LSTM Learning with Bayesian and Gaussian Processing for Anomaly Detection in Industrial IoT

Di Wu *Member, IEEE*, Zhongkai Jiang, Xiaofeng Xie, Xuetao Wei *Member, IEEE*, Weiren Yu *Member, IEEE*, and Renfa Li *Senior Member, IEEE*

***Abstract*—The data generated by millions of sensors in Industrial Internet of Things (IIoT) is extremely dynamic, heterogeneous, and large scale. It poses great challenges on the real-time analysis and decision making for anomaly detection in IIoT. In this paper, we propose a LSTM-Gauss-NBayes method, which is a synergy of the long short-term memory neural network (LSTM-NN) and the Gaussian Bayes model for outlier detection in IIoT. In a nutshell, the LSTM-NN builds model on normal time series. It detects outliers by utilising the predictive error for the Gaussian Naive Bayes model. Our method exploits advantages of both LSTM and Gaussian Naive Bayes models, which not only has strong prediction capability of LSTM for future time point data, but also achieves an excellent classification performance of Gaussian Naive Bayes model through the predictive error. We evaluate our approaches on 3 real-life datasets that involve both long-term and short-term time-dependency. Empirical studies demonstrate that our proposed techniques outperform the best-known competitors, which is a preferable choice for detecting anomalies.**

***Index Terms*—Industrial Internet of Things (IoT), anomaly detection, deep learning.**

I. Introduction

Industrial Internet of Things (IIoT) has been popularized and developed over the past years [1], such as food processing industry, smart cities and urban informatics. The data transmission and processing plays a key role in IIoT applications as large-scale data and information are produced by massive sensors in IIoT [2], [3]. Highly useful and valuable information could be derived to make intelligent automation and decisions for these IIoT applications. However, data anomalies inevitably appear due to the scale, computation and storage complexities [4], which could pose great risks on IIoT applications, especially safety-critical applications [5]. Thus, a surge of efficient techniques for detecting outliers are desired to ensure the quality of collected data.

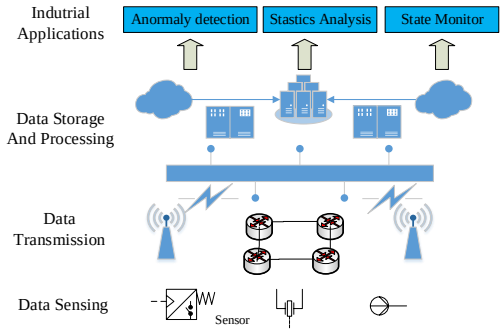


Fig. 1. The architecture of Industrial Internet of Things (IIoT).

In IIoT, time series data generated by massive sensors are becoming the most widespread [6]. As illustrated by the IIoT scenario in Figure 1, the data collected by different types of industrial sensors are transmitted to nearby edge nodes or remote data servers through heterogeneous communication and networking technologies. These IIoT data are stored and processed on demand or constantly for various industrial applications, such as anomaly detection, statistic analysis, and state monitoring. In comparison with the traditional Internet, the sampling frequency, measurement location and transmission rate during data sensing and transmission have a huge influence on the quality of raw data. These intrinsic characteristics of IIoT also make its data dynamic, large scale, time-dependent and high-dimensional, presenting strong correlation for learning meaningful information. Therefore, efficient data processing and learning techniques can not only present an intelligent analysis to support various industrial applications, but also diagnose the state of IIoT during data sensing and transmission through mining data features and their correlation.

Time-dependency is a very important feature for the IIoT data, which has great influence on data prediction and analysis [6]. For example, the data at current time point are likely to be related to previous time point or a time point in the long past. This is called short-term and long-term time dependency [7]. Such dependencies indicate that the abnormal occurrence of current time point may also be related to the data at previous time point, so we can make good use of this feature for anomaly detection. The anomalies we want to detect and tackle in IIoT fall into two categories in general. One is hardware anomalies, especially generated from different types of sensors, with potential problems such

Manuscript received December 01, 2017; revised May 04, 2018, January 30, 2019 and October 03, 2019; accepted November 05, 2019. Date of publication XXXX XX, 2019; date of current version XXXX XX, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61602168, Grant 61972145 and Grant 61932010, and in part by the HuXiang Youth Talent Program under Grant 2018RS3040. Paper was with no. TII-17-2875 and now no. TII-19-4519. (Corresponding author: Di Wu).

D. Wu is with the ExponentiAI Innovation Lab, and also with the Key Laboratory for Embedded and Network Computing of Hunan Province, Hunan University, Changsha 410082, China (e-mail: dwu@hnu.edu.cn).

Z. Jiang, X. Xie and R. Li are with the Department of Computer Engineering, and also with the Key Laboratory for Embedded and Network Computing of Hunan Province, Hunan University, Changsha 410082, China (e-mail: {peter_bon, xietls, lirenfa}@hnu.edu.cn).

X. Wei is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: xuetao.wei@gmail.com).

W. Yu is with the Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK (e-mail: ywr0708@hotmail.com).

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

Digital Object Identifier XX.XXXX/TII.XXXX.XXXXXXXX

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2952917, IEEE Transactions on Industrial Informatics

as environmental interference, device malfunction and reading errors. Another is software anomalies, impacted by program exception, transmission error and malicious attack, resulting in abnormal or manipulated data collection.

Traditional time series techniques detect variations of the distribution by employing EWMA (exponentially weighted moving average) and CUSUM (cumulative sum) within a given time interval [8]. The size of the time interval often requires to be determined in advance. One typical model for the time series approach is the autoregressive moving average model (ARIMA) [9], which transforms time series from non-stationary into stationary and predicts future values from past and present values of time series. However, it provides low accuracy and is limited to the short-term prediction only. There are also some anomaly detection algorithms based on the distance, such as [10]. Although these algorithms have low complexity, they are dependent on the threshold so that the accuracy of anomaly detection is not high and stable. A simple feed-forward neural network was proposed in time series data processing [11]. It predefined sliding windows to build features in order to make use of the relation between time series. However, it still has too much dependency on the parameters. Moreover, there exist several sequence models dealing with sequential data, *e.g.*, Kalman filtering [12], conditional random field model [13], Markov process [14], which are lacking the ability to learn long-term dependency. Though recurrent neural network (RNN) [15] can address the issue of long postponed tasks with no need to define time steps beforehand, it is easy for gradient explosion or vanishing gradient happen on time series tasks that take longer [16]. This makes it difficult to learn the time series of long-term dependency. Therefore, here we want to address the problem of long-term time dependency, which could cause instability and low accuracy in IIoT data anomaly detection.

In this article, we devise a LSTM-Gauss-NBayes method for outlier detection in IIoT. The LSTM-NN [17] is a variation of the RNN that can address the issue of gradient vanishing or explosion in an efficient manner by introducing a collection of memory units. First, our method uses the LSTM-NN model to predict the tendency of future time steps. Then, the stacked LSTM model is employed to learn normal time series. The aim to optimize the stacked LSTM prediction model is computing only the losses in the last sequence step. Meantime, with the aim to enhance the model extension capacity, the dropout [18] method is applied to the training phase of the model. This will enhance further the efficiency of model overfitting. We introduce the predicted error of future time steps into the Naive Bayes model [19] of normal distribution to find the outlier behavior, as illustrated in Figure 2. In general, our contributions are summarized as follows:

- The time-dependency is closely related to the outlier detection of IIoT data. The reason is that the occurrence of current anomalies is not only related to the current state, but also related to a certain time point in the past. Therefore, we first propose a stacked LSTM model to use its strong learning capability to deal with time series data with long-term time-dependency, short-term time-dependency and even weak time-dependency.
- An LSTM-Gauss-NBayes method is proposed for the anomaly detection in IIoT. We exploit advantages of LSTM's good prediction performance, and take advantage of predicted error to build Gaussian Naive Bayes model, which is well integrated into the excellent classification ability of the Bayesian model. Instead of simply combining Gaussian and Bayesian processing, the two models adopted here are connected by the predictive error generated from our LSTM prediction model specifically following the time series characteristics of IIoT data. Therefore, the integrated method can fully exploited the benefits of predicting residuals.
- A generic anomaly detection framework has been designed for learning and processing IIoT time series data. The framework is based on the LSTM-Gauss-NBayes method and could adapt to different types of IIoT data. We test our framework with comprehensive experiments in 3 real-world data sets. The results demonstrate that our approach outperforms the Stacked Bi-LSTM model, LSTM-NN model and MLP model, with on average a precision of 0.955 and a recall of 0.956.

The remainder of this article is organized as follows. Section II introduces our LSTM framework. Section III describes our outlier detection method. Section IV provides experimental results. Section VI concludes our article.

II. LSTM Learning Framework

A. Preliminary of LSTM Application

The LSTM-NN is a variation of the RNN. It can efficiently address the issue of gradient vanished or explosion by exploiting a collection of memory units. It enables the network to learn the appropriate time for 1) updating the memory unit with up-to-date information, and 2) forgeting the historical data from memory unit. At time t, the memory unit c_t contains the whole historical record till the present time, which relies on three "gates": the input gate i_t, the forget gate f_t, and the output gate o_t. The entry values of these gates are between 0 and 1. The LSTM network structure is quite suitable for datasets that include time dimensions (*e.g.,* medical sensored data, activity logs from web server, transactions in finance, or phone call records); only the present state and several past states are needed for network training. Since LSTM model can keep track of relationships and dependencies among many time-steps, it is widely adopted in a number of tasks for sequence learning. Due to a variety of LSTM applications in time-series data management and estimation [20], we propose a new framework which adopts the benefits of LSTM neural network and extends its structure specifically for the anomaly detection in industrial IoT.

B. Overview of Our Framework

In the proposed framework as show in Figure 2, we first process the initial data, with steps including data cleaning, data down-sample, and data normalization. Then, we divide the pre-processed data into training sets, validation sets and test sets, where the training sets and validation sets contain only the normal data. Later, these data sets are respectively

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2952917, IEEE Transactions on Industrial Informatics

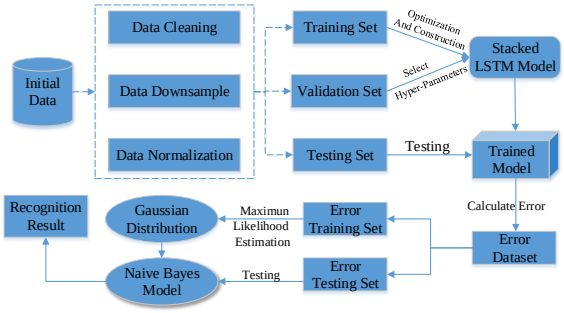
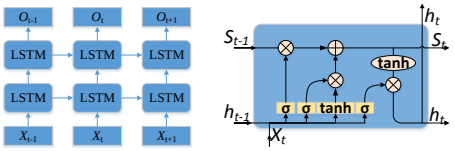


Fig. 2. Overview of our anomaly detection framework for IIoT time series data: LSTM Learning with Bayesian and Gaussian Processing.



(a) Stacked LSTM. (b) LSTM Internal Structure.

Fig. 3. (a) depicts the staked LSTM model that is unrolled (or unfolded) into an entire inter-linked network, where the LSTM cells in the hidden layer are inter-linked by recurrent connections. Through the feed-forward linkage, every cell of the lower LSTM hidden layer in the stacked LSTM layer is linked to every cell in the LSTM hidden layer above it. Besides, (b) illustrates the LSTM layer's inner structure, in which σ and tanh denote the function of activation. X_t represents the model input. h_t is the output of LSTM cell in the t-th time step and h_{t-1} is derived in the past sequence step. S_t represents the value of LSTM memory cell in the t-th time step.

used to optimize and construct the stacked LSTM model, select hyper-parameters, and obtain error data sets. Furthermore, the error data sets are split into two sets: error training and error test. We use the error training set to make the maximum likelihood estimation to obtain the parameters of the Gauss distribution. These parameters are fed to the Naive Bayes model to build a Gaussian Naive Bayes model. After that, classification results can be obtained when we import error test sets into this Gaussian Naive Bayes model.

We have used the proposed LSTM learning with Gaussian and Bayesian processing as a generic framework for anomaly detection in industrial IoT. The details of our method will be presented in following sections.

III. The LSTM-Gauss-NBayes Method

Using a LSTM-NN. In order to cope with the long-term or short-term time dependency in time series data from IIoT applications, we advocate to use a LSTM-NN structure. In this structure, the input layer is associated with a time series, and the amount of each hidden layer's LSTM cells is associated with the time step of time series. 2 hidden layers are utilized to form a stacked LSTM network, as depicted in Figure 3 (a), for outlier detection. The stacked LSTM network can enhance a model's performance on learning more complex features in comparison with single LSTM network. For the output layer, an inter-connected layer is built on the top LSTM layer, which is used to take the data from different time points into account, and evaluate their impact on the data at the next time point. These impact will be modeled and integrated as a predicted value, which is the output value at the next time point. For the LSTM cell, it controls the input, storage and output of data by introducing a set of gate mechanisms. As shown in Fig. 3(b), the LSTM gate units receive the output of the LSTM internal unit of the past time step and the input of present time step sample. However, if the previous layer of the LSTM cell layer is not the input layer, its various gate units accept both the output of its previous layer's LSTM internal unit at present time step and the output of this LSTM internal unit at past time step. Specifically, the three gates (input, forget, and output) update their internal values as follows:

The values of these internal structures in the Fig.3 (a) and (b) can be calculated with following steps in Eq. 1, where weight matrices W and bias vectors b are utilized to build connections between the input layer, output layer and memory block. $s_t^{(l)}$ stands for the memory cell's state of the t-th time steps of the l-th layer. $h_t^{(l)}$ is the output of memory cell at the t-th time step of the l-th layer. σ represents an entry-wise application of the sigmoid (logistic) function, ϕ denotes an element-wise application of the $\tanh$ function, and $\odot$ is the entry-wise Schur product. The three gates (input, output, and forget) are respectively represented by i, o, f, and g is the input node with a $\tanh$ activation.

$$\begin{aligned} g_t^{(l)} &= \phi(W_{gx}^{(l)} h_t^{(l-1)} + W_{gh}^{(l)} h_{t-1}^{(l)} + b_g^{(l)}) \\ i_t^{(l)} &= \sigma(W_{ix}^{(l)} h_t^{(l-1)} + W_{ih}^{(l)} h_{t-1}^{(l)} + b_i^{(l)}) \\ f_t^{(l)} &= \sigma(W_{fx}^{(l)} h_t^{(l-1)} + W_{fh}^{(l)} h_{t-1}^{(l)} + b_f^{(l)}) \\ o_t^{(l)} &= \sigma(W_{ox}^{(l)} h_t^{(l-1)} + W_{oh}^{(l)} h_{t-1}^{(l)} + b_o^{(l)}) \\ s_t^{(l)} &= g_t^{(l)} \odot i_t^{(l)} + s_{t-1}^{(l)} \odot f_t^{(l)} \\ h_t^{(l)} &= \phi(s_t^{(l)}) \odot o_t^{(l)} \end{aligned} \tag{1}$$

Data preprocessing. To process time series data, the down sampling technique is first employed to get the characteristic subsequence from the original time series, which enables a reduction in the dimensionality of the original time series and makes patterns easy-to-learn. Meanwhile, with the aim to accelerate the convergence of the model, a min-max normalization approach is applied for time series data normalization, which can be viewed as a linear map of the original time series data onto $[0, 1]$.

After completing above feature engineering for the IoT data, the next step is to construct a data set to contain the relationship between the time series and the time dependency of the IoT time series data. When predicting the data at the next time point, the model needs to use the data from the previous time point as reference, and the length of this time period is defined as the time step. The input structure of the LSTM model is generally a three-dimensional array as $[samples, timesteps, features]$. We propose a sliding regression method to construct the input for our LSTM model. That is, for the original time series $S = \{x_1, x_2, x_3, \cdots, x_i, \cdots, x_L\}$, where L is the length of the time series, and x_i represents the data of the i-th time point, we consider it as a d-dimensional vector and d represents the number of features. Given a sliding window T, which is set as our time step, the sliding regression

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2952917, IEEE Transactions on Industrial Informatics

is performed on the original time series to construct sequential samples as $\{(x_1, x_2, \cdots, x_i, \cdots, x_T), y^{(1)}\}$, $\{(x_2, x_3, \cdots, x_i, \cdots, x_{T+1}), y^{(2)}\}$, $\cdots\cdots$, $\{(x_{T+L}, x_{T+L-1}, \cdots, x_{T+L-i-1}, \cdots, x_{L-1}), y^{(T+L)}\}$, where $(x_1, x_2, \cdots, x_i, ...x_T)$ can be abbreviated as $x(1)$, namely the input of the first sample, and $y^{(1)}$ represents the label of the first sample, namely the target of the output. By doing so, we could use the data of the previous T time points to predict the data of the next time point.

Dividing the dataset, training models and obtaining the error dataset. The dataset is partitioned into a training set containing normal data, a validation set containing normal data, a test set containing normal data, and a test set containing abnormal data. Meantime, since abnormal samples are relatively small in the real IIoT time series, the stacked LSTM prediction model is only allowed for training by utilising the normal data training set, whose hyper-parameters are specified by the validation set. In addition, the test set containing normal data and the test set containing abnormal data are respectively put to the trained model. The prediction outcomes of both normal and abnormal data are derived, respectively. Then, the absolute gap between the real and predicted data can be computed, and the error dataset (that contains the error of normal and abnormal data) can be constructed.

Taking advantage of the error. Next, the error at every time point is taken from test sets containing both normal data and abnormal data as the numerical attribute of the prediction error data set. The error dataset is split to two sets (error training and error testing), in which the label value $y \in \{0,\ 1\}$ and 1 indicates abnormal. A Bernoulli model is created for the label value y as: $p\,(y) = \varphi^y(1-\varphi)^{1-y}$, where φ stands for the probability of the labels $y = 1$ in the error training set. At the same time, it is tacitly assumed that every numerical attribute in the error dataset follows the normal distribution. Actually, this assumption is often highly efficient and may yield stable outcomes. The associated normal PDF (probability density function) is built for the conditional probability of every attribute, as follows in Eq. 2:

$$\begin{aligned} p(x_j^{(i)}|y^{(i)}=1) &= \frac{1}{\sqrt{2\pi}\sigma_{ij}^{(1)}}\exp\big(-\frac{(x_j^{(i)}-\mu_{ij}^{(1)})^2}{2(\sigma_i^{(1)})^2}\big) \\ p(x_j^{(i)}|y^{(i)}=0) &= \frac{1}{\sqrt{2\pi}\sigma_{ij}^{(0)}}\exp\big(-\frac{(x_j^{(i)}-\mu_{ij}^{(0)})^2}{2(\sigma_{ij}^{(0)})^2}\big) \end{aligned} \tag{2}$$

where $x_j^{(i)}$ is the j-th attribute of the i-th sample in the error training set. $y^{(i)}$ stands for the label value of the i-th sample in the error training set. $\mu_{ij}^{(1)}$ and $\sigma_{ij}^{(1)}$ are respectively the means and variance of the j-th attribute of the i-th sample when its label value is 1. Otherwise, it is $\mu_{ij}^{(0)}$ and $\sigma_{ij}^{(0)}$ respectively when the sample's label value is 0.

Buliding Guassian Naive Bayes model. These parameters of the normal PDF can be obtained from the maximum likelihood estimate in the error training set as: $\mathrm{L}(\varphi, \mu_{ij}^{(1)}, \mu_{ij}^{(0)}, \sigma_{ij}^{(1)}, \sigma_{ij}^{(0)}) = \log\prod_{i=1}^{m}(\prod_{j=1}^{n} p(x_j^{(i)}|y^{(i)}; \varphi, \mu_{ij}^{(1)}, \mu_{ij}^{(0)}, \sigma_{ij}^{(1)}, \sigma_{ij}^{(0)}))p(y^{(i)}; \varphi)$. Then, the maximum likelihood estimation of these parameters is derived as follows:

$$\begin{aligned} \varphi &= \tfrac{1}{m}\textstyle\sum_{i=1}^{m} I\{y^{(i)}=1\} \\ \mu_{ij}^{(1)} &= \frac{\sum_{i=1}^{m} I\{y^{(i)}=1\}x_j^{(i)}}{\sum_{i=1}^{m} I\{y^{(i)}=1\}} \\ \mu_{ij}^{(0)} &= \frac{\sum_{i=1}^{m} I\{y^{(i)}=0\}x_j^{(i)}}{\sum_{i=1}^{m} I\{y^{(i)}=0\}} \\ \sigma_{ij}^{(1)} &= \frac{\sum_{i=1}^{m} I\{y^{(i)}=1\}(x_j^{(i)}-\mu_{ij}^{(1)})^2}{\sum_{i=1}^{m} I\{y^{(i)}=1\}} \\ \sigma_{ij}^{(0)} &= \frac{\sum_{i=1}^{m} I\{y^{(i)}=0\}(x_j^{(i)}-\mu_{ij}^{(0)})^2}{\sum_{i=1}^{m} I\{y^{(i)}=0\}} \end{aligned} \tag{3}$$

where $I\{\cdot\}$ is an indicator function. When the conditions inside the brackets are true, the value is 1, otherwise 0. $x_j^{(i)}$ represents the j-th attribute of the i-th sample in the error training set. $y^{(i)}$ stands for the label value of the i-th sample in the error training set.

Using the Naive Bayes model to calculate results. The proposed LSTM prediction model introduced above has considered two relations: 1) the impact of historical data on the current data, and 2) the impact of the current data on the later data. Since the prediction error is produced based on the two relations, we could assume that the prediction error generated at each time point is conditionally independent, and apply the Naive Bayes hypothesis here accordingly to calculate posterior probabilities by multiplying these conditional probabilities. In addition, making this assumption can facilitate the calculation of conditional probability and reduce the complexity of our model for the real-world application in IIoT scenario.

Specifically, for all the samples in the prediction error data set, we use these parameters presented in Eq. 3 to compute the conditional probability that an attribute of one sample which occurs in the presence of a certain class. Furthermore, according to the independent assumption of Naive Bayes, these conditional probabilities of different attributes of one sample can be multiplied together, which produces the conditional probability that a sample occurs in the presence of a certain class as: $p(x^{(i)}|y^{(i)} = 1) = \prod_{j=1}^{n} p(x_j^{(i)}|y^{(i)} = 1)$ and $p(x^{(i)}|y^{(i)} = 0) = \prod_{j=1}^{n} p(x_j^{(i)}|y^{(i)} = 0)$.

Thereafter, based on the Bayes equation, the abnormal probability of the category of every sample in the error test set can be computed as $p\,(y = 1|x) = \dfrac{p(x|y=1)p\,(y=1)}{p\,(x|y=1)\,p\,(y=1) + p(x|y=0)p\,(y=0)}$.

Incremental training. We use the incremental training mode [21] in our model. We first train the LSTM-Gauss-NBayes model offline. In our training method, the update for a layer of memory units in LSTM can be found in [22]. We select the adaptive gradient algorithm (Adagrad) [23] as the optimization method to minimize the mean squared error (MSE) loss, and the maximum number of iterations is set to be 10000 by heuristic rules [24]. Then, the backpropagation through time (BPTT) algorithm [25] is applied to update the model parameters. With the aim to avoid over-fitting, regularization techniques [26] and dropout are employed to enable a reduction in the complexity of the models. Then, we can deploy the trained model online and process the data online, using the incoming new data to train and update the model, which speeds up the learning efficiency.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2952917, IEEE Transactions on Industrial Informatics

Anomaly detection using Gaussian Naive Bayes. Algorithm 1 describes the process of applying Gaussian Naive Bayes to detect the error dataset generated by our stack-LSTM prediction model. In the framework of the LSTM, given m samples where each sample is a series of observations $(x_1, \ldots, x_t, \ldots, x_T)$, a prediction model is learnt to generate hypotheses $\hat{y}$ of the true values y. Here, t represents sequence steps, and T stands for the length of the sequence. We use the least squares loss function $loss(\hat{y}, y) = \frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$ as the cost function for this model, where $y^{(i)}$ stands for true value of the i-th sample, and $\hat{y}^{(i)}$ stands for predicted value of the i-th sample.

Algorithm 1 Sensor data anomaly detection algorithm using the Gaussian Naive Bayes

Input: Error Training data sets $E_{tra} \in \mathbb{R}^{m*n}$
Error Testing data sets E_{tes}
Output: Abnormal samples
for sample $(x^{(i)},\ y^{(i)})$ in E_{tra} **do**
  Build a Bernoulli model for the label value $y^{(i)}$ as: $p(y^{(i)}) = \varphi^{y^{(i)}}(1-\varphi)^{1-y^{(i)}}$
  for feature $x_j^{(i)}$ in $x^{(i)}$ **do**
    Establish the corresponding Gaussian probability density function for the conditional probability of each attribute $x_j^{(i)}$ as Eq. (2):
  end for
end for
for $j = 1$ to n **do**
  for $i = 1$ to m **do**
    Compute maximum likelihood estimation function for each feature $x_j^{(i)}$. Perform log transformation as: $\mathrm{L}(\varphi, \mu_{ij}^{(1)}, \mu_{ij}^{(0)}, \sigma_{ij}^{(1)}, \sigma_{ij}^{(0)}) = \log\prod_{i=1}^{m}(\prod_{j=1}^{n} p(x_j^{(i)}|y^{(i)}; \varphi, \mu_{ij}^{(1)}, \mu_{ij}^{(0)}, \sigma_{ij}^{(1)}, \sigma_{ij}^{(0)}))p(y^{(i)}; \varphi)$
    Derive the derivative for the likelihood function, and then get the solution of each unknown parameter in Eq. (3)
    Substitute the parameters to $p(x^{(i)}|y^{(i)} = 1)$ and $p(x^{(i)}|y^{(i)} = 0)$ with the independence principle of Naive Bayes
    Get the conditional probability of each sample
  end for
end for
for each test sample in E_{tes} **do**
  Calculate the probability of belonging to anomaly by the Bayesian formula $p(y = 1|x)$
  if $p(y = 1|x) > p(y = 0|x)$ **then**
    Mark x is an abnormal sample
  end if
end for

IV. Evaluation

We adopt Tensorflow, an open source platform for machine learning, to implement our proposed approaches, and utilise NVIDIA GTX1070 to speed up the training process for the model. We compare our models with these competitors:

- **Stacked Bidirectional LSTM (Stacked Bi-LSTM) [20].** We adopt an input layer, two forward LSTM units and two backward LSTM units consisting of a multiple hidden units as multiple hidden layers. At the output layer, the sigmoid function is used as a two classification layer.
- **LSTM Neural Network (LSTM NN) [17].** We construct it by using an input layer, two concealed layers with LSTM memory blocks and a classification layer. Its cost function is a cross entropy loss function.
- **MultiLayer Perceptron (MLP) [16].** We use an input layer, multiple concealed layers, and a classification layer to built it, where a multi-layer perceptron model is built with three concealed layers. The cross entropy loss function is also used as the cost function.

Moreover, based on experience and the amount of data in our experiment, each model is trained on 80% of the data and tested on 10% of the data. The rest of 10% is employed as a validation set. Meanwhile, the training of each model can be terminated early by setting the threshold of prediction error, which is a hyper-parameter debugged and selected according to the validation set. We utilise the validation set for choosing the hyper-parameters of these models. For example, the number of neurons in the hidden layers of the three models is selected by the validation set.

A. Dataset Description

To evaluate the performance on anomaly detection, we use our own real-life time time-series datasets: Power, Loop Sensor, and Land Sensor. Due to various (*e.g.,* cyclical, irregular) patterns of time-series data, Table I shows the autocorrelation coefficients (ACF) [9] of every dataset at different delay cases. We use the delay with value k to calculate the coefficient of current time step and past k time step, for the evaluation of these data sets on short-term and long-term time dependency. So the ACF can quantitatively describe the relationship between previous and present events.

TABLE I
DATASET DESCRIPTION.

Dataset	Autocorrelation coefficient(ACF)			Time dependency
	Delay k=1	Delay k=5	Delay k=10	
Power dataset	0.79	-0.78	0.56	long
Loop sensor dataset	0.71	0.4	0.05	short
Land sensor dataset	0.32	0.13	0.08	very weak

1) Power: This dataset consists of a user's power data over one year. Each 15 minutes is a time-stamp. We down-sample the raw data per week. Input samples are formed by the resulting data. By normal conditions, power consumption remains fairly high on weekdays (from Monday to Friday) and fairly low on Saturdays and Sundays. It can be discerned from Fig. 4 (a), the tendency of electricity consumption exhibits five peaks for the first five days, followed by some depressions for the following 2 days. Note that, since a user's power data often contains noise, the peak does not simultaneously appear for different days. If there were depressions on weekdays, or wave crest on weekends, it could be viewed as outlier because it violates the normal tendency of a user's electricity consumption, as depicted in Fig 4 (b). This could occur due to erroneous readings from the electric meter or manipulated consumption data from the user.

2) Loop Sensor: The dataset records the number of vehicles passing through near the stadium, which was collected by loop sensor once a game is held in the stadium. It can be observed from the Table I that, as the delay retains 5, the ACF remains smaller than 0.5. This implies that Loop Sensor is short-term time-dependent time series data. To serve the purpose for high-quality data analysis, we chose the data with just one hour before the game and two hours after the game. The results are shown in Figure 4 (c). We see that there is a tiny peak on Loop Sensor in the first half and the second half, and the wavelet valley is in the middle of it. After the wavelet valley, vehicle data grow dramatically. This agrees well with

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2952917, IEEE Transactions on Industrial Informatics

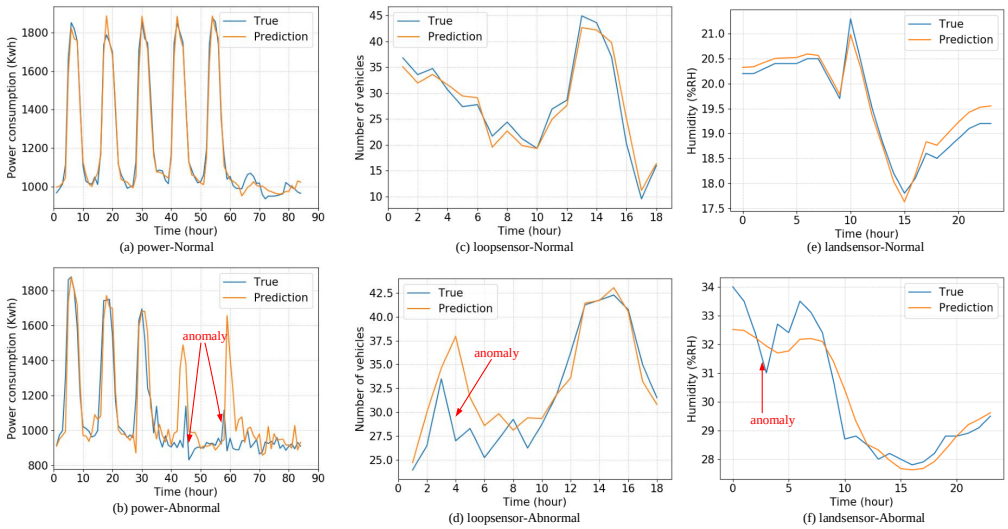


Fig. 4. LSTM Learning results of our model under three data sets. The orange lines represent the predicted results, and the blue lines represent the real results. In addition, the figures on the upper side show the learning results of normal samples, and the figures on the bottom side show the learning result of abnormal samples. The red arrow indicates potential anomalies.

a rapid increase in traffic after the game ends. This behavior is regarded as normal for this dataset. The anomaly outcomes illustrated in Fig 4 (d) as well.

3) Land Sensor: The dataset consists of land humidity data. Each 12 minutes makes a time-stamp. Fig. 4 (e) illustrates its normal results. Unlike previous two data sets, this is an irregular time series with rather weak time-dependency, as depicted in Table I, which is due to the fact that the land humidity is usually affected by some external factors, such as irrigation behaviors, natural rainfall, and weather changes. Though the values in Land Sensor may rise and fall at random in some range along with the time increasing, we could still conclude a general rule that the degree of land humidity varies slowly in the early hours during the morning, and decreases rapidly when the temperature increases during the daytime. Other normally external factors as mentioned above may impact land humidity temporarily, but still cannot greatly change the general trend. Therefore, our task for anomaly detection in land humility data set is to trace and detect irregularly abnormal changes violating the general rule through our model validation. To achieve this goal, likewise, the original time series is down-sampled by 5, with one-day humidity data as one sample, to train our model. We can discern in Figure 4 (f) that an outlier could have happened in the first part of Land Sensor, and its land humidity actually decreases significantly in the early hours of the morning. This may be an abnormal reading due to hardware malfunction, transmission error or malicious attack.

B. Performance Evaluation of Anomaly Detection

1) Evaluation Metrics: We use six metrics, that is Accuracy, Precision, Recall, F_1, ROC and AUC, to comprehensively evaluate the peroformance of our model. These metrics have been widely used [27], and the details of the metrics are listed following Eq. 4, where TP, TN, FP and FN are four classification results, respectively, denoting true positive (TP), true negative (TN), false positive (FP), and false negative (FN).

$$\begin{aligned} Accuracy &= \frac{TP+TN}{TP+FN+FP+TN} & Recall &= \frac{TP}{TP+FN} \\ Precision &= \frac{TP}{TP+FP} & F_1 &= \frac{2 \cdot Precision \cdot Recall}{Precision+Recall} \end{aligned} \tag{4}$$

Accuracy is employed to judge a model, aiming at data classification. As our objective is to find if the sample is anomaly, we also use two metrics (precision and recall) for our model evaluation. The Precision rate is utilised to determine if the classifier is able to accurately detect the abnormality. This is to say, its main focus is on detection of anomaly samples. Furthermore, the recall rate aims to determine if the classifier can detect all anomaly samples. F_β score is an integration of the precision and recall metrics; if $\beta < 1$, then it indicates more importance of the recall rate. In contrast, the precision rate has a larger influence on the quality assessment of the model. The F_1 metric provides a balanced overview of the algorithm performance.

The receiver operating characteristic (ROC) curve is a graph consisting of a false positive rate (FPR) on the x axis and true positive rate (TPR) on the y axis, where $TPR = \frac{TP}{TP+FN}$ and $FPR = \frac{FP}{FP+TN}$. We can obtain different TPR and FPR pairs by adjusting the classifier's classification threshold. The ROC data points are constituted by these data pairs and then form a ROC curve. The area under curve (AUC) is the area under the ROC curve, which reflects the performance of the classification model. The closer the AUC value is to 1, the better the classification is.

2) Results and Analysis: We use 4 classification metrics (Accuracy, Precision, Recall, F_1 to evaluate the performance of each model over 3 datasets. The results are shown in Table II, in which the highest scores for each metric on the same dataset are underlined.

On Power, it can be noticed in Table I that, as the delay reaches 10, the ACF is larger than 0.5. Therefore, it can be inferred that there is a long-term dependency for the time on this dataset, and its present data is connected to the data in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2952917, IEEE Transactions on Industrial Informatics

front of it in an inextricable manner. Nonetheless, as there is a certain periodicity on Power, *i.e.,* the electricity value often becomes higher on weekdays and lower on weekends, the periodic pattern is a key feature for time series data. Therefore, it is easy to build a good time series model with this feature. From Table II, we can observe the model having the hidden layer of the LSTM cell, in general, outperforms the general hidden layer in the outcomes of each of metrics. At the same time, the model with the bidirectional LSTM unit will perform better than the general LSTM NN model, but worse than our LSTM model with enhanced Gaussian and Bayesian processing. It can be noticed that the efficacy of the MLP model is satisfactory as well with the corresponding recall rate arriving at up to 88.2%.

On Loop Sensor, it is quite relevant to the time order as well. Nevertheless, in contrast to Power, the features of Loop Sensor are obscured, which makes it difficult to differentiate its anomaly time series data from its normal counterpart. Hence, we can see in Table II that the experimental results of the other three models are unsatisfactory in this data set. Since the Stacked Bi-LSTM model and LSTM NN model only simply apply the existing LSTM structure without specific design for the features of IoT data as mentioned in Section I, they cannot well detect this kind of vague dataset. In particular, the precision rate of outlier detection for the MLP model retains merely 79%. However, the proposed LSTM-Gauss-NBayes model on Loop Sensor outperforms the three competitors.

On Land Sensor, the data does not usually depend on the order of time of occurrence. Thus, it poses challenges for future values prediction based on past values only. We can observe from Table II that the F_1 scores of the LSTM-NN and MLP models are ineffective for ourlier detection on Land Sensor. The reason is that the features of Land Sensor are vague with its value constantly fluctuated. The Stacked Bi-LSTM model works better than the LSTM NN model due to the its bidirectional structure, but still worse than the Stacked LSTM model. This proves that bidirectional is not always necessary to process time-series data. In addition, it can also be found in Table II that MLP is better at precision rate of anomaly detection than the LSTM-NN, which is perhaps due to the weak time-dependency feature shown by Land Sensor.

TABLE II
EXPERIMENTAL RESULTS

Dataset	Method	Accuracy	Precision	Recall	F_1
Power dataset	LSTM-Gauss-NBayes	**0.969**	**1**	**0.941**	**0.962**
	Stacked Bi-LSTM	0.924	0.892	0.930	0.911
	LSTM NN	0.905	0.846	0.931	0.886
	MLP	0.873	0.843	0.925	0.882
Loop sensor dataset	LSTM-Gauss-NBayes	**0.952**	**0.931**	**0.975**	**0.953**
	Stacked Bi-LSTM	0.897	0.882	0.924	0.903
	LSTM NN	0.870	0.867	0.897	0.881
	MLP	0.823	0.789	0.818	0.803
Land sensor dataset	LSTM-Gauss-NBayes	**0.970**	**0.933**	**0.952**	**0.942**
	Stacked Bi-LSTM	0.905	0.920	0.872	0.895
	LSTM NN	0.820	0.864	0.782	0.821
	MLP	0.824	0.893	0.750	0.814

Due to the page limit, we only show the ROC curve under the power data set for the analysis of the ROC curves of different models. As shown in Fig. 5, the area under the LSTM-Gauss-Nbayes curve is larger than the area under the curve of the other two methods, and this can also be found from the value of the AUC. This shows that our method is far better than the other two methods. According to the definition of ROC curve, we can see that ROC continuously reduces the threshold of classification, and then calculates the values of TPR (True Positive Rate) and FPR (False Positive Rate). TPR shows the proportion of positive instances identified by the classifier to all positive instances. FPR shows the proportion of negative instances of the classifier that are considered to be positive for all negative instances. Analyzing the ROC curves of LSTM-Gauss-Nbayes, we can see that the TPR value quickly reaches 0.9 during the process of continuously lowering the threshold value. This shows that our approach is more robust. While the other two methods perform well, the growth rate of the TPR value is still slightly worse than ours.

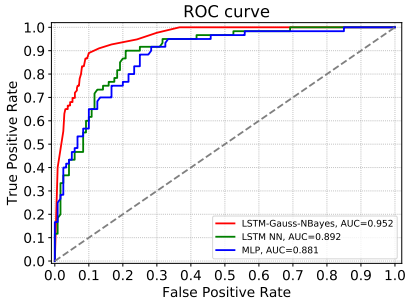


Fig. 5. ROC and AUC analysis for the three models in the power data set.

V. RELATED WORK

1) Applying Statistical Learning Method in Time Series Data: Traditional time series analysis uses mathematical statistical approaches for sequence analysis and future prediction. They follow basic principles by detecting continuous changes in the sequence and leveraging historical data to make a prediction about the tendencies development. Furthermore, the noise property of the time series is considered. Process effects can be influenced by uncertain factors, for which the weighted average approach is leveraged for statistical analysis for handling past data. An appealing model of the conventional approach for time series analytics is the auto-regressive moving average (ARIMA) model [9]. ARIMA transforms the time series from non-stationary to stationary via differential operations. Then the auto-correlation coefficient (ACF) and the Partial ACF (PACF) are proposed to deal with stationary time series. By analysing the auto-correlation graph and the partial auto-correlation graph, the optimal class and order can be derived, based on which the ARIMA model is built up.

Despite its simple and easy-to-use advantages, the ARIMA approach exhibits undesirable accuracy and it is not good for long-term prediction. Moreover, once the data is unstable, the performance of these the traditional statistical methods are very poor. Our LSTM prediction model not only accommodates the logical relationship between the time before and after, but also extracts the characteristics of the current time point. This avoids data instability.

2) Machine Learning Methods for Outlier Detection: Outlier detection aims to identify patterns in data that deviate from an a priori expected behavior [28]. Utilising graph theory

and spatiotemporal correlation analysis, [29] introduced a decentralised general outlier detection method. There have been several other approaches [30] using support vector machine (SVM) for outlier detection. Nevertheless, the methods based on SVM seem very sensitive to missing values. [10] considered the problem of distance-based outlier detection on uncertain datasets with normal distribution, and devise a cell-based method detect the outliers in a fast way. However, the algorithm performance is not resilient as different parameters selections will have a huge influence on the detection results. [31] introduced a new scheme to detect outliers in noisy data streams by employing a wavelet based soft-threshold filtering approach that can remove uncertainties in time series data streams, However, the method cannot always ensure high accuracy on a variety of real datasets. Recently, the work of [32] applied outlier detection in mobile computing via machine learning based clustering techniques.

These traditional machine learning methods only considered the characteristics of the current time point rather than the time dependence of the time series feature. Therefore, anomaly detection based on them for the Internet of Things time series data is not very effective.

3) Applying Neural Network in IoT: The neural network is an interconnected computing system and a back propagation neural network (BP-NN) [25] can learn, remember, store, extend, and extract features, tolerance faults and introspection. Complicated relationships can be extracted between input and output, even when the relationship itself is in flux. Recent years have witnessed grow interest for BP-NN to address the problem of identification and prediction [11], [18]. It has accomplished in many economic fields that the traditional economics approaches are unable to handle, *e.g.,* economic growth investigation, economic GNP forecasting, and stock prices prediction. Meanwhile, as an appealing prediction tool, BP-NN can guarantee high accuracy for nonlinear quantities estimation. Through the use of the time series relations between before and after, we can take historical observations as BP-NN's input, and future data as the BP-NN's output, which constructs a prediction model for time series data.

When an ordinary neural network is applied to IoT data, the relationship between the data at the time before and after can be manually constructed by a sliding window to process the IoT time series data [17], [21]. However, on one hand, this window value is not well defined. On the other hand, this method is still too simple and is only limited to short-term dependent time series data. Our method presents an LSTM prediction model with explicit internal LSTM unit structure, which constantly updates the internal state values while getting input data at each time point, thus ensuring the time-series data before and after the time point can keep a strong connection.

4) Application of Recurrent Neural Network (RNN) in IoT: Despite promising results achieved by BPNN to deal with time series data, in traditional NN, it is tacitly assumed the independence of all inputs and outputs, which seems not suitable for a number of scenarios. The RNN [15], [22] differs from the general feed-forward BPNN by memoising the past data and leveraging it to the computation of the present output, *i.e.,* the nodes between the concealed layers are not linked any more. The input of the concealed layer contains (i) the output of the input layer and (ii) the output of the concealed layer at the last time. In general, with the aim to tame the complexity of the model, we usually assumed that the current state is merely relevant to the past several states.

The simple RNN can build a dependency in theory among states of the time windows with long length. However, there is a high likelihood that the gradient explosion or gradient vanished will happen during the long-term time series processing task [16]. As a result, only short-term dependencies could be learned. Our model architecture contains the LSTM prediction model, so we can solve this problem of gradient explosion and gradient disappearance. In addition, while using the proposed LSTM model to produce the prediction error, we adopt the Gaussian distribution of the conditional probability of the error, and then rely on Naive Bayes' excellent classification performance to achieve the detection of abnormal data in IIoT.

VI. Conclusion

In this paper, we propose a LSTM-Gauss-NBayes method, which is a synergy of LSTM-NN and the Naive Bayes model with nomoral distribution for outlier detection in IIoT. We first use the training set to construct the stacked LSTM model. We next import the test set into the trained predictive model to build the error data set. Furthermore, the error training set is used to build Naive Bayes model of normal distribution for anomaly detection. Our extensive experiments show that our model achieves the promising results.

References

[1] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] Z. Qin, D. Wu, Z. Xiao, B. Fu, and Z. Qin, "Modeling and analysis of data aggregation from convergecast in mobile sensor networks for industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4457–4467, 2018.

[3] D. Wu, D. I. Arkhipov, M. Kim, C. L. Talcott, A. C. Regan, J. A. McCann, and N. Venkatasubramanian, "Addsen: Adaptive data processing and dissemination for drone swarms in urban sensing," *IEEE Transactions on Computers*, vol. 66, no. 2, pp. 183–198, 2017.

[4] D. S. Pham, S. Venkatesh, M. Lazarescu, and S. Budhaditya, "Anomaly detection in large-scale data stream networks," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 145–189, 2014.

[5] R. Mitchell and I. R. Chen, "Behavior-rule based intrusion detection systems for safety critical smart grid applications," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1254–1263, 2013.

[6] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, "Big iot data analytics: Architecture, opportunities, and open research challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017.

[7] J. G. D. Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International Journal of Forecasting*, vol. 22, no. 3, pp. 443 – 473, 2006.

[8] M. Basseville, I. V. Nikiforov *et al.*, *Detection of abrupt changes: theory and application*. Prentice Hall Englewood Cliffs, 1993, vol. 104.

[9] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

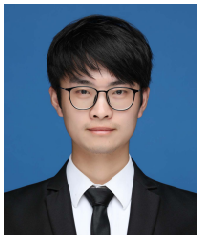
[10] S. A. Shaikh and H. Kitagawa, "Efficient distance-based outlier detection on uncertain datasets of gaussian distribution," *World Wide Web*, vol. 17, no. 4, pp. 511–538, 2014.

[11] B. Oancea and S. C. Ciucu, "Time series forecasting using neural networks," *CoRR*, vol. abs/1401.1333, 2014.

[12] S. S. Haykin, "Kalman filtering and neural networks," *Adaptive and Learning Systems for Signal Processing Communications and Control*, pp. 170 – 174, 2013.

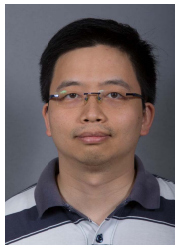
[13] J. Glass, M. Ghalwash, M. Vukicevic, and Z. Obradovic, "Extending the modelling capacity of gaussian conditional random fields while learning faster," in *AAAI*, 2016, pp. 1596–1602.
[14] M. S. Hwang, C. C. Yang, and S. F. Tzeng, "Fuzzy time series forecasting with a probabilistic smoothing hidden markov model," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 2, pp. 291–304, 2012.
[15] A. Graves *et al.*, *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, vol. 385.
[16] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *ICML*, 2013, pp. 1310–1318.
[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[18] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *Computer Science*, vol. 3, no. 4, pp. pgs. 212–223, 2012.
[19] F. V. Jensen, *An introduction to Bayesian networks*. UCL press London, 1996, vol. 210.
[20] M. Sun, G. Strbac, P. Djapic, and D. Pudjianto, "Preheating quantification for smart hybrid heat pumps considering uncertainty," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 8, pp. 4753–4763, 2019.
[21] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, "Panfis: A novel incremental learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 55–68, Jan 2014.
[22] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell, "Learning to diagnose with lstm recurrent neural networks," in *ICLR*, 2016.
[23] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.
[24] P. Joulani, A. Gyrgy, and C. Szepesvri, "Delay-tolerant online convex optimization: Unified analysis and adaptive-gradient algorithms," in *AAAI*, 2016.
[25] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, vol. 1, no. 4, pp. 339–356, 1988.
[26] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *NIPS*, 2016.
[27] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *SIAM International Conference on Data Mining*, 2003, pp. 25–36.
[28] V. P. Illiano and E. C. Lupu, "Detecting malicious data injections in wireless sensor networks: A survey," *ACM Computing Surveys (CSUR)*, vol. 48, no. 2, p. 24, 2015.
[29] P.-Y. Chen, S. Yang, and J. A. McCann, "Distributed real-time anomaly detection in networked industrial sensing systems," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3832–3842, 2015.
[30] H. Martins, L. Palma, A. Cardoso, and P. Gil, "A support vector machine based technique for online detection of outliers in transient time series," in *Asian Control Conference (ASCC)*. IEEE, 2015, pp. 1–6.
[31] J. Ma, L. Sun, H. Wang, Y. Zhang, and U. Aickelin, "Supervised anomaly detection in uncertain pseudoperiodic data streams," *ACM Transactions on Internet Technology (TOIT)*, vol. 16, no. 1, p. 4, 2016.
[32] M. S. Parwez, D. Rawat, and M. Garuba, "Big data analytics for user activity analysis and user anomaly detection in mobile wireless network," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2058–2065, 2017.

Di Wu (M'14) received his Ph.D. degrees in computer science from the University of California, Irvine, USA, in 2013. He was a Researcher with the Intel Collaborative Research Institute for Sustainable Connected Cites, a Research Associate with Imperial College London, a Visiting Researcher with IBM Research, and a Student Research Associate with the SRI International. He is currently an Associate Professor with Hunan University, China. His research interests include future networking, intelligent analytics, and smart architecture.

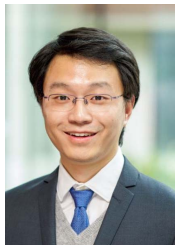


Zhongkai Jiang received the B.S. degree in Internet of Things Engineering from Suzhou University, China in 2016. He is currently working toward his M.S. degree in Computer Science and Technology at Hunan University, China. His research interests include machine learning and data mining.

Xiaofeng Xie received the B.S. degree in Network Engineering from Hunan University of Science and Technology, China in 2015, and the M.E. degree in Computer Technology from Hunan University, China in 2018. He is currently a Research Scientist with the Vivo Communication Technology, China. His research interests include machine learning and IoT data analytics.



Xuetao Wei (M'14) received his Ph.D. degree in computer science from the University of California, Riverside, USA, in 2013. From January 2014 to August 2019, he was an Assistant Professor and then promoted to Associate Professor at the University of Cincinnati, USA. He is currently an Associate Professor with the Southern University of Science and Technology, China. His research interests span the areas of blockchain, Internet of Things and security, which have been supported by federal and state funding agencies, including NSF and DARPA.



Weiren Yu (M'13) received his Ph.D. degree in computer science from the University of New South Wales, Australia, in 2014. He was a Research Associate with Imperial College London, and then a Lecturer of Computer Science with Aston University. He is currently an Assistant Professor of Computer Science with the University of Warwick, and an Honorary Research Fellow with Imperial College London. His current research spans graph data management, data mining, and Internet of Things. He is a Fellow of Higher Education Academy.

Renfa Li (M'05-SM'10) received his Ph.D. degree in electronic engineering from Huazhong University of Science and Technology, Wuhan, China, in 2003. He is currently a Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. He is the Director of the Key Laboratory for Embedded and Network Computing of Hunan Province, China, and an expert committee member of National Supercomputing Center in Changsha, China. His major interests include computer architectures, embedded computing systems, cyber-physical systems, and Internet of things.