Developing an Unsupervised Real-time Anomaly Detection Scheme for Time Series with Multi-seasonality

Wentai Wu, *Student Member, IEEE*, Ligang He, *Member, IEEE*, Weiwei Lin, Yi Su, Yuhua Cui, Carsten Maple, and Stephen Jarvis, *Member, IEEE*

***Abstract*—On-line detection of anomalies in time series is a key technique used in various event-sensitive scenarios such as robotic system monitoring, smart sensor networks and data center security. However, the increasing diversity of data sources and the variety of demands make this task more challenging than ever. Firstly, the rapid increase in unlabeled data means supervised learning is becoming less suitable in many cases. Secondly, a large portion of time series data have complex seasonality features. Thirdly, on-line anomaly detection needs to be fast and reliable. In light of this, we have developed a prediction-driven, unsupervised anomaly detection scheme, which adopts a backbone model combining the decomposition and the inference of time series data. Further, we propose a novel metric, Local Trend Inconsistency (LTI), and an efficient detection algorithm that computes LTI in a real-time manner and scores each data point robustly in terms of its probability of being anomalous. We have conducted extensive experimentation to evaluate our algorithm with several datasets from both public repositories and production environments. The experimental results show that our scheme outperforms existing representative anomaly detection algorithms in terms of the commonly used metric, Area Under Curve (AUC), while achieving the desired efficiency.**

***Index Terms*—time series, seasonality, anomaly detection, unsupervised learning**

I. Introduction

Time series data sources have been of interest in a vast variety of areas for many years – the nature of time series data was examined in a seminal study by Yule [1] and the techniques were applied to areas such as econometric [2] and oceanographic data [3] since the 1930s. However, in an era of hyperconnectivity, big data and machine intelligence, new technical scenarios are emerging such as autonomous driving, edge computing and Internet of Things (IoT). Analysis of such system poses new challenges to the detection of anomalies in time series data. Further, for a wide range of systems which require 24/7 monitoring services, it has become crucial to have the detection techniques that can provide early, reliable reports of anomalies. In cloud data centers, for example, a distributed monitoring system usually collects a variety of log data from the virtual machine level to the cluster level on a regular basis and sends them to a central detection module, which then analyzes the aggregated time series to detect any anomalous events including hardware failures, unavailability of services and cyber attacks. This requires a reliable on-line detector with strong sensitivity and specificity. Otherwise, the inefficient detection may cause unnecessary maintenance costs.

Several classes of schemes have been applied to the problem of anomaly detection for time series data. In certain cases decent results can be achieved by these traditional methods such as outlier detection [4][8][9][10], pattern (segment) extraction [12][13][14][15], sequence mapping [18][20][21]. However, we are facing a growing number of new scenarios and applications which produce large volumes of time series data with unprecedented complexity, posing challenges that traditional anomaly detection methods cannot address effectively. First, more and more time series data are being produced without labels since data labeling/annotation is usually very time-consuming and costly. Sometimes it is also unrealistic or impossible to acquire reliable labels when their correctness has to be guaranteed. Second, some applications may produce multi-channel series with complex features such as multi-period seasonality (i.e., multiple seasonal, such as yearly or monthly, patterns within one channel), long periodicity, fairly unpredictable channels and different seasonality between channels. As a result, learning these patterns requires effective seasonality discovery and strong ability for generalization. Third, the process is commonly required to be fast enough to support instant reporting or alarming once unexpected situation occurs. The capability of on-line detection is especially important in a wide range of event-sensitive scenarios such as medical and industrial process control systems.

In this paper, we propose a predictive solution to detecting anomalies effectively in time series with complex seasonality. The fundamental idea is to inspect the data samples as they arrive and match the data samples with an ensemble of forecasts made chronologically. Specifically, our solution is comprised of an augmented forecasting model and a novel detection algorithm that exploits the predictions of local sequences made by the underlying forecasting model. We build a frame-to-sequence Gated Recurrent Unit (GRU) network while extending its input with seasonal terms extracted by decomposing the time series of each sample channel. The integration of the seasonal features can alleviate negative impact from anomalous samples in the training data since the anomalous samples have minor impact on the long-term

Corresponding author: Ligang He. W. Wu, L. He, Y. Su and S. Jarvis are with the Department of Computer Science, University of Warwick. W. Lin is with the School of Computer Science and Engineering at the South China University of Technology. Y Cui is with the Research Institute of Worldwide Byte Information Security. C. Maple is with the Warwick Manufacturer Group, University of Warwick

periodic patterns. Because of the above reasons, our prediction framework does not require the labels (i.e., specifying which data are normal or abnormal) or uncontaminated training data (i.e., the training data only contain normal samples).

After predicting local sequences (i.e., the output of the forecasting model), we propose a novel method to weight the ensemble of different forecasts based on the reliability of their forecast sources and make it a chronological process to fit the on-line detection. The weight of each forecast is determined dynamically during the process of detection by scoring each forecast source (i.e., the forecast made based on this data source), which reflects how likely the predictions made by a forecast source is trustworthy. Based on the above ensemble, we propose a new metric, termed Local Trend Inconsistency (LTI), for measuring the deviation of an actual sequence from the predictions in real-time, and assigns an anomaly score to each of the newly arrived data points (which we also call frames) in order to quantify the probability that a frame is anomalous.

We also propose a method to map the LTI value of a frame to its Anomaly Score (AS) by a logistic-shaped function. The mapping further differentiates anomalies and normal data. In order to determine the logistic mapping function, we propose a method to automatically determine the optimal values of the fitting parameters in the logistic mapping function. The AS value of a frame in turn becomes the weight of its impact on the detection of future frames. This makes our LTI metric robust to the anomalous frames in the course of detection and significantly mitigates the potential impact of anomalous samples on the detection results of the future frames. This feature also enables our algorithm to work chronologically without maintaining a large reference database or caching too many historical data frames. To the best of our knowledge, the existing prediction-driven detection schemes do not take into account the reliability of the forecast sources.

The main contributions of our work are as follows:

- We first build a frame-to-sequence forecasting model integrating a GRU network with time series decomposition (using Prophet, an additive time series model developed by Facebook [29]) to enable the contamination-tolerant training on multi-seasonal time series data without any labels.
- We propose a new metric termed Local Trend Inconsistency (LTI), and based on this metric we further propose an unsupervised detection algorithm to score the probability of data anomaly. An practical method is also proposed for fitting the scoring function.
- We mathematically present the computation of LTI in the form of matrix operations and prove the possibility of parallelization for further speeding up the detection procedure.
- We conducted extensive experiments to evaluate the proposed scheme on two public datasets from the UCI data repository and a more complex dataset from a production environment. The result shows that our solution outperforms the existing algorithms significantly with low detection overhead.

The rest of this paper is organized as follows: Section 2 discusses a number of studies related to anomaly detection. In Section 3, we introduce Local Trend Inconsistency as the key metric in our unsupervised anomaly detection scheme. We then systematically present our unsupervised anomaly detection solution in Section 4, including the backbone model for prediction and a scoring algorithm for anomaly detection. We present and analyze the experimental results in Section 5, and finally conclude this paper in Section 6.

II. Related Work

The term *anomaly* refers to a data point that significantly deviates from the rest of the data which are assumed to follow some distribution or pattern. There are two main categories of approaches for anomaly detection: novelty detection and outlier detection. While novelty detection (e.g. classification methods [39][40][41][42]) requires the training data to be classified, outlier detection (e.g., clustering, principal component analysis [20] and feature mapping methods [43][44]) does not need a prior knowledge of classes (i.e., labels) and thus is also known as unsupervised anomaly detection. The precise terminology and definitions of these terminology may vary in different sources. We use the same taxonomy as Ahmed et al. did in reference [45] whilst in the survey presented by Hodge and Austin [38] unsupervised detection is classified as a subtype of outlier detection. The focus of our work is on unsupervised anomaly detection since we aim to design a more generic scheme and thus do not need to assume the labels are unavailable.

In the detection of time series anomalies, we are interested in discovering abnormal, unusual or unexpected records. In a time series, an anomaly can be detected within the scope of a single record or as a subsequence/pattern. Many classical algorithms can be applied to detect single-record anomaly as an outlier, such as the One Class Support Vector Machine (OCSVM) [4], a variant of SVM that exploits a hyperplane to separate normal and anomalous data points. Zhang et al. [5] implemented a network performance anomaly detector using OCSVM with Radial Basic Function (RBF), which is a commonly used kernel for SVM. Maglaras and Jiang [6] developed an intrusion detection module based on K-OCSVM, the core of which is an algorithm that performs K-means clustering iteratively on detected anomalies. Shang et al. [7] applies Particle Swarm Optimization (PSO) to find the optimal parameters for OCSVM, which they applied to detect the abnormalities in TCP traffic. In addition, Radovanović et al. [9] investigated the correlation between hub points and outliers, providing a useful guidance on using reverse nearest-neighbor counts to detect anomalies. Liu et al. [8] found that anomalies are susceptible to the property of ”isolation” and thus proposed Isolation Forest (iForest), an anomaly detection algorithm based on the structure of random forest. Taking advantage of iForest's flexibility, Calheiros et al. [10] adapted it to dynamic failures detection in large-scale data centers. For anomalous sequence or pattern detection, there are a number of classical methods available such as box modeling [11], symbolic sequence matching [18] and pattern extraction

[14][15]). For example, Huang et al. [19] proposed a scheme to identify the anomalies in VM live migrations by combining the extended Local Outlier Factor (LOF) and Symbolic Aggregate ApproXimation (SAX).

Recent advance in machine learning techniques inspires prediction-driven solutions for intelligent surveillance and detection systems (e.g., [48][49]). A prediction-driven anomaly detection scheme is often a sliding window-based scheme, in which future data values are predicted and then the predictions are compared against the actual values when the data arrive. This type of anomaly detection schemes has been attracting much attention recently thanks to the remarkable performance of recurrent neural networks (RNNs) in prediction/forecasting tasks. Filonov et al. [33] proposed a fault detection framework that relies on a Long Short Term Memory (LSTM) network to make predictions. The set of predictions along with the measured values of data are then used to compute error distribution, based on which anomalies are detected. Similar methodologies are used by [34] and [24]. LSTM-AD [34] is also a prediction scheme based on multiple forecasts. In LSTM-AD the abnormality of data samples is evaluated by analyzing the prediction error and the corresponding probability in the context of an estimated Gaussian error distribution obtained from the training data. However, the drawback of LSTM-AD is that it is prone to the contamination of training data. Therefore, when the training data contains both normal and anomalous data, the accuracy of the prediction model is likely to be affected, which consequently make the anomaly detection less reliable.

Malhotra et al. [23] adopt a different architecture named encoder-decoder, which is based on the notion that only normal sequences can be reconstructed by a well-trained encoder-decoder network. A major limitation of their model is that an unpolluted training set must be provided. As revealed by Pascanu et al. [25], RNNs may struggle in learning complex seasonal patterns in time series particularly when some channels of the series have long periodicity (e.g., monthly and yearly). A possible solution to that is decomposing the series before feeding into the network. Shi et al. [35] proposed a wavelet-BP (Back Propagation) neural network model for predicting the wind power. They decompose the input time series into the frequency components using the wavelet transform and build a prediction network for each of them. To forecast time series with complex seasonality, De Livera et al. [37] adopt a novel state space modeling framework that incorporates the seasonal decomposition methods such as the Fourier representation. A similar model was implemented by Gould et al. [36] to fit hourly and daily patterns in utility loads and traffic flows data.

Ensuring low overhead is essential for real-time anomaly detection. For example, Gu et al. [16] proposed an efficient motif (frequently repeated patterns) discovery framework incorporating an improved SAX indexing method as well as a trivial match skipping algorithm. Their experimental results on the CPU host load series show excellent time efficiency. Zhu et al. [17] propose a new method for locating similar subsequences as well as a parallel approach using GPUs to accelerate Dynamic Time Warping (DTW) for time series pattern discovery. Similarly, parallel algorithms (e.g., [50][51][52]) have been applied to several forms of machine learning models for efficiency boost.

III. Local Trend Inconsistency

In this section, we first introduce a series of basic notions and frequently-used symbols, then define a couple of distance metrics, and finally present the core concept in our anomaly detection scheme - *Local Trend Inconsistency* (*LTI*).

In some systems, more than one data collection device is deployed to gather information from multiple variables relating to a common entity simultaneously, which consequently generates multi-variate time series. In this paper we call them multi-channel time series.

***Definition** 1:* A *channel* is the full-length sequence of a single variable that comprises the feature space of a time series.

For the sake of convenience, we define a frame as follows. This concept of a frame is inspired by, but is more general than, a frame in video processing (since a video clip can be reckoned as a time series of images.)

***Definition** 2:* A *frame* is the data record at a particular point of time in a series. A frame is a vector in a multi-channel time series, or a scalar value in a single-channel time series.

Most of previous schemes detect anomalies by analyzing the data items in a time series as separate frames. However, in our approach we attempt to conduct the analysis from the perspective of local sequences.

***Definition** 3:* A *local sequence* is a fragment of the target time series; a local sequence at frame x is defined as a fragment of the series spanning from a previous frame to frame x.

For clarity, we list all the symbols frequently used in this paper in Table I.

TABLE I
LIST OF SYMBOLS

Symbol	Description
X	A time series X
$X(t)$	The tth frame of time series X
X_c	The cth channel of time series X
$X_c(t)$	The cth component of the tth frame of time series X
$x^{(i)}$	The ith feature of frame x
$\hat{x}_k^{(i)}$	The ith feature of the frame x predicted by frame k
S	An actual local sequence from the target time series
S_k	A local sequence predicted by frame k
$S(i)$	The ith frame in local sequence S
$S(i,j)$	An actual local sequence spanning from frame i to j
$S_k(i,j)$	A local sequence predicted by k spanning from frame i to j

Euclidean Distance and Dynamic Time Warping (DTW) Distance are commonly used to measure the distance between two vectors. However, the scale of Euclidean Distance largely depends on the dimensionality, i.e., vector length. DTW distance can measure the sequence similarity, but cannot produce the length-independent results. With the relatively high time complexity ($O(n^2m)$ for m-dimensional sequences of length n), DTW is often applied to the sequence-level analysis, in which the target is a sequence of frames or a pattern of varying length. However, our work aims to perform the frame-wise,

on-line detection, i.e., detect whether a frame is anomalous as the frame arrives.

Therefore, in this paper we use a modified form of Euclidean distance, called Dimension-independent Frame Distance ($DFDist$) as formulated in Eq. (1), to measure the distance between two frames $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$DFDist(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - y^{(i)})^2 \tag{1}$$

where m is the number of dimensions (i.e., number of channels) and x_i and y_i are the ith component of frame x and frame y, respectively. We do not square root the result. This does not impact the effectiveness of our approach, but makes it easier to handle when we transform all computations into matrix operations at the later stage of the processing. Also, the desired scale (i.e., $DFDist \in [0, 1]$) of the distance still holds for normalized data.

With $DFDist$, we can further measure the distance between two local sequences of the same length. The desired metric for sequence distance should be independent on the length of the sequences as we want to have a unified scale for any pair of sequences. We formulate the Length-independent Sequence Distance ($LSDist$) between two sequences S_X and S_Y of the same length in Eq. (2), where L is the length of the two local sequences.

$$LSDist(S_X, S_Y) = \frac{1}{L}\sum_{i=1}^{L} DFDist(S_X(i), S_Y(i)) \tag{2}$$

Although the definition of $LSDist$ already provides a unified scale of distance, the temporal information of the time series data is neglected. Assuming we are detecting the anomaly of the event at time t, we need to compare the local sequence at frame t with a ground truth sequence (assume there is one) to see if anything goes wrong in the latest time window. If we use $LSDist$ as the metric, then every time point is regarded as being equally important. However, this does not practically comply with the rule of time decay, namely, the most recent data point typically has the greatest reference value and also the greatest impact on what will happen in the next time point. Therefore, we refine $LSDist$ by weighting each term and adding a normalization factor. The Weighted Length-independent Sequence Distance ($WLSDist$) is defined in Eq. (3), where d_i is the weight of time decay for frame i and D_L is the normalization factor (so that $WLSDist$ remains in the same scale as $LSDist$).

$$WLSDist(S_X, S_Y) = \frac{\sum_{i=1}^{L} d_i \cdot DFDist(S_X(i), S_Y(i))}{D_L} \tag{3}$$

Time decay is applied on the basis that the two sequences are chronologically aligned. In this paper, we use the exponentially decaying weights, which is similar to the exponential moving average method [46]:

$$d_i = e^{-(L-i)}, i = 1, 2, ..., L \tag{4}$$

where i denotes the frame index and $L - i$ is the temporal distance (with $i = L$ being the current frame). Hence, the corresponding normalization factor D_L in Eq. (3) is the summation of a geometric series of length L:

$$D_L = \sum_{i=1}^{L} e^{-(L-i)} = \frac{1 - e^{-L}}{1 - e^{-1}} \tag{5}$$

where L is the sequence length.

Ideally it is easy to identify the anomalies by calculating $WLSDist$ between the target (such as local sequence or frame) and the ground truth. However, this approach is not feasible if the labels are unavailable (i.e., there is no ground truth). A possible solution is to replace the ground truth with expectation, which is obtained typically by using time series forecasting methods [22][34], which is the basic idea of the so-called prediction-driven anomaly detection schemes. However, a critical problem with such a prediction-driven scheme is the reliability of forecast. On the one hand, the prediction error is inevitable. On the other hand, the predictions made based on the historical frames, which may include anomalous frames, can be unreliable. This poses a great challenge for prediction-driven anomaly detection schemes.

Envisaging the above problems, we propose a novel, reliable prediction scheme, which makes use of multi-source forecasting. Unlike previous studies that use frame-to-frame predictors, our scheme makes a series of forecast at different time points (i.e, from different sources) by building a frame-to-sequence predictor. The resulting collection of forecasts form a *common* expectation from multiple sources for the target. When the target arrives and if it deviates from the common expectation, it is deemed that the target is likely to be an anomaly. This is the underlying principle of our unsupervised anomaly detection.

In order to quantitatively measure how far the target deviates from the collection of expectations obtained from multiple sources, we propose a metric we term the Local Trend Inconsistency (LTI). LTI takes into account the second challenging issue discussed above (i.e, there may exist anomalous frames in history) by weighting the prediction made based on a source (i.e., a frame at a previous time point) with the probability of the source being normal.

For a frame t (i.e., by which we refer to the frame arriving at time point t), $LTI(t)$ is formally defined in Eq. (6), where $S(i+1, t)$ is the actual sequence from frame $i+1$ to frame t, and $S_i(i+1, t)$ is the sequence of the same span predicted by frame i (i.e., prediction made when frame i arrives). L is the length of the prediction window, which is a hyper-parameter determining the maximum length of the predicted sequence and also the number of sources that make the predictions (i.e., the number of predictions/expectations) of the same target. $P(i)$ denotes the probability of frame i being normal.

Z_t is the normalization factor for frame t defined as the sum of all the probabilistic weights shown in Eq. (7). Z_t is used to normalize the value of $LTI(t)$ to the range of $[0, 1]$.

$$LTI(t) = \frac{1}{Z_t} \sum_{i=t-L}^{t-1} P(i) \cdot WLSDist\big(S(i+1,t), S_i(i+1,t)\big) \tag{6}$$

$$Z_t = \sum_{i=t-L}^{t-1} P(i) \tag{7}$$

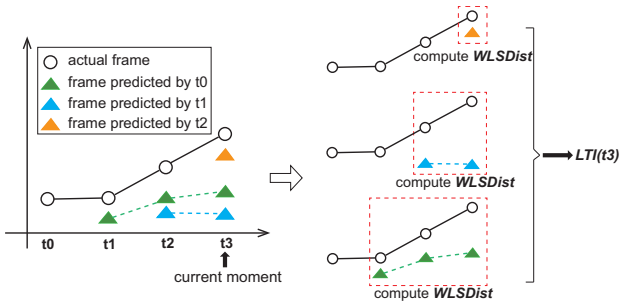


Fig. 1. An example demonstrating the calculation of Local Trend Inconsistency with the max probe length L equal to 3

Fig. 1 illustrates how LTI is calculated in a case where $L = 3$ (i.e., the length of the prediction window is 3). Based on the actual data arriving at t_0 (actual data are represented by circles), our scheme predicts the frames at three future time points, i.e., t_1, t_2 and t_3, which are depicted as green triangles in the left part of Fig. 1. When the time elapses to t_1, the data at t_1 arrives and our scheme predicts the data at the time points of t_2, t_3 and t_4 (in the figure we only plot the predictions up to the time point t_3), which are colored blue. Similarly, when the time elapses to t_2, the data at t_2 arrives and our scheme forecasts the data at the time points of t_3, t_4 and t_5 (colored orange).

Now assume we want to calculate $LTI(t_3)$ to gauge the abnormality of the data arriving at time t_3. As shown in Fig. 1, at time t_3, we know the actual local sequence from t_0 to t_3, i.e., $S(t_0, t_3)$ (corresponding to the term $S(i+1,t)$ in Eq. 6), and also we have made the following three predictions, which are the forecasts at three different time points:

- $S_0(t_1, t_3)$: the predicted local sequence from t_1 to t_3, which is predicted at time t_0;
- $S_1(t_2, t_3)$: the predicted local sequence from t_2 to t_3, which is predicted at time t_1;
- $S_2(t_3)$: the prediction of frame t_3 made at time t_2.

$LTI(t_3)$ is then obtained by i) calculating the weighted distances (i.e., $WSLDist$ in Eq. 3) between the predicted sequences and the corresponding actual sequence up to time t_3, i.e., the distances between $S_0(t_1, t_3)$ and $S(t_1, t_3)$ (shown at the bottom right of Fig. 1), between $S_1(t_2, t_3)$ and $S(t_2, t_3)$ (middle right of Fig. 1), and between $S_2(t_3)$ and $S(t_3)$ (top right of Fig. 1); ii) calculating the weighted sum (the weight is $P(i)$) of the distances obtained in last step, and iii) normalizing the weighted sum (i.e. divided by Z_t in Eq. (7)).

This multi-source prediction establishes the common expectation for the data values. How far the actual data deviates from the predicted data, which is measured by the distance between them, is used to quantify the abnormality of the given data.

The whole process can be formulated using matrix operations. Assume we are detecting anomaly at frame t and the size of the prediction window is L. For brevity let $df_k(t)$ denote the distance between frame t and a forecast of the frame made at time k (i.e., $DFDist(t, \hat{t}_k)$). We first define the frame-distance matrix $\mathbf{D_F}$:

$$\mathbf{D_F} = \begin{bmatrix} \mathbf{D_F}^{(t-L)} \\ \mathbf{D_F}^{(t-L+1)} \\ \vdots \\ \mathbf{D_F}^{(t-1)} \end{bmatrix}$$

where

$$\mathbf{D_F}^{(u)} = \begin{bmatrix} df_u(u+1) \\ df_u(u+2) \\ \vdots \\ df_u(t) \end{bmatrix}^{\mathrm{T}}$$

Then we define two diagonal normalization matrices $\mathbf{N_1}$ and $\mathbf{N_2}$ as follows:

$$\mathbf{N_1} = \begin{bmatrix} \frac{1}{D_L} & & & 0 \\ & \frac{1}{D_{L-1}} & & \\ & & \ddots & \\ 0 & & & \frac{1}{D_1} \end{bmatrix}$$

$$\mathbf{N_2} = \begin{bmatrix} \frac{1}{Z_t} & & & 0 \\ & \frac{1}{Z_t} & & \\ & & \ddots & \\ 0 & & & \frac{1}{Z_t} \end{bmatrix}$$

where D_L and Z_t are defined in (5) and (7), respectively. For convenience let $ds_k(t)$ denote $WLSDist\big(S(k+1,t), S_k(k+1,t)\big)$. Hence we can derive the matrix of weighted local sequence distances denoted as $\mathbf{D_S}$:

$$\mathbf{D_S} = \begin{bmatrix} ds_{t-L}(t) \\ ds_{t-L+1}(t) \\ \vdots \\ ds_{t-1}(t) \end{bmatrix} = \mathbf{N_1 D_F T}$$

where $\mathbf{T}$ is the time decay vector defined as:

$$\mathbf{T} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_L \end{bmatrix}$$

where d_i is computed via Eq. (4). Now we assume the probability of being normal is already known for each of frame t's predecessors (i.e., $P(t-1), P(t-2), ...$), and we put them together into a $1 \times L$ matrix $\mathbf{P}$:

$$\mathbf{P} = \begin{bmatrix} P(t-L) & P(t-L+1) & \cdots & P(t-1) \end{bmatrix}$$

Then we can reformulate $LTI(t)$ as below:

$$LTI(t) = \mathbf{P N_2 D_S} = \mathbf{P N_2 N_1 D_F T} \tag{8}$$

Through the use of matrices to formulate the calculation of LTI, we can know that the calculation can be performed efficiently in parallel. The Degree of Parallelism (DoP) of its

calculation can be higher than L. This is because the DoP for calculating the L terms in Eq. (6) can be L apparently (the calculation of every term is independent on each other). The calculation of each term can be further accelerated (including the calculations of $WLSDist$ and $DFDist$) by parallelizing the matrix multiplication. For example, with a number of $L \times L$ processes (i.e., a grid of processes) and exploiting the Scalable Universal Matrix Multiplication Algorithm (SUMMA) [47], we can achieve a roughly L^2 speedup in the multiplication of any two matrices with the dimension size of L, which helps reduce the time complexity of computing $\mathbf{N_1 D_F}$ from $O(L^3)$ to $O(L)$. Further, with the resulting $\mathbf{N_1 D_F}$ the computation of $\mathbf{N_1 D_F T}$ and $\mathbf{P N_2}$ can be performed in parallel as both of them are vector-matrix multiplication requiring only L processes and have time complexity of $O(L^2/L) = O(L)$. Finally multiplying the resulting matrices of $\mathbf{P N_2}$ (dimension=$1 \times L$) and $\mathbf{N_1 D_F T}$ (dimension=$L \times 1$) consumes $O(L)$. Note that the matrix D_F contains $L \times L$ entries of frame distance, each of which is calculated using Eq. (1). Therefore, updating D_F (upon a new frame arrives) is an operation with the complexity of $O(L^2 m/L^2) = O(m)$, where m is the frame dimension. Consequently, the time complexity of computing $LTI(t)$ in parallel is $O(m + L)$ in theory.

IV. Anomaly Detection with LTI

Our anomaly detection scheme is based on LTI (Local Trend Inconsistency) as LTI can effectively indicate how significantly the series deviates locally from the common expectation established by multi-source prediction.

As can be seen from Eq. (6), there are still two problems to be solved in calculating LTI. First, a mechanism is required to make reliable predictions of local sequences. Second, we need an algorithm to quantify the probabilistic factors (in matrix $\mathbf{P}$) as they are not known apriori.

In this section, we first introduce the backbone model we build for achieving accurate frame-to-sequence forecasting. The model is designed to learn the complex patterns in multi-seasonal time series with tolerance to pollution in the training data. Then we illustrate how to make use of the predictions (from multiple source frames) made to compute LTI. Finally, we propose an anomaly scoring algorithm that uses a scoring function to chronologically calculate anomaly probability for each frame based on LTI.

A. Prediction Model

To effectively learn and accurately predict local sequences in multi-seasonal time series, we adopt a combinatorial backbone model composed of a decomposition module and an inference module.

Recurrent Neural Network (RNN) is an ideal network to implement the inference module of our prediction model. RNNs (including mutations such as Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU)) are usually applied as end-to-end models (e.g., [26] [27]). However, a major limitation of them is the difficulty in learning complex seasonal patterns in multi-seasonal time series. Even though the accuracy may be improved by stacking more hidden layers and increasing back propagation distance (through time) during training, it could cause prohibitive training cost.

In view of this, we propose to include the seasonal features of the input data explicitly as the input of the neural network. This is achieved by conducting time series decomposition before running the prediction model, which is the purpose of the decomposition module. The resulting seasonal features can be regarded as the outcome of feature engineering. Technically speaking, seasonal features are essentially the "seasonal terms" decomposed from each channel of the target time series. We use *Prophet* [29], a framework based on the decomposable time series model [28], to extract the channel-wise seasonal terms. Let X_c denote the c-th channel of time series X, and $X_c(t)$ the t-th record of the channel. The outcome of time series decomposition for channel c is formulated as below:

$$X_c(t) = g_c(t) + s_c(t) + h_c(t) + \epsilon \tag{9}$$

where $g_c(t)$ is the trend term that models non-periodic changes, $s_c(t)$ represents the seasonal term that quantifies the seasonal effects. $h_c(t)$ reflects the effects of special occasions such as holidays, and ϵ is the error term that is not accommodated by the model. For simplicity, we in this paper only consider daily and weekly seasonal terms as additional features for the inference module of our model. *Prophet* relies on Fourier series to model multi-period seasonality, which enables the flexible approximation of any periodic patterns with arbitrary length. The underlying details can be referred to [29].

Separating seasonal terms from original frame values and using them as additional features effectively improve RNN from the following perspectives. First, explicit input of seasonal terms helps reduce the difficulty of learning complex seasonal terms in RNN. The extracted seasonal terms quantify seasonal effects. Second, time cost of training is expected to decrease as we can apply the Truncated Back Propagation Through Time (TBPTT) with a distance much shorter than the length of periodicity. Besides, the series decomposition process is very efficient, which will be demonstrated later by experiments. The top part of Fig. 2 shows the architecture of our backbone prediction model. In the prediction model, a stacked GRU network is implemented as the inference module, which takes as input the raw features of a frame concatenated with its seasonality features. We demonstrate the effectiveness of this backbone model in Section V-A.

B. Computing LTI based on Predictions

When we calculate Local Trend Inconsistency (LTI) in Eq. (6), we are actually measuring the distance between a local sequence and an ensemble of its predictions by a well trained backbone prediction model. The workflow of our online anomaly detection method includes three main steps: i) feed every arriving frame into the prediction model and continuously gather its output of predicting future frames, ii) organize the frame predictions by their sources (i.e., the frames which made the forecast) and concatenate them into local sequences, and iii) compute LTI of the newly arrived frame according to Eq. (6). Fig. 2 demonstrates the entire process, in

which LTI of a frame is converted to a score of abnormality using the algorithms to be introduced later.

C. Anomaly Scoring

In theory, the values of $LTI(t)$ can be directly used to score frame t in terms of its abnormality. However, the range of this metric is application-specific. So we further develop a measure that can represent the probability of data anomaly. Specifically, we define a logistic mapping function to convert the value of $LTI(t)$ to a probabilistic value:

$$\Phi(x) = \frac{1}{1 + e^{-k(x-x_0)}} \tag{10}$$

where k is the logistic growth rate and x_0 the x-value of the function's midpoint.

The left part of Fig. 3 shows the shapes of $\Phi(\cdot)$ with different values of k when x_0 is set to 0.5. The shape of $\Phi(\cdot)$ becomes steeper as k increases. We will introduce how to determine the optimal values of k and x_0 later.

Now we define the probabilistic anomaly score of frame t as below:

$$AS(t) = \Phi(LTI(t)) \tag{11}$$

The reason why we use Eq. (10) to map $LTI(t)$ to $AS(t)$ are three folds. First, we find that the $LTI(t)$ values are clustered together closely (top right of Fig. 3), which means that the difference in $LTI(t)$ values between normal and abnormal frames are not significant. This makes it difficult to differentiate them in practice although we can do so in theory. The right part of Fig. 3 illustrates the situation where we map raw $LTI(t)$ values to $AS(t)$. It can be seen from the figure that the value of anomaly scores are better dispersed leaving a clearer divide between normal data and (potential) anomalies. For example, the red line we draw separates out roughly 10 percent of potential anomalies with high scores. Second, as discussed in the previous section, our scheme makes a series of forecast from different sources for the target, which establishes a common expectation for the target. The challenge is that there may exist anomalous sources, from which the forecast made is unreliable. Thus we have to differentiate the quality of the predictions by specifying large weights (i.e., the $P(i)$ in Eq. 6) for normal sources and small weights for the sources that are likely to be abnormal. With the function $\Phi(\cdot)$ to disperse the $LTI(t)$ values (by mapping them into $AS(t)$, the impact difference between normal and abnormal frames is magnified. Last but not the least, we find that the actual values of $LTI(t)$ depend on particular applications that our detection scheme is applied to. After mapping, the $AS(t)$ values becomes less application-dependent, making it possible to set a universal anomaly threshold. This is similar to the scenario of determining the unusual events if the samples follow the normal distribution: the values lying beyond two standard deviations from the mean are often regarded as unusual.

Considering the second reason discussed above, we replace $P(i)$ in Eq. (6) with $1 - AS(i)$ where $i = t - L, t - L + 1, ..., t - 1$. Consequently, $LTI(t)$ is reformulated as:

$$LTI(t) = \frac{1}{Z_t} \sum_{i=t-L}^{t-1} (1 - AS(i)) \cdot WLSDist\big(S(i+1,t), S_j(i+1,t)\big) \tag{12}$$

where Z_t is the normalization factor reformulated as $\sum_{i=t-L}^{t-1}(1 - AS(i))$ and $1 - AS(i)$ represents the probability that frame i is normal.

The function $\Phi(\cdot)$ contains two parameters, k and x_0. The values of these two parameters need to be set before the function can be used to calculate the anomaly. Since x_0 is supposed to the midpoint of x, we set x_0 to be $\mathrm{mean}(LTI)$. We set k to $c/\mathrm{stdev}(LTI)$ ($\mathrm{stdev}(LTI)$ is the standard deviation of LTI, and c is a constant multiplier). The purpose of the mapping function is to disperse the LTI values that are densely clustered. On the one hand, the standard deviation $\mathrm{stdev}(LTI)$ can be used to represent how densely the LTI values reside around the mean. The lower the value of $\mathrm{stdev}(LTI)$, the more closely the LTI values are clustered. On the other hand, k represents how steep the middle slope of the logistic mapping function is. The greater k is, the steeper the logistic mapping function is. The more densely clustered the LTI values are, the steeper the logistic function needs to be in order to disperse those values. Therefore, for a set of LTI values with lower deviation, a bigger value should be set for k.

Instead of setting the values of k and x_0 manually, we propose an automated approach in this work to determine their values. More specifically, we design an iterative algorithm. The algorithm runs on a reference time series which is a portion of the training data. The algorithm is outlined in Algorithm 1.

Algorithm 1: Iterative procedure for unparameterizing $\Phi(\cdot)$

Input : prediction span L, reference series length r, predicted local sequences $S_i(i+1, i+L)$ for $i \in [0, r-1]$
Output: k, x_0
$k \leftarrow 1.0$, $x_0 \leftarrow 0.5$
$AS(i) \leftarrow 0$ for all $i \in [0, r-1]$
while *convergence criterion is not satisfied* **do**
  for $t \leftarrow L$ **to** $r - 1$ **do**
    compute $LTI(t)$ via Eq. (12)
    compute $AS(t)$ via Eq. (11)
  end
  $k \leftarrow \frac{c}{\mathrm{stdev}(LTI)}$, $x_0 \leftarrow \mathrm{mean}(LTI)$
end

In Algorithm 1, parameters k and x_0 are set to 1.0 and 0.5 initially, respectively. Note that it does not matter much what the initial values of k and x_0 are. When Algorithm 1 is run on the reference time series, LTI for each frame of the reference series is calculated. The values of k and x_0 will converge to $c/\mathrm{stdev}(LTI)$ and $\mathrm{mean}(LTI)$ eventually. In the

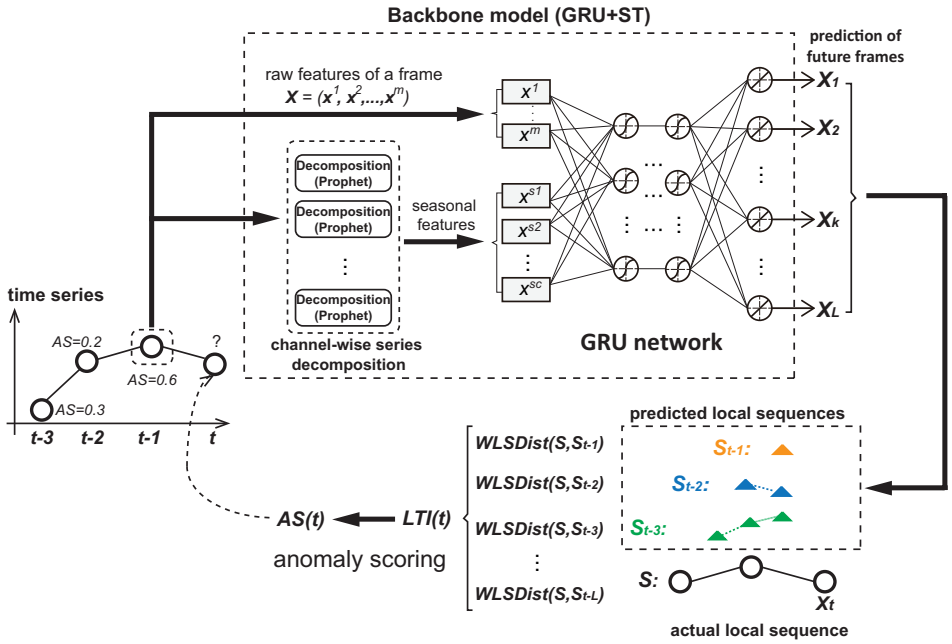


Fig. 2. An overview of the proposed prediction-driven anomaly detection framework for the time series, which uses a seasonality augmented GRU network as the backbone model to support the abnormality scoring based on Local Trend Inconsistency (LTI).

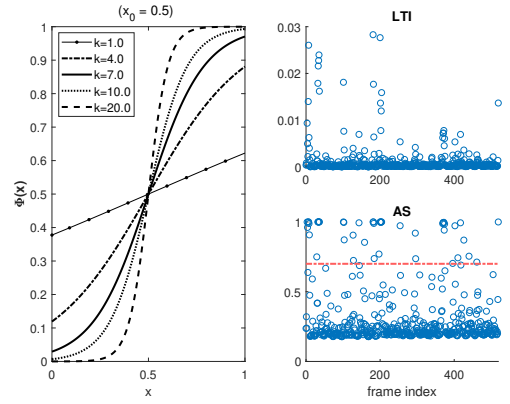


Fig. 3. The mapping function $\Phi(\cdot)$ we use for anomaly scoring (left), and the dispersion effect by mapping LTI values (top right) to anomaly scores (bottom right) with $\Phi(\cdot)$.

Algorithm 2: Anomaly Detection based on LTI

Input : current frame t, prediction span L, previous frames from $t-L$ to $t-1$, $AS(i)$ for $i \in [t-L, t-1]$

Output: $AS(t)$

for $i \leftarrow t-L$ **to** $t-1$ **do**

  use the proposed prediction model to forecast $S_i(i+1,t)$

  compute $WLSDist\big(S(i+1t), S_i(i+1,t)\big)$

end

compute $LTI(t)$

compute $AS(t)$

algorithm, we set a convergence criterion, in which both k and x_0 change by less than 0.1% since last update. In each loop, the algorithm computes $LTI(t)$ and $AS(t)$ along the reference series for each frame t. After each loop, we update k and x_0 and check if the criterion is met.

With the anomaly scoring function $AS(\cdot)$ and the trained backbone model for the target series, we now present our Anomaly Detection based on Local Trend Inconsistency (AD-LTI). Assume we are detecting the anomaly for frame t, the pseudo-code of our on-line detection procedure is described in Algorithm 2.

The information required for detection at frame t includes frame t itself, anomaly scores of previous frames, and the predicted local sequences ending at t, which is the output of our backbone prediction model. To analyze the time complexity of Algorithm 2, let m denote the number of dimensions of a frame (i.e., channels of the time series) and L the prediction span, which is a hyper-parameter shared by the backbone prediction model and the detection algorithm. Without parallelization, it takes $O(m)$ to calculate $DFDist$ between each pair of frames, so the time cost for obtaining $WLSDist$ between two local sequences is $O(Lm)$. Therefore, the time complexity of detection at a single frame t is $O(L^2m)$ since L sources of forecast are used (see Eq. 12). As analyzed in Section III, the complexity can be reduced to $O(m+L)$ with the proper parallelization.

V. Experiments

In this section, we first evaluate the effectiveness of our backbone prediction model. Then we compare AD-LTI with the existing anomaly detection algorithms in sensitivity and specificity (using the AUC metric).

We set up our experiments on a machine equipped with a dual-core CPU (model: Intel Core i5-8500, 3.00 GHz),

a GPU (model: GTX 1050 Ti) and 32GB memory. The inference module of our backbone model is implemented on Pytorch (version: 1.0.1) platform and the decomposition module is implemented using Prophet (version: 0.4) released by Facebook. We select three datasets for evaluation. CalIt2 and Dodgers Loop Sensor are two public datasets published by the University of California Irving (UCI) and available in the UCI machine learning repository. Another dataset we use is from the private production environment of a cyber-security company, which is the collaborator of this project. This dataset collects the server logs from a number of clusters (owned by other third-party enterprises) on a regular basis. The dataset is referred to as the Server Log dataset in this paper.

CalIt2 Dataset

CalIt2 is a multivariate time series dataset containing 10080 observations of two data streams corresponding to the counts of in-flow and out-flow of a building on UCI campus. The purpose is to detect the presence of an event such as a conference and seminar held in the building. The timestamps are contained in the dataset. The original data span across 15 weeks (2520 hours) and is half-hourly aggregated. We truncated the last 120 hours and conducted a simple processing on the remaining 2400 hours of data by making it hourly-aggregated. The CalIt2 dataset is provided with annotations that label the date, start time and end time of events over the entire period. There are 115 anomalous frames (4.56% contamination rate) in total. In our experiment, labels are omitted during training (because our prediction model forecasts local sequences of frames) and will only be used for evaluating detecting results.

Server Log Dataset

The Server Log dataset is a multi-channel time series with a fixed interval between two consecutive frames. The dataset spans from June 29th to September 4th, 2018 (1620 hours in total). The raw data is provided to us in form of separate log files, each of which stores the counts of a Linux server event on an hourly basis. The log files record the invocations of five different processes, which include CROND, RSYSLOGD, SESSION, SSHD and SU. Each process represents a channel of observing the server. We pre-processed the data by aggregating all the files to form a five-channel time series. Fig. 4 shows the time series of all five channels.

Currently, the company relies on security technicians to observe the time series and spot the potential anomalies, which might be caused by the security attacks. The aim of this project is to develop the automated method to spot the potential anomalies and quantify them at real time as the process invocations are being logged in the server. Anomalous events such as external cyber attacks exist in the Server Log dataset, but the labels are not available. We acquired the manual annotations for the test set from the technicians in the company. Totally 76 frames are labeled as anomalies in the test set, equivalent to a contamination rate of 14.6%.

Dodgers Loop Sensor Dataset

Dodgers Loop Sensor is also a public dataset available in the UCI data repository. The data were collected at the Glendale on-ramp for the 101 North freeway in Los Angeles. The sensor is close enough to the stadium for detecting unusual traffic after a Dodgers game, but not so close and heavily used by the game traffic. Traffic observations were taken over 25 weeks (from Apr. 10 to Oct. 01, 2005) with date and timestamps provided for both data records and events (i.e., the start and end time of games). The raw dataset contains 50400 records in total. We pro-processed the data to make it an hourly time series dataset.

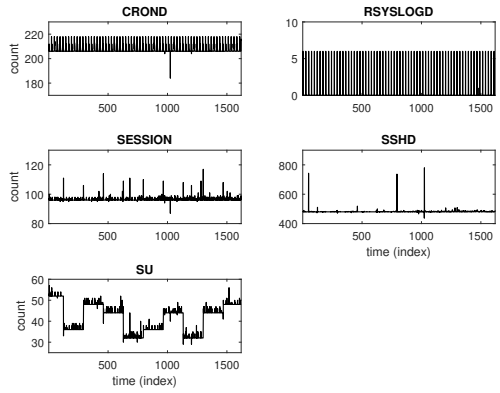


Fig. 4. The Server Log time series dataset

A. Evaluating Backbone Model

We trained our prediction model on the datasets separately to evaluate its accuracy as well as the impact of seasonal terms extracted by the decomposition module. We split the datasets into training, validation and test sets. On CalIt2, the first 1900 frames were used for training and the following 500 for testing. On the Server Log dataset, 1100 frames for training and 520 for test. On Dodgers Loop, 3000 records for training and 1000 for test. 300, 300, and 500 frames were used for validation on CalIt2, Server Log and Dodgers Loop, respectively.

The proposed model uses *Prophet* to implement the decomposition module and a stacked GRU network to implement the prediction module. We extracted daily and weekly terms for each channel. More specifically, for each channel we generated two mapping lists after fitting the data by *Prophet*. One list contains the readings at each of 24 hours in a day, while the other list includes the readings at each of 7 days in a week. Fig. 5 shows an example of the mapping lists.

The values of seasonal terms are different for CalIt2, Server Log and Dodgers datasets, but the resulting mapping lists share the same format as the example shown in Fig. 5.

Based on the mapping lists and the timestamp field provided in the data we build our prediction network with seasonal features as additional input. Table II shows the network structures adopted for each of the datasets, where L is the maximum length of local sequences as a hyper-parameter. $tanh$ is used as the activation function and Mean Square Error (MSE) loss as the loss function. Dropout is not enabled and we set a weight decay of $6e-6$ during the training to prevent over-fitting. We use Adam [30] as the optimizer with the initial learning rate set to 0.001.

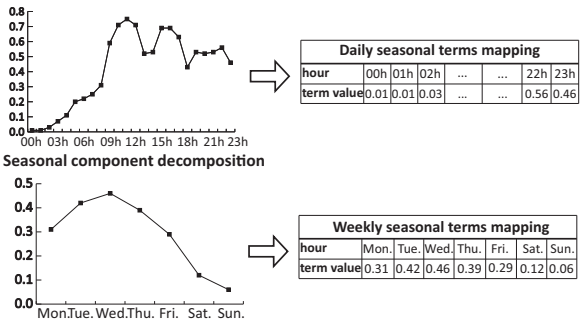


Fig. 5. An example of seasonal terms mapping in which the numerical values quantify seasonal impacts

TABLE II
NETWORK STRUCTURES OF THE INFERENCE PART FOR PREDICTING LOCAL SEQUENCE OF LENGTH L

Dataset	type	# of features (raw+seasonal)	topology
CalIt2	GRU	2+4	$[6, 20\times2, 2L]$
Server log	GRU	5+10	$[15, 20\times3, 5L]$
Dodgers	GRU	1+2	$[3, 20\times2, L]$

In order to evaluate the impact of the concatenated seasonal features, we also implemented a baseline GRU network with the same structure and hyper-parameters as our inference module except that the seasonal features are not included. We also consider the impact of a critical hyper-parameter, *time_steps*, in training the inference networks. The larger the *time_steps*, the longer the gradients back-propagate through time and the more time-consuming the training process becomes. We set different values of *time_steps* for the training of both our inference network and the baseline network to investigate the impact of seasonal features. The prediction span L is fixed to 5 (hours). The results are summarized in Table III.

In Table III, the decomposition time and training time refer to the fitting/training time spent by the decomposition module and the inference module, respectively. We evaluated three cases where *time_steps* takes different values of 24 (daily seasonality length), 72 or 168 (weekly seasonality length). Mean squared error (MSE) is calculated on the normalized test data to reflect the model quality. From the results we can first observe that it only takes the decomposition module of our model a few seconds to extract the seasonal terms from all the channels. More importantly, we find that augmenting the GRU model with seasonal terms (ST) makes the backbone model (GRU+ST) more complicated in structure, but it does not increase the training cost while resulting in much better accuracy – it outperforms the baseline GRU network (without Seasonal Terms) significantly in accuracy (i.e., lower error). The accuracy increases by more than 20 percent on CalIt2 and by from 35 to nearly 50 percent on the Server Log dataset.

B. Evaluating AD-LTI

In this section we evaluate our unsupervised anomaly detection algorithm AD-LTI. We also implement a number of representative related algorithms for comparison. These baseline algorithms include One Class Support Vector Machine (OCSVM) [4], Isolation Forest (iForest) [8], Piecewise Median Anomaly Detection [32], LSTM-based Fault Detection (LSTM-FD) [33] and LSTM-AD, which is LSTM-based anomaly detection scheme using multiple forecasts [34].

OCSVM is a mutation of SVM for unsupervised outlier detection. OCSVM shares the same theoretical basis as SVM while using an additional argument ν as an anomaly ratio-related parameter. Isolation forest is an outlier detection approach based on random forest in which isolation trees are built instead of decision trees. An a priori parameter cr is required to indicate the contamination rate. Both OCSVM and Isolation Forest are embedded in the Scikit-learn package [31]. Piecewise Median Anomaly Detection is a window-based algorithm that splits the series into fixed-size windows within which anomalies are detected based on a decomposable series model. LSTM-FD is a typical prediction-driven approach that detects anomalies in time series by simply analyzing (prediction) error distribution. They adopt a frame-to-frame LSTM network as their backbone model. Similar to our approach, LSTM-AD also uses a multi-source prediction scheme (we discussed its working in Section II).

We use the AUC metric to measure the effectiveness. Area Under the Curve, abbreviated as AUC, is a commonly used metric for comprehensively assessing the performance of binary classifiers. "The curve" refers to the Receiver Operator Characteristic (ROC) Curve, which is generated by plotting the true positive rate (y-axis) against the false positive rate (x-axis) based on the dynamics of decisions made by the target classifier (the anomaly detector in our case). The concept of ROC and AUC can reveal the effectiveness of a detection algorithm from the perspectives of both specificity and sensitivity. Another reason why we choose AUC is because it is a threshold-independent metric. AD-LTI does not perform classification but presents the detection results in the form of probability. Hence metrics such as precision and recall cannot be calculated unless we consider the threshold as an extra parameter, which violates our aim of designing a generic scheme.

We evaluate AD-LTI and the baseline algorithms on these three datasets. Parameters for baseline algorithms are set to the default or the same as in the original papers if they were suggested. For LSTM-FD, LSTM-AD and AD-LTI, $time\_steps$ is set to 72 (hours).

As shown in Fig. 6, Fig. 7 and Fig. 8, we draw three groups of 1-D heatmaps to compare the detection decisions made by each algorithm (labelled on the y-axis) with the ground truth on each test dataset. Normal and anomalous frames are marked by green and red, respectively, on the map of ground truth. Frames are also marked by each anomaly detection algorithm with scores, which are reflected using a range of colors from green to red. Anomaly events are sparse in Calit2 dataset (Fig. 6) while comparatively more anomalous data point exist in the Server Log (Fig. 7). From the figures, we can see that most of the "hotspots" are captured by our scheme and its false alarm rate is comparatively low. Notably we also observe that OCSVM produces a large number of false alarms on Calit2 but fails to spot most of the anomaly frames on the Server Log dataset. The Piecewise method misses a lot of anomalies, while the iForest method tends to mistakenly label a large portion

TABLE III

COMPARING GRU+ST (THE PROPOSED BACKBONE MODEL AUGMENTED WITH SEASONAL FEATURES) WITH THE VANILLA GRU IN ACCURACY, WHICH IS INDICTED BY THE LOWEST TEST MSE (MEAN SQUARE ERROR) ACHIEVED UNDER DIFFERENT TRAINING SETTINGS OF $time\_steps$ (ts). IN EACH GROUP OF COMPARISON, BOTH MODELS HAVE CONVERGED AND TRAINED FOR THE SAME NUMBER OF EPOCHS.

		Calit2 Dataset		Server Log Dataset		Dodgers Loop Dataset	
		GRU+ST	GRU	GRU+ST	GRU	GRU+ST	GRU
	seasonal term decomp. time	2.7s	-	6.6s	-	2.9s	-
ts=24	Test MSE	0.0068	0.0092	0.0020	0.0039	0.0098	0.0113
	Training time to converge	173.7s	176.7s	460.4s	464.9s	550.2s	552.3s
ts=72	Test MSE	0.0066	0.0089	0.0013	0.0020	0.0066	0.0085
	Training time to converge	180.2s	185.6s	468.3s	436.9s	628.6s	632.9s
ts=168	Test MSE	0.0067	0.0085	0.0018	0.0033	0.0072	0.0086
	Training time to converge	169.8s	174.5s	421.0s	446.5s	709.9s	752.9s

of normal data as anomalies. LSTM-AD produced the results close to our method on the Dodgers dataset, but rendered a large portion of false alarms on other two datasets. To give a more intuitive view, we plot the ROC curves of AD-LTI and the baseline algorithms on the test data in Fig. 9, Fig. 10 and Fig. 11.

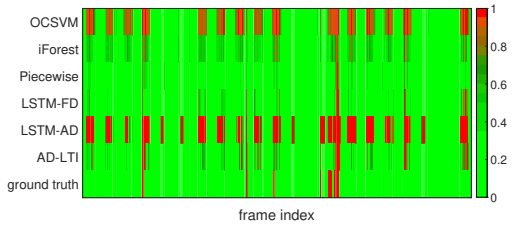


Fig. 6. Heatmaps of detection decisions made by AD-LTI and baseline algorithms compared with the ground truth on CalIt2 dataset

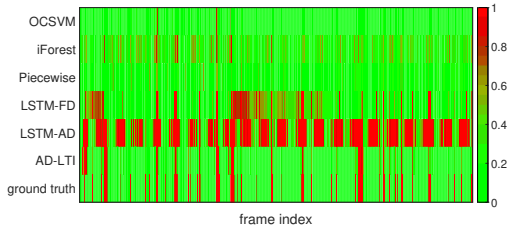


Fig. 7. Heatmaps of detection decisions made by AD-LTI and baseline algorithms compared with the ground truth on Server Log dataset

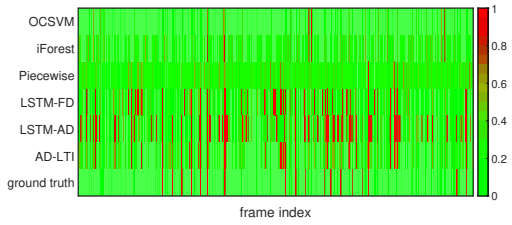


Fig. 8. Heatmaps of decision results by AD-LTI and baseline algorithms compared with the ground truth on Dodgers Loop dataset

From the ROC curves we can observe that AD-LTI produced the most reliable decisions as its curve is the closest to the top-left corner for all of the three datasets, especially on the Server Log Dataset (see Fig. 10), which features the complex seasonality in each channel. The detection difficulty on the Server Log dataset appears to be harder for other existing

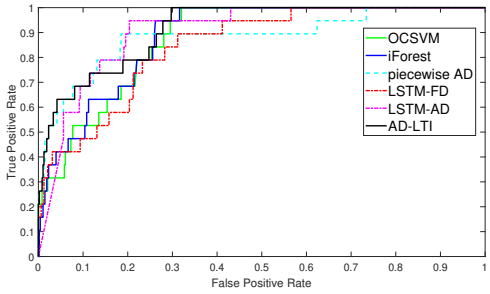


Fig. 9. ROC curves of anomaly detection algorithms on Calit2 dataset

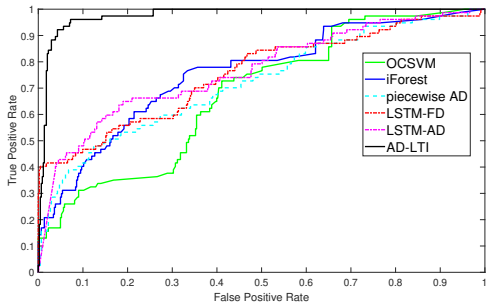


Fig. 10. ROC curves of anomaly detection algorithms on Server Log dataset

algorithms (the reason is explained later) - none of other algorithms achieve high true positive rate at a low false positive rate. We further calculate the corresponding AUC for each algorithm on both datasets. The resulting AUC values are

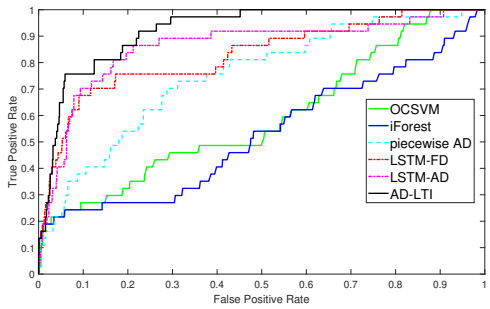


Fig. 11. ROC curves of anomaly detection algorithms on Dodgers Loop dataset

shown in Table IV.

TABLE IV
COMPARING THE AUC VALUES OF ANOMALY DETECTION ALGORITHMS ON CALIT2, SERVER LOG AND DODGERS LOOP DATASETS WHEREIN ACTUAL CONTAMINATION RATES (CR) ARE APPROXIMATELY 0.05, 0.15 AND 0.10, RESPECTIVELY.

	CalIt2	Server Log	Dodgers Loop
OCSVM [4](default)	0.876	0.677	0.591
OCSVM (nu = CR)	0.708	0.672	0.525
iForest [8](default)	0.891	0.756	0.535
iForest (cr = CR)	0.877	0.761	0.518
Piecewise AD [32]	0.833	0.721	0.751
LSTM-FD [33]	0.847	0.755	0.829
LSTM-AD [34]($L = L*$)	0.900	0.793	0.859
AD-LTI ($L = L*$)	**0.935**	**0.977**	**0.923**

As shown in Table IV, AD-LTI achieves the highest AUC values of 0.93, 0.977 and 0.923 on CalIt2, Server Log dataset and the Dodgers Loop datasets, respectively. On CalIt2, the AUC values of the baseline algorithms are between 0.8 and 0.9 with the only exception of OCSVM when nu is set to 0.05 - the approximately actual anomaly rate (0.046, precisely) for CalIt2. This to some degree indicates that OCSVM is sensitive to parameters. Anomaly detection is much more challenging on the Server Log dataset due to the increase in the number of channels, and the complexity in seasonality and uncertainty (e.g., channel *SU* is fairly unpredictable). As the result shows, the AUC values for all existing algorithms drop below 0.8 with the best of them, Isolation Forest, reaching 0.761 (with the contamination ratecr set to 0.15), which could float as it is a randomized algorithm. However, the actual contamination rate is hardly a priori knowledge in practical scenarios. We also observed that prediction-driven approaches (LSTM-FD, LSTM-AD and AD-LTI) significantly outperformed others on the Dodgers Loop dataset – this is mainly because of the presence of strong noise in the traffic data. The proposed AD-LTI algorithm makes the most reliable decisions in all of the tested scenarios. The main reasons are two-fold: from one perspective, the underlying backbone model for AD-LTI is very accurate with the complement of seasonal features that effectively captures complex seasonality and mitigates the noise in raw data. From another perspective, AD-LTI is robust in scoring each frame because we leverage multi-source forecasting and weight each prediction based on the confidence of the prediction source.

AD-LTI has an important hyper-parameter L, which determines both the prediction length for the backbone model and the maximum probe length for computing LTI. We evaluated our algorithm against LSTM-AD (which is also based on multiple forecasts) with different L values to investigate the impact of L on detection reliability and time efficiency. The result is summarized in Table V.

From Table V we can see our method outperformed LSTM-AD and also observe different impacts of the probe window length L on different datasets. On CalIt2 and Dodgers, the impact of L on the detection reliability (revealed by AUC) is subtle, while on the Server Log dataset very large L values show obvious negative effect on our scheme. The reasons behind these results are partly because as L becomes bigger (L is set to 20 or above), the prediction made by the backbone model becomes less accurate, and partly because of the dilution of local information. In comparison, LSTM-AD is much more susceptible to the hyper-parameter L. Besides, as expected a longer probe length leads to the increased overhead in detection, which can be mitigated by running the scheme in parallel. Empirically, we recommend setting L to a value between 5 and 20 considering both detection reliability and efficiency.

VI. CONCLUSION

On-line detection of anomalies in time series has been crucial in a broad range of information and control systems that are sensitive to unexpected events. In this paper, we propose an unsupervised, prediction-driven approach to reliably detecting anomalies in time series with complex seasonality. We first present our backbone prediction model, which is composed of a time series decomposition module for seasonal feature extraction, and an inference module implemented using a GRU network. Then we define Local Trend Inconsistency, a novel metric that measures abnormality by weighting local expectations from previous records. We then use a scoring function along with a detection algorithm to convert the LTI value into the probability that indicates a record's likelihood of being anomalous. The whole process can leverage the matrix operations for parallelization. We evaluated the proposed detection algorithm on three different datasets. The result shows that our scheme outperformed several representative anomaly detection schemes commonly used in practice.

In the future we plan to focus on extending our work to address new challenges in large-scale, information-intensive distributed systems such as edge computing and IoT. We aim to refine our method with scenario-oriented designs, for instance, detection in asynchronized streams sent by distributed sensors, and build a robust monitoring mechanism in order to support intelligent decisioning in these types of systems.

ACKNOWLEDGEMENT

This work is partially supported by Worldwide Byte Security Co. LTD, and is supported by National Natural Science Foundation of China (Grant Nos. 61772205, 61872084), Guangdong Science and Technology Department (Grant No. 2017B010126002), Guangzhou Science and Technology Program key projects (Grant Nos. 201802010010, 201807010052, 201902010040 and 201907010001), and the Fundamental Research Funds for the Central Universities, SCUT (Grant No. 2019ZD26).

REFERENCES

[1] Yule, G. U. (1926). Why do we sometimes get nonsense-correlations between Time-Series?–a study in sampling and the nature of time-series. Journal of the royal statistical society, 89(1), 1-63.

[2] Frisch, R., & Waugh, F. V. (1933). Partial time regressions as compared with individual trends. Econometrica: Journal of the Econometric Society, 387-401.

[3] Seiwell, H. R. (1949). The principles of time series analyses applied to ocean wave data. Proceedings of the National Academy of Sciences of the United States of America, 35(9), 518.

TABLE V
AUC VALUES AND DETECTION OVERHEADS (IN MS PER FRAME) USING LSTM-AD AND AD-LTI UNDER DIFFERENT SETTINGS OF PROBE LENGTH L. BOTH METHODS USE MULTIPLE FORECASTS WITH EACH FRAME BEING PREDICTED FOR L TIMES.

Algorithm	L	Calit2 Dataset		Server Log Dataset		Dodgers Loop Dataset	
		AUC	overhead(ms/frame)	AUC	overhead(ms/frame)	AUC	overhead(ms/frame)
LSTM-AD [34]	$L = 5$	0.900	0.126	0.793	0.129	0.859	0.123
	$L = 10$	0.883	0.142	0.753	0.148	0.778	0.142
	$L = 20$	0.847	0.174	0.596	0.183	0.813	0.174
	$L = 30$	0.813	0.206	0.505	0.215	0.815	0.205
AD-LTI	$L = 5$	0.911	0.189	0.977	0.282	0.912	0.196
	$L = 10$	0.912	0.353	0.925	0.399	0.923	0.316
	$L = 20$	0.935	0.706	0.845	0.784	0.906	0.707
	$L = 30$	0.912	1.125	0.784	1.461	0.915	1.110

[4] Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., & Platt, J. C. (2000). Support vector method for novelty detection. Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS'99), pp. 582-588.

[5] Zhang, R., Zhang, S., Lan, Y., & Jiang, J. (2008). Network anomaly detection using one class support vector machine. In Proceedings of the International MultiConference of Engineers and Computer Scientists (Vol. 1).

[6] Maglaras, L. A., & Jiang, J. (2014, August). Ocsvm model combined with k-means recursive clustering for intrusion detection in scada systems. In 10th International conference on heterogeneous networking for quality, reliability, security and robustness (pp. 133-134). IEEE.

[7] Shang, W., Zeng, P., Wan, M., Li, L., & An, P. (2016). Intrusion detection algorithm based on OCSVM in industrial control system. Security and Communication Networks, 9(10), 1040-1049.

[8] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2012). Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data (TKDD), 6(1).

[9] Radovanović, M., Nanopoulos, A., & Ivanović, M. (2014). Reverse nearest neighbors in unsupervised distance-based outlier detection. IEEE transactions on knowledge and data engineering, 27(5), 1369-1382.

[10] Calheiros, R. N., Ramamohanarao, K., Buyya, R., Leckie, C., & Versteeg, S. (2017). On the effectiveness of isolation-based anomaly detection in cloud data centers. Concurrency and Computation: Practice and Experience, 29(2017)e4169. doi: 10.1002/cpe.4169

[11] Chan, P. K., & Mahoney, M. V. (2005, November). Modeling multiple time series for anomaly detection. In Fifth IEEE International Conference on Data Mining (ICDM'05) (pp. 8-pp). IEEE.

[12] Ye, L., & Keogh, E. (2009, June). Time series shapelets: a new primitive for data mining. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 947-956). ACM.

[13] Zakaria, J., Mueen, A., & Keogh, E. (2012, December). Clustering time series using unsupervised-shapelets. In 2012 IEEE 12th International Conference on Data Mining (pp. 785-794). IEEE.

[14] Yeh, C. C. M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H. A., ... & Keogh, E. (2018). Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. Data Mining and Knowledge Discovery, 32(1), 83-123.

[15] Hou, L., Kwok, J. T., & Zurada, J. M. (2016, February). Efficient learning of time series shapelets. In 13th AAAI Conference on Artificial Intelligence.

[16] Gu, Z., He, L., Chang, C., Sun, J., Chen, H., & Huang, C. (2017). Developing an efficient pattern discovery method for CPU utilizations of computers. International Journal of Parallel Programming, 45(4), 853-878.

[17] Zhu, H., Gu, Z., Zhao, H., Chen, K., Li, C. T., & He, L. (2018). Developing a pattern discovery method in time series data and its GPU acceleration. Big Data Mining and Analytics, 1(4), 266-283.

[18] Wei, L., Kumar, N., Lolla, V. N., Keogh, E. J., Lonardi, S., & Chotirat (Ann) Ratanamahatana. (2005, June). Assumption-Free Anomaly Detection in Time Series. In SSDBM (Vol. 5, pp. 237-242).

[19] Huang, T., Zhu, Y., Wu, Y., Bressan, S., & Dobbie, G. (2016). Anomaly detection and identification scheme for VM live migration in cloud infrastructure. Future Generation Computer Systems, 56, 736-745.

[20] Hyndman, R. J., Wang, E., & Laptev, N. (2015, November). Large-scale unusual time series detection. In 2015 IEEE international conference on data mining workshop (ICDMW) (pp. 1616-1619). IEEE.

[21] Li, J., Pedrycz, W., & Jamal, I. (2017). Multivariate time series anomaly detection: A framework of Hidden Markov Models. Applied Soft Computing, 60, 229-240.

[22] Chauhan, Sucheta and Vig, Lovekesh. Anomaly detection in ECG time signals via deep long short-term memory networks. In Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on, pp. 1–7. IEEE, 2015.

[23] Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). LSTM-based encoder-decoder for multi-sensor anomaly detection. arXiv preprint arXiv:1607.00148.

[24] Ahmad, S., Lavin, A., Purdy, S., & Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. Neurocomputing, 262, 134-147.

[25] Pascanu, R., Mikolov, T., & Bengio, Y. (2013, February). On the difficulty of training recurrent neural networks. In International conference on machine learning (pp. 1310-1318).

[26] Tang, X. (2019). Large-Scale Computing Systems Workload Prediction Using Parallel Improved LSTM Neural Network. IEEE Access, 7, 40525-40533.

[27] Chen, S., Li, B., Cao, J., & Mao, B. (2018). Research on Agricultural Environment Prediction Based on Deep Learning. Procedia computer science, 139, 33-40.

[28] Harvey, A. & Peters, S. (1990), Estimation procedures for structural time series models, Journal of Forecasting, Vol. 9, 89-108.

[29] Taylor, S. J., & Letham, B. (2018). Forecasting at scale. The American Statistician, 72(1), 37-45.

[30] Kingma, D. and Ba, J. (2015) Adam: A Method for Stochastic Optimization. Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015).

[31] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.

[32] Vallis, O., Hochenbaum, J., & Kejariwal, A. (2014). A novel technique for long-term anomaly detection in the cloud. In 6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 14).

[33] P. Filonov, A. Lavrentyev, A. Vorontsov, Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model, NIPS Time Series Workshop 2016, Barcelona, Spain, 2016.

[34] Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015, April). Long short term memory networks for anomaly detection in time series. In Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 15'), pp. 89-94.

[35] Shi, H., Yang, J., Ding, M., & Wang, J. (2011). A short-term wind power prediction method based on wavelet decomposition and BP neural network. Automation of Electric Power Systems, 35(16), 44-48.

[36] Gould, P. G., Koehler, A. B., Ord, J. K., Snyder, R. D., Hyndman, R. J., & Vahid-Araghi, F. (2008). Forecasting time series with multiple seasonal patterns. European Journal of Operational Research, 191(1), 207-222.

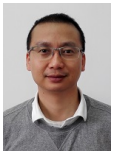
[37] De Livera, A. M., Hyndman, R. J., & Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. Journal of the American Statistical Association, 106(496), 1513-1527.

[38] Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. Artificial intelligence review, 22(2), 85-126.

[39] Janssens, O., Slavkovikj, V., Vervisch, B., Stockman, K., Loccufier, M., Verstockt, S., ... & Van Hoecke, S. (2016). Convolutional neural network based fault detection for rotating machinery. Journal of Sound and Vibration, 377, 331-345.

[40] Ince, T., Kiranyaz, S., Eren, L., Askar, M., & Gabbouj, M. (2016). Real-time motor fault detection by 1-D convolutional neural networks. IEEE Transactions on Industrial Electronics, 63(11), 7067-7075.
[41] Sabokrou, M., Fayyaz, M., Fathy, M., Moayed, Z., & Klette, R. (2018). Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. Computer Vision and Image Understanding, 172, 88-97.
[42] Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. L. (2014, June). Time series classification using multi-channels deep convolutional neural networks. In International Conference on Web-Age Information Management (pp. 298-310). Springer, Cham.
[43] Rajan, J. J., & Rayner, P. J. (1995). Unsupervised time series classification. Signal processing, 46(1), 57-74.
[44] Längkvist, M., Karlsson, L., & Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters, 42, 11-24.
[45] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.
[46] Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. International journal of forecasting, 20(1), 5-10.
[47] Van De Geijn, R. A., & Watts, J. (1997). SUMMA: Scalable universal matrix multiplication algorithm. Concurrency: Practice and Experience, 9(4), 255-274.
[48] Chen, J., Li, K., Deng, Q., Li, K., & Philip, S. Y. (2019). Distributed Deep Learning Model for Intelligent Video Surveillance Systems with Edge Computing. IEEE Transactions on Industrial Informatics.
[49] Chen, J., Li, K., Bilal, K., Metwally, A. A., Li, K., & Yu, P. (2018). Parallel protein community detection in large-scale PPI networks based on multi-source learning. IEEE/ACM transactions on computational biology and bioinformatics.
[50] Chen, J., Li, K., Bilal, K., Li, K., & Philip, S. Y. (2018). A bi-layered parallel training architecture for large-scale convolutional neural networks. IEEE transactions on parallel and distributed systems, 30(5), 965-976.
[51] Duan, M., Li, K., Liao, X., & Li, K. (2017). A parallel multiclassification algorithm for big data using an extreme learning machine. IEEE transactions on neural networks and learning systems, 29(6), 2337-2351.
[52] Chen, C., Li, K., Ouyang, A., Tang, Z., & Li, K. (2017). Gpu-accelerated parallel hierarchical extreme learning machine on flink for big data. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(10), 2740-2753.

Wentai Wu received the Bachelor and Master degrees in computer science from South China University of Technology in 2015 and 2018, respectively. Currently, he is a Ph.D. candidate supervised by Dr. Ligang He with the Department of Computer Science, the University of Warwick, United Kingdom. His research interests mainly include parallel and distributed computing, distributed learning, energy-efficient computing and predictive analytics.



Ligang He is a Reader in the Department of Computer Science at the University of Warwick, United Kingdom. He has published over 130 articles in journals and conferences, such as the IEEE TC, TPDS, TACO, IPDPS, ICPP, SC, VLDB. His research interests focus on parallel and distributed processing, high performance computing, cloud computing and bigdata processing.

Weiwei Lin received his B.S. and M.S. degrees from Nanchang University in 2001 and 2004, respectively, and the PhD degree in Computer Application from South China University of Technology in 2007. Currently, he is a professor in the School of Computer Science and Engineering, South China University of Technology. His research interests include distributed systems, cloud computing, big data computing and AI application technologies. He has published more than 80 papers in refereed journals and conference proceedings.

Yi Su is a PhD student in the Department of Computer Science at the University of Warwick, United Kingdom. Her research interest is in big data analysis.

Yuhua Cui received his M.S. degree from Shandong University, China and presently he serves as the Vice Dean of the Research Institute of Worldwide Information Security. His research interests mainly include computer security, cyber security and their applications.

Carsten Maple is Principal Investigator of the NCSC-EPSRC Academic Centre of Excellence in Cyber Security Research at the University and Professor of Cyber Systems Engineering in WMG. He is a co-investigator of the PETRAS National Centre of Excellence for IoT Systems Cybersecurity where he leads on Transport & Mobility. He has published over 250 peer-reviewed papers and is a co-author of the UK Security Breach Investigations Report 2010, supported by the Serious Organised Crime Agency and the Police Central e-crime Unit. Carsten is also a co-author of Cyberstalking in the UK, a report supported by the Crown Prosecution Service and Network for Surviving Stalking. He has advised executive and non-executive directors of public sector organisations and multibillion pound private organisations.

Stephen Jarvis studied at London, Oxford and Durham Universities before taking his first lectureship at the Oxford University Computing Laboratory. Here he worked on the development of performance tools for the BSP programming library, as well as teaching at Brasenose, Lincoln and Keble Colleges. After a short secondment to Microsoft Research in Cambridge, he joined the University of Warwick, rising to Professor in 2009. Professor Jarvis acted as Director of Research from 2008 to 2013, leading the Department to rank 2nd (out of 89 UK Computing Departments) in the 2014 UK Research Excellence Framework (REF). In 2013 he was appointed Chair of Department. Professor Jarvis has been a Visiting Exchange Professor at New York University since 2017 and is currently a member of the Board of Trustees at the Alan Turing Institute, the UK's national institute for data science. He is presently Deputy Pro Vice Chancellor (Research) at the University of Warwick.