International Journal of Data Science and Analytics (2019) 8:137–164
https://doi.org/10.1007/s41060-019-00186-0

REGULAR PAPER


# dLSTM: a new approach for anomaly detection using deep learning with delayed prediction

**Shigeru Maya**[1] · **Ken Ueno**[1] · **Takeichiro Nishikawa**[1]

Received: 24 May 2017 / Accepted: 27 April 2019 / Published online: 15 May 2019


**Abstract**
In this paper, we propose *delayed Long Short-Term Memory* (dLSTM), an anomaly detection method for time-series data. We first build a predictive model from normal (non-anomalous) training data, then perform anomaly detection based on the prediction error for observed data. However, there are multiple states in the waveforms of normal data, which may lower prediction accuracy. To deal with this problem, we utilize multiple prediction models based on LSTM for anomaly detection. In this scheme, the prediction accuracy strongly depends on the method of selecting a proper predictive model from multiple possible models. We propose a novel method to determine the proper predictive model for anomaly detection. Our approach provides multiple predicted value candidates in advance and selects the one that is closest to the measured value. We delay the model selection until the corresponding measured values are acquired. Using this concept for anomaly detection, dLSTM selects the proper predictive model to enhance prediction accuracy. In our experimental evaluation using real and artificial data, dLSTM detects anomalies more accurately than methods in comparison.

**Keywords** Anomaly detection · Deep learning · LSTM · Time-series data

## 1 Introduction

### 1.1 Background

For manufacturers, it is crucial to monitor production facilities to continually ensure that they are working correctly. It is therefore useful to automatically monitor the status of equipments and detect any anomalies. However, there are many kinds of failures, and it would be impractical to construct anomaly detection models suited to all kinds of anomalies.

To address this problem, anomaly detection typically uses models based only on normal (non-anomalous) data. In this approach, a model is constructed for the normal state of equipment. To detect anomalies, the degree of deviation between observed data and the normal state is then computed using this model. An advantage of this method is that anomalous data are not needed to build the model.

In this paper, we measure deviation from the normal state using the prediction error. If the prediction error is small, it is highly probable that the current state is similar to the normal state, while a large prediction error suggests that the current state is abnormal. This assumption is schematically illustrated in Fig. 1. Time-series data change its behavior after an abnormal event occurs (see Fig. 1a). Figure 1b shows the prediction errors at each timestamp, demonstrating that prediction error increases immediately after an anomaly.

In practice, sensor data are often used to monitor the status of equipment. Unfortunately, sensor data tend to contain noise, such as electrical noise on electromagnetic wave-signal lines. Also, sensor data waveforms are complicated in that there are multiple states in the normal data. We call the case in which there are multiple outputs for the same input *multi-mode*. Figure 2 shows this situation, with normal data in the training dataset. These normal data take one of two values, 1.0 and 3.0. Let State A and State B to be normal states setting the value to be 1.0 and 3.0, respectively. We study the learning behavior of this time-series data using predictive models. In Fig. 2, the green box indicates input

✉ Shigeru Maya
shigeru1.maya@toshiba.co.jp

Ken Ueno
ken.ueno@toshiba.co.jp

Takeichiro Nishikawa
takeichiro.nishikawa@toshiba.co.jp

[1] System Engineering Lab., Corporate Research & Development Center, Toshiba Corporation, Kawasaki-shi, Japan

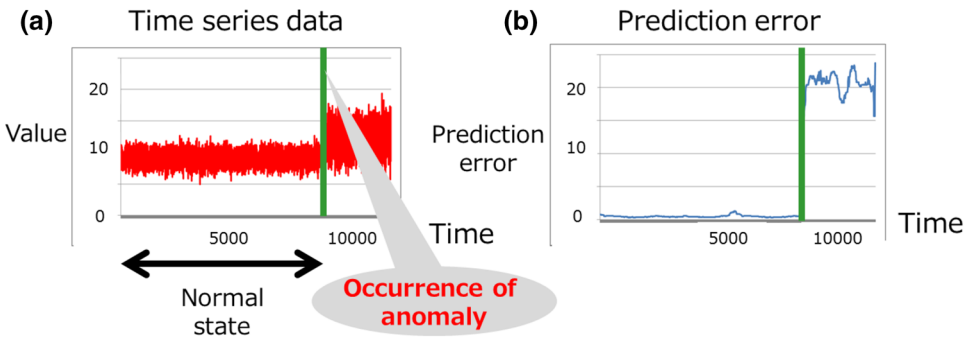


**Fig. 1** Anomaly detection in time-series data. **a** Time-series data containing noise. **b** Prediction error at each timestamp

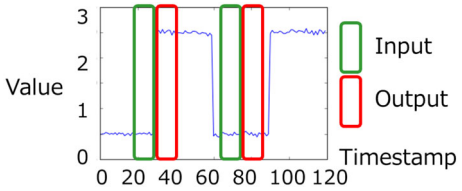


**Fig. 2** An example of a *multi-mode* case. The input–output relationship is not unique

data and the red box shows the output data to be predicted. In this figure, we show two pairs of input–output relationships (State A–State B and State A–State A). Although the two subsequences in the green boxes are similar, the corresponding output is different. In this case, the input-output relationship is not unique. Therefore, precisely predicting the output is challenging. Because these features can be seen even in normal data, the prediction error does not always decrease, thus handling sensor data is a challenging task.

Time-series data often include seasonal or trend components. The behavior itself is therefore different and easy to distinguish. In contrast, multi-mode data are characterized by having multiple outputs for the same input. Since input and output data are not in one-to-one correspondence, it is difficult to identify the proper behavior of the output and overcoming this problem would allow practical use of this method in many industrial fields. To build a precise predictive model, it is important to represent the complicated input–output relationship. Several previous papers [1,12] have used deep learning for anomaly detection. Deep learning can handle complicated data by embedding multiple nonlinear activation functions.

Recently, long short-term memory (LSTM) [7] has also been used in anomaly detection [1,12]. LSTM has an advantage over incorporating the context of the sequence data. If we apply LSTM to time-series data, we can incorporate time dependency.

Although deep learning is a powerful technique, sensor data are often too complicated to capture using only one predictive model due to their noise and multi-mode characteristics. In this paper, we thus propose a new approach, *delayed Long Short-Term Memory* (dLSTM), that uses multiple predictive models embedded in LSTM to address the problem. A gating function [19] identifies a prediction model for the input among a collection of models. Our aim is also to select the proper prediction model like a gating function. Although a gating function attempts to identify a prediction model, it is difficult to select a proper one when we have several outputs for the same input (multi-mode). In multi-kernel learning for one-class classification, the aim is to obtain the proper weight for each kernel to represent the output. However, it is still difficult to derive the proper weight if we have multiple output for the same input.

If we can choose the proper model from among multiple prediction models, we can more flexibly predict future states and reduce prediction error, thereby clarifying the difference in prediction error between normal and abnormal states.

When we have multiple outputs for the same input within normal data, it is highly probable that prediction values of normal data should be similar to at lease one of outputs. It follows that the prediction values of abnormal data should be different from any of the output. In our proposed method, we select the prediction model with least prediction error. When the prediction error is high even if we select the one with least prediction error, the data is different from any of the output and it should be abnormal. We make use of this particular problem of anomaly detection to select the proper predictive model. Our approach uses prediction error to measure deviation from the normal state. Prediction error is defined as the difference between predicted and measured values. Since measured values are necessary to compute prediction error, we only need to obtain the predictive value by the time we obtain the measured value. In other words, we can calculate the predicted value as soon as we acquire a measured value. By using this problem setting, we delay the timing of determining the predicted value until measured values are acquired, and we regard the predicted value as the output of the predictive model whose output is nearest to the measured value.

In our approach, the prediction error can be regarded as the difference between the measured value and the output nearest to the measured value. To the best of our knowledge, this is

the first approach of its kind in anomaly detection area. By referring to the measured value, dLSTM switches the proper predictive model quickly and can adapt to complicated time-series data.

### 1.2 Novelty of this study

We propose a novel anomaly detection method for time-series data by utilizing LSTM. LSTM is a powerful tool for capturing the context of sequential data, and is often used in fields such as natural language processing and speech recognition [17,22]. Previous papers [1,12] have used LSTM to detect anomalies from prediction errors.

These papers show how LSTM can capture long-term correlations in sequences and emphasize that one of the advantages of LSTM is that prior knowledge such as window size ($w$) is unnecessary. Therefore, LSTM is useful to obviate the need for a pre-specified window size, thereby obtaining competitive performance. To further improve performance, we need to explicitly tackle "multi-mode" problems. Ensemble learning is a good option for handling multi-mode output, therefore we developed it to making use of the particular problem of anomaly detection.

Our method is unique in that we consider the particular problem setting of anomaly detection. We focus on the setting in which we can delay the timing of prediction until we obtain the corresponding measured value. By making use of the measured value, we can select the proper predictive model and enhance prediction accuracy. To the best of our knowledge, this is a novel application of LSTM for anomaly detection. Moreover, this paper empirically demonstrates that our proposed method more clearly detects anomalies than do existing methods.

### 1.3 Related works

This section summarizes application of the method to time-series data for anomaly detection. The simple anomaly detection method is to detect anomalies by comparing data with known anomalies [4]. However, this method cannot identify unknown abnormal patterns.

Recently, online anomaly detection has been proposed. For example, Takeuchi et al. [23] proposed|*Change Finder* to simultaneously detect anomalies and outliers. Hill et al. [6] applied an autoregressive model to detecting anomalies according to deviation from normal data, and it is broadly assumed that anomaly scores rise sharply with general anomaly detection. Miyaguchi et al. [14] proposed a method that can be applied when scores rise gradually. Online anomaly detection is extremely fast, but the model for online anomaly detection is relatively simple and does not deal well with irregular and complicated waveforms.

To monitor manufacturing equipments, data are usually acquired from sensor devices and a large amount of normal data are often recorded. In this type of approach, we begin by learning a model from a large amount of normal data, and then we perform anomaly detection by reference to the model. The advantage of this approach, introduced below, is that it can handle many anomaly types. Hayton et al. [5] proposed a method that uses a one-class support vector machine that can represent nonlinear relations. Although the one-class support vector machine builds a model from normal data, it does not utilize multiple models to handle multi-mode conditions.

Recently, deep learning has been widely applied to handling the complex features of time-series data. In particular, LSTM [7] is a well-known deep learning method based on recurrent neural networks, and is embedded in many methods [1,12]. LSTM was originally applied to natural language processing [22] and speech recognition [17].

Anomaly detection methods using normal data can be classified into methods based on prediction errors and those based on reconstruction errors. When the prediction error is used, a model is built to predict future values of time-series data and to perform anomaly detection by checking the prediction error from observed data [1,12]. When the reconstruction error is used, a model is built to reconstruct the time-series data, often using an autoencoder [11]. Then, we perform anomaly detection by checking the reconstruction error for observed data. These deep learning methods use only one predictive model, and it is difficult to apply to multi-mode cases. We fundamentally improve this approach by using multiple models.

The input–output relation is deterministic when deep learning are applied to, which makes it difficult to deal with multi-mode cases. One proposed approach extends anomaly detection to use of a probabilistic model with deep learning [21]. Even in this approach, however, estimating accurate output distributions from similar inputs is a challenging task.

Another approach to dealing with multi-mode outputs is the ensemble approach, such as the anomaly detection methods proposed by Rayana and Akoglu [15]. However, this approach is best suited to network data. Maruyama et al. [13] proposed an anomaly detection method that dynamically selects models from data. Prediction errors are characterized by the property that they cannot be computed until both measured and predicted values are acquired. The main advantage of this property is that we can delay the timing of obtaining predictive values until we obtain the measured values. Ensemble approaches do not focus on this property of prediction errors.

### 1.4 Organization of this paper

The remainder of this paper is organized as follows. Section 2 introduces a framework for anomaly detection, and Sect. 3 describes our proposed method for time-series anomaly detection. Section 4 introduces the setting for evaluation, and Sect. 5 shows experimental results. Section 6 concludes this paper.

## 2 Anomaly detection based on data from a normal state

This section introduces a framework for building a model based on only normal data for detecting anomalies in observed data.

In this paper, we use a training dataset that contains only normal data and an observed dataset. We assume that there is only one anomaly occurrence and that all data after the anomaly are abnormal. We consider univariate time-series data here for simplicity, but multivariate time-series data can be approached in the same way by vectorizing the data. Our proposed method can be divided into three steps. In Step 1 (Fig. 3, Step 1), we learn the behavior of normal data from the training dataset using deep learning. Specifically, we construct a prediction model that predicts normal data as accurately as possible to learn its behavior. In Step 2 (Fig. 3, Step 2), we use the prediction model built in Step 1 to sequentially predict values in the observed dataset. However, since sensor data include noise, there is a high possibility that this noise affects the prediction error. To mitigate the influence of noise, in Step 3 (Fig. 3, Step 3) we apply a low-pass filter and then calculate an anomaly score representing the degree of anomaly. Note that dLSTM is only related to Step 1.

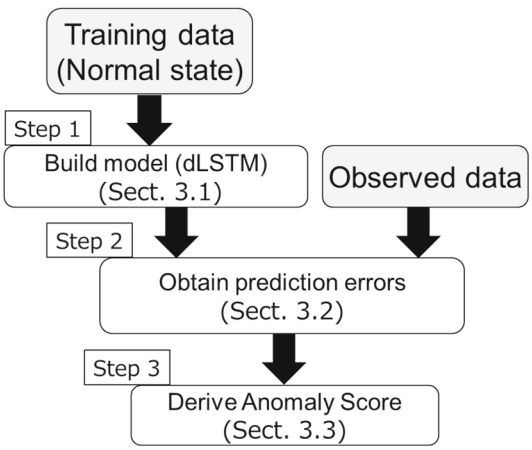


**Fig. 3** Overview of our proposed method. Step 1: Build models using only normal data. Step 2: Obtain the prediction error for the observed dataset. Step 3: Calculate the anomaly score from the prediction error

## 3 Predicting time-series data within the normal state

This section describes dLSTM and its application to obtaining the prediction error and anomaly score based on the prediction error. Section 3.1 describes our proposed prediction method, dLSTM. Section 3.2 describes the method for obtaining the prediction error for the observed dataset. Section 3.3 shows how to obtain an anomaly score for the observed dataset.

### 3.1 Building dLSTM model: step 1

In this section, we sequentially predict training data of size $T_1$, which includes only normal data within a window size $w$ to capture the normal state. Here, we denote the time between $a$ and $b$ (inclusive) as $a : b$. We consider the case of obtaining the predicted values for time $t + 1 : t + w$.

We first describe the procedure for obtaining the prediction error, and then derive the objective function. Anomaly detection is generally performed based not on the predicted value, but on the prediction error. Since we need both predicted and measured values to derive the prediction error, it follows that we can delay the timing for obtaining the predicted value until the corresponding measured values are available. Figure 4 illustrates our approach. We predict the values sequentially, considering the information of the predicted value at time $i$ ($t + 1 \leq i \leq t + w$). In previous approach, values between $t+1$ and $t+w$ are predicted deterministically at time $t$ [1,12] and green ones are predicted values. In contrast, our approach provides candidate values denoted by gray circles at time $t$ and determines each predicted value denoted by blue circles at the time that the corresponding measured value is obtained (time $i$). Focusing on time $i$ ($t + 1 \leq i \leq t + w$), the delay period is the period between $t$ and $i$, because our method and the conventional method obtain the predicted value at times $i$ and $t$, respectively. Note that there are multiple candidate values (gray circles) in dLSTM and single predicted value (green circle) in previous approach at each time stamp. The feature of our prediction error is that it includes the information of a measured value when we determine the predicted value. On the other hand, we do not include the information of a measured value to derive the predicted value in the previous approach.

Our approach presents candidates of predicted values at time $t$ from multiple predictive models at each timestamp within $t + 1 : t + w$. Then, we determine the predicted values at time $i$, with $i \in [t + 1 : t + w]$ among candidates.

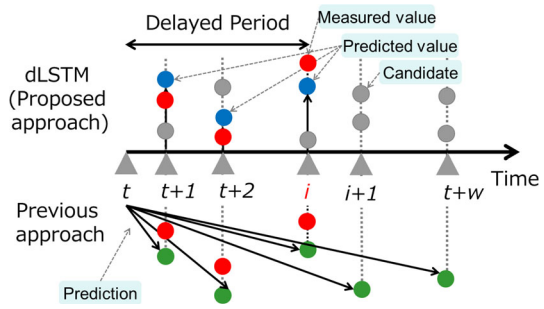


**Fig. 4** Schematic diagram of dLSTM compared with the previous approach when obtaining the predicted value at time $i$. Red circles indicate measured values, gray circles show candidate values, and blue circles show values selected from among the candidates as predicted values for dLSTM. The green circles show predicted values by the previous approach. Our approach determines predictive values from candidates obtained at time $t$, where the period from $t$ to $i$ is the delay period. See Fig. 5 for further details on our proposed prediction method (color figure online)

We provide multiple predictive models and denote the number of prediction models by $N$. We denote the measured and predicted values for time $t+1:t+w$ by $\boldsymbol{X}_{t+1:t+w} \in \mathbb{R}^w$ and $\boldsymbol{Y}_{t+1:t+w} \in \mathbb{R}^w$. Let $f_n$ be the $n$ th predictive model and let the corresponding predicted values be $\boldsymbol{Y}^n_{t+1:t+w} = f_n(\boldsymbol{X}_{t-w+1:t})$, where $\boldsymbol{Y}^n_{t+1:t+w} = [y^n_{t+1}, \ldots, y^n_{t+w}]$. In our approach, we determine the predicted value $(y_i)$ at time $i$ from the candidates of the predicted value $(y^n_i)$. In this paper, we use deep learning as a prediction model $f_n$. Because multiple nonlinear activation functions are used, complex input–output relations can be captured using deep learning. Analyses of sequential data in fields such as speech recognition and natural language processing often embed LSTM. In LSTM, cells store internal states and consider past information by sequentially updating that internal state. It is thus possible to consider time-dependent effects over long periods of time, such as deterioration of devices as they age, through incorporation into models using LSTM. Below, we show an example of predicting time-series data through 5-layer deep learning.

$$\boldsymbol{Z}^n_{1,t} = g_1(\boldsymbol{W}^n_1 \boldsymbol{X}_{t-w+1:t} + \boldsymbol{b}^n_1) \tag{1}$$

$$\boldsymbol{Z}^n_{2,t} = \mathrm{LSTM}(g_2(\boldsymbol{W}^n_2 \boldsymbol{Z}^n_{1,t} + \boldsymbol{b}^n_2)) \tag{2}$$

$$\boldsymbol{Z}^n_{3,t} = g_3(\boldsymbol{W}^n_3 \boldsymbol{Z}^n_{2,t} + \boldsymbol{b}^n_3)) \tag{3}$$

$$\boldsymbol{Y}^n_{t+1:t+w} = \boldsymbol{W}^n_4 \boldsymbol{Z}^n_{3,t} + \boldsymbol{b}^n_4 \tag{4}$$

Note that $g_1$, $g_2$, and $g_3$ are activation functions, such as sigmoid functions or a rectified linear unit (ReLU). We express $\boldsymbol{W}^n_j$ and $\boldsymbol{b}^n_j$ as the weight matrix and noise vector of the $j$th layer, respectively. There are $N$ output types based on the prediction models described above, and we compare the output of each prediction model $(y^n_i)$. These are candidates for the final predicted value $(y_i)$. In our proposed method, the

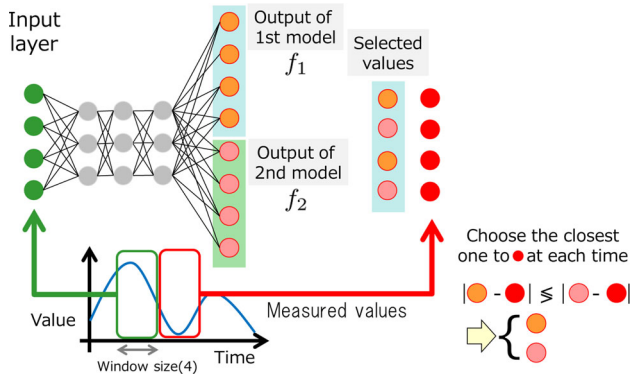


**Fig. 5** Schematic diagram of dLSTM with number of models $N = 2$ and window size $w = 4$

final predicted value at time $i$ is determined as the prediction model output that is closest to the corresponding measured value (i.e., the output giving the smallest prediction error). Let $x_t$ be the measured value at time $t$, that is, $\boldsymbol{X}_{t+1:t+w} = [\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+2}, \ldots, \boldsymbol{x}_{t+w}]$. Then the final predicted value $y_i$ at time $i$ is calculated as

$$y_i = y^{n^*}_i, \text{ where } n^* = \arg\min_n (y^n_i - x_i)^2, \tag{5}$$

where $\boldsymbol{Y}_{t+1:t+w} = [y_{t+1}, y_{t+2}, \ldots, y_{t+w}]$. From Eq. (5), for $n \in \{1, 2, \ldots, N\}$, we obtain the inequality

$$(y^{n^*}_i - x_i)^2 \leq (y^n_i - x_i)^2. \tag{6}$$

It can be seen that the prediction accuracy is improved by using measured values. We repeat this procedure from time $t+1$ to time $t+w$ to obtain the final predicted values $(\boldsymbol{Y}^n_{t+1:t+w})$. Figure 5 illustrates our approach for $N = 2$. The case where there are multiple outputs from the same input is called the multi-mode case. If each of the predictive models corresponds to the outputs, one of the prediction errors should be small when applied to all predictive models. In other words, if predictive performance deteriorates even when the model with the best prediction performance is applied, it is certain that data deviates from the normal and is thus abnormal. Therefore, we use the predictive model with the least predictive error.

We next describe the objective function of our proposed method. Even normal data sometimes include outliers, thus it is conceivable that general performance will decrease if we build a prediction model that is forced to respond to both normal data and to outliers. We thus improve generalization performance by deleting outliers from the training data. The deep learning framework applies minibatch training. We set the size of a minibatch to be $B$. It follows that $Bw$ predicted values are calculated in each minibatch training.

We express the mean and standard deviation of $(y_t - x_t)$ as $\mu$ and $\sigma$, respectively $(1 \leq t \leq Bw)$. We regard the value

at time $t$ as an outlier when it does not satisfy

$$\mu - 3\sigma < y_t - x_t < \mu + 3\sigma. \tag{7}$$

Therefore, we classify whether measured values in the training dataset are outliers when we obtain measured values, and eliminate these outliers.

From the above, the prediction error for each minibatch in this case is

$$\text{loss}(t) = \frac{\sum_{i=t+1}^{t+W} \mathbb{I}_{\mu-3\sigma<(x_i-y_i)<\mu+3\sigma} \times (x_i - y_i)^2}{\sum_{i=t+1}^{t+W+1} \mathbb{I}_{\mu-3\sigma<(x_i-y_i)<\mu+3\sigma}}, \tag{8}$$

where $\mathbb{I}$ is an indicator function that returns 1 if the condition is satisfied and 0 otherwise. The objective loss function is

$$\text{Loss}(t) = \sum_{j=1}^{B} \text{loss}(t + (j-1)w). \tag{9}$$

Note that Loss($t$) corresponds to the prediction error between time $t$ and $t + Bw$. For each measured value, our proposed method selects the model with the least prediction error, and parameters of only the corresponding model are updated using back propagation so that prediction error decreases. As a result, it is possible to generate prediction models more specific to the measured values.

Although LSTM can ideally incorporate all past time dependencies into the model, in practice, available memory limits the amount of past data that can be considered. Therefore, *truncated back propagation through time* (tBPTT) is often used to forget past information beyond a constant multiple of the window size. In this paper, we use Adam [8] for the back propagation method and forget past information beyond $L$ window sizes.

The following summarizes Algorithm 1, corresponding to Step 1. In lines 7–21 of Algorithm 1, we derive a loss value and update parameters based on the loss in each minibatch.

**Algorithm 1** Algorithm for building the predictive model of dLSTM (Step 1).

1: INPUT: Training dataset (normal data) for $\boldsymbol{X} \in \mathbb{R}^{T_1}$.
2: OUTPUT: Parameters for predictive dLSTM models.
3: • STEP 1
4: Initialize dLSTM parameters.
5: **for** $m = 1 \rightarrow$# of iterations $M$. **do**
6: Set $S = 0$ (Initialize the cumulative loss function).
7: **for** $j = 0 \rightarrow \text{int}(T_1/w) - 1$. **do**
8: Set $t = wj$.
9: **for** $n = 1 \rightarrow N$. **do**
10: Obtain the $n$th candidate ($\boldsymbol{Y}^n_{t+1:t+wj}$) according to Eq. (4).
11: **end for**
12: **for** $i = t + 1 \rightarrow t + wj$. **do**
13: Choose the predicted value $y_i$ at time $i$ from candidates according to Eq. (5).
14: **end for**
15: Set the loss value Loss($t$) according to Eq. (9).
16: $S \leftarrow S + \text{Loss}(t)$ (Update the cumulative loss function).
17: **if** $j$ is a multiple of $L$ **then**
18: Update the parameters of dLSTM using tBPTT to decrease the value of $S$.
19: $S = 0$ (Reset $S$).
20: **end if**
21: **end for**
22: **end for**

## 3.2 Calculation of prediction error: step 2

This section describes the method of obtaining the prediction error for observed data of size $T_2$. When applying to the observed data, we use the dLSTM built in Step 1. The prediction error at time $i$ is represented as $(x_i - y_i)^2$, where $y_i$ is from Eq. (5). For each window size $w$, we prepare $N$ predictive models and choose the one whose output is closest to the corresponding measured value. We can thus determine the prediction error as soon as we acquire the measured value. We build the predictive models and adapt their parameters in Step 1. However, as we sequentially read the observed dataset, we can update the internal state of LSTM. Moreover, which model we choose depends on the observed dataset, and we dynamically switch the proper predictive model according to the measured value. This allows capturing complicated time-series datasets. Algorithm 2 shows the algorithm for Step 2.

## 3.3 Deriving the anomaly score: step 3

This section describes a method for deriving an anomaly score based on the prediction error. Measured values from sensors are often affected by noise. Therefore, low-pass filters are often used to remove these effects. In this paper, we use a popular type of low-pass filter called a median filter for each filtering window size ($l$). The anomaly score $S(t)$ at time $t$ is obtained as

$$S(t) = \text{median}_{i \in [t-l+1, t-l+2, \ldots, t]} (y_i - x_i)^2. \tag{10}$$

When $l$ is 10,000 and the data frequency is 10 Hz, a common situation in manufacturing fields, $S(t)$ is derived based on data within the previous 1000 s ($\approx$ 20 min). Most parameters for predictive models are learned from normal data in the training dataset. Therefore, $S(t)$ measures the deviation of data from the normal state at time $t$ even after the occurrence of anomaly $S(t)$. Setting a large filtering window stabilizes performance by removing noise influences. However, small filtering windows are advantageous in that they allow prompter detection of anomalies. Algorithm 2 shows the algorithm for Step 3.

Generally, there are two *error* types: false positives, where normal data are classified as abnormal, and false negatives,

**Algorithm 2** Algorithm for obtaining the anomaly score (Steps 2, 3).

INPUT: dLSTM parameters; window size: $w$, filtering window size: $l$, and observed dataset $\boldsymbol{X} \in \mathbb{R}^{T_2}$.
OUTPUT: Anomaly score at time $t$ ($S(t)$).
• STEP 2
Load the dLSTM parameters learned in Step 1.
**for** $i = 1 \rightarrow \text{int}(T_2/w)$ **do**
  **for** $j = 1 \rightarrow w$ **do**
    Obtain the predicted value $y_{wi+j}$ at time $wi + j$ according to Eq. (5)
  **end for**
**end for**
•STEP 3
**for** $t = 1 \rightarrow T_2$ **do**
  Obtain anomaly score $S(t)$ according to Eq. (10).
**end for**

where abnormal data are classified as normal In manufacturing, it is likely that false positives are more problematic than false negatives. When the likelihood of false positives is high, alarms are issued too often, increasing human resource demands. It may cause the case that we are not afford to investigate the equipment when the true anomaly happens. On the other hand, false negatives happen when setting a large filtering window, because the anomaly score does not increase rapidly until the period corresponding to the length of the filtering window elapses after the occurrence of the anomaly. In this case, the influence of false negatives are limited to products produced in this period and we can assess the impact of false negative.

## 4 Experimental settings

This section introduces settings for comparing performances using artificial and real data.

### 4.1 Model architecture of dLSTM

This section introduces the network architecture for our proposed method. Our method uses multiple prediction models, setting the number of models $N$ to 2 or 10. Furthermore, we ensure that part of the architecture is shared among $N$ predictive models. We can reduce the number of parameters by partly sharing architectures and efficiently learning the parameters. Each predictive model shares structures for the first seven layers and independently updates the parameters for the last two layers. Note that the first layer is the input layer and the last layer is the output layer. Therefore, each predictive model is composed of nine layers with sizes $[w, \lfloor \frac{3w}{4} \rfloor, \lfloor \frac{2w}{3} \rfloor, \lfloor \frac{w}{2} \rfloor, \lfloor \frac{w}{3} \rfloor, \lfloor \frac{w}{2} \rfloor, \lfloor \frac{2w}{3} \rfloor, \lfloor \frac{3w}{4} \rfloor, w]$. LSTM is embedded between the second and third layers and between the third and fourth layers. For the activation function, we embed ReLU between the layer pairs #2·#3, #3·#4, #6·#7, and #7·#8. For layers without embedded ReLU, we do not embed any kind of activation function. To enhance generalization performance, we incorporate a regularization term for the weight matrix $\boldsymbol{W}_i$ for each layer with L2 regularization, using a parameter $\lambda = 0.05$. Figure 6 shows an overview of the network architecture.

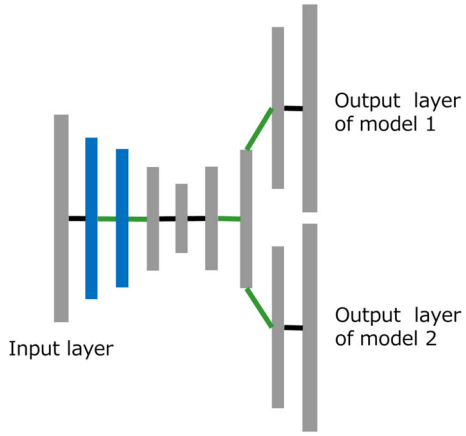


**Fig. 6** Schematic figure of dLSTM for $N = 2$. The green line represents embedding of ReLU and the blue layers are LSTM layers (color figure online)

### 4.2 Comparative methods

As described in Sect. 1.3, anomaly detection using deep learning can be divided into two approaches: those based on prediction error and those based on reconstruction error. Our proposed method (dLSTM) is based on prediction error. In this section, we introduce two comparative methods using prediction error and three methods using reconstruction error.

We first consider methods using prediction error. dLSTM has two main features, use of multiple predictive models and delayed prediction. To verify the effectiveness of multiple models and delayed prediction, we provide two comparative methods, Single and Predet. Single is essentially same as the same as in [1,12] using one predictive model. In contrast, Predet provides multiple predictive models. However, Predet determines which model to choose before acquiring a measured value.

We next deal with methods using reconstruction error. Autoencoder is a neural network that reconstructs input as accurately as possible. We capture waveforms of normal data using autoencoder and derive an anomaly score based on the reconstruction error. We thus assume a high probability that anomalies occur when the reconstruction error is large. Many autoencoders have been proposed. We introduce three typical autoencoders as comparative methods, namely the stacked, contractive, and variational autoencoders.

For all methods, we set the window size $w$ to 100, $L$ (used for tBPTT) to 15, and $B$ (for minibatch) to 200. The number of iterations is 350 unless otherwise noted.

See Sect. 5.6 for a discussion of methods not based on deep learning.

#### 4.2.1 Single

Our proposed method utilizes multiple predictive models. The Single comparison model is a single predictive model comprising eleven layers. The layer sizes are $[w, \lfloor \frac{3w}{4} \rfloor, \lfloor \frac{2w}{3} \rfloor, \lfloor \frac{w}{2} \rfloor, \lfloor \frac{w}{3} \rfloor, \lfloor \frac{w}{4} \rfloor, \lfloor \frac{w}{3} \rfloor, \lfloor \frac{w}{2} \rfloor, \lfloor \frac{2w}{3} \rfloor, \lfloor \frac{3w}{4} \rfloor, w]$. LSTM is embedded between the second and third layers and between the third and fourth layers. LSTM is multiply embedded in [1,12], and ReLU is embedded between the layer pairs #2·#3, #3·#4, #8·#9, and #9·#10.

#### 4.2.2 Predet

One main feature of the proposed method is that the prediction timing is delayed. The predetermined predictive model (Predet) is a comparison method corresponding to the case where multiple prediction models are used but prediction is predetermined, not delayed. Let the output of the $n$ th predictive model be $\boldsymbol{V}_t^n = f_n(\boldsymbol{X}_{t-w+1:t})$, where $f_n(\cdot)$ is deep learning with the same architecture used in our proposed method. We determine which model to choose according to the variable $\boldsymbol{H}_t = h(\boldsymbol{X}_{t-w+1:t}) \in [0, 1]^w$, which is obtained by the deep learning embedded sigmoid function $h(\cdot)$ in the last layer to constrain the output to within the range [0, 1]. The final predictive $y_i$ value at time $i$ $(t + 1 \leq i \leq t + w)$ is then obtained by $y_i = y_i^n$, where $n$ satisfies the inequality $\left(\frac{n-1}{N} < \boldsymbol{H}_t^{i-t} \leq \frac{n}{N}\right)$ according to the variable $\boldsymbol{H}_t^{i-t}$, which is the $(i - t)$ th value of $\boldsymbol{H}_t$. This method is one example of an ensemble approach.

#### 4.2.3 CAE

Contractive autoencoders (CAE) [16] can enhance generalization performance by incorporating a penalty term corresponding to the Frobenius norm of the Jacobian matrix of an activation function. Let $\boldsymbol{x} \in \mathbb{R}^{\mathrm{d}_x}$, $f(\cdot)$ and $\boldsymbol{h} \in \mathbb{R}^{\mathrm{d}_h}$ respectively be the input, an activation function, and the output resulting from applying the activation function to the input. Its formulation is $\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$. In [16], a sigmoid function $(f(z) = \frac{1}{1+\exp^{-z}})$ is used as an activation function and the corresponding penalty term is

$$\sum_{i}^{\mathrm{d}_h} (\boldsymbol{h}_i(1 - \boldsymbol{h}_i)) \sum_{j}^{\mathrm{d}_x} \boldsymbol{W}_{i,j}^2. \tag{11}$$

We set the regularization parameter to be 0.01. The sizes of the layers are $[w, \lfloor \frac{3w}{4} \rfloor, w]$ and use a sigmoid function as an activation function between the #1 and #2 layers.

#### 4.2.4 SAE

The stacked autoencoder (SAE) is a feed-forward multilayer neural network. Embedding multiple nonlinear activation functions allows capturing complex sensor data. This method was inspired by Sakurada and Yairia [18]. The layer sizes are $[w, \lfloor \frac{w}{2} \rfloor, \lfloor \frac{w}{4} \rfloor, \lfloor \frac{w}{4} \rfloor, \lfloor \frac{3w}{4} \rfloor, w]$, and ReLU is embedded between the layer pairs #2·#3, #3·#4, and #4·#5.

#### 4.2.5 VAE

The variational autoencoder (VAE) [9] is a probabilistic model. It first produces mean and logarithm-of-variance values. It then provides a random variable based on these values and applies the nonlinear activation function to obtain the desired output. Unlike CAE and SAE, VAE is not a deterministic autoencoder. We first describe the architecture for obtaining the mean and logarithm-of-variance values. The layer sizes are $[w, \lfloor \frac{3w}{4} \rfloor, \lfloor \frac{2w}{3} \rfloor, \lfloor \frac{w}{2} \rfloor, \lfloor \frac{w}{3} \rfloor, \lfloor \frac{w}{4} \rfloor]$. ReLU is embedded between the layer pairs #2·#3, #3·#4, and #4·#5. We next describe the network used to reconstruct the input data. The layer sizes are $[\lfloor \frac{w}{4} \rfloor, \lfloor \frac{w}{3} \rfloor, \lfloor \frac{w}{2} \rfloor, \lfloor \frac{2w}{3} \rfloor, \lfloor \frac{3w}{4} \rfloor, w]$. ReLU is embedded between the layer pairs #2·#3, #3·#4, and #4·#5.

### 4.3 Artificial datasets

This section shows how artificial datasets are generated. Let $\boldsymbol{x}_t$ be the measured value at time $t$. As preprocessing, we convert $\boldsymbol{x}_t$ to $\frac{\boldsymbol{x}_t - \mu_{\mathrm{train}}}{\sigma_{\mathrm{train}}}$, where $\mu_{\mathrm{train}}$ and $\sigma_{\mathrm{train}}$ are respectively the mean and the standard deviation of the training data for each dataset. For each dataset, there is only one occurrence of anomaly and all the data after the occurrence of the anomaly are abnormal data.

#### 4.3.1 Sin-data

Sensor data exhibit periodicity with a slightly varying period. We generate the first artificial dataset (sin-data) by slightly changing the period $T$ of a sine curve. For the normal state, $T$ is the integer part of the variable taken from the normal distribution $\mathcal{N}(50, 5)$. We add noise from the normal distribution $\mathcal{N}(0, 0.3)$ at each timestamp. Therefore, the value for each period $T$ at time $t$ is

$$5\sin\left(\frac{2\pi t}{T}\right) + \epsilon, \tag{12}$$

where $\epsilon$ is a variable following $\mathcal{N}(0, 0.3)$.

For the abnormal state, the period $T$ is the integer part of the variable from the normal distribution $\mathcal{N}(40, 5)$, and the other parts are identical to the normal state. The training dataset is generated by repeating Eq. (12) 10,000 times for the

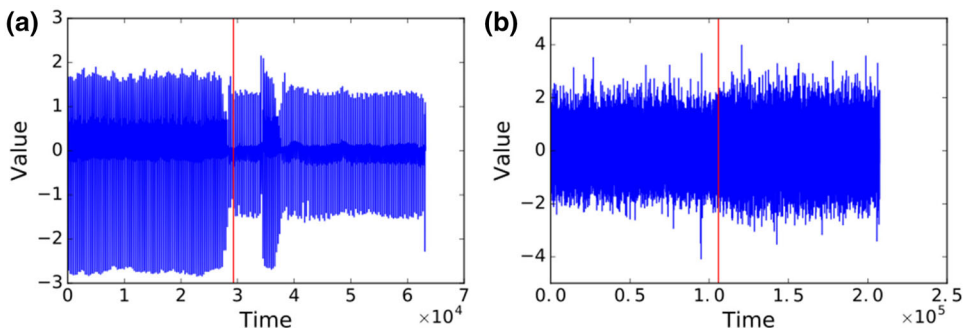


**Fig. 7** **a** ECG dataset. **b** Uwave dataset

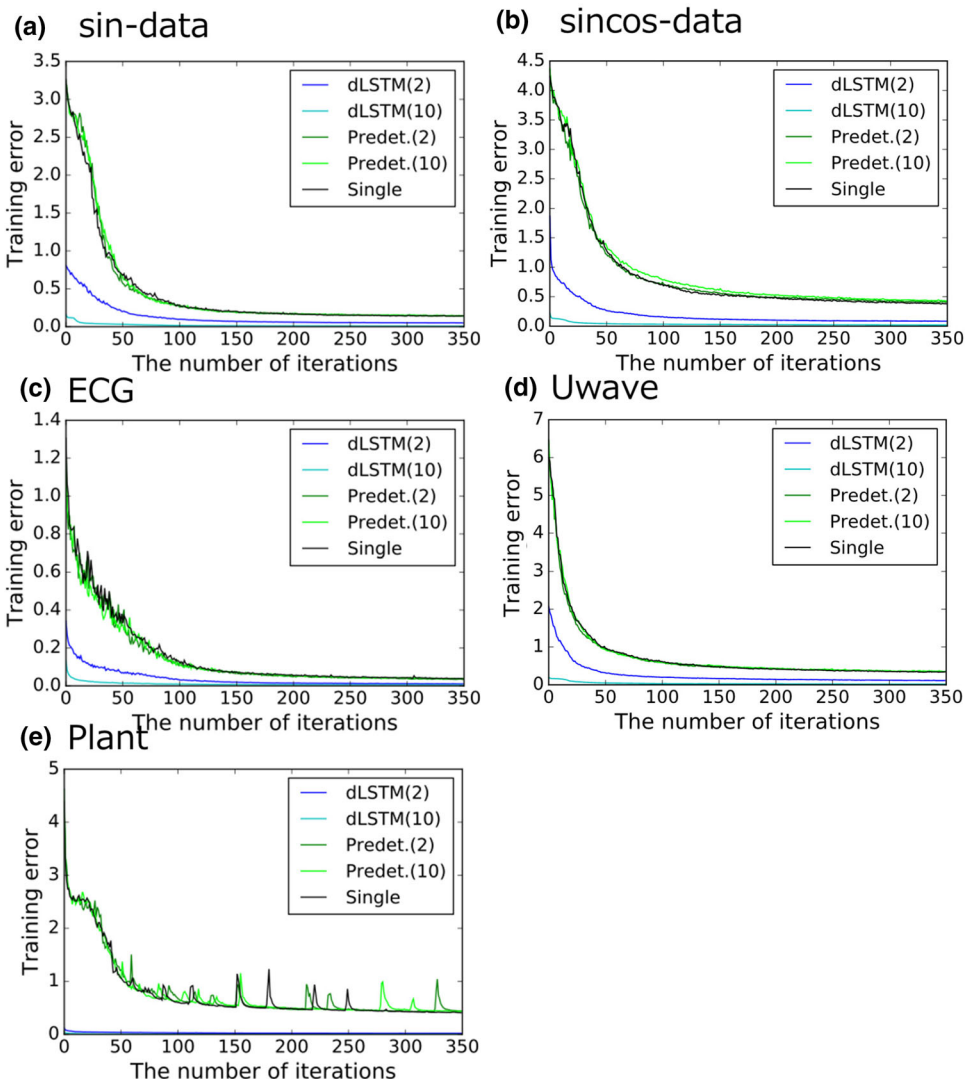


**Fig. 8** Relations between number of iterations and prediction error (training error) for **a** sin-data. **b** sincos-data, **c** ECG, **d** Uwave, and **e** plant

normal state. The observed dataset is generated by repeating Eq. (12) 5000 times for the normal state and then repeating Eq. (12) 5000 times for the abnormal state.

#### 4.3.2 Sincos-data

We next show the second artificial dataset (sincos-data). Here, we assume a more complicated case by combining a sine curve and a cosine curve and adding noise. The specific generation method is as follows:

$$A\sin\left(\frac{2\pi}{T_3}t\right)+\epsilon,\ 0<t\leq T_3, \tag{13}$$

$$A\cos\left(\frac{2\pi}{T_4}t\right)+\epsilon,\ T_3<t\leq T_3+T_4, \tag{14}$$

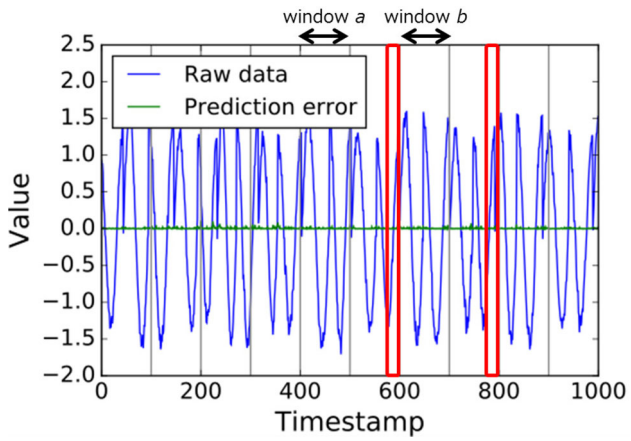


**Fig. 9** Sincos-data and prediction errors by dLSTM(2). Gray lines represent the window $w$

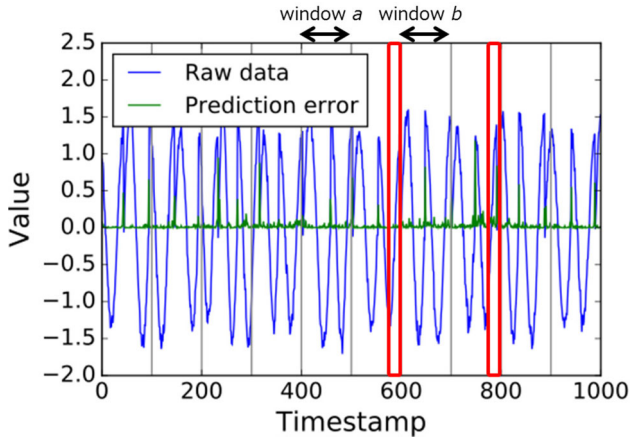


**Fig. 10** Sincos-data and prediction errors by Single. Gray lines represent the window $w$

$$B\sin\left(\frac{2\pi}{T_5}t\right)+\epsilon,\ T_3+T_4<t\leq\sum_{j=3}^{5}T_j, \tag{15}$$

$$B\cos\left(\frac{2\pi}{T_6}t\right)+\epsilon,\ \sum_{j=3}^{5}T_j<t\leq\sum_{j=3}^{6}T_j, \tag{16}$$

where, $\epsilon$ is a variable independently following $\mathcal{N}(0, 0.3)$.

For the normal state, we respectively set $A$ and $B$ to 5 and 6, and $T_3$, $T_4$, $T_5$, and $T_6$ are the integer parts of the variable independently taken from $\mathcal{N}(50, 5)$.

For the abnormal state, we respectively set $A$ and $B$ to 6 and 7, and $T_3$, $T_4$, $T_5$, $T_6$ are the integer parts of the variable independently taken from $\mathcal{N}(40, 5)$.

To generate the training dataset, we repeat from Eq. (13) to Eq. (16) 2500 times with normal state parameters. To generate the observed dataset, we repeat from Eqs. (13) to (16) 1250 times with normal state parameters, then repeat from Eqs. (13) to (16) 1250 times with abnormal state parameters.

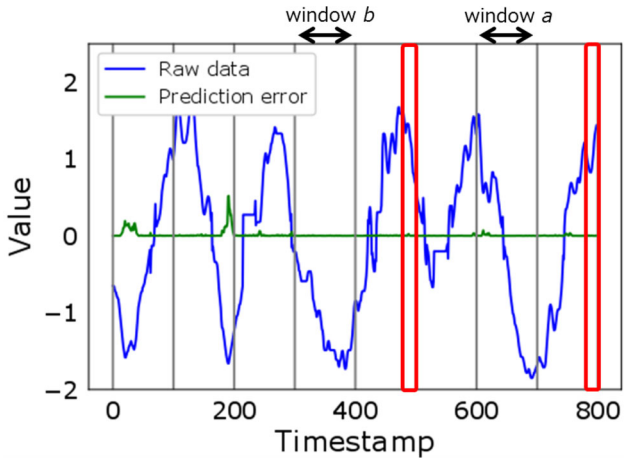


**Fig. 11** Uwave data and prediction errors by dLSTM(10). Gray lines represent the window $w$

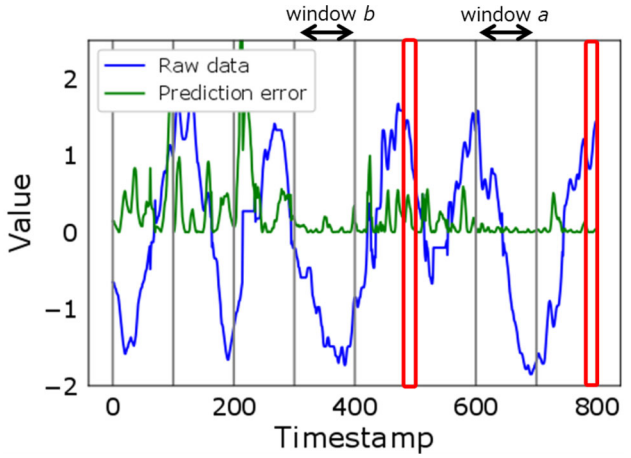


**Fig. 12** Uwave data and prediction errors by Single. Gray lines represent the window $w$

## 4.4 Real datasets

This section introduces the real datasets. Let $\boldsymbol{x}_t$ be the measured value at time $t$. As preprocessing, we convert $\boldsymbol{x}_t$ to $\frac{\boldsymbol{x}_t-\mu_{\text{train}}}{\sigma_{\text{train}}}$, where $\mu_{\text{train}}$ and $\sigma_{\text{train}}$ are respectively the mean and standard deviation of the training data for each dataset. For each dataset, there is only one anomaly occurrence, and all data after the anomaly are abnormal.

### 4.4.1 ECG

The real dataset considered here is an ECG dataset provided by PhysioBank.[1] Among the many PhysioBank datasets, we chose the 102nd record of the MIT-BiH Arrhyhmia Database (mitdb). The mitdb is widely used for testing anomaly detection [3,10,20].

[1] https://physionet.org/cgi-bin/atm/ATM.

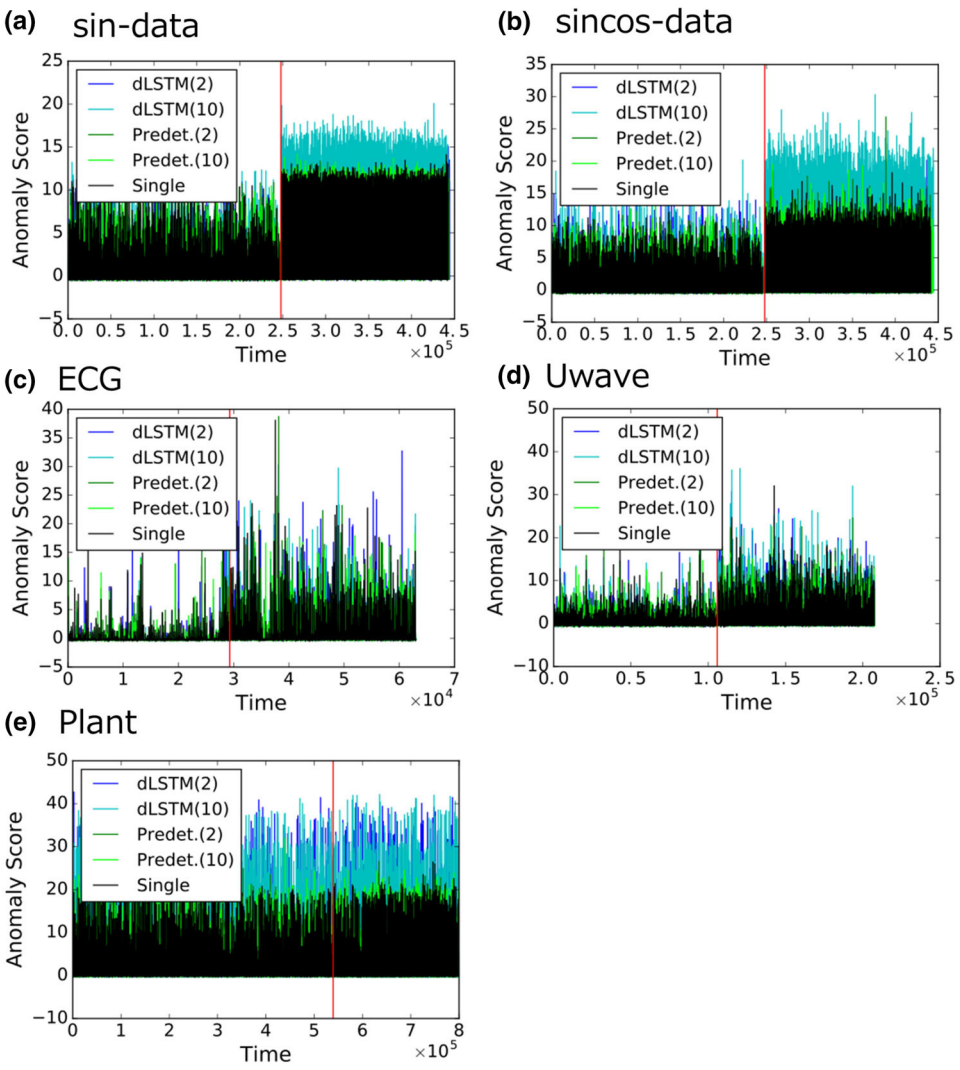


**Fig. 13** Anomaly score at each time after normalization ($R(t)$) using the prediction error and setting the size of the filtering window $l$ to 1 for **a** sin-data, **b** sincos-data, **c** ECG, **d** Uwave, and **e** plant. The red line represents the time when the anomaly occurred (color figure online)

According to an annotation to this dataset, a pacemaker is used most of the time, but from time 29,276 to time 62,531 the dataset corresponds to regular heartbeats. We regard the pacemaker heartbeat as the normal state and regular heartbeats as the abnormal state, and aim to detect regular heartbeats.

We generate the training and observed datasets by extracting data from the original dataset after time 63,001 and before time 63,000, respectively. The observed dataset thus includes both normal and abnormal states (see Fig. 7a). The red line represents the time when the abnormal state occurred. According to Fig. 7a, the amplitude greatly differs before and after the anomaly.

#### 4.4.2 Uwave

The Uwave dataset is generated by processing a dataset provided by UCR[2] [2]. Among the many UCR datasets, we chose UWaveGestureLibraryALL, because this dataset is relatively large and suited to deep learning. This dataset was originally designed for classification tasks. In order to adapt it to anomaly detection tasks, we generated the training dataset by concatenating all data with the label "7" from UWaveGestureLibraryALL_TEST dataset. We also generated the observed dataset by first concatenating all data with the label "7" and then concatenating all data with the label "2" from UWaveGestureLibraryALL_TRAIN dataset. Therefore, the training and observed datasets are univariate datasets, and we regard the original data labeled "7" as normal data and original data labeled "2" as abnormal data. Figure 7b shows the observed dataset. The red line shows the time when an anomaly occurred. In Fig. 7b, the amplitude increases after the anomaly occurs.

[2] http://www.cs.ucr.edu/~eamonn/time_series_data/.

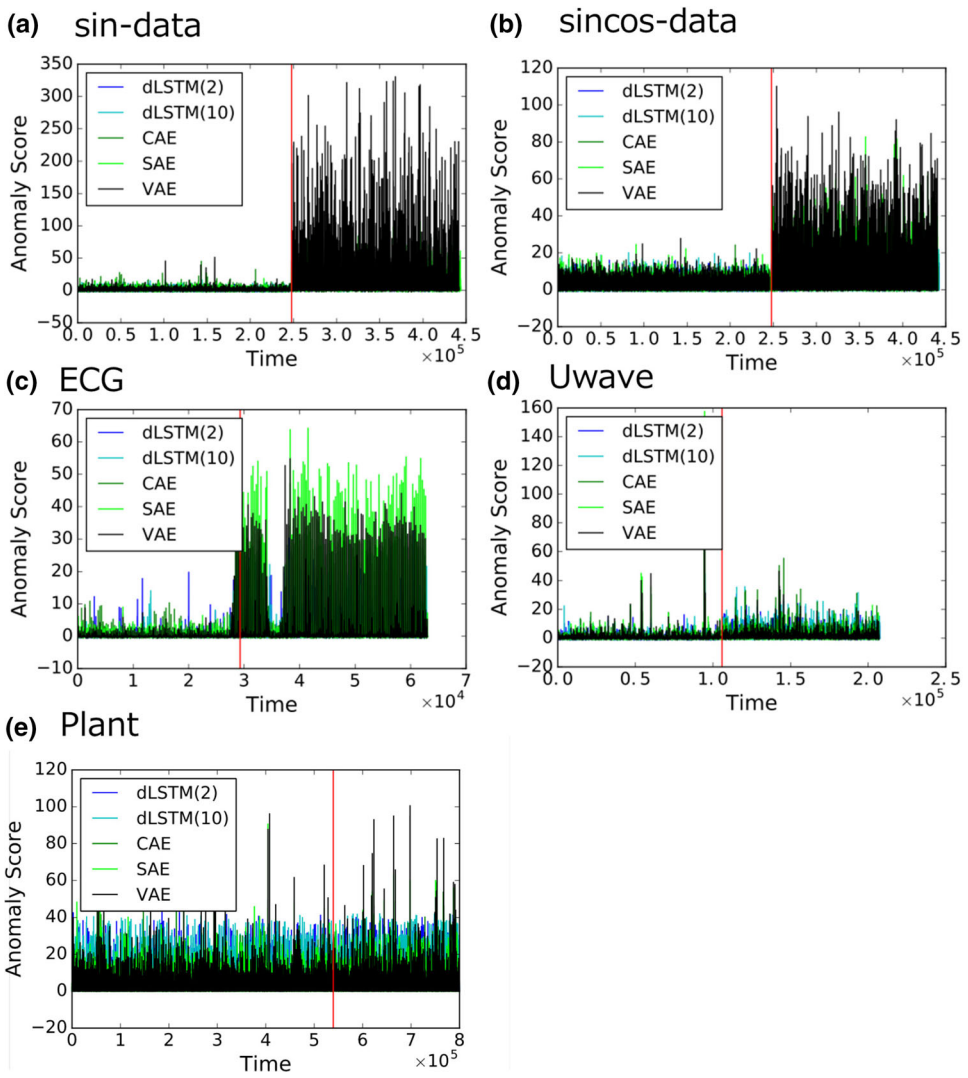


**Fig. 14** Anomaly score at each time after normalization ($R(t)$) using the reconstruction error setting the size of the filtering window $l$ to 1 for **a** sin-data, **b** sincos-data, **c** ECG, **d** Uwave, and **e** plant. The red line represents the time when the anomaly occurred (color figure online)

#### 4.4.3 Plant

This dataset contains sensor data from monitoring the equipment states. Only one failure occurred during measurements with this device, at time 1,036,835. We generated training and observed datasets based on the original dataset up to time 497,545 and after time 497,546, respectively. We must omit further description of this dataset to maintain confidentiality.

## 5 Results

This section evaluates the performance of dLSTM using the various datasets. We denote our proposed method with $N$ predictive models as dLSTM($N$) and denote the Predet method in a similar manner.

### 5.1 Prediction accuracy performance

As described in Sect. 1.1, sensor data contain noise and multi-mode. Therefore, accurate predictions are challenging. dLSTM thus provides multiple predictive models with delayed predictions. Figure 8 shows prediction errors for the training dataset. This figure shows that the prediction error for dLSTM is greatly small and that dLSTM captures training data more accurately than do the comparison methods for all datasets.

We next check the effect of noise and multi-mode data. Figures 9 and 10 show raw data for sincos-data and the prediction error for the training dataset when dLSTM(2) and Single were applied, respectively. From Fig. 9, sincos-data contains noise, and prediction error from dLSTM(2) is consistently low as compared with Single. Focusing on Fig. 10, although raw data within windows $a$ and $b$ are similar, the following subsequences seem to be dissimilar, especially within the

**Table 1** Median values for normalized anomaly score $R(t)$ in the anomaly state (m-score)

| Dataset | dLSTM(2) | dLSTM(10) | Predet(2) | Predet(10) | Single | CAE | SAE | VAE |
|---|---|---|---|---|---|---|---|---|
| sin-data | 0.300 | 0.152 | 0.213 | 0.241 | 0.360 | **0.509** | − 0.051 | 0.143 |
| sincos-data | 0.345 | 0.186 | 0.355 | 0.375 | **0.439** | 0.346 | 0.006 | 0.163 |
| ECG | − 0.148 | − 0.114 | − 0.16 | − 0.128 | − 0.113 | − 0.134 | − 0.094 | − **0.044** |
| Uwave | − 0.114 | − 0.256 | − 0.011 | − 0.03 | **0.021** | − 0.13 | − 0.109 | − 0.136 |
| Plant | − 0.051 | − **0.045** | − 0.096 | − 0.093 | − 0.095 | − 0.135 | − 0.124 | − 0.106 |

Filtering window $l$ size is 1, and the number of iterations $M$ is 350. The best performance is highlighted in bold

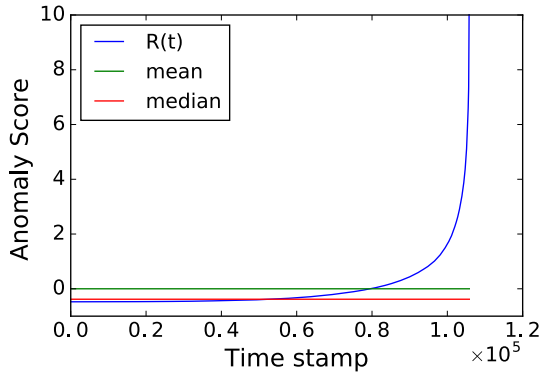


**Fig. 15** Sorted normalized anomaly scores within training data from Uwave dataset. The green and red lines respectively show the mean and median (color figure online)

red boxes. Figures 11 and 12 illustrate raw data from Uwave dataset and the prediction error for the training dataset when dLSTM(10) and Single were applied. Similar observations can be seen in this real dataset. Therefore, with the existence of multi-mode large prediction errors occur within the red boxes for these datasets. dLSTM performed well in the same region by using multiple models with delayed prediction and effectively handled multi-mode data.

## 5.2 Anomaly detection performance

This section introduces the results for performance of anomaly detection. To compare the performance of different methods, we normalized anomaly scores within the observed dataset using the mean and standard deviation of anomaly scores for normal data in each observed dataset and method. We normalized the anomaly score $S(t)$ at time $t$ by converting it to $R(t) \equiv \frac{S(t)-\mu_{S(t)}}{\sigma_{S(t)}}$, where $\mu_{S(t)}$ and $\sigma_{S(t)}$ are respectively the mean and standard deviation of $S(t)$ in the normal state. Note that while $S(t)$ takes a positive value, $R(t)$ can be negative. Normalizing the $R(t)$ values in the normal state for all methods makes it easy to compare differences between normal and abnormal anomaly scores. For example, if $R(t) = 3$, then observed data at time $t$ differs from the nor-

**Table 2** Median values for normalized anomaly score $R(t)$ in the anomaly state (m-score)

| Dataset | dLSTM(2) | dLSTM(10) | Predet(2) | Predet(10) | Single | CAE | SAE | VAE |
|---|---|---|---|---|---|---|---|---|
| sin-data | 22.091 | $93.685^{1st}$ | 12.846 | 15.390 | 18.805 | $26.215^{2nd}$ | 13.815 | 16.116 |
| sincos-data | 26.164 | $98.845^{1st}$ | 17.831 | 18.180 | 19.916 | $39.998^{2nd}$ | 38.472 | 34.865 |
| ECG | 4.100 | 1.863 | 1.434 | $6.401^{2nd}$ | $7.832^{1st}$ | 0.832 | 5.731 | 4.563 |
| Uwave | $4.075^{2nd}$ | 2.454 | 3.940 | 3.726 | $4.673^{1st}$ | 3.021 | 1.086 | 1.639 |
| Plant | $11.039^{1st}$ | $4.264^{2nd}$ | 1.691 | 1.732 | 1.932 | 0.668 | 2.127 | 1.632 |

Size of the filtering window $l$ is 1000, number of iterations $M$ is 350

**Table 3** Median values for normalized anomaly score $R(t)$ in the anomaly state (m-score). Size of the filtering window $l$ is 10,000, number of iterations $M$ is 350

| Dataset | dLSTM(2) | dLSTM(10) | Predet(2) | Predet(10) | Single | CAE | SAE | VAE |
|---|---|---|---|---|---|---|---|---|
| sin-data | 72.764 | $1319.163^{1st}$ | 49.487 | 56.195 | 77.105 | $84.246^{2nd}$ | 47.687 | 52.432 |
| sincos-data | 95.246 | $352.573^{1st}$ | 66.227 | 59.976 | 62.515 | 126.617 | $129.818^{2nd}$ | 106.388 |
| ECG | 32.265 | $296.942^{1st}$ | $84.353^{2nd}$ | 57.349 | 74.054 | 2.604 | 15.878 | 11.396 |
| Uwave | 12.145 | $65.758^{1st}$ | 13.791 | 12.136 | $15.870^{2nd}$ | 9.971 | 6.914 | 8.575 |
| Plant | $15.770^{1st}$ | 4.584 | 7.137 | 7.209 | 7.102 | 3.167 | $9.378^{2nd}$ | 5.602 |

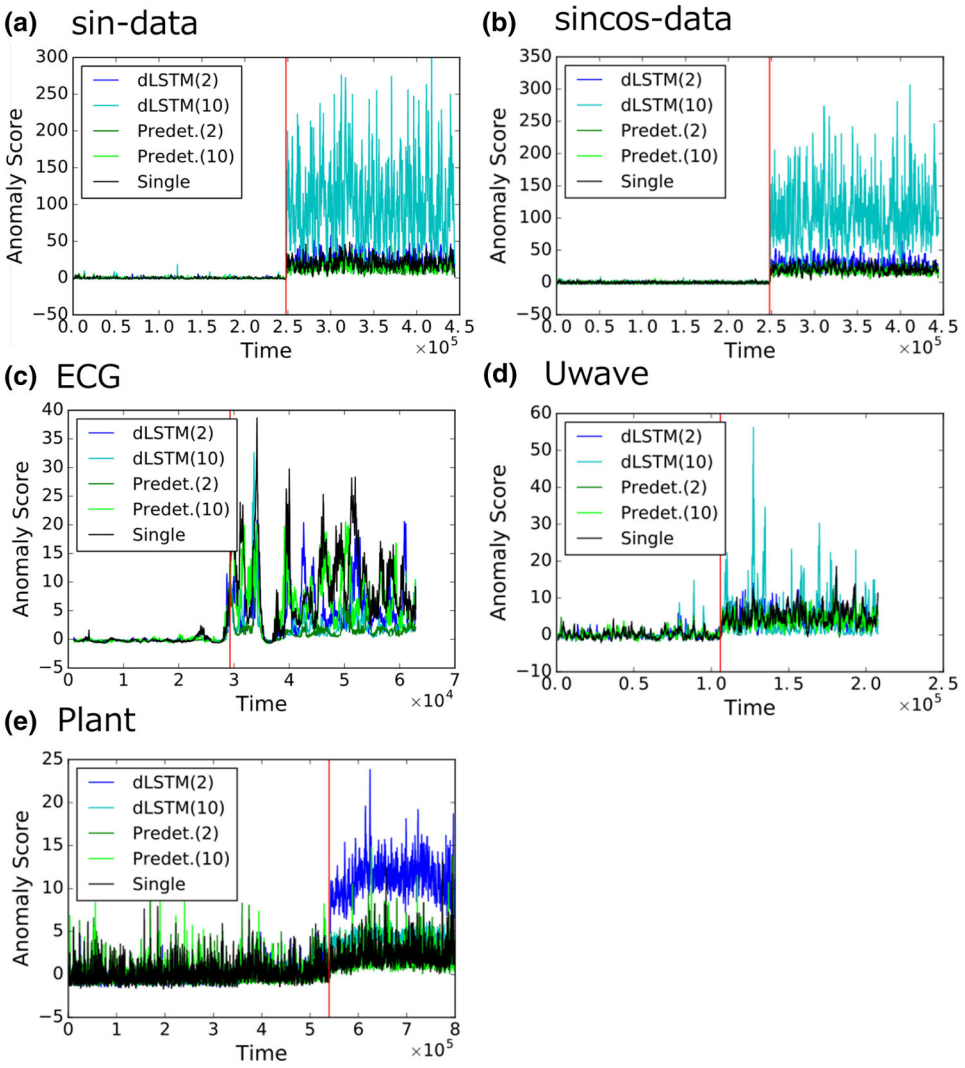


**Fig. 16** Anomaly score at each time after normalization ($R(t)$) using the prediction error, setting the size of the filtering window $l$ to 1000, for **a** sin-data, **b** sincos-data, **c** ECG, **d** Uwave, and **e** plant. The red line represents the time when the anomaly occurred (color figure online)

mal state by $3\sigma$. It follows that the anomaly detection method performs well if $R(t)$ values are large following an anomaly. To quantitatively evaluate $R(t)$, we provide a performance measure named the *m-score* $\in \mathbb{R}$. The m-score value is the median of normalized anomaly scores $R(t)$ within the abnormal state. When an anomaly occurs at time $t_1$ and the size of the observed dataset is $T_2$, the corresponding value m-score is

$$\text{m-score} = \text{median}_{i\in[t_1,t_1+1,\ldots,T_2]} R(i). \tag{17}$$

If the m-score is large, the method clearly distinguishes between abnormal and normal states.

We next evaluated the performance in terms of the filtering window size ($l$). As described in Sect. 3.3, the filtering window size is crucial for anomaly detection.

We first show the results when setting the filtering size window to 1, which is equivalent to executing only Steps 1 and 2. Figures 13 and 14 illustrate the performance of each method when setting the size of the filtering window $l$ to 1. Table 1 describes the median value of normalized anomaly scores within the abnormal state (m-score), setting the size of the filtering window $l$ to 1. Note that elements in Table 1 can take negative values.

In Figs. 13 and 14, we can visually identify the change to anomalous scores in sin-data and sincos-data. However, from Table 1, the median values in sin-data and sincos-data are greatly small (generally less than 0.5), and there is a high possibility that the anomaly cannot be detected. This implies that Step 3 is necessary for anomaly detection.

Several median values in Table 1 were negative values. To consider the cause of these negative values, we derive the m-score by two procedures, normalizing the anomaly score $R(t)$ and calculating the median of $R(t)$. We focus on the second procedure. If the distribution of the anomaly score is symmetric, the mean and the median values should be the same. Figure 15 shows the sorted normalized anomaly score $R(t)$ for Uwave in the training data. The scores have a long-

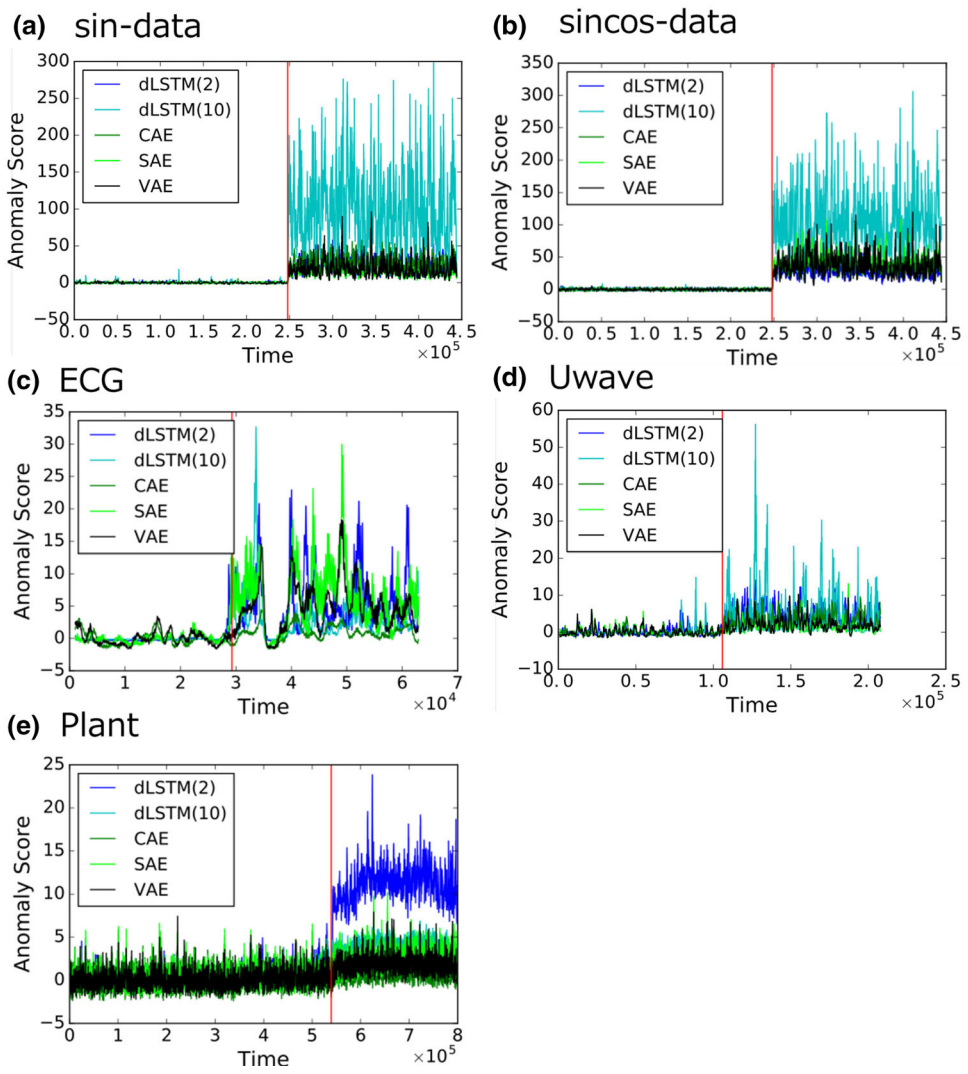


**Fig. 17** Anomaly score at each time after normalization ($R(t)$) using the reconstruction error, setting the size of the filtering window $l$ to be 1000, for **a** sin-data, **b** sincos-data, **c** ECG, **d** Uwave, and **e** plant. The red line represents the time when the anomaly occurred (color figure online)

tail distribution with mean and median values of 0.00 and $-0.38$, respectively. This distribution is asymmetric with a smaller median than mean, causing some values in Table 1 to take negative values. This also implies that there are outliers even in the training dataset from the perspective of deep learning. Because the median is generally a more robust and conservative measure than the mean, we adopt median values as a performance measure of $R(t)$.

## 5.3 Evaluation of step 3 effects

We evaluated the performance when setting a large filtering window size (1000 or 10,000).

Tables 2 and 3 describe the median normalized anomaly score $R(t)$ for each method, setting the filtering window $l$ to 1000 and 10,000. Values marked 1st and 2nd respectively represent the largest and second-largest values in the dataset.

### 5.3.1 Filtering window: 1000

This section describes the results when setting the filtering window size to be 1000. According to Table 2, our proposed approaches (dLSTM(2) and dLSTM(10)) worked better than did the comparative methods for the artificial and Plant datasets. For the artificial datasets (sin-data and sincos-data), dLSTM(10) was significantly better, and dLSTM(2) showed the best performance for the Plant dataset. Meanwhile, Single produced the best performance for the ECG and Uwave datasets. However, the difference in m-score between Single and dLSTM is at most about 4 (7.832 (Single ECG) $-$ 4.100 (dLSTM(2) ECG)), and this difference is sufficiently small as compared with the cases when the artificial and Plant datasets are applied. This implies that there is no significant difference in performance among methods when ECG and Uwave are applied.

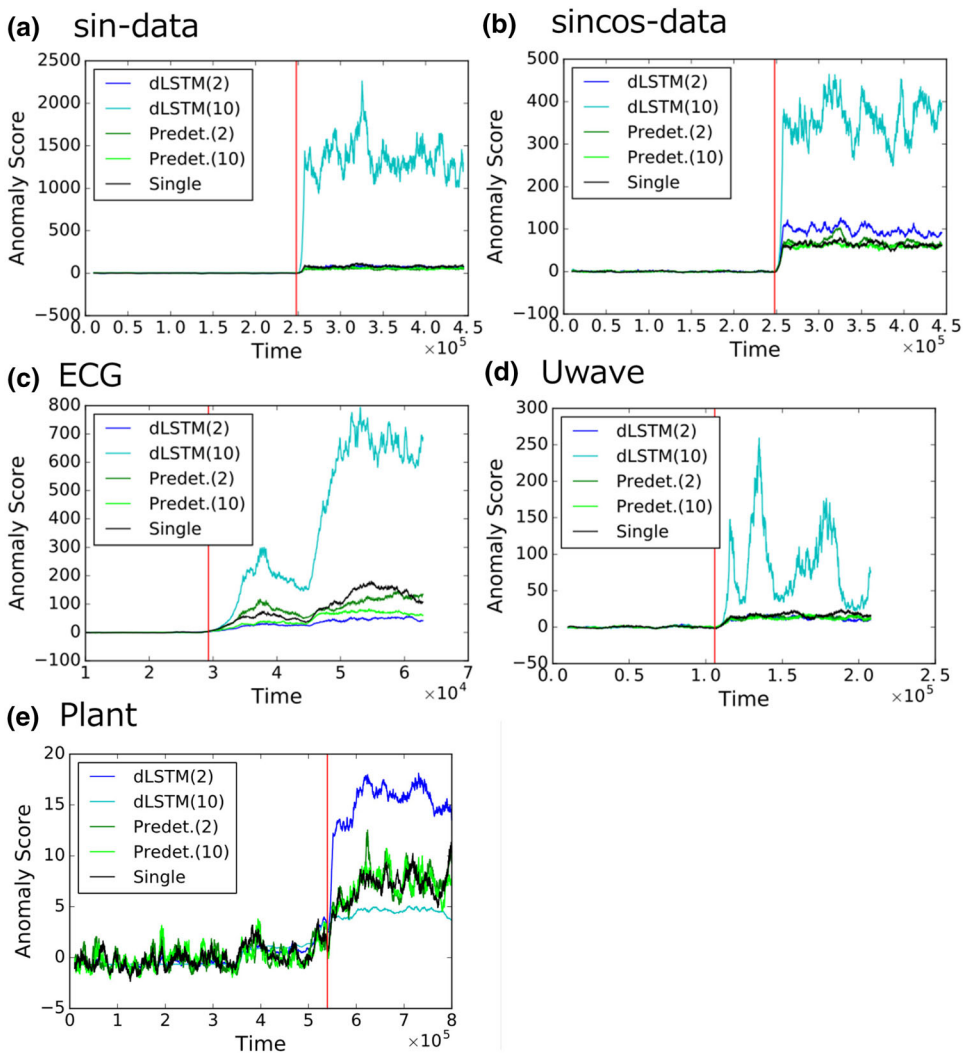


**Fig. 18** Anomaly score at each time after normalization ($R(t)$) using the prediction error, setting the size of the filtering window $l$ to be 10,000, for **a** sin-data, **b** sincos-data, **c** ECG, **d** Uwave, and **e** plant. The red line represents the time when the anomaly occurred (color figure online)

Figures 16 and 17 compare the performances of the methods. These figures correspond to Table 2. Although there is no obvious difference between the methods as in the case of ECG and Uwave, we find that dLSTM is superior, as for other datasets. This is because both ECG and Uwave are biological data, and thus subject to noise. Removing the impact of noise is necessary to obtain positive performance, and setting a large filtering window is a workable alternative. Although waveforms for sincos-data are discontinuous according to Eqs. (13) to (16), and despite Plant being a real dataset thus potentially complex, the impact of noise is not as large as in the biological data. Therefore, dLSTM outperformed methods in comparison when setting the filtering window to be only 1000.

#### 5.3.2 Filtering window: 10,000

This section shows the results when setting the filtering window size to 10,000. According to Table 3, dLSTM(10) worked better than the comparative methods for all datasets except for Plant, where dLSTM(2) showed the best performance. Because setting a large filtering window size in Step 3 effectively mitigates the impact of noise, the m-score values improved for every method and dataset pair. In real operations, we set the threshold and alarm if $R(t)$ exceeds the threshold. Because there is a clear difference in $R(t)$ between the normal and abnormal state, dLSTM is advantageous and the detection performance does not change significantly even when thresholds are slightly changed. We refer to a method as being *robust* if its performance does not significantly change when the threshold is slightly changed.

Figures 18 and 19 compare the performances of the methods. These figures correspond to Table 3. The behavior of $R(t)$ is stable as compared with the case when the filtering window size is 1000, and dLSTM outperformed other methods in terms of m-score.

Our approach generally outperformed the comparative methods. As Tables 2 and 3 show, dLSTM(10) showed the

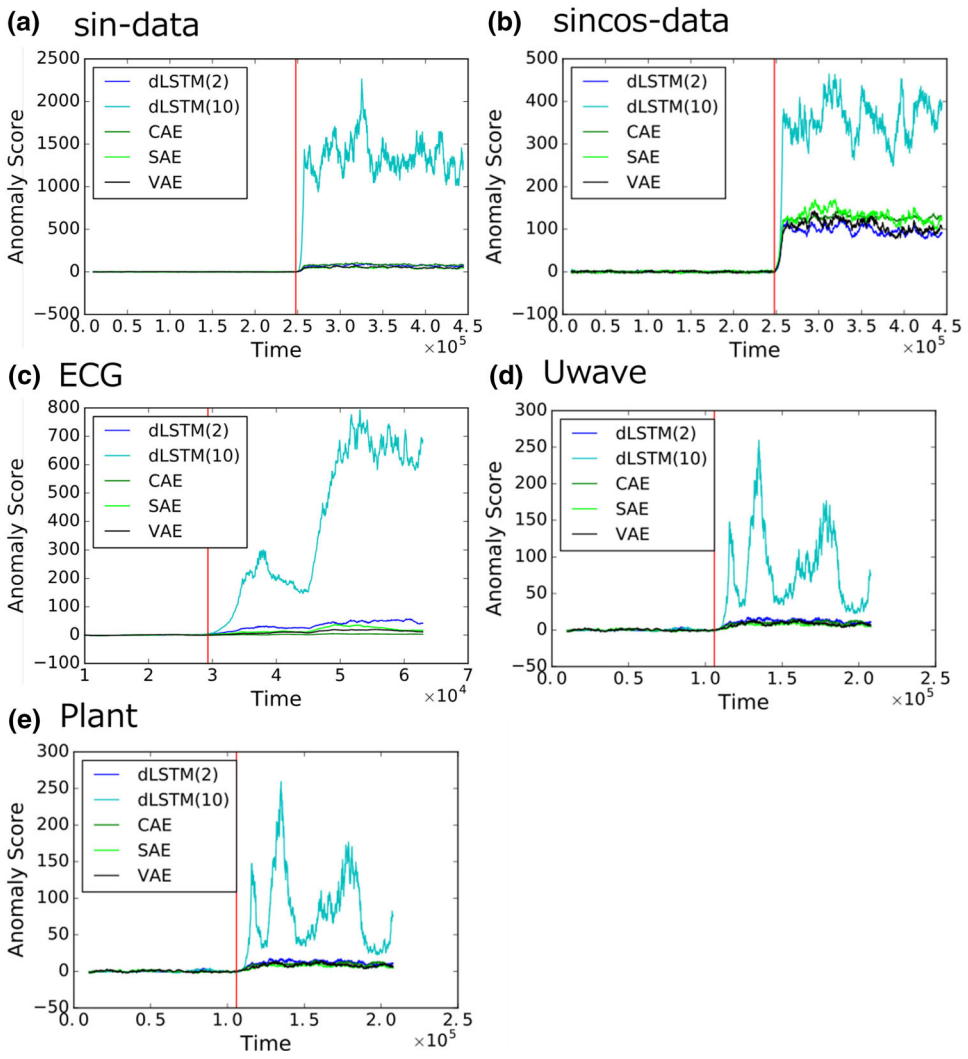


**Fig. 19** Anomaly score at each time after normalization ($R(t)$) using the reconstruction error setting the size of the filtering window $l$ to be 10,000, for **a** sin-data, **b** sincos-data, **c** ECG, **d** Uwave, and **e** plant. The red line represents the time when the anomaly occurred (color figure online)

best performance for 6 out of 10 measures, while either dLSTM(2) or dLSTM(10) showed the best performance for 8 out of 10 measures, when separately counting different values of $l$.

#### 5.3.3 The effect of filtering window size

We next consider the effect of large value for the filtering window size. Figure 20 shows an enlarged view of Figs. 16 and 18, and 21 shows an enlarged view of Figs. 17 and 19. Red lines represent times when an anomaly occurred, and orange lines represent 1000 timestamps after an anomaly occurred. A large filtering window is helpful in removing the negative impact of noise, and anomaly score values are stable when the filtering window size is 10,000. Note that $R(t)$ grows gradually after an anomaly, since we take the median of the latest $l$ values of the prediction error.

One side effect of setting a large filtering window size is that it takes longer to increase the anomaly score. Except for the (e) Plant data, anomalies can be recognized within 1000 timestamps after their occurrence when setting the filtering window size to be 1000, because the anomaly score grows within this period. However, we cannot recognize changes in anomaly score indicated by the orange lines when setting the filtering window size to be 10,000.

Although the small filtering window size is helpful in promptly detecting anomalies, its performance can be high, depending on the threshold. This is because the m-score is small when the filtering window size is 1000, making it more sensitive to the threshold. To aim for stable operation and to reduce false-positive alarms, setting a large filtering window is a powerful option.

### 5.4 Evaluation of threshold robustness

This section quantitatively evaluates the threshold effects for anomaly score $R(t)$. We prepared six performance measures:

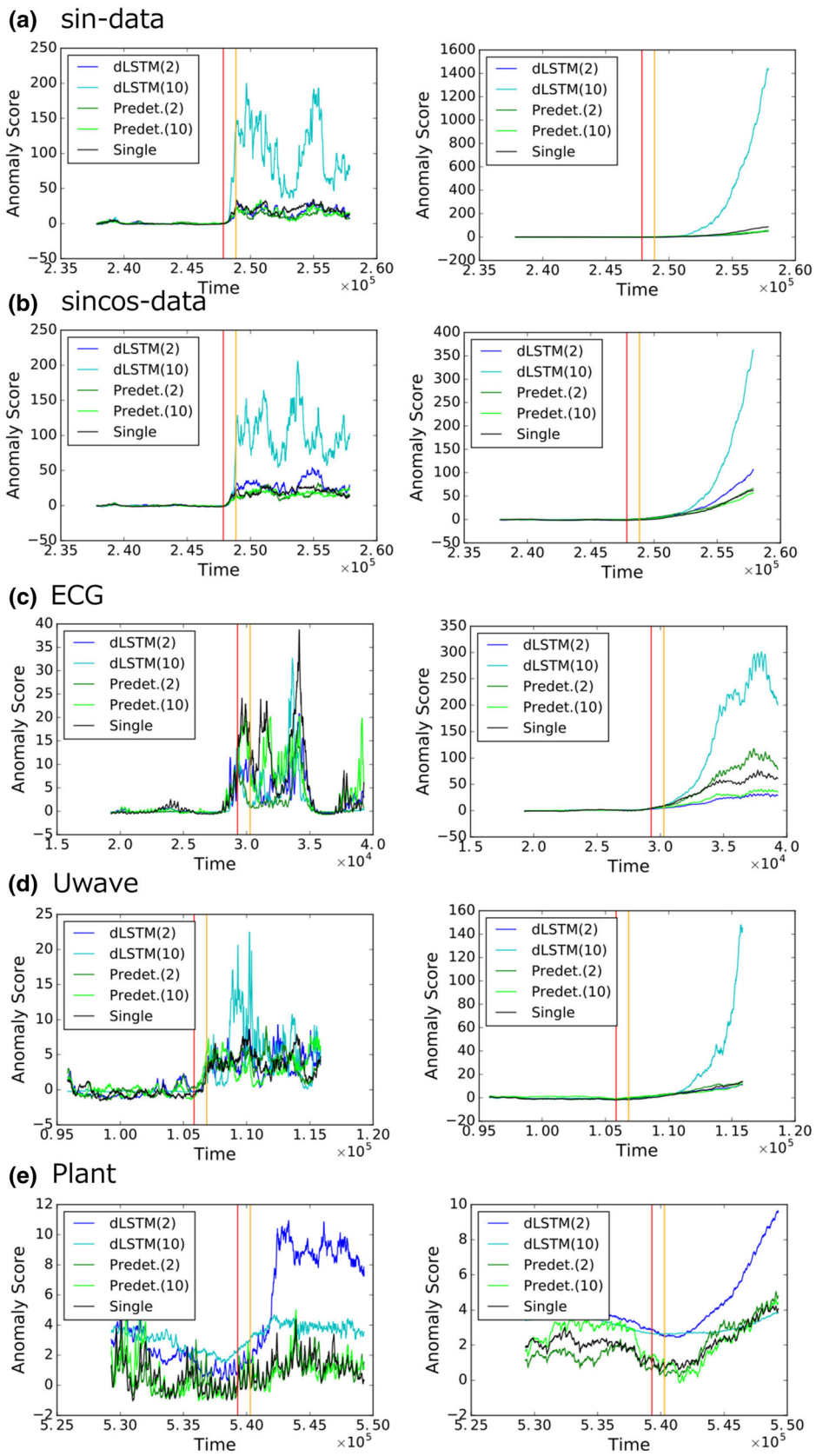


**Fig. 20** Enlarged view of anomaly scores at each time after normalization using prediction error ($R(t)$). Left and right figures respectively correspond to filtering window $l$ sizes of 1000 and 10,000 for **a** sin-data, **b** sincos-data, **c** ECG, **d** Uwave, and **e** plant. The red and orange lines respectively represent times when the anomaly occurred and 1000 timestamps after (color figure online)

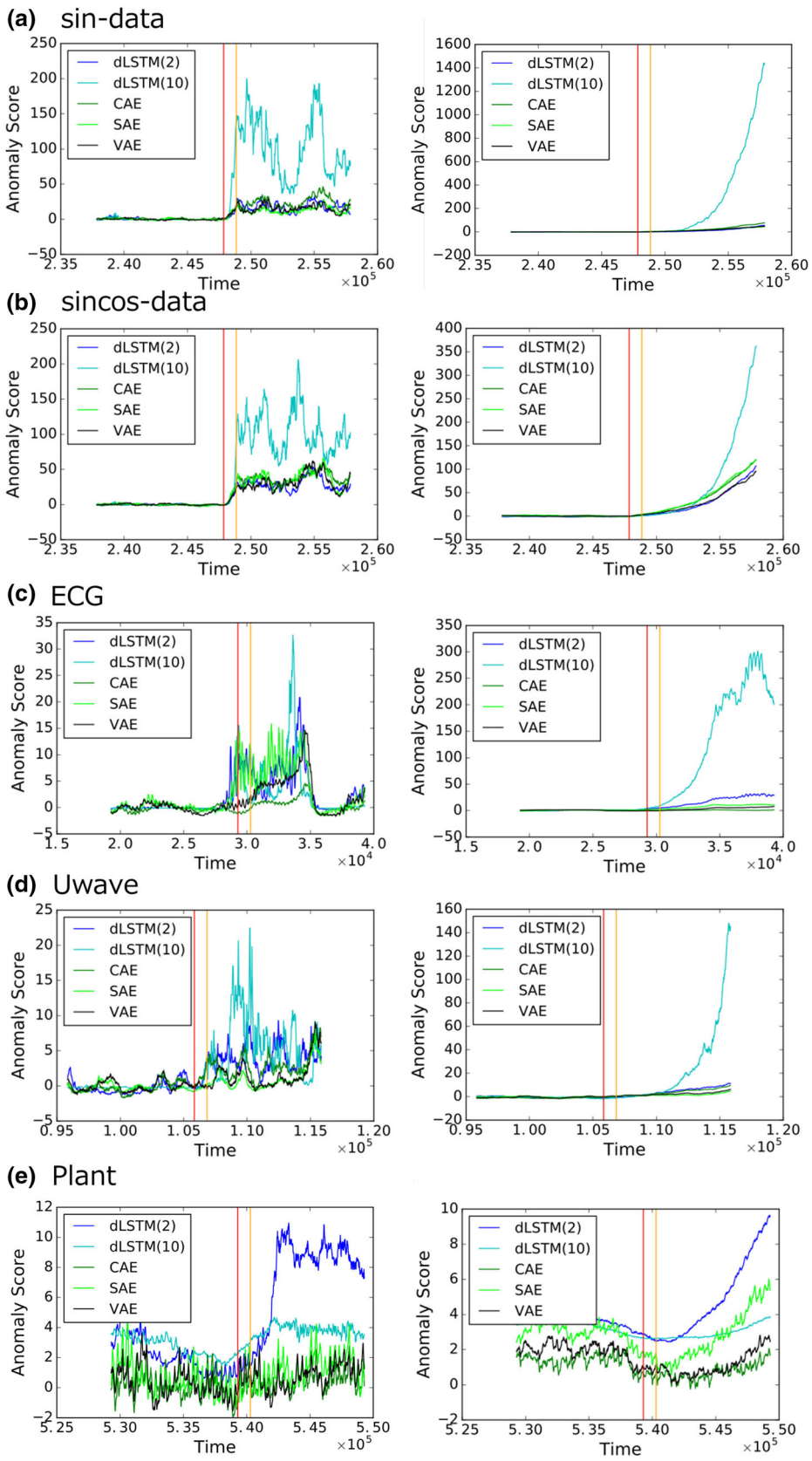


**Fig. 21** Enlarged view of anomaly score at each time after normalization using reconstruction error ($R(t)$). The left and right figures correspond to setting the size of the filtering window $l$ to 1000 and 10000, respectively. **a** sin-data, **b** sincos-data, **c** ECG, **d** Uwave, and **e** plant. The red and orange lines respectively represent times when the anomaly occurred and 1000 timestamps after (color figure online)

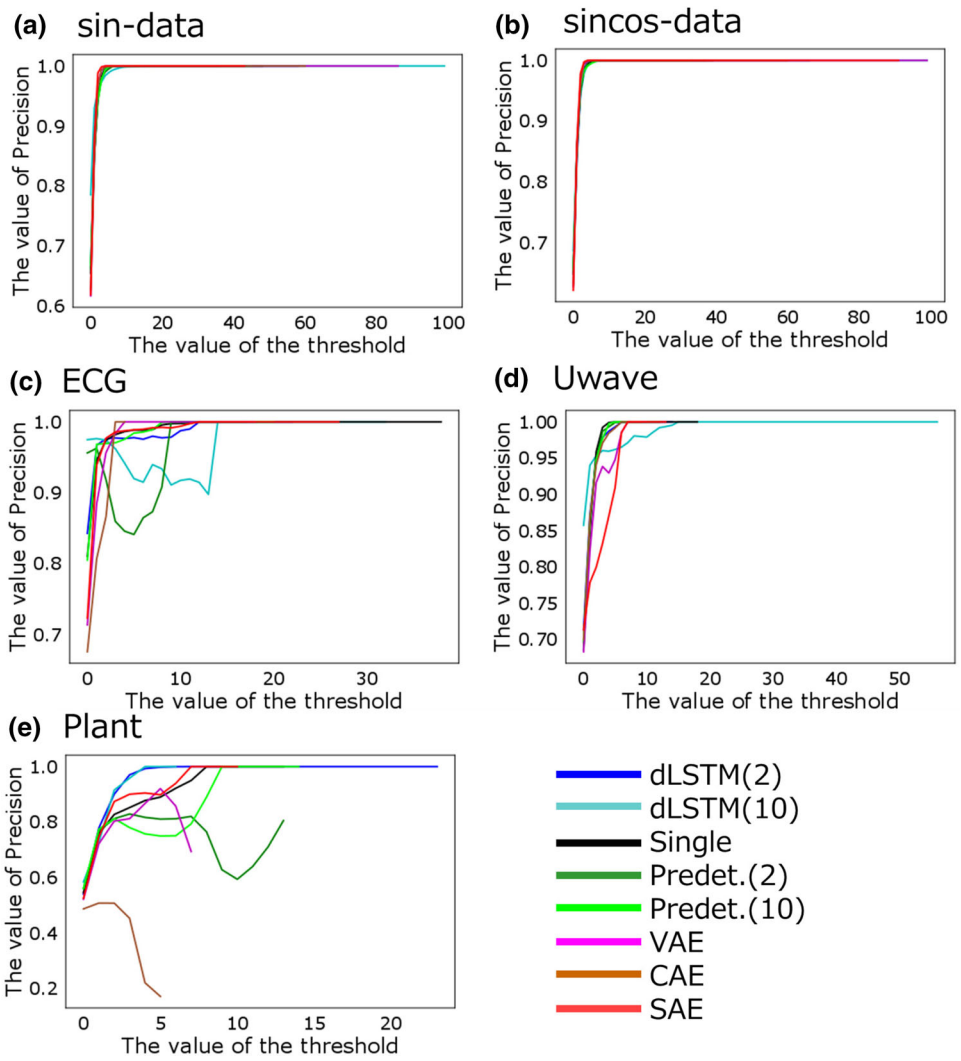


**Fig. 22** Values for precision setting the filtering window $l$ size to 1000

*Precision*, *Recall*, *F-measure*, *False Positive Number* (FPN), *Overlooking Period* (OP), and *Confidence Margin* (CM).

We first varied the threshold from 0 to 100 and obtained the resulting values for Precision, Recall, and F-measure based on $R(t)$ and the threshold. If $R(t)$ exceeds the threshold, we classify the data at time $t$ as abnormal. We regard an abnormal data point as a positive sample and a normal data point as a negative one. Under a small threshold, most data are classified as abnormal and the value of Recall is high. Figures 22, 23, 24, 25, 26 and 27 show values for Precision, Recall, and F-measure setting the filtering window size as 1000 or 10,000 against the threshold. According to Figs. 22 and 23, a large filtering window size helps stabilize the performance of Precision by mitigating the impact of noise. The behavior of Precision is similar to that of other methods, and there are only small differences in performance. In contrast, Figs. 24, 25, 26 and 27 show that both of our proposed methods (dLSTM(2) and dLSTM(10)) produced high values for Recall and F-measure in most cases. For example, in Fig. 27e, dLSTM(10) performed best among all methods in terms of F-measure. If the threshold is between 5 and 10, the F-measure is over 0.9, supporting the results in Table 2 showing the effectiveness of our proposed methods. Since the F-measure considers the trade-off between Precision and Recall, it is an important practical indicator and our proposed method achieved the stable performance by varying threshold value.

We next introduce FPN and OP. FPN represents the number of timestamps exceeding the threshold for normal data within the observed dataset. A lower FPN value denotes fewer false positives, and OP represents how rapidly we detect anomalies. The definition of OP is the time period between the occurrence of an anomaly and the first time after the anomaly when the anomaly score exceeds the threshold in the observed data. A lower OP value denotes prompt anomaly detection. Let $T_2$, $t_1$, and $C \in \mathbb{R}$ respectively be the size of the observed dataset, the time when the anomaly happened, and the threshold. Mathematical definitions of FPN and OP are

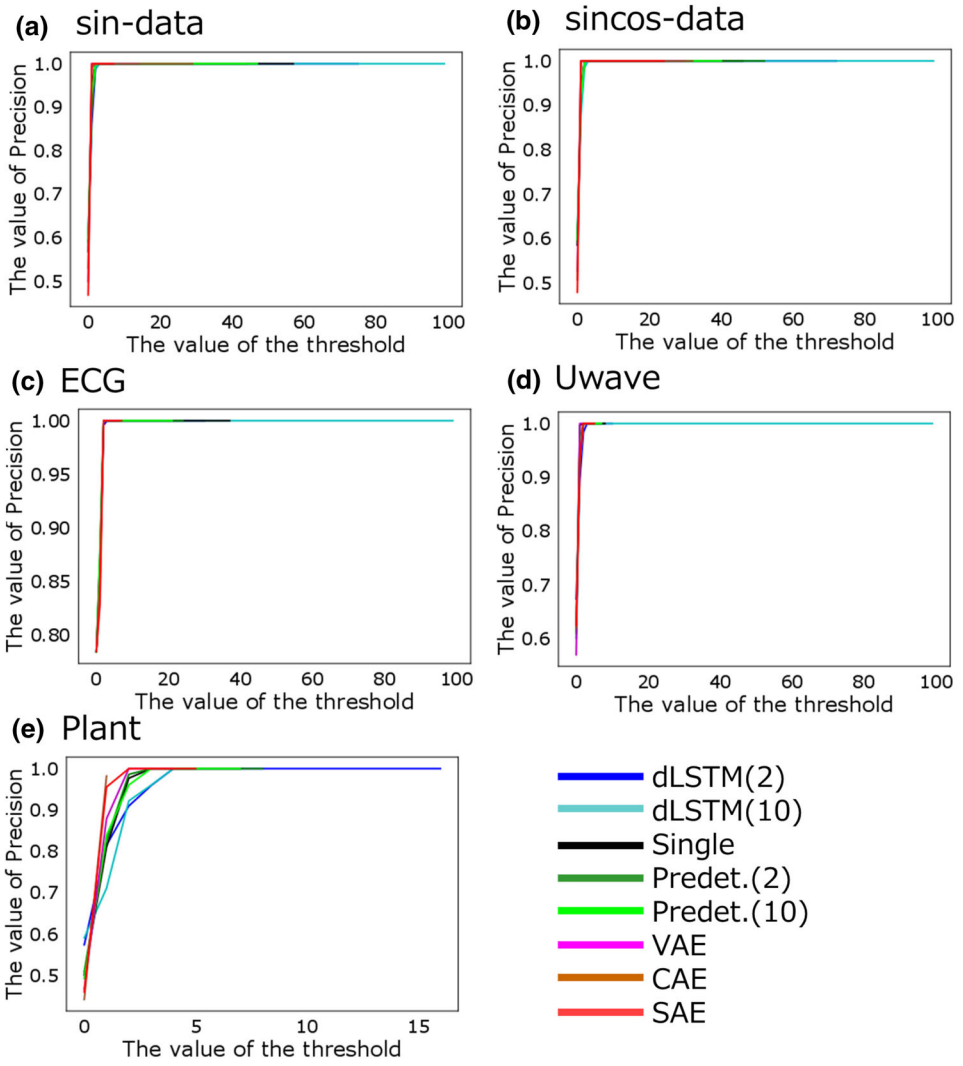


**Fig. 23** Values for precision setting the filtering window $l$ size to 10,000

$$\mathrm{FPN} = \sum_{t=1}^{t_1} \mathbb{I}_{R(t)>C}, \tag{18}$$

$$\mathrm{OP} = \min_k \{k | k > t_1 \wedge R(k) > C\}. \tag{19}$$

We varied the threshold from 3 to 99 and calculated mean FPN and OP values for each method and dataset. Tables 4, 5, 6 and 7 summarize the results. Values marked 1st represent the smallest values in the dataset.

Tables 4 and 6 show that either dLSTM(2) or dLSTM(10) detected anomalies in most cases, regardless of the size of the filtering window. Although the corresponding mean FPN values are larger than in the comparison methods (Tables 5 and 7), the difference is at most about 110 (110.082 (Plant dSLTM(2)) −0.000 (Plant VAE)) and this value is significantly small considering the size of the normal data within the observed dataset (539289 = 1036835 − 49746). dLSTM thus promptly detects anomalies while maintaining a low false-positive rate for each filtering window size.

We next consider valid threshold ranges. CM is the margin between normal and abnormal data when 1% false positives and false negatives are allowed.

Let Percentile$(\boldsymbol{x}, q)$, $t_1$, $\boldsymbol{x} \in \mathbb{R}^{T_2}$, and $l$ respectively be the function to compute the $q$% percentile, the time when the anomaly occurred, the observed dataset with size $T_2$, and the filtering window size. The definition of CM is

$$\mathrm{CM} = \mathrm{Percentile}(\boldsymbol{x}_{t_2+l:T_2}, 1) - \mathrm{Percentile}(\boldsymbol{x}_{1:t_1-l}, 99). \tag{20}$$

The method is robust against the threshold as the value of CM is large. If CM takes a negative value, it follows that it is impossible to maintain false positives and negatives below 1%, regardless of the threshold.

Tables 8 and 9 summarize the CM for each method and dataset. Table 8 shows that all CM values are below 0 for

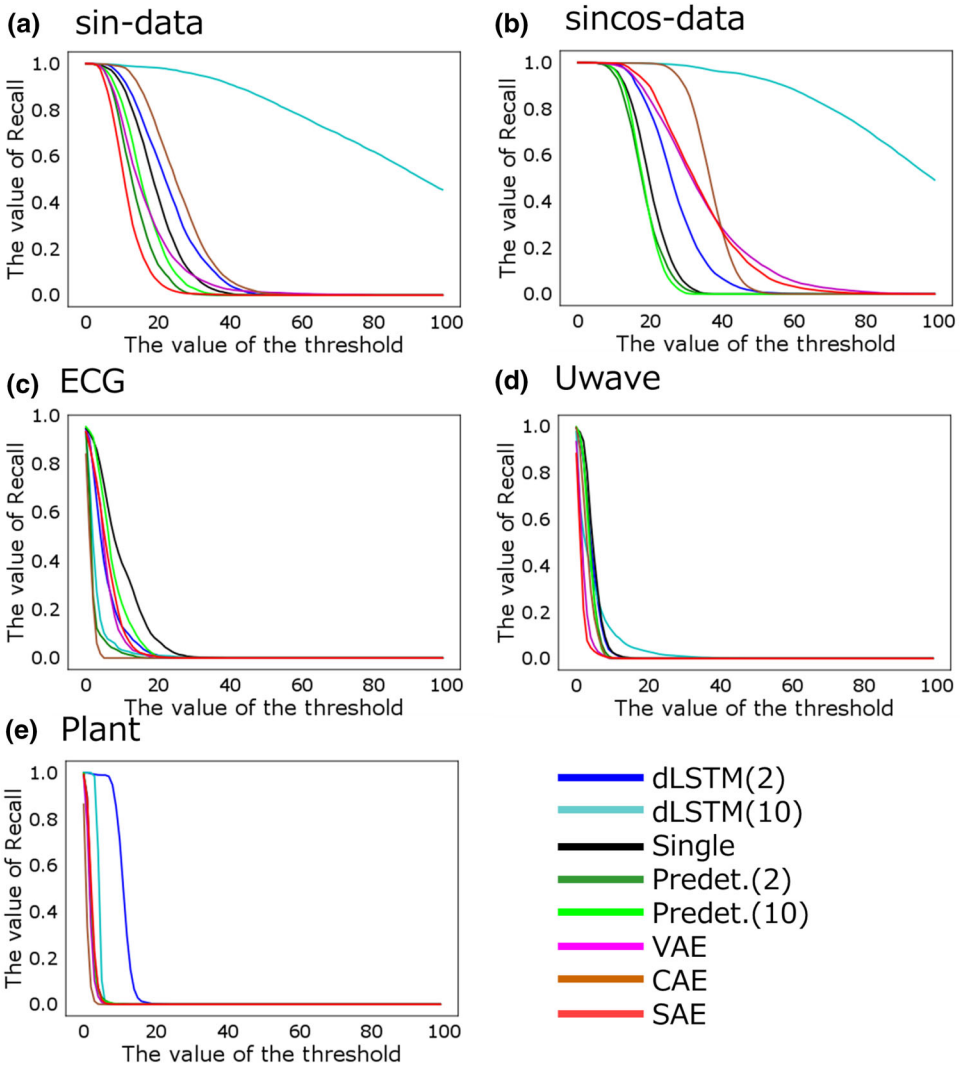


**Fig. 24** Values for recall setting the filtering window $l$ size to 1000

ECG and Uwave when the filtering window size is 1000. It is thus impossible to set a proper threshold. This result also suggests that a large filtering window size is necessary. dLSTM showed the positive performance for other datasets. Focusing on Plant, only dLSTM(2) took a positive value and achieved acceptable levels (false positives and negative were both below 1%).

dLSTM produced the best results when setting the filtering window size to 10,000. Therefore, dLSTM is superior in that it shows positive results in many cases, even when the threshold value is changed.

From the viewpoint of FPN, OP, and CM, dLSTM promptly detects anomalies while maintaining a low number of false positives, and its performance is robust against thresholds.

### 5.5 Computation time

This section considers computation times. We implemented all methods in Chainer and computed two computation times on a single GPU (Nvidia M40). The first computation time is that required for building the model for each iteration, corresponding to Step 1. The second computation time is that required to obtain the prediction (or reconstruction) error for each window size $w$, corresponding to Step 2. We respectively refer these as the "building time" and the "inference time." Table 10 summarizes the results when applying sin-data. For building times, methods that do not embed LSTM (CAE, SAE, and VAE) are computationally efficient. GPUs are inherently efficient at parallel processing. How-

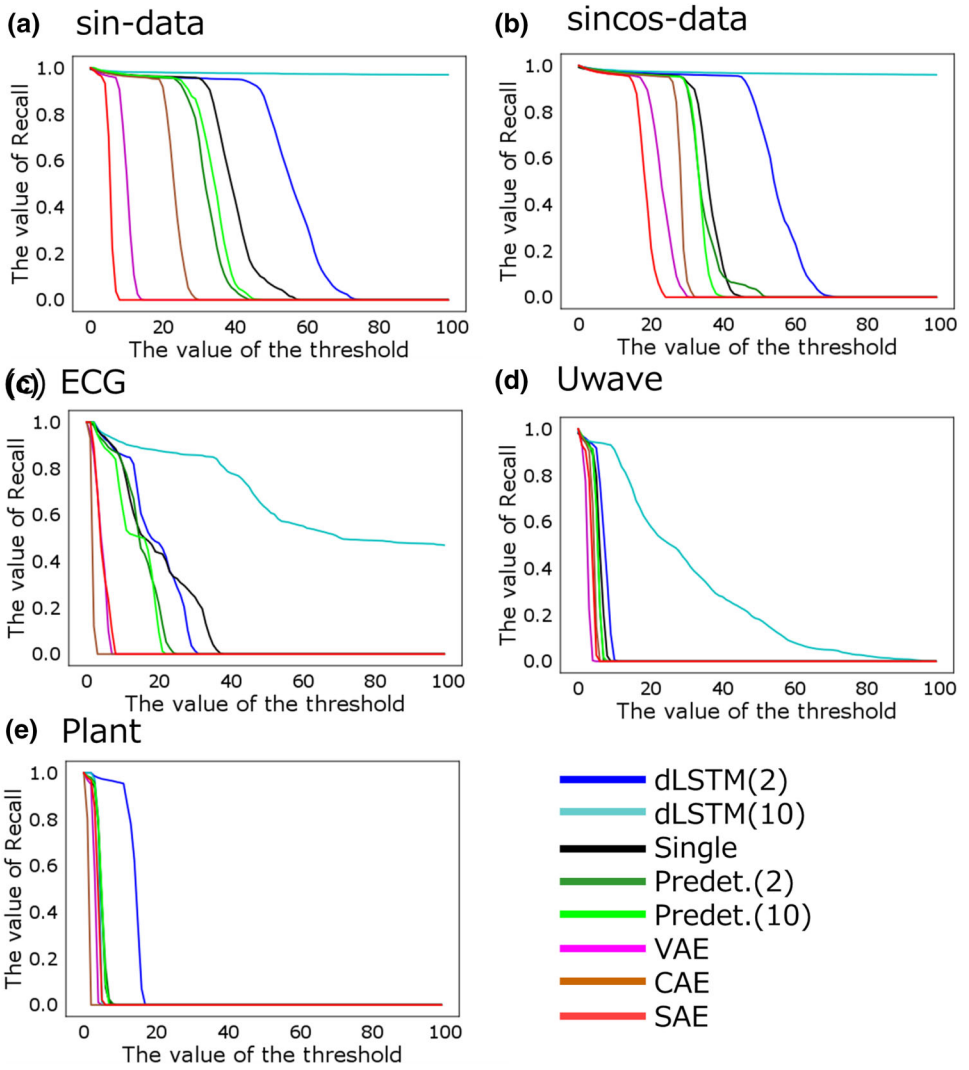


**Fig. 25** Values for recall setting the filtering window $l$ size to 10,000

ever, LSTM requires sequential processing of time-series data to perform back propagation, and GPUs are not suited to this task. This suggests that methods using LSTM, including dLSTM, are time-consuming. However, once we build a precise model, we can constantly utilize this model until an abnormality occurs. Creating accurate models is thus a top priority, even if creating accurate models takes a long time.

Next, we focus on the inference time. The table also shows that between dLSTM and other methods, differences in inference time are at most about five times. There is no need to perform back propagation to calculate the inference times. Therefore, the performance of inference time for dLSTM did not decrease as much as compared with CAE, SAE, or VAE. Since the observed dataset is continuously acquired, processing speed is important. This poses no particular problem so long as the speed for obtaining the prediction error is faster than that needed to acquire the observed dataset. The proposed method can process data with $w = 100$ timestamps in 0.133 s, which is sufficient for manufacturing companies in most cases.

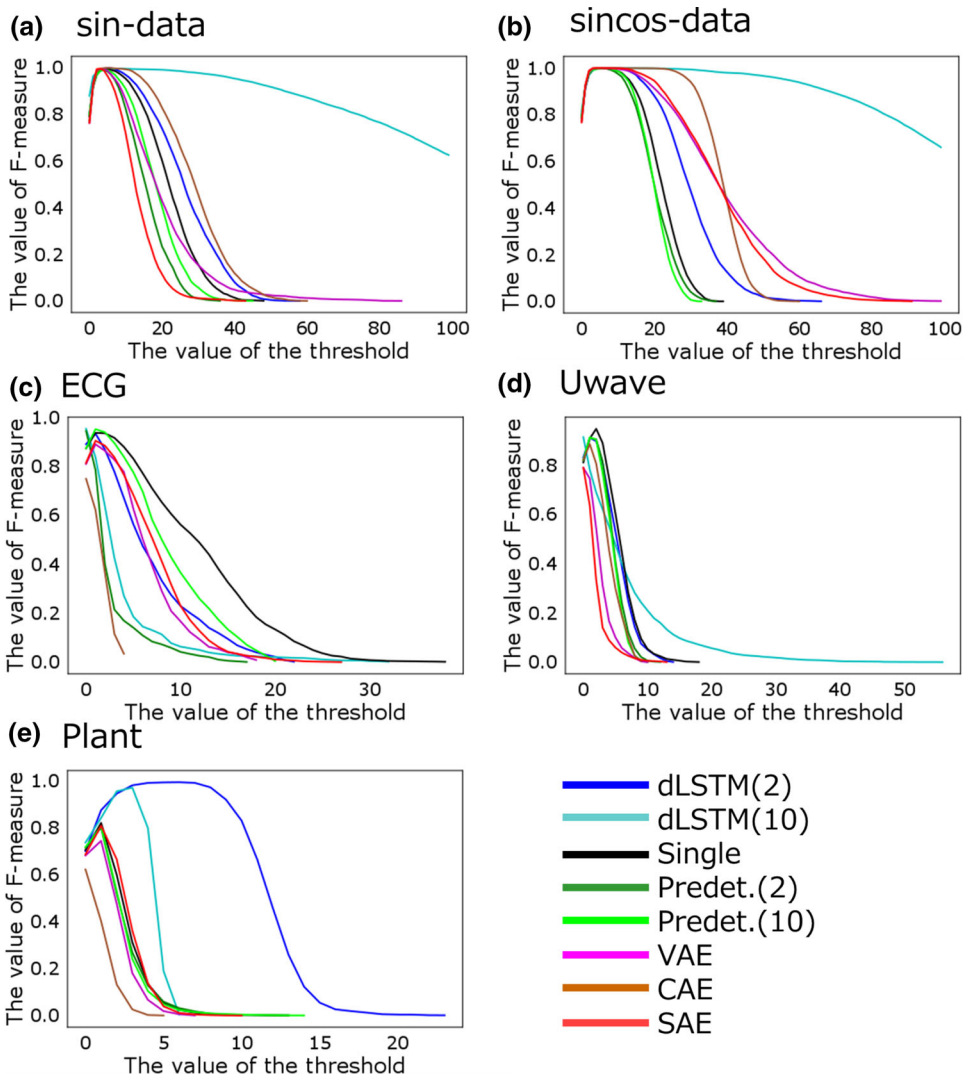


**Fig. 26** Values for F-measure setting the filtering window $l$ size to 1000

### 5.6 Results on non-deep learning methods

The methods considered so far are based on deep learning. This section thus addresses a non-deep learning method. Online change point detection methods such as Change Finder [23] are widely used for anomaly detection. The AR model is embedded in this model to capture the behavior of time-series data. Figures 28 and 29 show the results of applying Change Finder to the Plant dataset. The values suggest that we cannot detect the occurrence of anomalies. One reason for this is that online methods such as Change Finder should activate only when the state of time-series data changes. In the problem setting, there was only one time when the state changed, corresponding to the time when the anomaly happened. In contrast, methods based on deep learning obtain anomaly scores that measure data deviation from the normal state. Therefore, these methods should continuously activate so long as the state remains abnormal following anomaly occurrence, making anomaly detection easy. Furthermore, AR models can capture only linear relations. Since real data contains noise and periodicity is not constant, real data is more varied. Figure 30 shows raw data and predic-

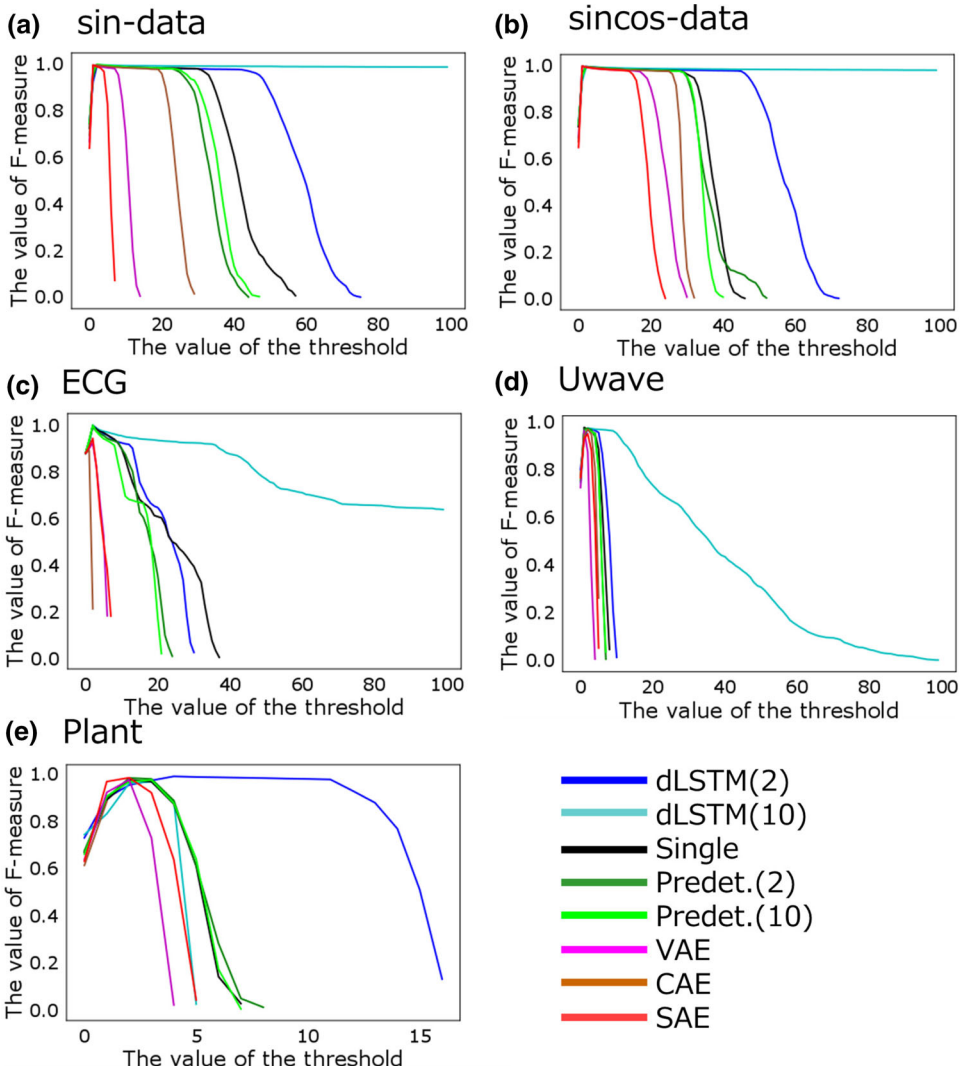


**Fig. 27** Values for F-measure setting the filtering window $l$ to 10,000

**Table 4** Mean OP values with filtering window $l$ size 1000

| Dataset | dLSTM(2) | dLSTM(10) | Predet(2) | Predet(10) | Single | CAE | SAE | VAE |
|---|---|---|---|---|---|---|---|---|
| sin-data | 91166.227 | 664.691[1st] | 137052.072 | 120031.309 | 106726.515 | 83884.959 | 108036.093 | 41155.175 |
| sincos-data | 76314.258 | 868.124[1st] | 131125.598 | 138563.474 | 125136.216 | 74744.67 | 14434.041 | 15755.052 |
| ECG | 27118.742 | 23956.155 | 28755.402 | 27412.34 | 21861.474[1st] | 33035.515 | 26466.866 | 29395.34 |
| Uwave | 90424.371 | 53161.175[1st] | 93958.598 | 94098.588 | 90561.052 | 94672.897 | 95177.34 | 94672.041 |
| Plant | 211960.299[1st] | 250608.062 | 244954.454 | 234232.629 | 241073.959 | 254073.907 | 243577.948 | 249700.175 |

tion error when a linear model ($\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$) is applied to sincos-data. Comparing Fig. 30 with Figs. 9 and 10, the prediction error becomes too large to capture such complicated relationships using only linear transformations, this method is not suited to this problem setting.

## 6 Conclusions

In this paper, we proposed dLSTM as an anomaly detection method based on prediction error for time-series data. Time-series data are often corrupted by noise and have multi-mode, deteriorating the prediction error. It is thus difficult to correctly perform anomaly detection. The proposed method provides multiple prediction models with delayed prediction. Since we can delay the timing for obtaining predictive values

**Table 5** Mean FPN values with filtering window $l$ size 1000

| Dataset | dLSTM(2) | dLSTM(10) | Predet(2) | Predet(10) | Single | CAE | SAE | VAE |
|---|---|---|---|---|---|---|---|---|
| sin-data | 60.814 | 160.062 | 76.577 | 65.928 | 44.289 | 21.598 | 9.320[1st] | 28.835 |
| sincos-data | 38.062 | 70.423 | 43.443 | 70.670 | 52.381 | 15.639 | 13.113[1st] | 27.546 |
| ECG | 20.866 | 20.691 | 24.608 | 23.206 | 19.093 | 0.969[1st] | 14.773 | 1.866 |
| Uwave | 26.361 | 78.495 | 14.567 | 18.722 | 9.268[1st] | 24.320 | 27.072 | 25.753 |
| Plant | 107.680 | 114.01 | 177.588 | 197.000 | 132.072 | 59.784[1st] | 100.247 | 84.093 |

**Table 6** Mean FPN values with filtering window $l$ size 10,000

| Dataset | dLSTM(2) | dLSTM(10) | Predet(2) | Predet(10) | Single | CAE | SAE | VAE |
|---|---|---|---|---|---|---|---|---|
| sin-data | 19710.299 | 4178.619[1st] | 75579.062 | 56905.412 | 13743.351 | 12650.000 | 72940.918 | 65043.567 |
| sincos-data | 7205.598 | 5850.763[1st] | 21533.082 | 61789.588 | 51561.536 | 6273.536 | 6111.340 | 7162.763 |
| ECG | 21416.093 | 3080.619[1st] | 4405.062 | 15552.670 | 7674.505 | 33304.649 | 26438.619 | 29929.155 |
| Uwave | 88199.897 | 8059.649[1st] | 87416.969 | 89694.979 | 84333.371 | 91852.031 | 95093.68 | 91943.134 |
| Plant | 221745.701[1st] | 253332.474 | 238183.701 | 242677.000 | 243789.247 | 254046.814 | 236747.351 | 248515.000 |

**Table 7** Mean FPN values with filtering window $l$ size 10,000

| Dataset | dLSTM(2) | dLSTM(10) | Predet(2) | Predet(10) | Single | CAE | SAE | VAE |
|---|---|---|---|---|---|---|---|---|
| sin-data | 0.000[1st] | 21.691 | 17.186 | 4.165 | 0.000[1st] | 16.381 | 0.000[1st] | 3.351 |
| sincos-data | 3.340 | 22.227 | 0.000[1st] | 18.258 | 0.000[1st] | 17.66 | 0.000[1st] | 0.000[1st] |
| ECG | 0.041 | 4.474 | 4.928 | 3.567 | 3.588 | 0.000[1st] | 0.000[1st] | 0.000[1st] |
| Uwave | 16.062 | 5.845 | 0.000[1st] | 0.000[1st] | 0.000[1st] | 0.000[1st] | 0.000[1st] | 0.000[1st] |
| Plant | 186.196 | 110.082 | 19.062 | 83.784 | 45.938 | 9.577 | 55.948 | 0.000[1st] |

**Table 8** CM with filtering window $l$ size of 1000

| Dataset | dLSTM(2) | dLSTM(10) | Predet(2) | Predet(10) | Single | CAE | SAE | VAE |
|---|---|---|---|---|---|---|---|---|
| sin-data | 3.572 | 8.432[1st] | 0.436 | 0.550 | 1.650 | 7.662 | 1.898 | 1.441 |
| sincos-data | 8.857 | 26.636[1st] | 4.135 | 5.349 | 5.128 | 23.715 | 12.808 | 8.777 |
| ECG | − 1.107 | − 0.05 | − 0.271 | − 1.335 | − 2.109 | − 4.174 | − 2.736 | − 4.329 |
| Uwave | − 3.217 | − 5.863 | − 2.556 | − 2.870 | − 2.084 | − 3.359 | − 3.958 | − 4.019 |
| Plant | 3.546[1st] | − 0.370 | − 3.245 | − 3.476 | − 3.112 | − 3.496 | − 3.080 | − 3.249 |

**Table 9** CM with filtering window $l$ size of 10,000

| Dataset | dLSTM(2) | dLSTM(10) | Predet(2) | Predet(10) | Single | CAE | SAE | VAE |
|---|---|---|---|---|---|---|---|---|
| sin-data | 54.799 | 1005.518[1st] | 34.492 | 38.126 | 58.884 | 68.342 | 31.557 | 36.416 |
| sincos-data | 77.713 | 261.492[1st] | 55.075 | 49.344 | 48.66 | 112.531 | 101.551 | 79.588 |
| ECG | 23.940 | 150.594[1st] | 50.799 | 29.697 | 37.485 | 0.255 | 10.659 | 7.361 |
| Uwave | 5.267 | 21.638[1st] | 7.240 | 7.923 | 10.370 | 4.342 | 1.883 | 2.078 |
| Plant | 8.817[1st] | 1.060 | 1.785 | 1.484 | 1.122 | − 0.600 | 4.000 | 1.050 |

**Table 10** Computation times for each model [s]

| | dLSTM(2) | dLSTM(10) | Predet(2) | Predet(10) | Single | CAE | SAE | VAE |
|---|---|---|---|---|---|---|---|---|
| Building time | 97 | 235 | 109 | 255 | 89 | 0.187 | 0.157 | 0.339 |
| Inference time | 0.133 | 0.118 | 0.101 | 0.117 | 0.0774 | 0.0404 | 0.0367 | 0.0317 |

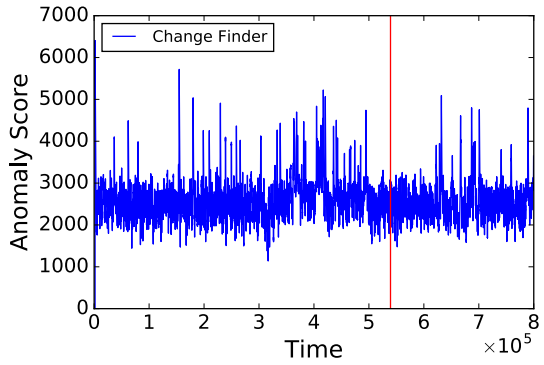


**Fig. 28** Anomaly score of Change Finder setting the smoothing parameter to 1000. The red line represents the timestamp where the anomaly occurred (color figure online)

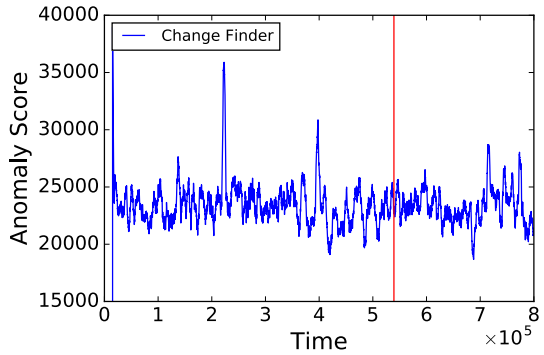


**Fig. 29** Anomaly score of Change Finder setting the smoothing parameter to be 10,000. The red line represents the timestamp where the anomaly occurred (color figure online)

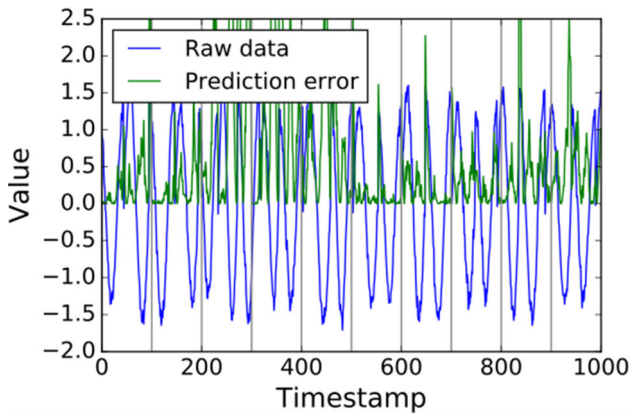


**Fig. 30** Sincos-data and prediction errors in a linear model. Gray lines represent the window $w$

until the corresponding measured values are acquired according to the particular problem setting for anomaly detection, we make use of measured values to select the most suited prediction model. Our approach provides multiple predicted value candidates in advance and selects that closest to the measured value as the predicted value. By using the measured values, we made prediction errors small and clearly indicated differences in anomaly scores for normal and abnormal states.

In experiments using artificial and real datasets, the proposed method performed well and was robust against thresholds. The proposed method can be flexibly combined with many other predictive models for time-series data.



## References

1. Chauhan, S., Vig, L.: Anomaly detection in ECG time signals via deep long short-term memory networks. In: Proceedings of IEEE International Conference on Data Science and Advanced Analytics, pp. 1–7 (2015)
2. Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The ucr time series classification archive (2015). https://www.cs.ucr.edu/~eamonn/time_series_data/
3. Chuah, M.C., Fu, F.: ECG anomaly detection via time series analysis. In: Proceedings of International Workshops on Frontiers of High Performance Computing and Networking, pp. 123–135 (2007)
4. Gaddam, S.R., Phoha, V.V., Balagani, K.S.: K-means+ ID3: a novel method for supervised anomaly detection by cascading k-means clustering and ID3 decision tree learning methods. IEEE Trans. Knowl. Data Eng. **19**(3), 345–354 (2007)
5. Hayton, P., Utete, S., King, D., King, S., Anuzis, P., Tarassenko, L.: Static and dynamic novelty detection methods for jet engine health monitoring. Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. **365**, 493–514 (2007)
6. Hill, D.J., Minsker, B.S.: Anomaly detection in streaming environmental sensor data: a data-driven modeling approach. Environ. Model. Softw. **25**(9), 1014–1022 (2010)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
8. Kingma, D., Ba, J.: Adam: a method for stochastic optimization (2014). arXiv preprint arXiv:1412.6980

9. Kingma, D., Welling, M.: Auto-encoding variational Bayes. In: Proceedings of the 2nd International Conference on Learning Representations(ICLR) (2014)
10. Ma, J., Sun, L., Wang, H., Zhang, Y., Aickelin, U.: Supervised anomaly detection in uncertain pseudoperiodic data streams. ACM Trans. Internet Technol. **16**(1), 1–20 (2016)
11. Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., Shrof, G.: LSTM-based encoder-decoder for multi-sensor anomaly detection. In: Presented at ICML 2016 Anomaly Detection Workshop (2016). arXiv preprint arXiv:1607.00148
12. Malhotra, P., Vig, L., Shroff, G., Agarwal, P.: Long short term memory networks for anomaly detection in time series. Proc. Eur. Symp. Artif. Neural Netw. **23**, 89–94 (2015)
13. Maruyama, Y., Yamanishi, K.: Dynamic model selection with its applications to computer security. In: Information Theory Workshop, pp. 82–87 (2004). https://doi.org/10.1109/ITW.2004.1405279
14. Miyaguchi, K., Yamanishi, K.: Online detection of continuous changes in stochastic processes. Int. J. Data Sci. Anal. **3**(3), 213–229 (2017)
15. Rayana, S., Akoglu, L.: Less is more: building selective anomaly ensembles. ACM Trans. Knowl. Discov. Data (TKDD) **10**(4), 42 (2016)
16. Rifai, S., Vincent, P., Muller, X., Glorot, X., Bengio, Y.: Contractive auto-encoders: explicit invariance during feature extraction. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 833–840 (2011)
17. Sak, H., Senior, A.W., Beaufays, F.: Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: Proceedings of the Annual Conference of International Speech Communication Association, pp. 338–342 (2014)
18. Sakurada, M., Yairia, T.: Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, p. 4. ACM (2014)
19. Saligrama, F.N.V.: Dynamic model selection for prediction under a budget (2017). arXiv preprint arXiv:1704.07505
20. Sivaraks, H., Ratanamahatana, C.A.: Robust and accurate anomaly detection in ECG artifacts using time series motif discovery. Comput. Math. Methods Med. **2015**, 1–20 (2015)
21. Söelch, M., Bayer, J., Ludersdorfer, M., van der Smagt, P.: Variational inference for on-line anomaly detection in high-dimensional time series (2016). arXiv preprint arXiv:1602.07109
22. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
23. Takeuchi, J., Yamanishi, K.: a unifying framework for detecting outliers and change points from non-stationary time series data. IEEE Trans. Knowl. Data Eng. **18**(4), 482–492 (2006)