optimal model training and testing method can be briefly described as follows.

Optimal method: i) Model training: sort $\{\kappa_i | 1 \leq i \leq n\}$ in ascending order and find the BSV in the training data set by Proposition 2; ii) Testing: compare the two summations of kernels as shown in Eqn. (18).

However, when the training set is large, it still costs high computational time to calculate the summations of kernels.

C. Approximate Training and Testing for IQSSVM

To make the model training and testing more efficient, we propose Lemma 2 to demonstrate that the summation of kernels is equivalent to a much simpler formulation.

Lemma 2: Suppose x_0 is a vector in the input space, and it satisfies $\Phi(x_0) = \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i)$ in the feature space. Then

$$\sum_{i=1}^{n} k(x, x_i) = nk(x, x_0). \tag{19}$$

Proof: See the Appendix.

According to Lemma 2, the calculation in Proposition 2 as well as in Proposition 3 can be significantly simplified, respectively, as follows:

Let κ'_i denote the original kernel $k(x_0, x_i)$. Sorting $\{\kappa'_i | 1 \leq i \leq n\}$ in an ascending order, we can obtain a sequence $\{\kappa'_{i'} | 1 \leq i' \leq n\}$.

Proposition 4: The $(j'+1)$-th training record $x_{j'+1}$, $j' = \lfloor \nu n \rfloor$, in the sequence $\{\kappa'_{i'} | 1 \leq i' \leq n\}$ is a BSV.

Proof: See the Appendix.

Proposition 5: The new incoming data x is an anomaly if and only if $k(x, x_0) < k(x_{j'+1}, x_0)$, where $j' = \lfloor \nu n \rfloor$.

Proof: See the Appendix.

Based on Proposition 4 and Proposition 5, the training/testing processes only need to calculate the kernels between x_0 and each training/testing data, whereas Proposition 2 and Proposition 3 require the calculations of the kernels across all training and testing vectors. Thus, the time and space complexity can be reduced considerably.

Unfortunately, it is not easy to exactly find the point x_0 in the input space, especially when the mapping function is non-linear. Nevertheless, we can use the mean value $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ to approximate the point x_0, and improve the efficiency of our optimal method by the following approximate method.

Approximate method: i) Model training: use $\tilde{\kappa'}_i$ to denote $k(x_i, \bar{x})$, and sort $\{\tilde{\kappa'}_i | 1 \leq i \leq n\}$ in ascending order to obtain the approximate BSV $x_{j'+1}$, where $j' = \lfloor \nu n \rfloor$; ii) Testing: the approximate decision function $f'(x)$ is designed as follows:

$$f'(x) = sgn(k(x, \bar{x}) - \tilde{\kappa'}_{j'+1}). \tag{20}$$

$f'(x) = -1$ indicates that x is an anomaly; otherwise, x is normal.

To evaluate the effect of approximating x_0 by $\bar{x}$, we compare the performance differences between our approximate method and our optimal method under different parameters in Section VI. The experimental results show that the optimal method and the approximate method achieve nearly the same accuracy in their respective best cases.

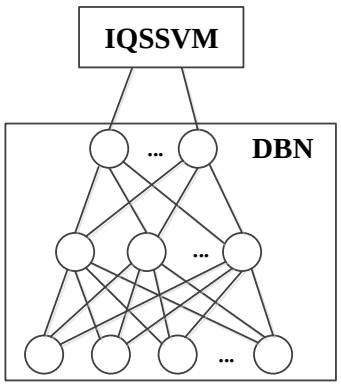


Fig. 7. The DBN-IQSSVM model.

V. Efficient Anomaly Detection for High-dimensional Sensing Data

In this section, we first integrate DBN and IQSSVM (termed as the DBN-IQSSVM model) and then propose our optimal and approximate anomaly detection algorithms, respectively, based on the new model.

A. Modeling

The unsupervised learning-based architecture is presented in Fig. 7, where the IQSSVM is stacked above the output layer of DBN.

Fig. 8(a) and 8(b) illustrate the optimal training and testing steps for the DBN-IQSSVM model, respectively. In the optimal model training stage, training data $\mathbf{X}$ with m dimensions are input into the bottom layer of DBN to learn the parameters $\{w, b, c\}$ in each RBM. Meanwhile, DBN extracts the features of the training data, $\hat{\mathbf{X}}$ with r dimensions, on the top layer and passes the features to IQSSVM to find the BSV $\hat{x}_{j'+1}$ based on Proposition 2. IQSSVM outputs $\kappa_{j'+1}$, the summation of kernels between $\hat{x}_{j'+1}$ and all training data, as the threshold for testing. In the optimal testing stage, the dimension of the testing data x is reduced by the trained DBN, and the summation of the kernels between the testing data and all training data is then calculated. Finally, the decision function Eqn. (18) is used to judge whether or not the testing data is an anomaly.

We also design approximate training and testing mechanisms, as shown in Fig. 8(c) and 8(d), respectively, to reduce the time complexity. In the approximate training stage, the IQSSVM first calculates the mean value $\bar{x}$ of the training data and then computes the kernels between each training data and $\bar{x}$, instead of computing the kernels across all pairs of training data. In the approximate testing stage, IQSSVM only computes the kernel of the compressed testing data $\hat{x}$ and $\bar{x}$ and compares it with $\tilde{\kappa'}_{j'+1}$ to judge whether or not the testing data x is an anomaly.

B. Algorithm Details

To simplify presentation, we write the pseudo-code for training and testing together, but would like to note that in practice, the training is done only once with the training dataset $\mathbf{X}$, but the testing is for each incoming data x_t in a streaming manner.

Algorithm 1: Optimal Anomaly Detection Algorithm based on DBN-IQSSVM

Input: $\mathbf{X}$, x_t, ν, σ
Output: y_t

1 $[w, b, c, \hat{\mathbf{X}}] \leftarrow trainDBN(\mathbf{X})$ ▷ Training
2 $\mathbf{K} \leftarrow kernel(\hat{\mathbf{X}}, \hat{\mathbf{X}})$
3 $\kappa \leftarrow sum(\mathbf{K}, column)$
4 $[\hat{x}_{\lfloor \nu n \rfloor + 1}, \kappa_{thd}] \leftarrow sort(\kappa, ascend)$
5 $\hat{x}_t \leftarrow DBN(w, b, c, x_t)$ ▷ Testing
6 $\mathbf{K}_t \leftarrow kernel(\hat{x}_t, \hat{\mathbf{X}})$
7 $\kappa_t \leftarrow sum(\mathbf{K}_t)$
8 **if** $\kappa_t < \kappa_{thd}$ **then**
9 $\quad y_t \leftarrow -1$
10 **else**
11 $\quad y_t \leftarrow 1$
12 **return** y_t

The pseudo-code of the optimal anomaly detection algorithm based on DBN-IQSSVM is presented in Alg. 1. It inputs the training data set $\mathbf{X}$, testing data x_t, the predefined anomaly fraction ν and the kernel parameter σ, and outputs the label of the testing data y_t. y_t has binary values 1 and -1, representing normal and anomaly, respectively. In the training stage, Alg. 1 first trains the DBN model to optimize the parameters w, b and c, while extracting features from the training set $\mathbf{X}$ (line 1). Then it computes the kernel matrix $\mathbf{K}$ across all training vectors (line 2), and sums its columns to obtain a sequence κ (line 3). According to Proposition 2, Alg. 1 sorts the sequence κ in ascending order (line 4), and finds the BSV, $\hat{x}_{\lfloor \nu n \rfloor + 1}$, as well as the threshold κ_{thd}, where $\kappa_{thd} = \kappa_{\lfloor \nu n \rfloor + 1}$ (line 4). In the testing stage, Alg. 1 first extracts the features of testing data x_t with the trained DBN (line 5). Then it computes the kernels on $\hat{x}_t$ and all training vectors $\mathbf{X}$ (line 6) and sums the kernels to obtain κ_t (line 7). Finally, it compares κ_t with the threshold κ_{thd}. If κ_t is smaller than κ_{thd}, x_t will be considered as an anomaly, and y_t will be assigned to -1 (line 9). Otherwise, y_t will be set to 1, indicating x_t is a normal data (line 11).

The most time-consuming step in Alg. 1 is computing the kernels across all pairs of training vectors (line 2). Thus the computational complexity for Alg. 1 is $O(n^2)$, where n is the number of the training vectors.

The pseudo-code of the approximate anomaly detection algorithm based on DBN-IQSSVM is presented in Alg. 2. Its inputs and outputs are the same with Alg. 1. In the training stage, Alg. 2 first calculates the mean value of $\hat{\mathbf{X}}$ after the dimension reduction by DBN (line 2). Then it computes the kernels between $\bar{x}$ and all training vectors to form a sequence $\tilde{\kappa'}$ (line 3). Alg. 2 sorts the sequence $\tilde{\kappa'}$ in ascending order (line 4), and then finds the approximate BSV as well as the corresponding threshold $\tilde{\kappa'}_{thd}$ according to Proposition 4 (line 4). In the testing stage, Alg. 2 computes the kernel between $\bar{x}$ and $\hat{x}_t$ (i.e., $\tilde{\kappa'}_t$), and then compares it with the threshold $\tilde{\kappa'}_{thd}$ (line 6). If $\tilde{\kappa'}_t$ is smaller than $\tilde{\kappa'}_{thd}$, x_t will

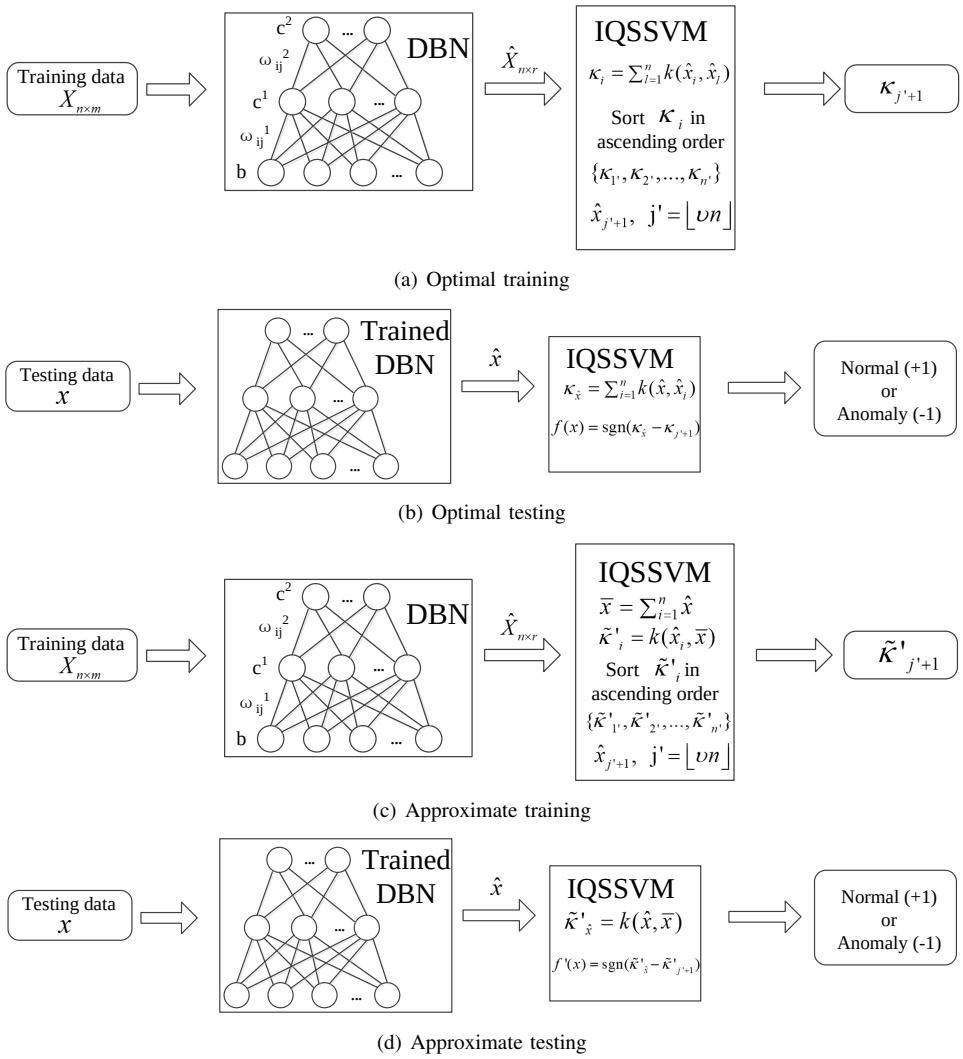


Fig. 8. Model training and testing for DBN-IQSSVM.

Algorithm 2: Approximate Anomaly Detection Algorithm based on DBN-IQSSVM

Input: $\mathbf{X}$, x_t, ν, σ
Output: y_t
1 $[w, b, c, \hat{\mathbf{X}}] \leftarrow trainDBN(\mathbf{X})$ ▷ Training
2 $\bar{x} \leftarrow mean(\hat{\mathbf{X}})$
3 $\tilde{\kappa'} \leftarrow kernel(\hat{\mathbf{X}}, \bar{x})$
4 $[\hat{x}_{\lfloor \nu n \rfloor + 1}, \tilde{\kappa'}_{thd}] \leftarrow sort(\tilde{\kappa'}, ascend)$
5 $\hat{x}_t \leftarrow DBN(w, b, c, x_t)$ ▷ Testing
6 $\tilde{\kappa'}_t \leftarrow kernel(\hat{x}_t, \bar{x})$
7 **if** $\tilde{\kappa'}_t < \tilde{\kappa'}_{thd}$ **then**
8 $\quad y_t \leftarrow -1$
9 **else**
10 $\quad y_t \leftarrow 1$
11 **return** y_t

be considered as an anomaly, and y_t will be assigned to -1 (line 8). Otherwise, y_t will be set to 1 (line 10).

The most time-consuming steps in Alg. 2 are the DBN training (line 1) and the computing of kernels between $\bar{x}$ and all training vectors (line 3). Both of them require the time complexity of $O(n)$. Therefore, the time complexity of Alg. 2 is $O(n)$, where n is the number of the training vectors.

VI. Performance Evaluation

A. Setup

1) Datasets: In our experiments, we used four real data sets from UCI [24] to verify the performance of the proposed method. The detailed information (including the full names and the characteristics) of the data sets are given in Sec. III-A2, and the basic statistics of these datasets are listed in Tab. III. "Min" means the minimum value in each data set, and "Max" means the maximum value in each data set.

For all datasets, we selected $500 \sim 2000$ consecutive records, the first 80% records for training and the remaining 20% records for testing. We normalized all datasets to the range $[0, 1]$, and then randomly selected 5% and 10% of the records, respectively, from the training sets and testing sets, to make

TABLE III
BASIC STATISTICS OF THE DATASETS

Datasets	FCT	GAS	DSA	HAR
Dimension	54	128	315	561
Min	-134	-2.95×10^2	-35.27	-2.72
Max	6.89×10^3	1.91×10^5	41.04	5.07
Mean	1.55×10^2	2.35×10^3	0.18	-0.50

them "anomalies"[2]: For each selected record, we randomly replaced $q\%$ of its dimensions' values by random values drawn from the uniform distribution function $U(0, 1)$. We name q as the *fraction of anomalous attributes*, and varied q from 20% to 100% to verify the performance of the methods under different degrees of anomalies.

2) Model Settings: The parameters of DBN (including the number of epochs, units on the hidden layers, and learning rate [27]) were set based on the best performance of all algorithms (actually, the structure of the DBN does not noticeably affect the performance of anomaly detection according to our experimental results). All the evaluated methods use RBF kernel in their training and testing phase, as RBF outperforms other kernel functions (such as linear kernel and polynomial kernel function) in our experiments. As the parameter σ in the RBF kernel as well as the pre-defined anomaly ratio ν may influence the performance of the models, we vary both parameters to observe the performance of the algorithms under different settings of these parameters.

B. Evaluation Metrics

We compared our optimal algorithm (denoted by O-DBN-IQSSVM) as well as the approximate algorithm (denoted by A-DBN-IQSSVM) with QSOCSVM-based algorithm (denoted by DBN-QSOCSVM) and the SOCSVM-based algorithm (denoted by DBN-SOCSVM) [2] from two aspects.

Accuracy: In the experimental results, the receiver operating characteristic (ROC) curve and the corresponding area under the curve (AUC) are used to measure the overall accuracy of all algorithms. False positive rate (FPR) and true positive rate (TPR) are also presented to reflect the accuracy of the algorithms from different angles. The formulations of FPR and TPR are as follows:

$$FPR = \frac{|\hat{\mathbf{A}} \cap \mathbf{N}|}{|\mathbf{N}|}, \quad TPR = \frac{|\hat{\mathbf{A}} \cap \mathbf{A}|}{|\mathbf{A}|}. \tag{21}$$

Where $\mathbf{A}$ and $\hat{\mathbf{A}}$ represent the sets of true anomalies and inferred anomalies, respectively; $\mathbf{N}$ denotes the set of normal instances.

Speed: We record the training/testing times of the algorithms over different scales of datasets. Because all algorithms require the same amount time for the DBN training and testing, we only record the time required for their OCSVM training and testing. All experiments were implemented in MATLAB (R2019b) and run on a server with dual Intel Core i5 CPU at 2.5 GHz and 32 GB RAM.

Note that, DBN-QSOCSVM is also our own suggested improvement over DBN-SOCSVM, which can be treated roughly as the same as our optimal solution O-DBN-IQSSVM (except that O-DBN-IQSSVM simplifies the calculations and improves DBN-SOCSVM with the capability of processing sequential data). We compare our optimal and approximate algorithms with DBN-QSOCSVM to show the improvements on processing sequential data.

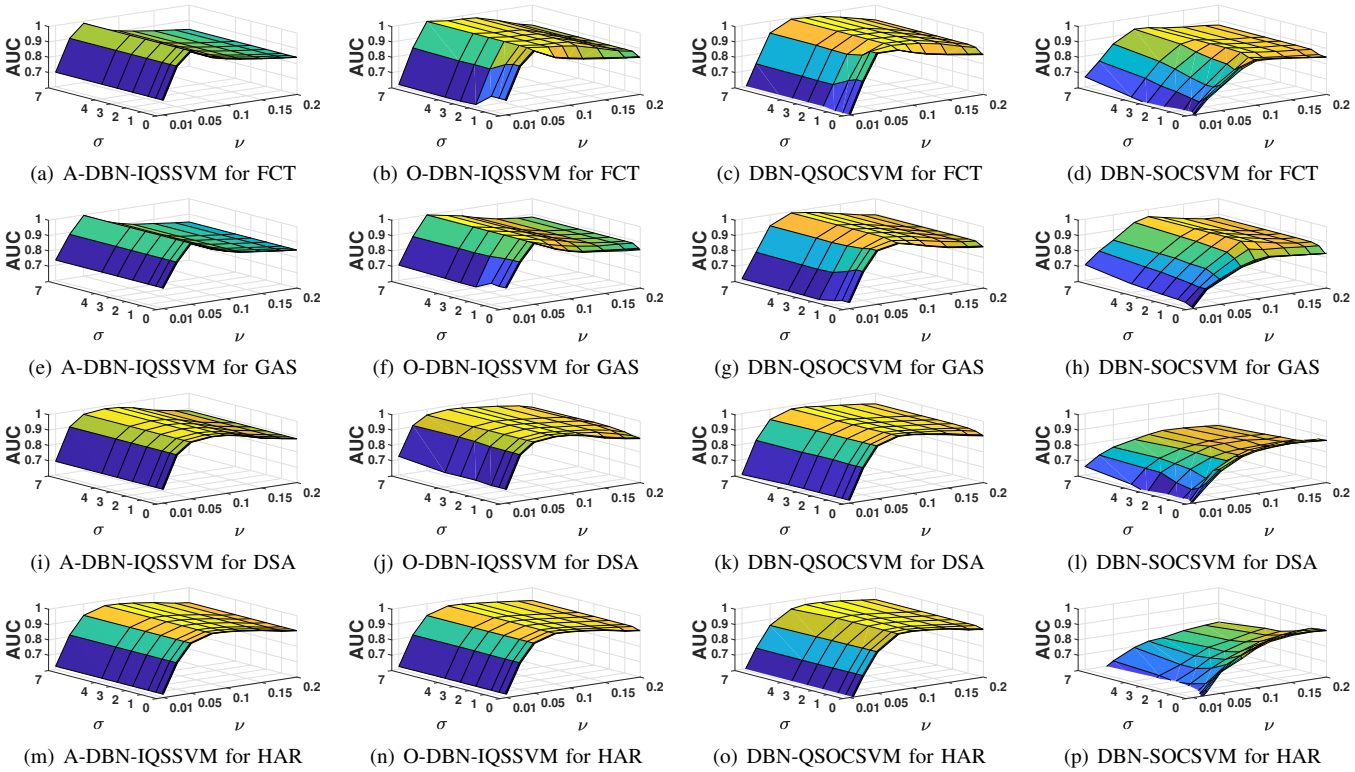
The results presented in following sections were averaged over 30 runs.

C. Results

Fig. 9 plots the variations of AUCs in different kernel parameter σ and different anomaly rate ν. From the figures, we can see that (1) the AUC of A-DBN-IQSSVM always remains stable when the value of σ varies, whereas the AUCs of O-DBN-IQSSVM and DBN-QSOCSVM fluctuate slightly in some cases (in FCT and GAS datasets). DBN-SOCSVM changes greatly in all cases. This demonstrates that it is important for DBN-SOCSVM to optimize the kernel parameter in the training stage, whereas the optimization of the parameter is not quite necessary for our methods; (2) the deviation of A-DBN-IQSSVM from O-DBN-IQSSVM depends on the deviation of $\bar{x}$ from x_0. A-DBN-IQSSVM presents stable performance because it only needs to compute the kernel function once to obtain $k(x_i, \bar{x})$ for each instance, whereas O-DBN-IQSSVM needs to compute the kernels among all instances to obtain the summation $\sum_{l=1}^{n} k(x_i, x_l)$ (i.e., $nk(x_i, x_0)$) for each instance. However, despite the variation of the parameters, the *best* performances of the two methods are almost the same; (3) the performance of all four methods heavily depends on the anomaly rate ν. For our A-DBN-IQSSVM and O-DBN-IQSSVM, the optimal value of ν is always very close to the actual anomaly rate of the training set (i.e., 0.05). And for DBN-SOCSVM, the optimal value of ν presents quite different values for different data sets. For example, in FCT and GAS, the optimal ν for DBN-SOCSVM is 0.1, whereas in DSA and HAR, the optimal ν is 0.2. As the training and testing datasets are input together to DBN-QSOCSVM, the optimal value of ν is between 0.05 (the anomaly rate of the training set) and 0.1 (the anomaly rate of the testing set).

Fig. 10 compares the accuracy of the algorithms over different fractions of anomalous attributes q. The kernel parameter σ, and the anomaly rate ν are assigned according to the best performances of the four algorithms. The figures show that (1) as the fraction of anomalous attributes increases, all algorithms generally present better accuracy. This is because the greater the fraction of anomalous attributes, the more evident the anomalies; (2) O-DBN-IQSSVM and A-DBN-IQSSVM have almost the same accuracy, and both of them have much higher overall accuracy (AUCs) and much lower FPRs than DBN-SOCSVM. In particular, for cases where anomalies have only 20% anomalous attributes (i.e., the anomalies are quite obscure), our method still have averagely 85% AUC and 2.6% false positive rate, whereas the average AUC of DBN-SOCSVM

[2]To avoid human bias, we replace the selected data with random numbers. In this case, "anomalies" does not necessarily mean extreme values but rather mean that the values have been (randomly) modified.



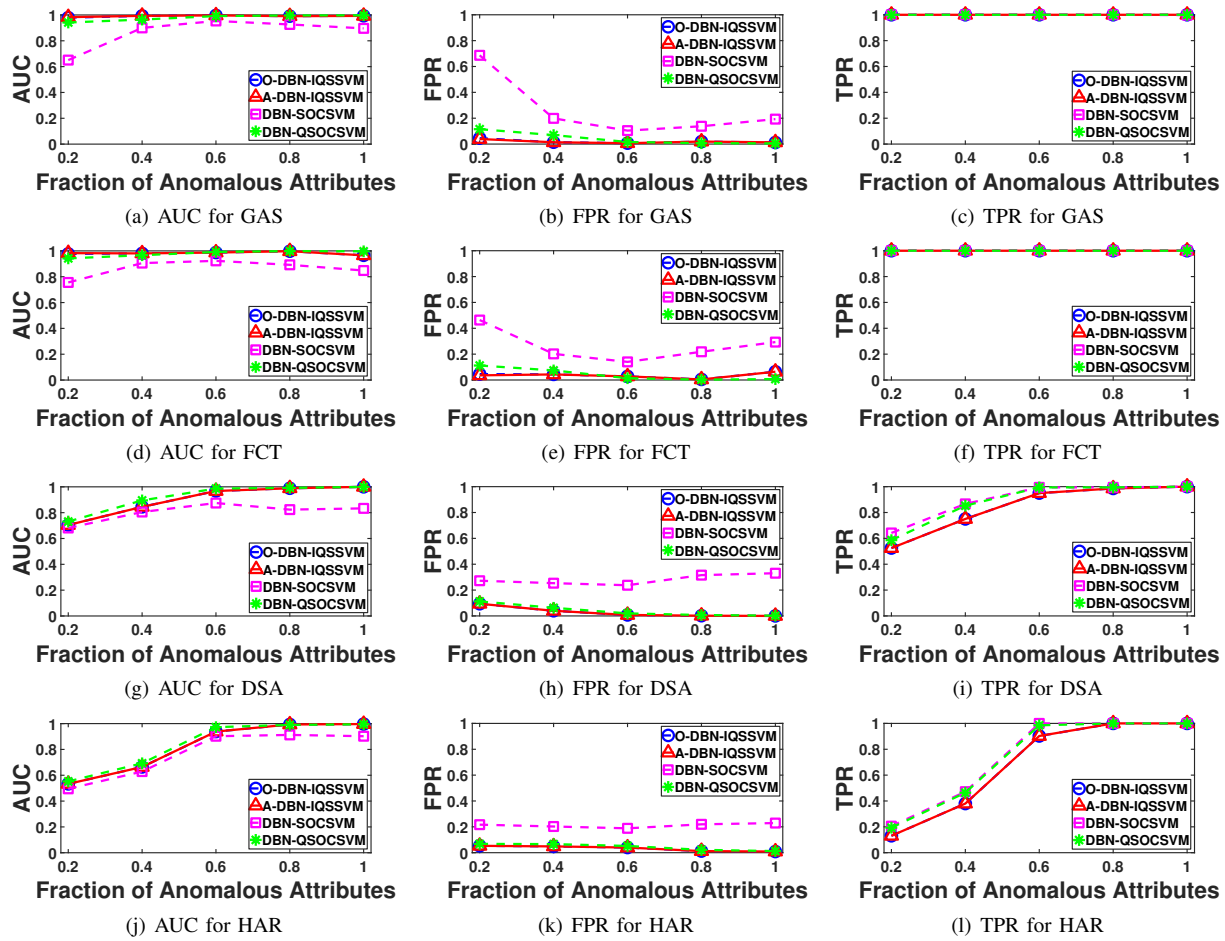
(a) A-DBN-IQSSVM for FCT (b) O-DBN-IQSSVM for FCT (c) DBN-QSOCSVM for FCT (d) DBN-SOCSVM for FCT

(e) A-DBN-IQSSVM for GAS (f) O-DBN-IQSSVM for GAS (g) DBN-QSOCSVM for GAS (h) DBN-SOCSVM for GAS

(i) A-DBN-IQSSVM for DSA (j) O-DBN-IQSSVM for DSA (k) DBN-QSOCSVM for DSA (l) DBN-SOCSVM for DSA

(m) A-DBN-IQSSVM for HAR (n) O-DBN-IQSSVM for HAR (o) DBN-QSOCSVM for HAR (p) DBN-SOCSVM for HAR

Fig. 9. AUCs of the algorithms over different σ and ν.



(a) AUC for GAS (b) FPR for GAS (c) TPR for GAS

(d) AUC for FCT (e) FPR for FCT (f) TPR for FCT

(g) AUC for DSA (h) FPR for DSA (i) TPR for DSA

(j) AUC for HAR (k) FPR for HAR (l) TPR for HAR

Fig. 10. Accuracy of the algorithms over different fraction of anomalous attributes.

is 64% and the FPR is 39%; (3) The relatively higher TPR and FPR of DBN-SOCSVM indicate that this method tends to find much more anomalies than the ground truth; (4) The overall accuracy of DBN-QSOCSVM is a little bit higher than that of O-DBN-IQSSVM and A-DBN-IQSSVM in some cases. Because the DBN-QSOCSVM model is formed based on both training and testing data, whereas the models of O-DBN-IQSSVM and A-DBN-IQSSVM are built just on the training data; (5) For the data sets with periodic variations and relatively low dimensions (i.e., GAS and FCT), our methods can perform accurately even when the fraction of anomalous attributes is 20%. For DSA and HAR, our methods can achieve a high accuracy when the fraction of anomalous attributes exceeds 60%. As the dimension of DSA is lower than that of HAR, our methods perform relatively better in DSA than in HAR.

Tab. IV records the averaged training and testing times of the algorithms (in millisecond). The table shows that A-IQSSVM runs dramatically faster than the other three algorithms. Roughly speaking, it can accelerate the computing speed by three orders of magnitude. SOCSVM requires the most computing time, as it requires to solve a quadratic programming problem in the training step. O-IQSSVM can save the computing time by about $30\% \sim 45\%$. Note that the time of SOCSVM recorded in Tab. IV excludes the time for tuning up parameters, so the actual time for SOCSVM would be even higher. As QSOCSVM solves a linear programming problem to identify anomalies, the total time of training and testing is relatively shorter than the total time of SOCSVM. However, as the training and testing processes cannot be separately executed in QSOCSVM, it will always require the total time of training and testing to test the new data. Therefore, compared with the other three algorithms, QSOCSVM will cost much more computing time when the testing data arrives sequentially.

D. Summary

In summary, the performance of DBN-SOCSVM depends heavily on both σ and ν. It is important for DBN-SOCSVM to search for the optimal values for the two parameters (with the help of labelled training data) before applying them to anomaly detection. This also indicates that the training of DBN-SOCSVM may become supervised (if we want to find the optimal parameters) and requires much more computing time. For A-DBN-IQSSVM, O-DBN-IQSSVM and DBN-QSOCSVM, they only need to estimate an appropriate anomaly rate ν according to the actual environment, then all of them will present their optimal performance. When in the best performance of each method, the overall accuracy of our methods can improve DBN-SOCSVM by 12% and the overall false positive rate can be reduced by 85%. Our approximate algorithm has almost the same accuracy with the optimal algorithm, but has extremely low running time. A-DBN-IQSSVM and O-DBN-IQSSVM just sacrifice no more than 2% overall accuracy of DBN-QSOCSVM, but significantly improve its capability by processing sequential testing data.

VII. Conclusions

This paper presents a novel OCSVM-based approach of anomaly detection for high-dimensional sensing data. Based on the analysis of the data compressed by DBN, a new anomaly detection model as well as a new form of training and testing mechanism based on the model are proposed. Different from the state-of-the-art OCSVM-based anomaly detection technique, DBN-SOCSVM to be specific, the training of our model is designed to find a BSV in the training set, and the testing is to compare the new incoming data with the BSV, using the properties proved in this paper. We designed both the optimal and approximate methods for training and testing of our model. Anomaly detection based on approximate training and testing can reduce the computational complexity from quadratic to linear. With the experiments over four real-world high-dimensional datasets, we compared our methods with the state-of-art anomaly detection method. Experimental results confirm the following three advantages of our methods over DBN-SOCSVM: i) robust performance (i.e., resilient to parameter changes), ii) higher overall accuracy, and iii) faster running speed. Finally, the proposed new form of training and testing methods can be extended from anomaly detection to broader applications that use OCSVM-based techniques.

Acknowledgment

This work is supported by Anhui Provincial Natural Science Foundation (No. 2008085MF203), the National Natural Science Foundation of China (No. 61402013, No. 31671589, No. 61828202), the Open Foundation of State Key Laboratory of Networking and Switching Technology (SKLNST-2018-1-10), and the Natural Sciences and Engineering Research Council of Canada (No. RGPIN-2018-03896)

References

[1] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[2] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[3] N. Shahid, I. H. Naqvi, and S. B. Qaisar, "Characteristics and classification of outlier detection techniques for wireless sensor networks in harsh environments: a survey," *Artificial Intelligence Review*, vol. 43, no. 2, pp. 193–228, 2015.

[4] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[5] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*, 2018, pp. 4393–4402.

[6] J. T. Andrews, E. J. Morton, and L. D. Griffin, "Detecting anomalous data using auto-encoders," *International Journal of Machine Learning and Computing*, vol. 6, no. 1, p. 21, 2016.

[7] P. Laskov, C. Schäfer, I. Kotenko, and K.-R. Müller, "Intrusion detection in unlabeled data with quarter-sphere support vector machines," *Praxis der Informationsverarbeitung und Kommunikation*, vol. 27, no. 4, pp. 228–236, 2004.

[8] T. Yu, X. Wang, and A. Shami, "Recursive principal component analysis-based data outlier detection and sensor data aggregation in iot systems," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2207–2216, 2017.

[9] R. K. Gunupudi, M. Nimmala, N. Gugulothu, and S. R. Gali, "Clapp: A self constructing feature clustering approach for anomaly detection," *Future Generation Computer Systems*, vol. 74, pp. 417–429, 2017.

[10] J. W. Branch, C. Giannella, B. Szymanski, R. Wolff, and H. Kargupta, "In-network outlier detection in wireless sensor networks," *Knowledge and information systems*, vol. 34, no. 1, pp. 23–54, 2013.

TABLE IV
THE COMPUTING TIMES OF THE ALGORITHMS (MILLISECONDS)

Datasets	Scales	A-IQSSVM		O-IQSSVM		QSOCSVM	SOCSVM	
		Train	Test	Train	Test	Train&test	Train	Test
GAS	500	0.35	0.11	40.59	10.23	132.73	134.39	73.12
	1000	0.43	0.13	162.67	42.50	609.41	696.10	272.67
	2000	0.60	0.20	598.38	146.94	3323.03	3177.51	1077.72
FCT	500	0.27	0.08	36.67	9.24	114.39	109.55	62.04
	1000	0.42	0.12	157.04	39.27	573.53	519.77	244.93
	2000	0.65	0.22	629.25	174.54	3742.32	2782.13	1012.89
DSA	500	0.26	0.08	37.11	8.81	114.78	109.79	59.06
	1000	0.39	0.12	146.82	35.20	543.40	515.90	215.01
	2000	0.68	0.23	708.53	188.66	4092.56	3113.79	1109.67
HAR	500	0.24	0.08	34.81	8.94	109.81	112.89	56.78
	1000	0.37	0.11	139.76	35.30	533.01	593.52	206.73
	2000	0.58	0.18	570.18	140.95	3236.56	3276.67	799.41

[11] R. Huang, X. Qiu, and L. Rui, "Simple random sampling-based probe station selection for fault detection in wireless sensor networks," *Sensors*, vol. 11, no. 3, pp. 3117–3134, 2011.

[12] Y. Zhang, N. A. Hamm, N. Meratnia, A. Stein, M. Van De Voort, and P. J. Havinga, "Statistics-based outlier detection for wireless sensor networks," *International Journal of Geographical Information Science*, vol. 26, no. 8, pp. 1373–1392, 2012.

[13] O. Ghorbel, W. Ayedi, H. Snoussi, and M. Abid, "Fast and efficient outlier detection method in wireless sensor networks," *IEEE sensors journal*, vol. 15, no. 6, pp. 3403–3411, 2015.

[14] F. Riahi and O. Schulte, "Model-based outlier detection for object-relational data," in *2015 IEEE Symposium Series on Computational Intelligence*. IEEE, 2015, pp. 1590–1598.

[15] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[16] Z. Huan, C. Wei, and G.-H. Li, "Outlier detection in wireless sensor networks using model selection-based support vector data descriptions," *Sensors*, vol. 18, no. 12, p. 4328, 2018.

[17] X. Deng, P. Jiang, X. Peng, and C. Mi, "An intelligent outlier detection method with one class support tucker machine and genetic algorithm toward big sensor data in internet of things," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4672–4683, 2019.

[18] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. C. Bezdek, "Quarter sphere based distributed anomaly detection in wireless sensor networks," in *IEEE International Conference on Communications*, 2007.

[19] M. M. Aboelwafa, K. G. Seddik, M. H. Eldefrawy, Y. Gadallah, and M. Gidlund, "A machine learning-based technique for false data injection attacks detection in industrial iot," *IEEE Internet of Things Journal*, 2020.

[20] R. Chalapathy, A. K. Menon, and S. Chawla, "Anomaly detection using one-class neural networks," *arXiv preprint arXiv:1802.06360*, 2018.

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

[22] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[23] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[24] "The uci machine learning repository." [Online]. Available: http://archive.ics.uci.edu/ml/datasets.html

[25] B. Scholkopf, K.-K. Sung, C. J. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing support vector machines with gaussian kernels to radial basis function classifiers," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2758–2765, 1997.

[26] P. Cheng and M. Zhu, "Lightweight anomaly detection for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 11, no. 8, p. 653232, 2015.

[27] G. E. Hinton, "A practical guide to training restricted boltzmann machines," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 599–619.

Yan Qiao received the Ph.D. degree in computer science from Beijing University of Posts and Telecommunications in 2012. She was a Post-Doctoral Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. She is currently a Visiting Scholar in the Department of Computer Science, University of Victoria, Canada. She is also an associate professor with the School of Computer Science and Information Engineering, Hefei University of Technology, China. She now focuses on network monitoring, anomaly detection for Internet of Things.

Kui Wu received the B.Sc. and M.Sc. degrees in computer science from Wuhan University, China, in 1990 and 1993, respectively, and the Ph.D. degree in computing science from the University of Alberta, Canada, in 2002. He joined the Department of Computer Science, University of Victoria, Canada, in 2002, where he is currently a Full Professor. His research interests include network performance analysis, mobile and wireless networks, network performance evaluation, and anomaly detection for Internet of Things.

Peng Jin is now a postgraduate in School of Information and Computer, Anhui Agriculture University, Cina. His research areas include machine learning and anomaly detection.