Unsupervised Anomaly Detection With LSTM Neural Networks

Tolga Ergen and Suleyman Serdar Kozat, *Senior Member, IEEE*

***Abstract*— We investigate anomaly detection in an unsupervised framework and introduce long short-term memory (LSTM) neural network-based algorithms. In particular, given variable length data sequences, we first pass these sequences through our LSTM-based structure and obtain fixed-length sequences. We then find a decision function for our anomaly detectors based on the one-class support vector machines (OC-SVMs) and support vector data description (SVDD) algorithms. As the first time in the literature, we jointly train and optimize the parameters of the LSTM architecture and the OC-SVM (or SVDD) algorithm using highly effective gradient and quadratic programming-based training methods. To apply the gradient-based training method, we modify the original objective criteria of the OC-SVM and SVDD algorithms, where we prove the convergence of the modified objective criteria to the original criteria. We also provide extensions of our unsupervised formulation to the semisupervised and fully supervised frameworks. Thus, we obtain anomaly detection algorithms that can process variable length data sequences while providing high performance, especially for time series data. Our approach is generic so that we also apply this approach to the gated recurrent unit (GRU) architecture by directly replacing our LSTM-based structure with the GRU-based structure. In our experiments, we illustrate significant performance gains achieved by our algorithms with respect to the conventional methods.**

***Index Terms*— Anomaly detection, gated recurrent unit (GRU), long short-term memory (LSTM), support vector data description (SVDD), support vector machines (SVMs).**

I. Introduction

A. Preliminaries

ANOMALY detection [1] has attracted significant interest in the contemporary learning literature due to its applications in a wide range of engineering problems [2]–[4]. In this article, we study the variable length anomaly detection problem in an unsupervised framework, where we seek to find a function to decide whether or not each unlabeled variable length sequence in a given data set is anomalous. Note that although this problem is extensively studied in the literature and there exist different methods, e.g., supervised (or semisupervised) methods, that require the knowledge of data labels, we employ an unsupervised method due to the high cost of obtaining accurate labels in most real-life applications [1]. However, we also extend our derivations to the semisupervised and fully supervised frameworks for completeness.

In the current literature, a common and widely used approach for anomaly detection is to find a decision function that defines the model of normality [1], [5]. In this approach, one first defines a certain decision function and then optimizes the parameters of this function with respect to a predefined objective criterion, e.g., the one-class support vector machines (OC-SVMs) and support vector data description (SVDD) algorithms [6], [7]. However, algorithms based on this approach examine time series data over a sufficiently long time window to achieve an acceptable performance [1], [8], [9]. Thus, their performances significantly depend on the length of this time window so that this approach requires careful selection for the length of the time window to provide a satisfactory performance [8], [10]. To enhance performance for time series data, Fisher kernel and generative models are introduced [11]–[14]. However, the main drawback of the Fisher kernel model is that it requires the inversion of the Fisher information matrix, which has a high computational complexity [11], [12]. On the other hand, in order to obtain an adequate performance from a generative model such as a hidden Markov model (HMM), one should carefully select its structural parameters, e.g., the number of states and topology of the model [13], [14]. Furthermore, the type of training algorithm has also considerable effects on the performance of generative models, which limits their usage in real-life applications [14]. Thus, neural networks, especially recurrent neural networks (RNNs)-based approaches are introduced, thanks to their inherent memory structure that can store "time" or "state" information [1], [15]. However, since the basic RNN architecture does not have control structures (gates) to regulate the amount of information to be stored [16], [17], a more advanced RNN architecture with several control structures, i.e., the long short-term memory (LSTM) network, is introduced [17], [18]. However, neural networks-based approaches cannot directly optimize an objective criterion for anomaly detection due to the lack of data labels in an unsupervised framework [1], [19]. Hence, they first predict a sequence from its past samples and then determine whether the sequence is an anomaly or not based on the prediction error, i.e., an anomaly is an event, which cannot be predicted from the past nominal data [1]. Thus, they require a probabilistic model for the prediction error and a threshold on the probabilistic model to detect anomalies, which results in challenging optimization problems and restricts their performance accordingly [1], [19], [20]. Furthermore, both the

Manuscript received May 30, 2018; revised December 25, 2018; accepted August 14, 2019. Date of publication September 13, 2019; date of current version August 4, 2020. This work was supported by Tubitak Project under Grant 117E153. *(Corresponding author: Tolga Ergen.)*

T. Ergen is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: ergen@stanford.edu).

S. S. Kozat is with the Department of Electrical and Electronics Engineering, Bilkent University, 06800 Ankara, Turkey (e-mail: kozat@ee.bilkent.edu.tr).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2935975

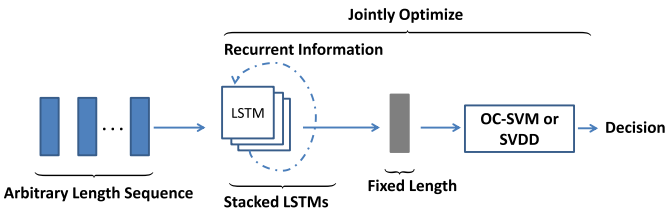

Fig. 1. Overall structure of our anomaly detection approach.

common and neural networks-based approaches can process only fixed-length vector sequences, which significantly limits their usage in real-life applications [1].

In order to circumvent these issues, we introduce novel LSTM-based anomaly detection algorithms for variable length data sequences. In particular, we first pass variable length data sequences through an LSTM-based structure to obtain fixed-length representations. We then apply our OC-SVM [6]-based algorithm and SVDD [7]-based algorithm for detecting anomalies in the extracted fixed-length vectors as illustrated in Fig. 1. Unlike the previous approaches in the literature [1], we jointly train the parameters of the LSTM architecture and the OC-SVM (or SVDD) formulation to maximize the detection performance. For this joint optimization, we propose two different training methods, i.e., a quadratic programming-based algorithm and gradient-based algorithm, where the merits of each different approach are detailed in the article. For our gradient-based training method, we modify the original OC-SVM and SVDD formulations and then provide the convergence results of the modified formulations to the original ones. Thus, instead of following the prediction-based approaches [1], [19], [20] in the current literature, we define proper objective functions for anomaly detection using the LSTM architecture and optimize the parameters of the LSTM architecture via these well-defined objective functions. Hence, our anomaly detection algorithms are able to process variable length sequences and provide high performance for time series data. Furthermore, since we introduce a generic approach in the sense that it can be applied to any RNN architecture, we also apply our approach to the gated recurrent unit (GRU) architecture [21], i.e., an advanced RNN architecture as the LSTM architecture, in our simulations. Through an extensive set of experiments, we demonstrate significant performance gains with respect to the conventional methods [6], [7], [10].

B. Prior Art and Comparisons

Several different methods have been introduced for the anomaly detection problem [1]. Among these methods, the OC-SVM [6] and SVDD [7] algorithms are generally employed due their high performance in real-life applications [22]. However, these algorithms provide inadequate performance for time series data due to their inability to capture time dependencies [8], [9]. In order to improve the performances of these algorithms for time series data, in [9], Zhang *et al.* convert time series data into a set of vectors by replicating each sample so that they obtain 2-D vector sequences. However, even though they obtain 2-D vector sequences, the second dimension does not provide additional information such that this approach still provides inadequate performance for time series data [8]. As another approach, the OC-SVM-based method in [8] acquires a set of vectors from time series data by unfolding the data into a phase space using a time delay embedding process [23]. More specifically, for a certain sample, they create an E dimensional vector by using the previous $E-1$ samples along with the sample itself [8]. However, in order to obtain satisfactory performance from this approach, the dimensionality, i.e., E, should be carefully tuned, which restricts its usage in real-life applications [24]. On the other hand, even though LSTM-based algorithms provide high performance for time series data, we have to solve highly complex optimization problems to get adequate performance [1]. For example, the LSTM-based anomaly detection algorithms in [10] and [25] first predict time series data and then fit a multivariate Gaussian distribution to the error, where they also select a threshold for this distribution. Here, they allocate a different set of sequences to learn the parameters of the distribution and threshold via the maximum likelihood estimation technique [10], [25]. Thus, the conventional LSTM-based approaches require careful selection of several additional parameters, which significantly degrades their performance in real-life [1], [10]. Furthermore, both the OC-SVM- (or SVDD) and LSTM-based methods are able to process only fixed-length sequences [6], [7], [10]. To circumvent these issues, we introduce generic LSTM-based anomaly detectors for variable length data sequences, where we jointly train the parameters of the LSTM architecture and the OC-SVM (or SVDD) formulation via a predefined objective function. Therefore, we not only obtain high performance for time series data but also enjoy joint and effective optimization of the parameters with respect to a well-defined objective function.

C. Contributions

Our main contributions are as follows.

1) We introduce LSTM-based anomaly detection algorithms in an unsupervised framework, where we also extend our derivations to the semisupervised and fully supervised frameworks.
2) As the first time in the literature, we jointly train the parameters of the LSTM architecture and the OC-SVM (or SVDD) formulation via a well-defined objective function, where we introduce two different joint optimization methods. For our gradient-based joint optimization method, we modify the OC-SVM and SVDD formulations and then prove the convergence of the modified formulations to the original ones.
3) Thanks to our LSTM-based structure, the introduced methods are able to process variable length data sequences. In addition, unlike the conventional methods [6], [7], our methods effectively detect anomalies in time series data without requiring any preprocessing.
4) Through an extensive set of experiments involving real and simulated data, we illustrate significant performance improvements achieved by our algorithms with respect to the conventional methods [6], [7], [10].

Moreover, since our approach is generic, we also apply it to the recently proposed GRU architecture [21] in our experiments.

D. Organization of the Article

The organization of this article is as follows. In Section II, we first describe the variable length anomaly detection problem and then introduce our LSTM-based structure. In Section III-A, we introduce anomaly detection algorithms based on the OC-SVM formulation, where we also propose two different joint training methods in order to learn the LSTM and SVM parameters. The merits of each different approach are also detailed. In a similar manner, we introduce anomaly detection algorithms based on the SVDD formulation and provide two different joint training methods to learn the parameters in Section III-B. In Section IV, we demonstrate performance improvements over several real-life data sets. Thanks to our generic approach, we also introduce GRU-based anomaly detection algorithms. Finally, we provide concluding remarks in Section V.

II. Model and Problem Description

In this article, all vectors are column vectors and denoted by boldface lower case letters. Matrices are represented by boldface uppercase letters. For a vector $\boldsymbol{a}$, $\boldsymbol{a}^T$ is its ordinary transpose and $||\boldsymbol{a}|| = \sqrt{\boldsymbol{a}^T\boldsymbol{a}}$ is the ℓ_2-norm. The time index is given as subscript, e.g., $\boldsymbol{a}_i$ is the i^{th} vector. Here, $\mathbf{1}$ (and $\mathbf{0}$) is a vector of all ones (and zeros) and $\boldsymbol{I}$ represents the identity matrix, where the sizes are understood from the context.

We observe data sequences $\{\boldsymbol{X}_i\}_{i=1}^n$, i.e., defined as

$$\boldsymbol{X}_i = [\boldsymbol{x}_{i,1}\ \boldsymbol{x}_{i,2} \dots \boldsymbol{x}_{i,d_i}]$$

where $\boldsymbol{x}_{i,j} \in \mathbb{R}^p$, $\forall j \in \{1, 2, \dots d_i\}$ and $d_i \in \mathbb{Z}^+$ is the number of columns in $\boldsymbol{X}_i$, which can vary with respect to i. Here, we assume that the bulk of the observed sequences are normal and the remaining sequences are anomalous. Our aim is to find a scoring (or decision) function to determine whether $\boldsymbol{X}_i$ is anomalous or not based on the observed data, where $+1$ and -1 represent the outputs of the desired scoring function for nominal and anomalous data, respectively. As an example application for this framework, in host-based intrusion detection [1], the system handles operating system call traces, where the data consist of system calls that are generated by users or programs. All traces contain system calls that belong to the same alphabet; however, the co-occurrence of the system calls is the key issue in detecting anomalies [1]. For different programs, these system calls are executed in different sequences, where the length of the sequence may vary for each program. Binary encoding of a sample set of call sequences can be $\boldsymbol{X}_1 = 101011$, $\boldsymbol{X}_2 = 1010$, and $\boldsymbol{X}_3 = 1011001$ for $n = 3$ case [1]. After observing such a set of call sequences, our aim is to find a scoring function that successfully distinguishes the anomalous call sequences from the normal sequences.

In order to find a scoring function $l(\cdot)$ such that

$$l(\boldsymbol{X}_i) = \begin{cases} -1, & \text{if } \boldsymbol{X}_i \text{ is anomalous} \\ +1, & \text{otherwise} \end{cases}$$

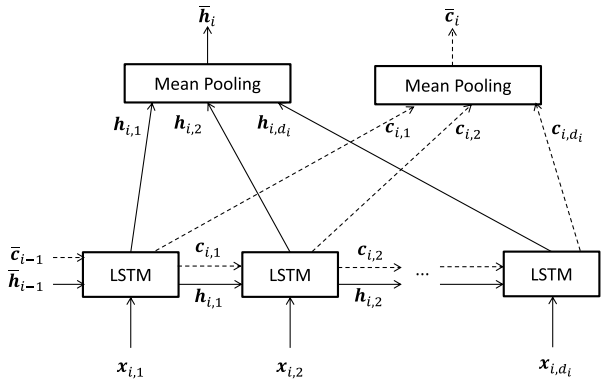

Fig. 2. Our LSTM-based structure for obtaining fixed-length sequences. Note that each LSTM block has the same parameters; however, we represent them as separate blocks for presentation simplicity.

one can use the OC-SVM algorithm [6] to find a hyperplane that separates the anomalies from the normal data or the SVDD algorithm [7] to find a hypersphere enclosing the normal data while leaving the anomalies outside the hypersphere. However, these algorithms can only process fixed-length sequences. Hence, we use the LSTM architecture [18] to obtain a fixed-length vector representation for each $\boldsymbol{X}_i$ as we previously introduced in [26]. Although there exist several different versions of LSTM architecture, we use the most widely employed architecture, i.e., the LSTM architecture without peephole connections [17]. We first feed $\boldsymbol{X}_i$ to the LSTM architecture as demonstrated in Fig. 2, where the internal LSTM equations are as follows [18]:

$$\boldsymbol{z}_{i,j} = g(\boldsymbol{W}^{(z)}\boldsymbol{x}_{i,j} + \boldsymbol{R}^{(z)}\boldsymbol{h}_{i,j-1} + \boldsymbol{b}^{(z)}) \quad (1)$$

$$\boldsymbol{s}_{i,j} = \sigma(\boldsymbol{W}^{(s)}\boldsymbol{x}_{i,j} + \boldsymbol{R}^{(s)}\boldsymbol{h}_{i,j-1} + \boldsymbol{b}^{(s)}) \quad (2)$$

$$\boldsymbol{f}_{i,j} = \sigma(\boldsymbol{W}^{(f)}\boldsymbol{x}_{i,j} + \boldsymbol{R}^{(f)}\boldsymbol{h}_{i,j-1} + \boldsymbol{b}^{(f)}) \quad (3)$$

$$\boldsymbol{c}_{i,j} = \boldsymbol{s}_{i,j} \odot \boldsymbol{z}_{i,j} + \boldsymbol{f}_{i,j} \odot \boldsymbol{c}_{i,j-1} \quad (4)$$

$$\boldsymbol{o}_{i,j} = \sigma(\boldsymbol{W}^{(o)}\boldsymbol{x}_{i,j} + \boldsymbol{R}^{(o)}\boldsymbol{h}_{i,j-1} + \boldsymbol{b}^{(o)}) \quad (5)$$

$$\boldsymbol{h}_{i,j} = \boldsymbol{o}_{i,j} \odot g(\boldsymbol{c}_{i,j}) \quad (6)$$

where $\boldsymbol{c}_{i,j} \in \mathbb{R}^m$ is the state vector, $\boldsymbol{x}_{i,j} \in \mathbb{R}^p$ is the input vector, and $\boldsymbol{h}_{i,j} \in \mathbb{R}^m$ is the output vector for the j^{th} LSTM unit in Fig. 2. In addition, $\boldsymbol{s}_{i,j}$, $\boldsymbol{f}_{i,j}$, and $\boldsymbol{o}_{i,j}$ is the input, forget, and output gates, respectively. Here, $g(\cdot)$ is set to the hyperbolic tangent function, i.e., tanh, and applies to input vectors pointwise. Similarly, $\sigma(\cdot)$ is set to the sigmoid function. $\odot$ is the operation for elementwise multiplication of two same-sized vectors. Furthermore, $\boldsymbol{W}^{(\cdot)}$, $\boldsymbol{R}^{(\cdot)}$, and $\boldsymbol{b}^{(\cdot)}$ are the parameters of the LSTM architecture, where the size of each is selected according to the dimensionality of the input and output vectors. Basically, in our LSTM architecture, $\boldsymbol{c}_{i,j-1}$ represents the cell state of the network from the previous LSTM block. This cell state provides an information flow between consecutive LSTM blocks. For the LSTM architecture, it is important to determine how much information we should keep in the cell state. Thus, in order to determine the amount of information to be kept, we use $\boldsymbol{f}_{i,j}$, which outputs a number between 0 and 1, and scales the cell state in (4). The next step is to determine how much new information

we should learn from the data. For this purpose, we compute $\boldsymbol{z}_{i,j}$, which contains new candidate values, via a tanh layer, where we control the amount of learning through $\boldsymbol{s}_{i,j}$. We then generate a new cell state information by multiplying old and new information with the forget and input gates, respectively, as in (4). Finally, we need to determine what we should output. In order to obtain the output, we use $\boldsymbol{c}_{i,j}$. However, we also need to determine which parts of the cell state we should keep for the output. Thus, we first compute $\boldsymbol{o}_{i,j}$ to filter certain parts of the cell state. Then, we push the cell state through a tanh layer and multiply it with the output gate to obtain the final output of an LSTM block as in (6).

After applying the LSTM architecture to each column of our data sequences as illustrated in Fig. 2, we take the average of the LSTM outputs for each data sequence, i.e., the mean pooling method. Through this, we obtain a new set of fixed-length sequences, i.e., denoted as $\{\bar{\boldsymbol{h}}_i\}_{i=1}^n$, $\bar{\boldsymbol{h}}_i \in \mathbb{R}^m$. Note that we also use the same procedure to obtain the state information $\bar{\boldsymbol{c}}_i \in \mathbb{R}^m$ for each $\boldsymbol{X}_i$ as demonstrated in Fig. 2. We emphasize that even though we do not use the mean state vector $\bar{\boldsymbol{c}}_i$ explicitly in Section III, all the calculations that include $\bar{\boldsymbol{h}}_i$ also requires the computation $\bar{\boldsymbol{c}}_i$ via the mean pooling method.

Remark 1: We use the mean pooling method in order to obtain the fixed-length sequences as $\bar{\boldsymbol{h}}_i = (1/d_i)\sum_{j=1}^{d_i} \boldsymbol{h}_{i,j}$. However, we can also use the other pooling methods. For example, for the last and max pooling methods, we use $\bar{\boldsymbol{h}}_i = \boldsymbol{h}_{i,d_i}$ and $\bar{\boldsymbol{h}}_i = \max_j \boldsymbol{h}_{i,j}$, $\forall i \in \{1, 2, \ldots n\}$, respectively. Our derivations can be straightforwardly extended to these different pooling methods.

III. Novel Anomaly Detection Algorithms

In this section, we first formulate the anomaly detection approaches based on the OC-SVM and SVDD algorithms. We then provide joint optimization updates to train the parameters of the overall structure.

A. Anomaly Detection With the OC-SVM Algorithm

In this section, we provide an anomaly detection algorithm based on the OC-SVM formulation and derive the joint updates for both the LSTM and SVM parameters. For the training, we first provide a quadratic programming-based algorithm and then introduce a gradient-based training algorithm. To apply the gradient-based training method, we smoothly approximate the original OC-SVM formulation and then prove the convergence of the approximated formulation to the actual one in Section III-A2.

In the OC-SVM algorithm, our aim is to find a hyperplane that separates the anomalies from the normal data [6]. We formulate the OC-SVM optimization problem for the sequence $\{\bar{\boldsymbol{h}}_i\}_{i=1}^n$ as follows [6]:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{n_\theta}, \boldsymbol{w}\in\mathbb{R}^m, \xi\in\mathbb{R}, \rho\in\mathbb{R}} \quad \frac{\|\boldsymbol{w}\|^2}{2} + \frac{1}{n\lambda}\sum_{i=1}^{n}\xi_i - \rho \tag{7}$$

$$\text{s. t.: } \boldsymbol{w}^T\bar{\boldsymbol{h}}_i \geq \rho - \xi_i, \quad \xi_i \geq 0 \ \forall i \tag{8}$$

$$\boldsymbol{W}^{(\cdot)T}\boldsymbol{W}^{(\cdot)} = \boldsymbol{I}, \boldsymbol{R}^{(\cdot)T}\boldsymbol{R}^{(\cdot)} = \boldsymbol{I} \text{ and } \boldsymbol{b}^{(\cdot)T}\boldsymbol{b}^{(\cdot)} = 1 \tag{9}$$

where ρ and $\boldsymbol{w}$ are the parameters of the separating hyperplane, $\lambda > 0$ is a regularization parameter, ξ is a slack variable to penalize misclassified instances, and we group the LSTM parameters $\{\boldsymbol{W}^{(z)}, \boldsymbol{R}^{(z)}, \boldsymbol{b}^{(z)}, \boldsymbol{W}^{(s)}, \boldsymbol{R}^{(s)}, \boldsymbol{b}^{(s)}, \boldsymbol{W}^{(f)}, \boldsymbol{R}^{(f)}, \boldsymbol{b}^{(f)}, \boldsymbol{W}^{(o)}, \boldsymbol{R}^{(o)}, \boldsymbol{b}^{(o)}\}$ into $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$, where $n_\theta = 4m(m+p+1)$. Since the LSTM parameters are unknown and $\bar{\boldsymbol{h}}_i$ is a function of these parameters, we also minimize the cost function in (7) with respect to $\boldsymbol{\theta}$.

After solving the optimization problem in (7)–(9), we use the scoring function

$$l(\boldsymbol{X}_i) = \text{sgn}(\boldsymbol{w}^T\bar{\boldsymbol{h}}_i - \rho) \tag{10}$$

to detect the anomalous data, where the sgn$(\cdot)$ function returns the sign of its input.

We emphasize that while minimizing (7) with respect to $\boldsymbol{\theta}$, we might suffer from overfitting and impotent learning of time dependencies on the data [27], i.e., forcing the parameters to null values, e.g., $\boldsymbol{\theta} = \boldsymbol{0}$. To circumvent these issues, we introduce (9), which constraints the norm of $\boldsymbol{\theta}$ to avoid overfitting and trivial solutions, e.g., $\boldsymbol{\theta} = \boldsymbol{0}$, while boosting the ability of the LSTM architecture to capture time dependencies [27], [28].

Remark 2: In (9), we use an orthogonality constraint for each LSTM parameter. However, we can also use other constraints instead of (9) and solve the optimization problem in (7)–(9) in the same manner. For example, a common choice of constraint for neural networks is the Frobenius norm [29], i.e., defined as

$$\|\boldsymbol{A}\|_F = \sum_i \sum_j \boldsymbol{A}_{ij}^2 \tag{11}$$

for a real matrix $\boldsymbol{A}$, where $\boldsymbol{A}_{ij}$ represents the element at the i^{th} column and j^{th} row of $\boldsymbol{A}$. In this case, we can directly replace (9) with a Frobenius norm constraint for each LSTM parameter as in (11) and then solve the optimization problem in the same manner. Such approaches only aim to regularize the parameters [28]. However, for RNNs, we may also encounter exponential growth or decay in the norm of the gradients while training the parameters, which significantly degrades capabilities of these architectures to capture time dependencies [27], [28]. Moreover, (9) also regularizes the parameters by bounding the norm of each column of the coefficient matrices as one. Thus, in this article, we put the constraint (9) in order to regularize the parameters while improving the capabilities of the LSTM architecture in capturing time dependencies [27], [28].

1) Quadratic Programming-Based Training Algorithm: Here, we introduce a training approach based on quadratic programming for the optimization problem in (7)–(9), where we perform consecutive updates for the LSTM and SVM parameters. For this purpose, we first convert the optimization problem to a dual form in the following. We then provide the consecutive updates for each parameter.

We have the following Lagrangian for the SVM parameters:

$$L(\boldsymbol{w}, \xi, \rho, \nu, \alpha) = \frac{\|\boldsymbol{w}\|^2}{2} + \frac{1}{n\lambda}\sum_{i=1}^{n}\xi_i - \rho - \sum_{i=1}^{n}\nu_i\xi_i - \sum_{i=1}^{n}\alpha_i(\boldsymbol{w}^T\bar{\boldsymbol{h}}_i - \rho + \xi_i) \tag{12}$$

where $\nu_i,\ \alpha_i \geq 0$ are the Lagrange multipliers. Taking derivative of (12) with respect to $\boldsymbol{w}$, ξ, and ρ and then setting the derivatives to zero give

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \bar{\boldsymbol{h}}_i \tag{13}$$

$$\sum_{i=1}^{n} \alpha_i = 1 \quad \text{and } \alpha_i = 1/(n\lambda) - \nu_i \quad \forall i. \tag{14}$$

Note that at the optimum, the inequalities in (8) become equalities if α_i and ν_i are nonzero, i.e., $0 < \alpha_i < 1/(n\lambda)$ [6]. With this relation, we compute ρ as

$$\rho = \sum_{j=1}^{n} \alpha_j \bar{\boldsymbol{h}}_j^T \bar{\boldsymbol{h}}_i \quad \text{for } \ 0 < \alpha_i < 1/(n\lambda). \tag{15}$$

By substituting (13) and (14) into (12), we obtain the following dual problem for the constrained minimization in (7)–(9):

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{n_\theta}, \boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \bar{\boldsymbol{h}}_i^T \bar{\boldsymbol{h}}_j \tag{16}$$

$$\text{s. t.:} \quad \sum_{i=1}^{n} \alpha_i = 1 \quad \text{and } 0 \leq \alpha_i \leq 1/(n\lambda) \quad \forall i \tag{17}$$

$$\boldsymbol{W}^{(\cdot)T}\boldsymbol{W}^{(\cdot)} = \boldsymbol{I}, \boldsymbol{R}^{(\cdot)T}\boldsymbol{R}^{(\cdot)} = \boldsymbol{I}$$
$$\text{and } \boldsymbol{b}^{(\cdot)T}\boldsymbol{b}^{(\cdot)} = 1 \tag{18}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^n$ is a vector representation for α_i's. Since the LSTM parameters are unknown, we also put the minimization term for $\boldsymbol{\theta}$ into (16) as in (7). By substituting (13) into (10), we have the following scoring function for the dual problem:

$$l(\boldsymbol{X}_i) = \text{sgn}\left(\sum_{j=1}^{n} \alpha_j \bar{\boldsymbol{h}}_j^T \bar{\boldsymbol{h}}_i - \rho\right) \tag{19}$$

where we calculate ρ using (15).

In order to find the optimal $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ for the optimization problem in (16)–(18), we employ the following procedure. We first select a certain set of the LSTM parameters, i.e., $\boldsymbol{\theta}_0$. Based on $\boldsymbol{\theta}_0$, we find the minimizing $\boldsymbol{\alpha}$ values, i.e., $\boldsymbol{\alpha}_1$, using the sequential minimal optimization (SMO) algorithm [30]. Now, we fix $\boldsymbol{\alpha}$ as $\boldsymbol{\alpha}_1$ and then update $\boldsymbol{\theta}$ from $\boldsymbol{\theta}_0$ to $\boldsymbol{\theta}_1$ using the algorithm for optimization with orthogonality constraints in [31]. We repeat these consecutive update procedures until $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ converge [32]. Then, we use the converged values in order to evaluate (19). Although the convergence of the algorithm is not guaranteed, it can be obtained by carefully tuning certain parameters, e.g., the learning rate, in most of real-life applications [32]. In the following, we explain these procedures in detail.

Based on $\boldsymbol{\theta}_k$, i.e., the LSTM parameter vector at the k^{th} iteration, we update $\boldsymbol{\alpha}_k$, i.e., the $\boldsymbol{\alpha}$ vector at the k^{th} iteration, using the SMO algorithm due to its efficiency in solving quadratic constrained optimization problems [30]. In the SMO algorithm, we choose a subset of parameters to minimize and fix the rest of parameters. In the extreme case, we choose only one parameter to minimize, however, due to (17), we must choose at least two parameters. To illustrate how the SMO algorithm works in our case, we choose α_1 and α_2 to update and fix the rest of the parameters in (16). From (17), we have

$$\alpha_1 = 1 - S - \alpha_2, \quad \text{where } \ S = \sum_{i=3}^{n} \alpha_i. \tag{20}$$

We first replace α_1 in (16) with (20). We then take the derivative of (16) with respect to α_2 and equate the derivative to zero. Thus, we obtain the following update for α_2 at the k^{th} iteration:

$$\alpha_{k+1,2} = \frac{(\alpha_{k,1} + \alpha_{k,2})(K_{11} - K_{12}) + M_1 - M_2}{K_{11} + K_{22} - 2K_{12}} \tag{21}$$

where $K_{ij} \triangleq \bar{\boldsymbol{h}}_i^T \bar{\boldsymbol{h}}_j$, $M_i \triangleq \sum_{j=3}^{n} \alpha_{k,j} K_{ij}$ and $\alpha_{k,i}$ represents the i^{th} element of $\boldsymbol{\alpha}_k$. Due to (17), if the updated value of α_2 is outside of the region $[0, 1/(n\lambda)]$, we project it to this region. Once α_2 is updated as $\alpha_{k+1,2}$, we obtain $\alpha_{k+1,1}$ using (20). For the rest of the parameters, we repeat the same procedure, which eventually converges to a certain set of parameters [30]. In this way, we obtain $\boldsymbol{\alpha}_{k+1}$, i.e., the minimizing $\boldsymbol{\alpha}$ for $\boldsymbol{\theta}_k$.

Following the update of $\boldsymbol{\alpha}$, we update $\boldsymbol{\theta}$ based on the updated $\boldsymbol{\alpha}_{k+1}$ vector. For this purpose, we employ the optimization method in [31]. Since we have $\boldsymbol{\alpha}_{k+1}$ that satisfies (17), we reduce the dual problem to

$$\min_{\boldsymbol{\theta}} \ \kappa(\boldsymbol{\theta}, \boldsymbol{\alpha}_{k+1}) = \frac{1}{2} \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{k+1,i} \alpha_{k+1,j} \bar{\boldsymbol{h}}_i^T \bar{\boldsymbol{h}}_j \tag{22}$$

$$\text{s.t.:} \boldsymbol{W}^{(\cdot)T}\boldsymbol{W}^{(\cdot)} = \boldsymbol{I}, \boldsymbol{R}^{(\cdot)T}\boldsymbol{R}^{(\cdot)} = \boldsymbol{I} \quad \text{and } \boldsymbol{b}^{(\cdot)T}\boldsymbol{b}^{(\cdot)} = 1. \tag{23}$$

For (22) and (23), we update $\boldsymbol{W}^{(\cdot)}$ as follows:

$$\boldsymbol{W}_{k+1}^{(\cdot)} = \left(\boldsymbol{I} + \frac{\mu}{2}\boldsymbol{A}_k\right)^{-1} \left(\boldsymbol{I} - \frac{\mu}{2}\boldsymbol{A}_k\right) \boldsymbol{W}_k^{(\cdot)} \tag{24}$$

where the subscripts represent the current iteration index, μ is the learning rate, $\boldsymbol{A}_k = \boldsymbol{G}_k(\boldsymbol{W}_k^{(\cdot)})^T - \boldsymbol{W}_k^{(\cdot)}\boldsymbol{G}_k^T$, and the element at the i^{th} row and the j^{th} column of $\boldsymbol{G}$ is defined as

$$\boldsymbol{G}_{ij} \triangleq \frac{\partial \kappa(\boldsymbol{\theta}, \boldsymbol{\alpha}_{k+1})}{\partial \boldsymbol{W}_{ij}^{(\cdot)}}. \tag{25}$$

Remark 3: For $\boldsymbol{R}^{(\cdot)}$ and $\boldsymbol{b}^{(\cdot)}$, we first compute the gradient of the objective function with respect to the chosen parameter as in (25). We then obtain $\boldsymbol{A}_k$ according to the chosen parameter. Using $\boldsymbol{A}_k$, we update the chosen parameter as in (24).

With these updates, we obtain a quadratic programming-based training algorithm (see Algorithm 1 for the pseudocode) for our LSTM-based anomaly detector.

2) Gradient-Based Training Algorithm: Although the quadratic programming-based training algorithm directly optimizes the original OC-SVM formulation without requiring any approximation, since it depends on the separated consecutive updates of the LSTM and OC-SVM parameters, it might not converge to even a local minimum [32]. In order to resolve this issue, in this section, we introduce a training method based on only the first-order gradients, which updates the parameters at the same time. However, since we require an approximation to the original OC-SVM formulation to apply this method,

Algorithm 1 Quadratic Programming-Based Training for the Anomaly Detection Algorithm Based on OC-SVM

Initialize the LSTM parameters as $\boldsymbol{\theta}_0$ and the dual OC-SVM parameters as $\boldsymbol{\alpha}_0$
Determine a threshold ϵ as convergence criterion
$k = -1$
do
  $k = k + 1$
  Using $\boldsymbol{\theta}_k$, obtain $\{\bar{\boldsymbol{h}}\}_{i=1}^{n}$ according to Fig. 2
  Find optimal $\boldsymbol{\alpha}_{k+1}$ for $\{\bar{\boldsymbol{h}}\}_{i=1}^{n}$ using (20) and (21)
  Based on $\boldsymbol{\alpha}_{k+1}$, obtain $\boldsymbol{\theta}_{k+1}$ using (24) and Remark 3
while $\left(\kappa(\boldsymbol{\theta}_{k+1}, \boldsymbol{\alpha}_{k+1}) - \kappa(\boldsymbol{\theta}_k, \boldsymbol{\alpha}_k)\right)^2 > \epsilon$
Detect anomalies using (19) evaluated at $\boldsymbol{\theta}_k$ and $\boldsymbol{\alpha}_k$

we also prove the convergence of the approximated formulation to the original OC-SVM formulation in this section.

Considering (8), we write the slack variable in a different form as follows:

$$G(\beta_{\boldsymbol{w},\rho}(\bar{\boldsymbol{h}}_i)) \triangleq \max\{0, \beta_{\boldsymbol{w},\rho}(\bar{\boldsymbol{h}}_i)\} \quad \forall i \tag{26}$$

where

$$\beta_{\boldsymbol{w},\rho}(\bar{\boldsymbol{h}}_i) \triangleq \rho - \boldsymbol{w}^T \bar{\boldsymbol{h}}_i.$$

By substituting (26) into (7), we remove the constraint (8) and obtain the following optimization problem:

$$\min_{\boldsymbol{w}\in\mathbb{R}^m, \rho\in\mathbb{R}, \boldsymbol{\theta}\in\mathbb{R}^{n_\theta}} \frac{\|\boldsymbol{w}\|^2}{2} + \frac{1}{n\lambda}\sum_{i=1}^{n} G(\beta_{\boldsymbol{w},\rho}(\bar{\boldsymbol{h}}_i)) - \rho \tag{27}$$

$$\text{s.t.: } \boldsymbol{W}^{(\cdot)T}\boldsymbol{W}^{(\cdot)} = \boldsymbol{I}, \boldsymbol{R}^{(\cdot)T}\boldsymbol{R}^{(\cdot)} = \boldsymbol{I} \quad \text{and } \boldsymbol{b}^{(\cdot)T}\boldsymbol{b}^{(\cdot)} = 1. \tag{28}$$

Since (26) is not a differentiable function, we are unable to solve the optimization problem in (27) using gradient-based optimization algorithms. Hence, we employ a differentiable function

$$S_\tau(\beta_{\boldsymbol{w},\rho}(\bar{\boldsymbol{h}}_i)) = \frac{1}{\tau}\log\left(1 + e^{\tau\beta_{\boldsymbol{w},\rho}(\bar{\boldsymbol{h}}_i)}\right) \tag{29}$$

to smoothly approximate (26), where $\tau > 0$ is the smoothing parameter and log represents the natural logarithm. In (29), as τ increases, $S_\tau(\cdot)$ converges to $G(\cdot)$ (see Fig. 3); hence, we choose a large value for τ.

Proposition 1: As τ increases, $S_\tau(\beta_{\boldsymbol{w},\rho}(\bar{\boldsymbol{h}}_i))$ uniformly converges to $G(\beta_{\boldsymbol{w},\rho}(\bar{\boldsymbol{h}}_i))$. As a consequence, our approximation $F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta})$ converges to the SVM objective function $F(\boldsymbol{w}, \rho, \boldsymbol{\theta})$, i.e., defined as

$$F(\boldsymbol{w}, \rho, \boldsymbol{\theta}) \triangleq \frac{\|\boldsymbol{w}\|^2}{2} + \frac{1}{n\lambda}\sum_{i=1}^{n} G(\beta_{\boldsymbol{w},\rho}(\bar{\boldsymbol{h}}_i)) - \rho.$$

Proof of Proposition 1: The proof of the proposition is given in Appendix A. □

With (29), we modify our optimization problem as follows:

$$\min_{\boldsymbol{w}\in\mathbb{R}^m, \rho\in\mathbb{R}, \boldsymbol{\theta}\in\mathbb{R}^{n_\theta}} F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta}) \tag{30}$$

$$\text{s.t.: } \boldsymbol{W}^{(\cdot)T}\boldsymbol{W}^{(\cdot)} = \boldsymbol{I}, \boldsymbol{R}^{(\cdot)T}\boldsymbol{R}^{(\cdot)} = \boldsymbol{I} \quad \text{and } \boldsymbol{b}^{(\cdot)T}\boldsymbol{b}^{(\cdot)} = 1 \tag{31}$$

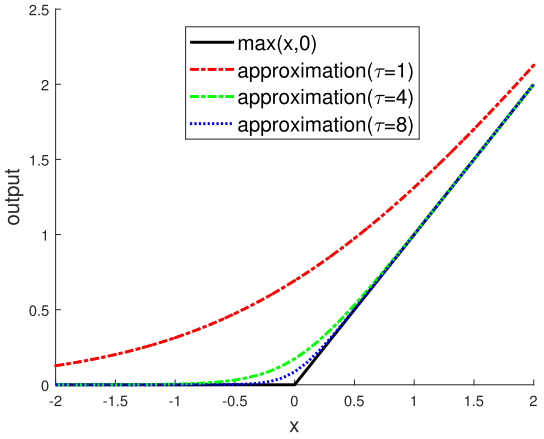

Fig. 3. Comparison of (26) with its smooth approximations.

where $F_\tau(\cdot,\cdot,\cdot)$ is the objective function of our optimization problem and defined as

$$F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta}) \triangleq \frac{\|\boldsymbol{w}\|^2}{2} + \frac{1}{n\lambda}\sum_{i=1}^{n} S_\tau(\beta_{\boldsymbol{w},\rho}(\bar{\boldsymbol{h}}_i)) - \rho.$$

To obtain the optimal parameters for (30) and (31), we update $\boldsymbol{w}$, ρ and $\boldsymbol{\theta}$ until they converge to a local or global optimum [31], [33]. For the update of $\boldsymbol{w}$ and ρ, we use the gradient descent algorithm [33], where we compute the first-order gradient of the objective function with respect to each parameter. We first compute the gradient for $\boldsymbol{w}$ as follows:

$$\nabla_{\boldsymbol{w}} F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta}) = \boldsymbol{w} + \frac{1}{n\lambda}\sum_{i=1}^{n} \frac{-\bar{\boldsymbol{h}}_i e^{\tau\beta_{\boldsymbol{w},\rho}(\bar{\mathrm{h}}_i)}}{1 + e^{\tau\beta_{\boldsymbol{w},\rho}(\bar{\mathrm{h}}_i)}}. \tag{32}$$

Using (32), we update $\boldsymbol{w}$ as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \mu\nabla_{\boldsymbol{w}} F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta})\Big|_{\substack{\boldsymbol{w}=\boldsymbol{w}_k \\ \rho=\rho_k \\ \boldsymbol{\theta}=\boldsymbol{\theta}_k}} \tag{33}$$

where the subscript k indicates the value of any parameter at the k^{th} iteration. Similarly, we calculate the derivative of the objective function with respect to ρ as follows:

$$\frac{\partial F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta})}{\partial \rho} = \frac{1}{n\lambda}\sum_{i=1}^{n} \frac{e^{\tau\beta_{\boldsymbol{w},\rho}(\bar{\mathrm{h}}_i)}}{1 + e^{\tau\beta_{\boldsymbol{w},\rho}(\bar{\mathrm{h}}_i)}} - 1. \tag{34}$$

Using (34), we update ρ as

$$\rho_{k+1} = \rho_k - \mu\frac{\partial F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta})}{\partial \rho}\Big|_{\substack{\boldsymbol{w}=\boldsymbol{w}_k \\ \rho=\rho_k \\ \boldsymbol{\theta}=\boldsymbol{\theta}_k}}. \tag{35}$$

For the LSTM parameters, we use the method for optimization with orthogonality constraints in [31] due to (31). To update $\boldsymbol{W}^{(\cdot)}$, we calculate the gradient of the objective function as

$$\frac{\partial F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta})}{\partial \boldsymbol{W}_{ij}^{(\cdot)}} = \frac{1}{n\lambda}\sum_{i=1}^{n} \frac{-\boldsymbol{w}^T\left(\partial\bar{\boldsymbol{h}}_i/\partial\boldsymbol{W}_{ij}^{(\cdot)}\right)e^{\tau\beta_{\boldsymbol{w},\rho}(\bar{\mathrm{h}}_i)}}{1 + e^{\tau\beta_{\boldsymbol{w},\rho}(\bar{\mathrm{h}}_i)}}. \tag{36}$$

We then update $\boldsymbol{W}^{(\cdot)}$ using (36) as

$$\boldsymbol{W}_{k+1}^{(\cdot)} = \left(\boldsymbol{I} + \frac{\mu}{2}\boldsymbol{B}_k\right)^{-1}\left(\boldsymbol{I} - \frac{\mu}{2}\boldsymbol{B}_k\right)\boldsymbol{W}_k^{(\cdot)} \tag{37}$$

where $\boldsymbol{B}_k = \boldsymbol{M}_k(\boldsymbol{W}_k^{(\cdot)})^T - \boldsymbol{W}_k^{(\cdot)}\boldsymbol{M}_k^T$ and

$$\boldsymbol{M}_{ij} \triangleq \frac{\partial F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta})}{\partial \boldsymbol{W}_{ij}^{(\cdot)}}. \tag{38}$$

Remark 4: For $\boldsymbol{R}^{(\cdot)}$ and $\boldsymbol{b}^{(\cdot)}$, we first compute the gradient of the objective function with respect to the chosen parameter as in (38). We then obtain $\boldsymbol{B}_k$ according to the chosen parameter. Using $\boldsymbol{B}_k$, we update the chosen parameter as in (37).

Remark 5: In the semisupervised framework, we have the following optimization problem for our SVM-based algorithms [34]:

$$\min_{\boldsymbol{\theta},\boldsymbol{w},\xi,\eta,\gamma,\rho} \left(\frac{\sum_{i=1}^{l}\eta_i + \sum_{j=l+1}^{l+k}\min(\gamma_j,\xi_j)}{(1/C)}\right) + \|\boldsymbol{w}\| \tag{39}$$

$$\text{s.t.: } y_i(\boldsymbol{w}^T\bar{\boldsymbol{h}}_i + \rho) \geq 1 - \eta_i, \quad \eta_i \geq 0,\ i = 1,\ldots,l \tag{40}$$

$$\boldsymbol{w}^T\bar{\boldsymbol{h}}_j - \rho \geq 1 - \xi_j, \quad \xi_j \geq 0 \qquad j = l+1,\ldots,l+k \tag{41}$$

$$-\boldsymbol{w}^T\bar{\boldsymbol{h}}_j + \rho \geq 1 - \gamma_j, \ \gamma_j \geq 0 \quad j = l+1,\ldots,l+k \tag{42}$$

$$\boldsymbol{W}^{(\cdot)T}\boldsymbol{W}^{(\cdot)} = \boldsymbol{I}, \boldsymbol{R}^{(\cdot)T}\boldsymbol{R}^{(\cdot)} = \boldsymbol{I} \quad \text{and} \quad \boldsymbol{b}^{(\cdot)T}\boldsymbol{b}^{(\cdot)} = 1 \tag{43}$$

where $\gamma \in \mathbb{R}$ and $\eta \in \mathbb{R}$ are slack variables as ξ, C is a tradeoff parameter, l and k are the number of the labeled and unlabeled data instances, respectively, and $y_i \in \{-1, +1\}$ represents the label of the i^{th} data instance.

For the application of quadratic programming-based training method in the semisupervised case, we apply all the steps from (12) to (25) for the optimization problem in (39)–(43). Similarly, we modify the equations from (26) to (38) according to (39)–(43) in order to get the gradient-based training method in the semisupervised framework. For the supervised implementations, we follow the same procedures with the semisupervised implementations for $k = 0$ case.

Hence, we complete the required updates for each parameter. The complete algorithm is also provided in Algorithm 2 as a pseudocode. Moreover, we illustrate the convergence of our approximation (29)–(26) in Proposition 1. Using Proposition 1, we then demonstrate the convergence of the optimal values for our objective function (30) to the optimal values of the actual SVM objective function (27) in Theorem 1.

Theorem 1: Let $\boldsymbol{w}_\tau$ and ρ_τ be the solutions of (30) for any fixed $\boldsymbol{\theta}$. Then, $\boldsymbol{w}_\tau$ and ρ_τ are unique and $F_\tau(\boldsymbol{w}_\tau, \rho_\tau, \boldsymbol{\theta})$ converges to the minimum of $F(\boldsymbol{w}, \rho, \boldsymbol{\theta})$.

Proof of Theorem 1: The proof of the theorem is given in Appendix B. □

B. Anomaly Detection With the SVDD Algorithm

In this section, we introduce an anomaly detection algorithm based on the SVDD formulation and provide the joint updates in order to learn both the LSTM and SVDD parameters. However, since the generic formulation is the same with the OC-SVM case, we only provide the required and distinct updates for the parameters and proof for the convergence of the approximated SVDD formulation to the actual one.

Algorithm 2 Gradient-Based Training for the Anomaly Detection Algorithm Based on OC-SVM

1: Initialize the LSTM parameters as $\boldsymbol{\theta}_0$ and the OC-SVM parameters as $\boldsymbol{w}_0$ and ρ_0
2: Determine a threshold ϵ as convergence criterion
3: $k = -1$
4: **do**
5: $k = k + 1$
6: Using $\boldsymbol{\theta}_k$, obtain $\{\bar{\boldsymbol{h}}\}_{i=1}^n$ according to Fig. 2
7: Obtain $\boldsymbol{w}_{k+1}$, ρ_{k+1} and $\boldsymbol{\theta}_{k+1}$ using (33), (35), (37) and Remark 4
8: **while** $\left(F_\tau(\boldsymbol{w}_{k+1}, \rho_{k+1}, \boldsymbol{\theta}_{k+1}) - F_\tau(\boldsymbol{w}_k, \rho_k, \boldsymbol{\theta}_k)\right)^2 > \epsilon$
9: Detect anomalies using (10) evaluated at $\boldsymbol{w}_k$, ρ_k and $\boldsymbol{\theta}_k$

In the SVDD algorithm, we aim to find a hypersphere that encloses the normal data while leaving the anomalous data outside the hypersphere [7]. For the sequence $\{\bar{\boldsymbol{h}}_i\}_{i=1}^n$, we have the following SVDD optimization problem [7]:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{n_\theta},\tilde{\boldsymbol{c}}\in\mathbb{R}^m,\xi\in\mathbb{R},R\in\mathbb{R}} R^2 + \frac{1}{n\lambda}\sum_{i=1}^{n}\xi_i \tag{44}$$

$$\text{s. t.: } \|\bar{\boldsymbol{h}}_i - \tilde{\boldsymbol{c}}\|^2 - R^2 \leq \xi_i, \quad \xi_i \geq 0\ \forall i \tag{45}$$

$$\boldsymbol{W}^{(\cdot)T}\boldsymbol{W}^{(\cdot)} = \boldsymbol{I}, \boldsymbol{R}^{(\cdot)T}\boldsymbol{R}^{(\cdot)} = \boldsymbol{I} \quad \text{and} \quad \boldsymbol{b}^{(\cdot)T}\boldsymbol{b}^{(\cdot)} = 1 \tag{46}$$

where $\lambda > 0$ is a tradeoff parameter between R^2 and the total misclassification error, R is the radius of the hypersphere, and $\tilde{\boldsymbol{c}}$ is the center of the hypersphere. In addition, $\boldsymbol{\theta}$ and ξ represent the LSTM parameters and the slack variable, respectively, as in the OC-SVM case. After solving the constrained optimization problem in (44)–(46), we detect anomalies using the following scoring function:

$$l(\boldsymbol{X}_i) = \text{sgn}(R^2 - \|\bar{\boldsymbol{h}}_i - \tilde{\boldsymbol{c}}\|^2). \tag{47}$$

1) Quadratic Programming-Based Training Algorithm: In this section, we introduce a training algorithm based on quadratic programming for (44)–(46). As in the OC-SVM case, we first assume that the LSTM parameters are fixed and then perform optimization over the SVDD parameters based on the fixed LSTM parameters. For (44) and (45), we have the following Lagrangian:

$$L(\tilde{\boldsymbol{c}}, \xi, R, \nu, \alpha) = R^2 + \frac{1}{n\lambda}\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\nu_i\xi_i - \sum_{i=1}^{n}\alpha_i(\xi_i - \|\bar{\boldsymbol{h}}_i - \tilde{\boldsymbol{c}}\|^2 + R^2) \tag{48}$$

where $\nu_i,\ \alpha_i \geq 0$ are the Lagrange multipliers. Taking derivative of (48) with respect to $\tilde{\boldsymbol{c}}$, ξ, and R and then setting the derivatives to zero yields

$$\tilde{\boldsymbol{c}} = \sum_{i=1}^{n}\alpha_i\bar{\boldsymbol{h}}_i \tag{49}$$

$$\sum_{i=1}^{n}\alpha_i = 1 \quad \text{and } \alpha_i = 1/(n\lambda) - \nu_i \quad \forall i. \tag{50}$$

Putting (49) and (50) into (48), we obtain a dual form for (44) and (45) as follows:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{n_\theta},\boldsymbol{\alpha}\in\mathbb{R}^n} \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\bar{\boldsymbol{h}}_i^T\bar{\boldsymbol{h}}_j - \sum_{i=1}^{n}\alpha_i\bar{\boldsymbol{h}}_i^T\bar{\boldsymbol{h}}_i \tag{51}$$

$$\text{s. t.: } \sum_{i=1}^{n}\alpha_i = 1 \quad \text{and } 0\le\alpha_i\le 1/(n\lambda) \quad \forall i \tag{52}$$

$$\boldsymbol{W}^{(\cdot)T}\boldsymbol{W}^{(\cdot)} = \boldsymbol{I}, \quad \boldsymbol{R}^{(\cdot)T}\boldsymbol{R}^{(\cdot)} = \boldsymbol{I} \quad \text{and } \boldsymbol{b}^{(\cdot)T}\boldsymbol{b}^{(\cdot)} = 1. \tag{53}$$

Using (49), we modify (47) as

$$l(\boldsymbol{X}_i) = \text{sgn}\left(R^2 - \sum_{k=1}^{n}\sum_{j=1}^{n}\alpha_k\alpha_j\bar{\boldsymbol{h}}_k^T\bar{\boldsymbol{h}}_j + 2\sum_{j=1}^{n}\alpha_j\bar{\boldsymbol{h}}_j^T\bar{\boldsymbol{h}}_i - \bar{\boldsymbol{h}}_i^T\bar{\boldsymbol{h}}_i\right). \tag{54}$$

In order to solve the constrained optimization problem in (51)–(53), we employ the same approach as in the OC-SVM case. We first fix a certain set of the LSTM parameters $\boldsymbol{\theta}$. Based on these parameters, we find the optimal $\boldsymbol{\alpha}$ using the SMO algorithm. After that, we fix $\boldsymbol{\alpha}$ to update $\boldsymbol{\theta}$ using the algorithm for optimization with orthogonality constraints. We repeat these procedures until we reach convergence. Finally, we evaluate (54) based on the converged parameters.

Remark 6: In the SVDD case, we apply the SMO algorithm using the same procedures with the OC-SVM case. In particular, we first choose two parameters, e.g., α_1 and α_2, to minimize and fix the other parameters. Due to (52), the chosen parameters must obey (20). Hence, we have the following update rule for α_2 at the k^{th} iteration:

$$\alpha_{k+1,2} = \frac{2(1-S)(K_{11}-K_{12}) + K_{22} - K_{11} + M_1 - M_2}{2(K_{11}+K_{22}-2K_{12})}$$

where $S = \sum_{j=3}^{n}\alpha_{k,j}$ and the other definitions are the same with the OC-SVM case. We then obtain $\alpha_{k+1,1}$ using (20). By this, we obtain the updated values $\alpha_{k+1,2}$ and $\alpha_{k+1,1}$. For the remaining parameters, we repeat this procedure until reaching convergence.

Remark 7: For the SVDD case, we update $\boldsymbol{W}^{(\cdot)}$ at the k^{th} iteration as in (24). However, instead of (25), we have the following definition for $\boldsymbol{G}$:

$$\boldsymbol{G}_{ij} = \frac{\partial\pi(\boldsymbol{\theta},\boldsymbol{\alpha}_{k+1})}{\partial\boldsymbol{W}_{ij}^{(\cdot)}}$$

where

$$\pi(\boldsymbol{\theta},\boldsymbol{\alpha}_{k+1}) \triangleq \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{k+1,i}\alpha_{k+1,j}\bar{\boldsymbol{h}}_i^T\bar{\boldsymbol{h}}_j - \sum_{i=1}^{n}\alpha_{k+1,i}\bar{\boldsymbol{h}}_i^T\bar{\boldsymbol{h}}_i$$

at the k^{th} iteration. For the remaining parameters, we follow the procedure in Remark 3.

Hence, we obtain a quadratic programming-based training algorithm for our LSTM-based anomaly detector, which is also described in Algorithm 3 as a pseudocode.

Algorithm 3 Quadratic Programming-Based Training for the Anomaly Detection Algorithm Based on SVDD

1: Initialize the LSTM parameters as $\boldsymbol{\theta}_0$ and the dual SVDD parameters as $\boldsymbol{\alpha}_0$
2: Determine a threshold ϵ as convergence criterion
3: $k = -1$
4: **do**
5: $k = k + 1$
6: Using $\boldsymbol{\theta}_k$, obtain $\{\bar{\boldsymbol{h}}\}_{i=1}^n$ according to Fig. 2
7: Find optimal $\boldsymbol{\alpha}_{k+1}$ for $\{\bar{\boldsymbol{h}}\}_{i=1}^n$ using the procedure in Remark 6
8: Based on $\boldsymbol{\alpha}_{k+1}$, obtain $\boldsymbol{\theta}_{k+1}$ using Remark 7
9: **while** $\left(\pi(\boldsymbol{\theta}_{k+1},\boldsymbol{\alpha}_{k+1}) - \pi(\boldsymbol{\theta}_k,\boldsymbol{\alpha}_k)\right)^2 > \epsilon$
10: Detect anomalies using (54) evaluated at $\boldsymbol{\theta}_k$ and $\boldsymbol{\alpha}_k$

2) Gradient-Based Training Algorithm: In this section, we introduce a training algorithm based on only the first-order gradients for (44)–(46). We again use the $G(\cdot)$ function in (26) in order to eliminate the constraint in (45) as follows:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{n_\theta},\tilde{\boldsymbol{c}}\in\mathbb{R}^m,R\in\mathbb{R}} R^2 + \frac{1}{n\lambda}\sum_{i=1}^{n}G(\Psi_{R,\tilde{\boldsymbol{c}}}(\bar{\boldsymbol{h}}_i)) \tag{55}$$

$$\text{s.t.: } \boldsymbol{W}^{(\cdot)T}\boldsymbol{W}^{(\cdot)} = \boldsymbol{I}, \boldsymbol{R}^{(\cdot)T}\boldsymbol{R}^{(\cdot)} = \boldsymbol{I} \text{ and } \boldsymbol{b}^{(\cdot)T}\boldsymbol{b}^{(\cdot)} = 1 \tag{56}$$

where

$$\Psi_{R,\tilde{\boldsymbol{c}}}(\bar{\boldsymbol{h}}_i) \triangleq \|\bar{\boldsymbol{h}}_i - \tilde{\boldsymbol{c}}\|^2 - R^2.$$

Since the gradient-based methods cannot optimize (55) due to the nondifferentiable function $G(\cdot)$, we employ $S_\tau(\cdot)$ instead of $G(\cdot)$ and modify (55) as

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{n_\theta},\tilde{\boldsymbol{c}}\in\mathbb{R}^m,R\in\mathbb{R}} F_\tau(\tilde{\boldsymbol{c}},R,\boldsymbol{\theta}) = R^2 + \frac{1}{n\lambda}\sum_{i=1}^{n}S_\tau(\Psi_{R,\tilde{\boldsymbol{c}}}(\bar{\boldsymbol{h}}_i)) \tag{57}$$

$$\text{s.t.: } \boldsymbol{W}^{(\cdot)T}\boldsymbol{W}^{(\cdot)} = \boldsymbol{I}, \quad \boldsymbol{R}^{(\cdot)T}\boldsymbol{R}^{(\cdot)} = \boldsymbol{I} \text{ and } \boldsymbol{b}^{(\cdot)T}\boldsymbol{b}^{(\cdot)} = 1 \tag{58}$$

where $F_\tau(\cdot,\cdot,\cdot)$ is the objective function of (57). To obtain the optimal values for (57) and (58), we update $\tilde{\boldsymbol{c}}$, R, and $\boldsymbol{\theta}$ till we reach either a local or a global optimum. For the updates of $\tilde{\boldsymbol{c}}$ and R, we employ the gradient descent algorithm, where we use the following gradient calculations. We first compute the gradient of $\tilde{\boldsymbol{c}}$ as

$$\nabla_{\tilde{\boldsymbol{c}}}F_\tau(\tilde{\boldsymbol{c}},R,\boldsymbol{\theta}) = \frac{1}{n\lambda}\sum_{i=1}^{n}\frac{2(\tilde{\boldsymbol{c}}-\bar{\boldsymbol{h}}_i)e^{\tau\Psi_{\tilde{\boldsymbol{c}},R}(\bar{\text{h}}_i)}}{1+e^{\tau\Psi_{\tilde{\boldsymbol{c}},R}(\bar{\text{h}}_i)}}. \tag{59}$$

Using (59), we have the following update:

$$\tilde{\boldsymbol{c}}_{k+1} = \tilde{\boldsymbol{c}}_k - \mu\nabla_{\tilde{\boldsymbol{c}}}F_\tau(\tilde{\boldsymbol{c}},R,\boldsymbol{\theta})\Big|_{\substack{\tilde{\boldsymbol{c}}=\tilde{\boldsymbol{c}}_k\\ R^2=R_k^2\\ \boldsymbol{\theta}=\boldsymbol{\theta}_k}} \tag{60}$$

where the subscript k represents the iteration number. Likewise, we compute the derivative of the objective function with respect to R^2 as

$$\frac{\partial F_\tau(\tilde{\boldsymbol{c}},R,\boldsymbol{\theta})}{\partial R^2} = 1 + \frac{1}{n\lambda}\sum_{i=1}^{n}\frac{-e^{\tau\Psi_{\tilde{\boldsymbol{c}},R}(\bar{\text{h}}_i)}}{1+e^{\tau\Psi_{\tilde{\boldsymbol{c}},R}(\bar{\text{h}}_i)}}. \tag{61}$$

With (61), we update R^2 as

$$R_{k+1}^2 = R_k^2 - \mu \frac{\partial F_\tau(\tilde{\boldsymbol{c}}, R, \boldsymbol{\theta})}{\partial R^2}\Big|_{\substack{\tilde{\boldsymbol{c}}=\tilde{\boldsymbol{c}}_k \\ R^2=R_k^2 \\ \boldsymbol{\theta}=\boldsymbol{\theta}_k}}. \tag{62}$$

For $\boldsymbol{\theta}$, the gradient calculation is as follows:

$$\frac{\partial F_\tau(\tilde{\boldsymbol{c}}, R, \boldsymbol{\theta})}{\partial \boldsymbol{W}_{ij}^{(\cdot)}} = \sum_{i=1}^{n} \frac{2\big(\partial \bar{\boldsymbol{h}}_i / \partial \boldsymbol{W}_{ij}^{(\cdot)}\big)^T (\bar{\boldsymbol{h}}_i - \tilde{\boldsymbol{c}}) e^{\tau \Psi_{\tilde{\boldsymbol{c}},R}(\bar{\mathrm{h}}_i)}}{n\lambda\big(1 + e^{\tau \Psi_{\tilde{\boldsymbol{c}},R}(\bar{\mathrm{h}}_i)}\big)}. \tag{63}$$

Using (63), we have the following update:

$$\boldsymbol{W}_{k+1}^{(\cdot)} = \left(\boldsymbol{I} + \frac{\mu}{2}\boldsymbol{B}_k\right)^{-1}\left(\boldsymbol{I} - \frac{\mu}{2}\boldsymbol{B}_k\right)\boldsymbol{W}_k^{(\cdot)} \tag{64}$$

where $\boldsymbol{B}_k = \boldsymbol{M}_k(\boldsymbol{W}_k^{(\cdot)})^T - \boldsymbol{W}_k^{(\cdot)}\boldsymbol{M}_k^T$ and

$$\boldsymbol{M}_{ij} \triangleq \frac{\partial F_\tau(\tilde{\boldsymbol{c}}, R, \boldsymbol{\theta})}{\partial \boldsymbol{W}_{ij}^{(\cdot)}}. \tag{65}$$

Remark 8: For $\boldsymbol{R}^{(\cdot)}$ and $\boldsymbol{b}^{(\cdot)}$, we first compute the gradient of the objective function with respect to the chosen parameter as in (65). We then obtain $\boldsymbol{B}_k$ according to the chosen parameter. Using $\boldsymbol{B}_k$, we update the chosen parameter as in (64).

Remark 9: In the semisupervised framework, we have the following optimization problem for our SVDD-based algorithms [35]:

$$\min_{\boldsymbol{\theta},\tilde{\boldsymbol{c}},R,\xi,\gamma,\eta} \; R^2 - C_1\gamma + C_2\sum_{i=1}^{l}\xi_i + C_3\sum_{j=l+1}^{l+k}\eta_j \tag{66}$$

$$\text{s.t.: } \|\bar{\boldsymbol{h}}_i - \tilde{\boldsymbol{c}}\|^2 - R^2 \le \xi_i, \quad \xi_i \ge 0 \; \forall_{i=1}^{l} \tag{67}$$

$$y_j(\|\bar{\boldsymbol{h}}_j - \tilde{\boldsymbol{c}}\|^2 - R^2) \le -\gamma + \eta_j \quad \eta_j \ge 0 \; \forall_{j=l+1}^{l+k} \tag{68}$$

$$\boldsymbol{W}^{(\cdot)T}\boldsymbol{W}^{(\cdot)} = \boldsymbol{I}, \boldsymbol{R}^{(\cdot)T}\boldsymbol{R}^{(\cdot)} = \boldsymbol{I} \text{ and } \boldsymbol{b}^{(\cdot)T}\boldsymbol{b}^{(\cdot)} = 1 \tag{69}$$

where $\eta \in \mathbb{R}$ is a slack variable as ξ, $\gamma \in \mathbb{R}$ is the margin of the labeled data instances, C_1, C_2, and C_3 are tradeoff parameters, k and l are the number of the labeled and unlabeled data instances, respectively, and $y_j \in \{-1, +1\}$ represents the label of the j^{th} data instance.

For the quadratic programming-based training method, we modify all the steps from (48) to (54), Remark 6 and Remark 7 with respect to (66)–(69). In a similar manner, we modify the equations from (55) to (65) according to (66)–(69) in order to obtain the gradient-based training method in the semisupervised framework. For the supervised implementations, we follow the same procedures with the semisupervised implementations for $l = 0$ case.

The complete algorithm is provided in Algorithm 4. In the following, we provide the convergence proof as in the OC-SVM case.

Theorem 2: Let $\tilde{\boldsymbol{c}}_\tau$ and R_τ^2 be the solutions of (57) for any fixed $\boldsymbol{\theta}$. Then, $\tilde{\boldsymbol{c}}_\tau$ and R_τ^2 are unique and $F_\tau(\tilde{\boldsymbol{c}}_\tau, R_\tau, \boldsymbol{\theta})$ converges to the minimum of $F(\tilde{\boldsymbol{c}}, R, \boldsymbol{\theta})$, i.e., defined as

$$F(\tilde{\boldsymbol{c}}, R, \boldsymbol{\theta}) \triangleq R^2 + \frac{1}{n\lambda}\sum_{i=1}^{n} G(\Psi_{R,\tilde{\boldsymbol{c}}}(\bar{\boldsymbol{h}}_i)).$$

Proof of Theorem 2: The proof of the theorem is given in Appendix C. □

Algorithm 4 Gradient-Based Training for the Anomaly Detection Algorithm Based on SVDD

1: Initialize the LSTM parameters as $\boldsymbol{\theta}_0$ and the SVDD parameters as $\tilde{\boldsymbol{c}}_0$ and R_0^2
2: Determine a threshold ϵ as convergence criterion
3: $k = -1$
4: **do**
5: $k = k + 1$
6: Using $\boldsymbol{\theta}_k$, obtain $\{\bar{\boldsymbol{h}}\}_{i=1}^n$ according to Fig. 2
7: Obtain $\tilde{\boldsymbol{c}}_{k+1}$, R_{k+1}^2 and $\boldsymbol{\theta}_{k+1}$ using (60), (62), (64) and Remark 8
8: **while** $(F_\tau(\tilde{\boldsymbol{c}}_{k+1}, R_{k+1}, \boldsymbol{\theta}_{k+1}) - F_\tau(\tilde{\boldsymbol{c}}_k, R_k, \boldsymbol{\theta}_k))^2 > \epsilon$
9: Detect anomalies using (47) evaluated at $\tilde{\boldsymbol{c}}_k$, R_k^2 and $\boldsymbol{\theta}_k$

IV. Simulations

In this section, we demonstrate the performances of the algorithms on several different data sets. We first evaluate the performances on a data set that contains variable length data sequences, i.e., the digit data set [36]. We then compare the anomaly detection performances on several different benchmark real data sets such as the occupancy [37], Hong Kong Exchange (HKE) rate [38], http [39], and Alcoa stock price [40] data sets. While performing experiments on real benchmark data sets, we also include the GRU-based algorithms in order to compare their performances with the LSTM-based ones. Moreover, we also measure the training times of the algorithms and perform an experiment to observe the effects of the orthogonality constraint in this section. Note that since the introduced algorithms have bounded functions, e.g., the sigmoid function in the LSTM architecture, for all the experiments in this section, we normalize each dimension of the data sets into $[-1, 1]$.

Throughout this section, we denote the LSTM-based OC-SVM anomaly detectors that are trained with the gradient and quadratic programming-based algorithms as "LSTM-GSVM" and "LSTM-QPSVM," respectively. In a similar manner, we use "LSTM-GSVDD" and "LSTM-QPSVDD" for the SVDD-based anomaly detectors. Moreover, for the labels of the GRU-based algorithms, we replace the LSTM prefix with GRU.

A. Anomaly Detection for Variable Length Data Sequences

In this section, we first evaluate the performances of the introduced anomaly detectors on the digit data set [36]. In this data set, we have the pixel samples of digits, which were written on a tablet by several different authors [36]. Since the speed of writing varies from person to person, the number of samples for a certain digit might significantly

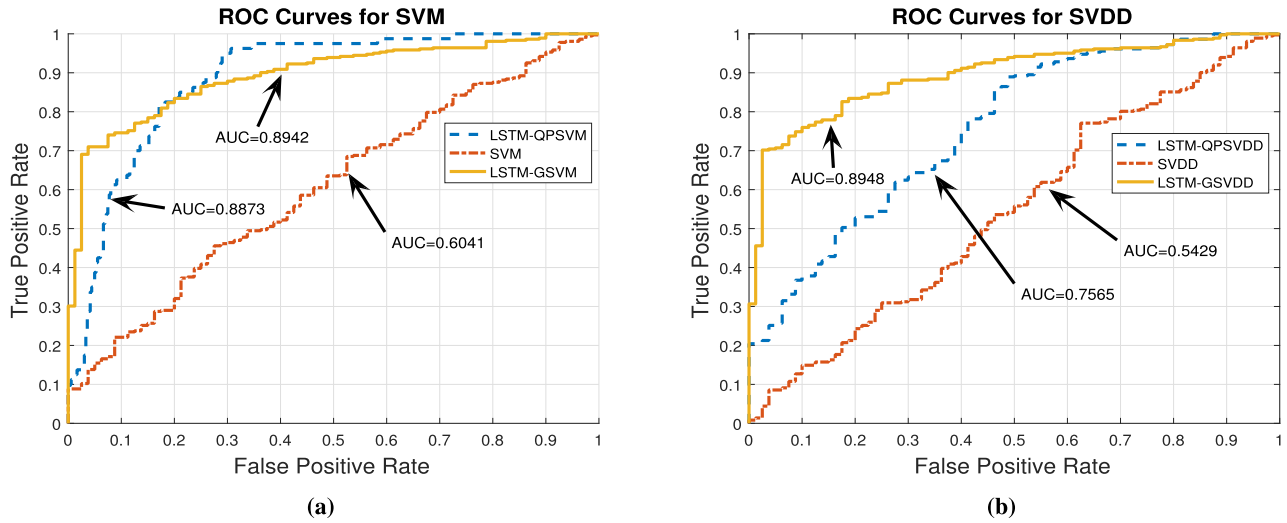

Fig. 4. ROC curves of the algorithms for the digit data set, where we consider digit "0" as normal and digit "9" as anomaly (a) for the SVM-based algorithms and (b) for the SVDD-based algorithms.

differ. The introduced algorithms are able to process such kind of sequences, thanks to their generic structure in Fig. 2. However, the conventional OC-SVM and SVDD algorithms cannot directly process these sequences [6], [7]. For these algorithms, we take the mean of each sequence to obtain a fixed-length vector sequence, i.e., 2-D in this case (two coordinates of a pixel). In order to evaluate the performances, we first choose a digit as normal and another digit as an anomaly. We emphasize that we randomly choose digits for illustration and obtain similar performances for the other digits. We then divide the samples of these digits into training and test parts, where we allocate 60% of the samples for the training part and 40% for the test part. In both the training and test parts, we select the samples so that 10% of the samples are anomalies. Then, using the training part, we optimize the parameters of each algorithm using twofold cross validation, where we also select a certain crucial parameter, e.g., μ. This procedure results in $\mu = 0.05$, 0.001, 0.05, and 0.01 for LSTM-GSVM, LSTM-QPSVM, LSTM-GSVDD, and LSTM-QPSVDD, respectively. Furthermore, we select the output dimension of the LSTM architecture as $m = 2$ and the regularization parameter as $\lambda = 0.5$ for all the algorithms. For the implementation of the conventional OC-SVM and SVDD algorithms, we use the LIBSVM library and their parameters are selected in a similar manner via built-in optimization tools of LIBSVM [41].

Here, we use the area under the receiver operating characteristic (ROC) curve as a performance metric [42]. In a ROC curve, we plot a true positive rate (TPR) as a function of false positive rate (FPR). Area under this curve, i.e., also known as AUC, is a well-known performance measure for anomaly detection tasks [42]. In Fig. 4(a) and (b), we illustrate the ROC curves and provide the corresponding AUC scores, where we label digit "0" and "9" as normal and anomaly, respectively. For the OC-SVM and SVDD algorithms, since we directly take the mean of variable length data sequences to obtain fixed-length sequences, they achieve significantly lower AUC scores compared to the introduced LSTM-based methods. Among the LSTM-based methods, LSTM-GSVM slightly outperforms LSTM-QPSVM. On the other hand, LSTM-GSVDD achieves significantly higher AUC than LSTM-QPSVDD. Since the quadratic programming-based training method depends on the separated consecutive updates of the LSTM and SVM (or SVDD) parameters, it might not converge to even a local minimum. However, the gradient-based method can guarantee convergence to at least a local minimum given a proper choice of the learning rate [33]. Thus, although these methods might provide similar performances as in Fig. 4(a), it is also expected to obtain much higher performance from the gradient-based method for certain cases as shown in Fig. 4(b). However, overall, the introduced algorithms provide significantly higher AUC than the conventional methods.

Besides the previous scenario, we also consider a scenario, where we label digit "1" and "7" as normal and anomaly, respectively. In Fig. 5(a) and (b), we illustrate the ROC curves and provide the corresponding AUC scores. As in the previous scenario, for both the SVM and SVDD cases, the introduced algorithms achieve higher AUC scores than the conventional algorithms. Among the introduced algorithms, LSTM-GSVM and LSTM-GSVDD achieve the highest AUC scores for the SVM and SVDD cases, respectively. Furthermore, the AUC score of each algorithm is much lower compared to the previous case due to the similarity between digits "1" and "7."

In addition to the digit data set, we perform another experiment that handles variable length data sequences. In this experiment, we evaluate the anomaly detection performances of the algorithms on a financial data set, i.e., the Ford stock price data set [43]. Here, we have daily stock price values. For our anomaly detection framework, we first artificially introduce anomalies via a Gaussian distribution with the mean and ten times the variance of the training data. We then select certain parts of the time series data by applying a variable length time windowing operation, thus we obtain variable length data

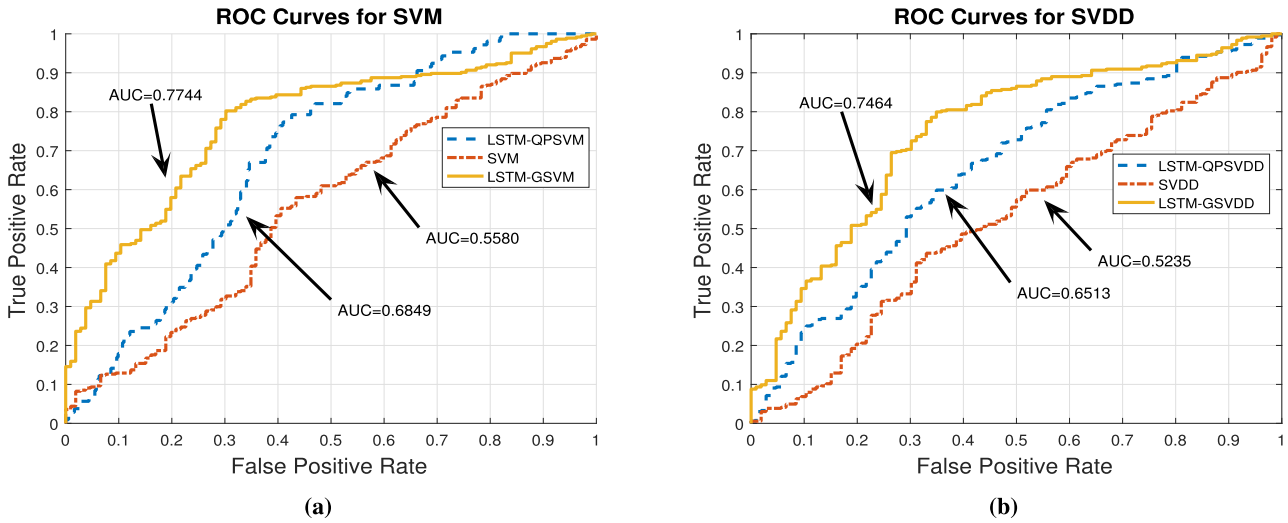

Fig. 5. ROC curves of the algorithms for the digit data set, where we consider digit "1" as normal and digit "7" as anomaly (a) for the SVM-based algorithms and (b) for the SVDD-based algorithms.

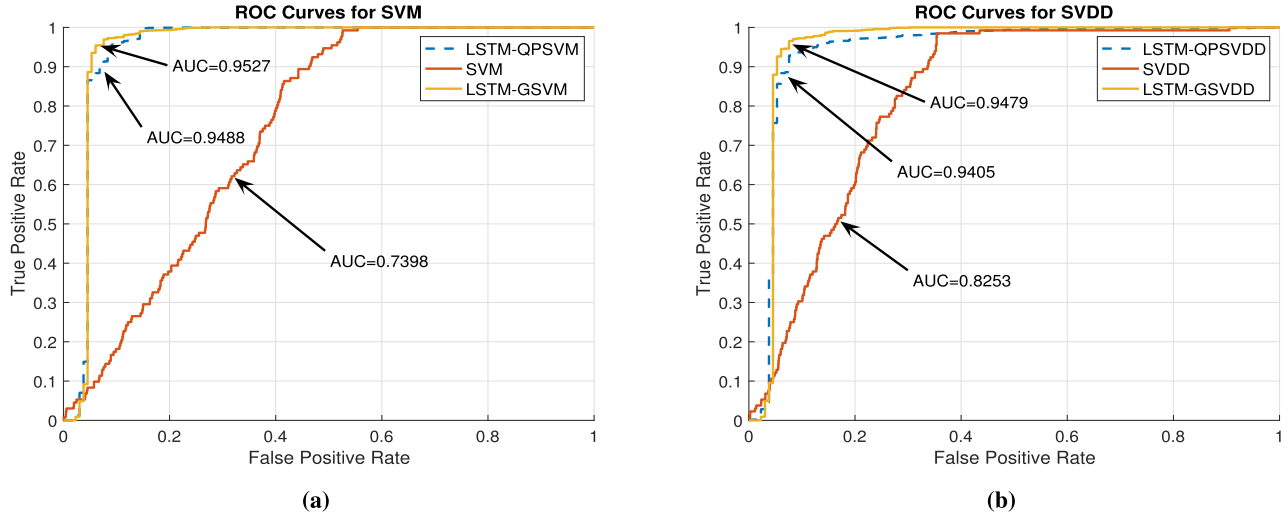

Fig. 6. ROC curves of the stock price data set for (a) SVM-based algorithms and (b) SVDD-based algorithms.

sequences. Moreover, unlike the previous cases, we choose $\mu = 0.01, 0.001, 0.001$, and 0.005 for LSTM-GSVM, LSTM-QPSVM, LSTM-GSVDD, and LSTM-QPSVDD, respectively.

In Fig. 6, we observe that the LSTM-based algorithms achieve considerably higher AUC scores than the SVM and SVDD algorithms. Among the LSTM-based methods, LSTM-GSVM slightly outperforms LSTM-QPSVM. Similarly, LSTM-GSVDD achieves slightly higher AUC than LSTM-QPSVDD. Moreover, as in the previous experiments, the gradient-based training method provides higher performance compared to the quadratic programming-based method, thanks to its learning capabilities.

B. Benchmark Real Data sets

In this section, we compare the AUC scores of each algorithm on several different real benchmark data sets. Moreover, we provide the training times and evaluate the effects of the orthogonality constraint on these data sets. Since our approach in this article is generic, in addition to the LSTM-based algorithms, we also implement our approach on the recently introduced RNN architecture, i.e., the GRU architecture, which is defined by the following equations [21]:

$$\tilde{\boldsymbol{z}}_{i,j} = \sigma(\boldsymbol{W}^{(\tilde{z})}\boldsymbol{x}_{i,j} + \boldsymbol{R}^{(\tilde{z})}\boldsymbol{h}_{i,j-1}) \tag{70}$$

$$\boldsymbol{r}_{i,j} = \sigma(\boldsymbol{W}^{(r)}\boldsymbol{x}_{i,j} + \boldsymbol{R}^{(r)}\boldsymbol{h}_{i,j-1}) \tag{71}$$

$$\tilde{\boldsymbol{h}}_{i,j} = g(\boldsymbol{W}^{(\tilde{h})}\boldsymbol{x}_{i,j} + \boldsymbol{r}_{i,j} \odot (\boldsymbol{R}^{(\tilde{h})}\boldsymbol{h}_{i,j-1})) \tag{72}$$

$$\boldsymbol{h}_{i,j} = \tilde{\boldsymbol{h}}_{i,j} \odot \tilde{\boldsymbol{z}}_{i,j} + \boldsymbol{h}_{i,j-1} \odot (\boldsymbol{1} - \tilde{\boldsymbol{z}}_{i,j}) \tag{73}$$

where $\boldsymbol{h}_{i,j} \in \mathbb{R}^m$ is the output vector and $\boldsymbol{x}_{i,j} \in \mathbb{R}^p$ is the input vector. Furthermore, $\boldsymbol{W}^{(\cdot)}$ and $\boldsymbol{R}^{(\cdot)}$ are the parameters of the GRU, where the sizes are selected according to the dimensionality of the input and output vectors. We then replace (1)–(6) with (70)–(73) in Fig. 2 to obtain GRU-based anomaly detectors. Note that in this section, we also include the LSTM-based anomaly detection approach in [10] and [25] as another benchmark performance criterion, especially for the experiments with time series data.

TABLE I
AUC SCORES OF THE ALGORITHMS FOR THE OCCUPANCY, HKE RATE, HTTP, AND ALCOA STOCK PRICE DATA SETS

Algorithms / Datasets	SVM	SVDD	LSTM	LSTM-QPSVM	LSTM-GSVM	LSTM-QPSVDD	LSTM-GSVDD	GRU-QPSVM	GRU-GSVM	GRU-QPSVDD	GRU-GSVDD
Occupancy	0.8676	0.6715	0.7444	0.8917	0.8957	0.7869	0.8609	0.8719	0.9059	0.7417	0.9109
HKE	0.8000	0.8500	0.9756	0.9467	0.9783	0.8560	0.9753	0.8481	0.9517	0.8794	0.9527
Http	0.9963	0.9993	0.9901	0.9992	0.9997	0.9994	0.9994	0.9986	0.9989	0.9999	0.9994
Alcoa	0.7197	0.9390	0.9658	0.9496	0.9515	0.9415	0.9507	0.7583	0.9422	0.9662	0.9394

1) Occupancy Detection: We first evaluate the performances of the algorithms on the occupancy data set [37]. In this data set, we have five features, which are relative humidity percentage, light (in lux), carbon dioxide level (in ppm), temperature (in Celsius), and humidity ratio, and our aim is to determine whether an office room is occupied or not based on the features. Here, we use the same procedure with Section IV-A to separate the test and training data. Moreover, using the training data, we select $\mu = 0.05$, 0.05, 0.001, and 0.01 for LSTM-GSVM, LSTM-QPSVM, LSTM-GSVDD, and LSTM-QPSVDD, respectively. Note that, for the GRU-based algorithms in this section, we use the same parameter setting with the LSTM-based algorithms. Furthermore, we choose $m = 5$ and $\lambda = 0.5$ for all of the experiments in this section in order to maximize the performances of the algorithms.

As can be seen in Table I, due to their inherent memory, both the LSTM- and GRU-based algorithms achieve considerably high-AUC scores compared to the conventional SVM and SVDD algorithms. Moreover, GRU-GSVDD achieves the highest AUC score among all the algorithms, where the LSTM-based algorithms (LSTM-GSVM and LSTM-QPSVM) also provide comparable AUC scores. Here, we also observe that the gradient-based training method provides higher AUC scores compared to the quadratic programming-based training method, which might stem from its separated update procedure that does not guarantee convergence to a certain local minimum.

2) Anomalous Exchange Rate Detection: Other than the occupancy data set, we also perform an experiment on the HKE rate data set in order to examine the performances for a real-life financial scenario. In this data set, we have the amount of Hong Kong dollars that one can buy for one US dollar each day. In order to introduce anomalies to this data set, we artificially add samples from a Gaussian distribution with the mean and ten times the variance of the training data. Furthermore, using the training data, we select $\mu = 0.01$, 0.005, 0.05, and 0.05 for LSTM-GSVM, LSTM-QPSVM, LSTM-GSVDD, and LSTM-QPSVDD, respectively.

In Table I, we illustrate the AUC scores of the algorithms on the HKE rate data set. Since we have time-series data, both the LSTM- and GRU-based algorithms naturally outperform the conventional methods, thanks to their inherent memory, which preserves sequential information. Moreover, since the LSTM architecture also controls its memory content via an output gate unlike the GRU architecture [21], we obtain the highest AUC scores from LSTM-GSVM. As in the previous cases, the gradient-based training method provides better performance than the quadratic programming-based training.

3) Network Anomaly Detection: We also evaluate the AUC scores of the algorithms on the http data set [39]. In this data set, we have 4 features, which are duration (number of seconds of the connection), network service, number of bytes from source to destination and from destination to source. Using these features, we aim to distinguish normal connections from network attacks. In this experiment, we select $\mu = 0.01$, 0.05, 0.001, and 0.01 for LSTM-GSVM, LSTM-QPSVM, LSTM-GSVDD, and LSTM-QPSVDD, respectively.

We demonstrate the performances of the algorithms on the http data set in Table I. Even though all the algorithms achieve high-AUC scores on this data set, we still observe that the LSTM- and GRU-based algorithms have higher AUC scores than the conventional SVM and SVDD methods. Overall, GRU-QPSVDD achieves the highest AUC score and the quadratic programming-based training methods perform better than the gradient-based training method on this data set. However, since the AUC scores are very high and close to each other, we observe only slight performance improvement for our algorithms in this case.

4) Anomalous Stock Price Detection: As the last experiment, we evaluate the anomaly detection performances of the algorithms on another financial data set, i.e., the Alcoa stock price data set [40]. In this data set, we have daily stock price values. As in the HKE rate data set, we again artificially introduce anomalies via a Gaussian distribution with the mean and ten times the variance of the training data. Moreover, we choose $\mu = 0.01, 0.001, 0.001$, and 0.005 for LSTM-GSVM, LSTM-QPSVM, LSTM-GSVDD, and LSTM-QPSVDD, respectively.

In Table I, we illustrate the AUC scores of the algorithms on the Alcoa stock price data set. Here, we observe that the GRU- and LSTM-based algorithms achieve considerably higher AUC scores than the conventional methods, thanks to their memory structure. Although the LSTM-based algorithms have higher AUC scores in general, we obtain the highest AUC score from GRU-QPSVDD. Moreover, as in the previous experiments, the gradient-based training method provides higher performance compared to the quadratic programming-based method thanks to its learning capabilities.

5) Constraint and Time Complexity Analysis: In Table II, we compare the performance of LSTM-GSVM under three different scenarios, i.e., using the orthogonality constraint, using the conventional ℓ_2 norm regularization constraint and a case without constraint. Note that since LSTM-GSVM

TABLE II

AUC SCORES OF LSTM-GSVM FOR THE ORTHOGONALITY CONSTRAINT IN (9), ℓ_2 NORM REGULARIZATION CONSTRAINT IN (11), AND NO CONSTRAINT CASES

Constraint Type \ Datasets	HKE	Alcoa	Occupancy	Http
Orthogonality Constraint	0.9783	0.9515	0.8957	0.9997
Regularization Constraint	0.9492	0.9435	0.8939	0.9996
No Constraint	0.9489	0.9435	0.8713	0.9994

TABLE III

TRAINING TIMES (IN SECONDS) OF THE ALGORITHMS. FOR THIS EXPERIMENT, WE USE A COMPUTER THAT HAS I5-6400 PROCESSOR, 2.7 GHz CPU, AND 16 GB RAM

Algorithms / Datasets	LSTM-GSVM	LSTM-QPSVM	LSTM-GSVDD	LSTM-QPSVDD
Alcoa	14.09	31.16	14.06	21.66
HKE	1.63	1.81	1.69	1.82
Occupancy	33.27	119.67	32.97	133.33
Http	20.86	32.57	21.06	36.57
Digit	22.28	23.30	22.53	23.61

provides high AUC scores for all the experiments, we choose it to perform this experiment. We observe that the case with the orthogonality constraint outperforms the other cases. Thus, we use it to improve our detection performance in this article. In addition to this, we measure the training times of the algorithms for all the data sets. In Table III, we observe that the gradient-based algorithms achieve significantly faster training performance compared to the quadratic programming-based methods due to the highly complicated structure of the quadratic programming optimization method.

V. CONCLUDING REMARKS

In this article, we study anomaly detection in an unsupervised framework and introduce LSTM-based algorithms. In particular, we have introduced a generic LSTM-based structure in order to process variable-length data sequences. After obtaining fixed-length sequences via our LSTM-based structure, we introduce a scoring function for our anomaly detectors based on the OC-SVM [6] and SVDD [7] algorithms. As the first time in the literature, we jointly optimize the parameters of both the LSTM architecture and the final scoring function of the OC-SVM (or SVDD) formulation. To jointly optimize the parameters of our algorithms, we have also introduced gradient and quadratic programming-based training methods with different algorithmic merits, where we extend our derivations for these algorithms to the semisupervised and fully supervised frameworks. In order to apply the gradient-based training method, we modify the OC-SVM and SVDD formulations and then provide the convergence results of the modified formulations to the actual ones. Therefore, we obtain highly effective anomaly detection algorithms, especially for time series data, that are able to process variable length data sequences. In our simulations, due to the generic structure of our approach, we have also introduced GRU-based anomaly detection algorithms. Through an extensive set of experiments, we illustrate significant performance improvements achieved by our algorithms with respect to the conventional methods [6], [7], [10] over several different real and simulated data sets.

APPENDIX A
PROOF OF PROPOSITION I

In order to simplify our notation, for any given $\boldsymbol{w}$, $\boldsymbol{\theta}$, $\boldsymbol{X}_i$, and ρ, we denote $\beta_{\boldsymbol{w},\rho}(\bar{\boldsymbol{h}}_i)$ as Ω. We first show that $S_\tau(\Omega) \geq G(\Omega)$, $\forall \tau > 0$. Since

$$\begin{aligned} S_\tau(\Omega) &= \frac{1}{\tau}\log(1+e^{\tau\Omega}) \\ &\geq \frac{1}{\tau}\log(e^{\tau\Omega}) \\ &= \Omega \end{aligned}$$

and $S_\tau(\Omega) \geq 0$, we have $S_\tau(\Omega) \geq G(\Omega) = \max\{0, \Omega\}$. Then, for any $\Omega \geq 0$, we have

$$\begin{aligned} \frac{\partial S_\tau(\Omega)}{\partial \tau} &= \frac{-1}{\tau^2}\log(1+e^{\tau\Omega}) + \frac{1}{\tau}\frac{\Omega e^{\tau\Omega}}{1+e^{\tau\Omega}} \\ &< \frac{-1}{\tau}\Omega + \frac{1}{\tau}\frac{\Omega e^{\tau\Omega}}{1+e^{\tau\Omega}} \\ &\leq 0 \end{aligned}$$

and for any $\Omega < 0$, we have

$$\begin{aligned} \frac{\partial S_\tau(\Omega)}{\partial \tau} &= \frac{-1}{\tau^2}\log(1+e^{\tau\Omega}) + \frac{1}{\tau}\frac{\Omega e^{\tau\Omega}}{1+e^{\tau\Omega}} \\ &< 0, \end{aligned}$$

thus, we conclude that $S_\tau(\Omega)$ is a monotonically decreasing function of τ. As the last step, we derive an upper bound for the difference $S_\tau(\Omega) - G(\Omega)$. For $\Omega \geq 0$, the derivative of the difference is as follows:

$$\frac{\partial(S_\tau(\Omega) - G(\Omega))}{\partial \Omega} = \frac{e^{\tau\Omega}}{1+e^{\tau\Omega}} - 1 < 0,$$

hence, the difference is a decreasing function of Ω for $\Omega \geq 0$. Therefore, the maximum value is $\log(2)/\tau$ and it occurs at $\Omega = 0$. Similarly, for $\Omega < 0$, the derivative of the difference is positive, which shows that the maximum for the difference occurs at $\Omega = 0$. With this result, we obtain the following bound:

$$\frac{\log(2)}{\tau} = \max_{\Omega}\big(S_\tau(\Omega) - G(\Omega)\big). \tag{74}$$

Using (74), for any $\epsilon > 0$, we can choose τ sufficiently large so that $S_\tau(\Omega) - G(\Omega) < \epsilon$. Hence, as τ increases, $S_\tau(\Omega)$ uniformly converges to $G(\Omega)$. By averaging (74) over all the data points and multiplying with $1/\lambda$, we obtain

$$\frac{\log(2)}{\lambda\tau} = \max_{\boldsymbol{w},\rho,\boldsymbol{\theta}}\big(F_\tau(\boldsymbol{w},\rho,\boldsymbol{\theta}) - F(\boldsymbol{w},\rho,\boldsymbol{\theta})\big)$$

which proves the uniform convergence of $F_\tau(\cdot,\cdot,\cdot)$ to $F(\cdot,\cdot,\cdot)$.

APPENDIX B
PROOF OF THEOREM I

We have the following Hessian matrix of $F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta})$ with respect to $\boldsymbol{w}$:

$$\nabla^2_{\boldsymbol{w}} F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta}) = \boldsymbol{I} + \frac{\tau}{n\lambda} \sum_{i=1}^{n} \frac{e^{\tau\beta \boldsymbol{w}_{,\rho}(\bar{\mathrm{h}}_i)}}{(1 + e^{\tau\beta \boldsymbol{w}_{,\rho}(\bar{\mathrm{h}}_i)})^2} \bar{\boldsymbol{h}}_i \bar{\boldsymbol{h}}_i^T,$$

which satisfies $\boldsymbol{v}^T \nabla^2_{\boldsymbol{w}} F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta}) \boldsymbol{v} > 0$ for any nonzero column vector $\boldsymbol{v}$. Hence, the Hessian matrix is positive definite, which shows that $F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta})$ is strictly convex function of $\boldsymbol{w}$. Consequently, the solution $\boldsymbol{w}_\tau$ is both global and unique given any ρ and $\boldsymbol{\theta}$. In addition, we have the following second-order derivative for ρ:

$$\frac{\partial^2 F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta})}{\partial \rho^2} = \frac{\tau}{n\lambda} \sum_{i=1}^{n} \frac{e^{\tau\beta \boldsymbol{w}_{,\rho}(\bar{\mathrm{h}}_i)}}{(1 + e^{\tau\beta \boldsymbol{w}_{,\rho}(\bar{\mathrm{h}}_i)})^2} > 0,$$

which implies that $F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta})$ is strictly convex function of ρ. As a result, the solution ρ_τ is both global and unique for any given $\boldsymbol{w}$ and $\boldsymbol{\theta}$.

Let $\boldsymbol{w}^*$ and ρ^* be the solutions of (27) for any fixed $\boldsymbol{\theta}$. From the proof of Proposition 1, we have

$$\begin{aligned} F_\tau(\boldsymbol{w}^*, \rho^*, \boldsymbol{\theta}) \geq F_\tau(\boldsymbol{w}_\tau, \rho_\tau, \boldsymbol{\theta}) &\geq F(\boldsymbol{w}_\tau, \rho_\tau, \boldsymbol{\theta}) \\ &\geq F(\boldsymbol{w}^*, \rho^*, \boldsymbol{\theta}). \end{aligned} \tag{75}$$

Using the convergence result in Proposition 1 and (75), we have

$$\begin{aligned} &\lim_{\tau\to\infty} F_\tau(\boldsymbol{w}_\tau, \rho_\tau, \boldsymbol{\theta}) \leq \lim_{\tau\to\infty} F_\tau(\boldsymbol{w}^*, \rho^*, \boldsymbol{\theta}) = F(\boldsymbol{w}^*, \rho^*, \boldsymbol{\theta}) \\ &\lim_{\tau\to\infty} F_\tau(\boldsymbol{w}_\tau, \rho_\tau, \boldsymbol{\theta}) \geq F(\boldsymbol{w}^*, \rho^*, \boldsymbol{\theta}) \end{aligned}$$

which proves the following equality:

$$\lim_{\tau\to\infty} F_\tau(\boldsymbol{w}_\tau, \rho_\tau, \boldsymbol{\theta}) = F(\boldsymbol{w}^*, \rho^*, \boldsymbol{\theta}).$$

APPENDIX C
PROOF OF THEOREM II

We have the following Hessian matrix of $F_\tau(\tilde{\boldsymbol{c}}, R, \boldsymbol{\theta})$ with respect to $\tilde{\boldsymbol{c}}$:

$$\nabla^2_{\tilde{\boldsymbol{c}}} F_\tau(\tilde{\boldsymbol{c}}, R, \boldsymbol{\theta}) = \sum_{i=1}^{n} \frac{2\boldsymbol{I}(\Omega_i + \Omega_i^2) + 4\tau\Omega_i(\tilde{\boldsymbol{c}} - \bar{\boldsymbol{h}}_i)(\tilde{\boldsymbol{c}} - \bar{\boldsymbol{h}}_i)^T}{n\lambda\left(1 + \Omega_i\right)^2},$$

where $\Omega_i = e^{\tau \Psi_{\tilde{\boldsymbol{c}}, R}(\bar{\mathrm{h}}_i)}$, which implies $\boldsymbol{v}^T \nabla^2_{\tilde{\boldsymbol{c}}} F_\tau(\tilde{\boldsymbol{c}}, R, \boldsymbol{\theta}) \boldsymbol{v} > 0$ for any nonzero column vector $\boldsymbol{v}$. Thus, the Hessian matrix is positive definite, which shows that $F_\tau(\boldsymbol{w}, \rho, \boldsymbol{\theta})$ is strictly convex function of $\tilde{\boldsymbol{c}}$. As a result, the solution $\tilde{\boldsymbol{c}}_\tau$ is both global and unique given any R and $\boldsymbol{\theta}$. In addition to this, we have the following second-order derivative for R^2:

$$\frac{\partial^2 F_\tau(\tilde{\boldsymbol{c}}, R, \boldsymbol{\theta})}{\partial (R^2)^2} = \frac{\tau}{n\lambda} \sum_{i=1}^{n} \frac{e^{\tau \Psi_{\tilde{\boldsymbol{c}}, R}(\bar{\mathrm{h}}_i)}}{\left(1 + e^{\tau \Psi_{\tilde{\boldsymbol{c}}, R}(\bar{\mathrm{h}}_i)}\right)^2} > 0,$$

which implies that $F_\tau(\tilde{\boldsymbol{c}}, R, \boldsymbol{\theta})$ is strictly convex function of R^2. Therefore, the solution R^2_τ is both global and unique for any given $\tilde{\boldsymbol{c}}$ and $\boldsymbol{\theta}$.

The convergence proof directly follows the proof of Theorem 1.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, p. 15, Jul. 2009. doi: 10.1145/1541880.1541882.

[2] N. Görnitz, L. A. Lima, K. Müller, M. Kloft, and S. Nakajima, "Support vector data descriptions and k-means clustering: One class?" *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 3994–4006, Sep. 2018.

[3] Z. Ghafoori, S. M. Erfani, S. Rajasegarar, J. C. Bezdek, S. Karunasekera, and C. Leckie, "Efficient unsupervised parameter estimation for one-class support vector machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 5057–5070, Oct. 2018.

[4] Q. Chen, R. Luley, Q. Wu, M. Bishop, R. W. Linderman, and Q. Qiu, "AnRAD: A neuromorphic anomaly detection framework for massive concurrent data streams," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1622–1636, May 2018.

[5] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire, "Novelty detection using level set methods," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 576–588, Mar. 2015.

[6] 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.

[7] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Mach. Learn.*, vol. 54, no. 1, pp. 45–66, Jan. 2004. doi: 10.1023/B:MACH.0000008084.60811.49.

[8] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 3, Jul. 2003, pp. 1741–1745.

[9] R. Zhang, S. Zhang, S. Muthuraman, and J. Jiang, "One class support vector machine for anomaly detection in the communication network performance data," in *Proc. 5th Conf. Appl. Electromagn., Wireless Opt. Commun. (ELECTROSCIENCE)*. Stevens Point, WI, USA: World Scientific and Engineering Academy and Society, 2007, pp. 31–37.

[10] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, *Long Short Term Memory Networks for Anomaly Detection in Time Series*. Louvain-la-Neuve, Belgium: Presses Universitaires de Louvain, 2015, p. 89.

[11] J. Zhao and L. Itti, "Classifying time series using local descriptors with hybrid sampling," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 3, pp. 623–637, Mar. 2016.

[12] R. C. Venkatesan and A. Plastino, "Fisher information framework for time series modeling," *Phys. A. Stat. Mech. Appl.*, vol. 480, pp. 22–38, Aug. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437117301863

[13] K. T. Abou-Moustafa, M. Cheriet, and C. Y. Suen, "Classification of time-series data using a generative/discriminative hybrid," in *Proc. 9th Int. Workshop Frontiers Handwriting Recognit.*, Oct. 2004, pp. 51–56.

[14] K. T. Abou-Moustafa. (2003). *A Generative-Discriminative Framework for Time-Series Data Classification*. [Online]. Available: https://spectrum.library.concordia.ca/2392/

[15] H. Debar, M. Becker, and D. Siboni, "A neural network component for an intrusion detection system," in *Proc. IEEE Comput. Soc. Symp. Res. Secur. Privacy*, May 1992, pp. 240–250.

[16] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[17] K. Greff, R. K. Srivastava, J. Koutnìk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[19] R. Kozma, M. Kitamura, M. Sakuma, and Y. Yokoyama, "Anomaly detection by neural network models and statistical time series analysis," in *Proc. IEEE Int. Conf. Neural Netw., IEEE World Congr. Comput. Intell.*, vol. 5, Jun. 1994, pp. 3207–3210.

[20] C. M. Bishop, "Novelty detection and neural network validation," *IEE Proc.-Vis., Image Signal Process.*, vol. 141, pp. 217–222, Aug. 1994.

[21] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: https://arxiv.org/abs/1412.3555

[22] Y. Wang, J. Wong, and A. Miner, "Anomaly intrusion detection using one class SVM," in *Proc. IEEE 5th Annu. SMC Inf. Assurance Workshop*, Jun. 2004, pp. 358–364.

[23] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a time series," *Phys. Rev. Lett.*, vol. 45, no. 9, p. 712, Sep. 1980.

[24] R. Zhang, S. Zhang, Y. Lan, and J. Jiang, "Network anomaly detection using one class support vector machine," in *Proc. Int. MultiConf. Eng. Comput. Scientists*, vol. 1, 2008, pp. 1–5.
[25] S. Chauhan and L. Vig, "Anomaly detection in ECG time signals via deep long short-term memory networks," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2015, pp. 1–7.
[26] T. Ergen and S. S. Kozat, "Online training of LSTM networks in distributed systems for variable length data sequences," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 5159–5165, Oct. 2018.
[27] L. Jing *et al.*, "Gated orthogonal recurrent units: On learning to forget," 2017, *arXiv:1706.02761*. [Online]. Available: https://arxiv.org/abs/1706.02761
[28] S. Wisdom *et al.*, "Full-capacity unitary recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4880–4888.
[29] M. Bai, B. Zhang, and J. Gao, "Tensorial recurrent neural networks for longitudinal data analysis," 2017, *arXiv:1708.00185*. [Online]. Available: https://arxiv.org/abs/1708.00185
[30] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Tech. Rep. MSR-TR-98-14, Apr. 1998, p. 21. [Online]. Available: https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/
[31] Z. Wen and W. Yin, "A feasible method for optimization with orthogonality constraints," *Math. Program.*, vol. 142, no. 1, pp. 397–434, Dec. 2013.
[32] A. Beck, "On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes," *SIAM J. Optim.*, vol. 25, no. 1, pp. 185–209, Jan. 2015.
[33] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ, USA: Wiley, 2003.
[34] K. P. Bennett and A. Demiriz, "Semi-supervised support vector machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 368–374.
[35] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld, "Toward supervised anomaly detection," *J. Artif. Intell. Res.*, vol. 46, no. 1, pp. 235–262, Jan. 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2512538.2512545
[36] D. Dua and C. Graff, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, Irvine, CA, USA, Tech. Rep., 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[37] L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and CO_2 measurements using statistical learning models," *Energy Buildings*, vol. 112, pp. 28–39, Jan. 2016.
[38] E. W. Frees. *Regression Modelling With Actuarial and Financial Applications*. Accessed: May 1, 2018. [Online]. Available: http://instruction.bus.wisc.edu/jfrees/jfreesbooks/Regression%20Modeling/BookWebDec2010/data.html
[39] S. Rayana. (2016). *ODDS Library*. [Online]. Available: http://odds.cs.stonybrook.edu
[40] *Summary for Alcoa Inc. Common Stock*. Accessed: May 1, 2018. [Online]. Available: http://finance.yahoo.com/quote/AA?ltr=1
[41] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm
[42] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognit.*, vol. 30, no. 7, pp. 1145–1159, 1997.
[43] *Summary for Ford Inc. Common Stock*. Accessed: May 1, 2018. [Online]. Available: http://finance.yahoo.com/quote/AA?ltr=1

Tolga Ergen received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2016 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Electrical Engineering Department, Stanford University, Stanford, CA, USA.

His current research interests include machine learning, optimization, and neural networks.

Suleyman Serdar Kozat (A'10–M'11–SM'11) received the B.S. degree (Hons.) from Bilkent University, Ankara, Turkey, in 1998, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 2001 and 2004, respectively.

He joined the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, as a Research Staff Member and later became a Project Leader with the Pervasive Speech Technologies Group, where he focused on problems related to statistical signal processing and machine learning. He was a Research Associate with the Cryptography and Anti-Piracy Group, Microsoft Research, Redmond, WA, USA. He is currently a Professor with the Electrical and Electronics Engineering Department, Bilkent University. He has coauthored more than 200 papers in refereed high impact journals and conference proceedings and holds several patent inventions (used in several different Microsoft and IBM products). He holds several patent inventions due to his research accomplishments with the IBM Thomas J. Watson Research Center and Microsoft Research. His current research interests include cyber security, anomaly detection, big data, data intelligence, adaptive filtering, and machine learning algorithms for signal processing.

Dr. Kozat received many international and national awards. He is the Elected President of the IEEE Signal Processing Society, Turkey Chapter.