Local Trend Inconsistency: A Prediction-driven Approach to Unsupervised Anomaly Detection in Multi-seasonal Time Series

Wentai Wu, Ligang He, *Member, IEEE*, and Weiwei Lin

Abstract—On-line detection of anomalies in time series is a key technique 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 demands are making this task more challenging than ever. First, the rapid increase of unlabeled data makes supervised learning no longer suitable in many cases. Second, a great portion of time series have complex seasonality features. Third, on-line anomaly detection needs to be fast and reliable. In view of this, we in this paper adopt an unsupervised prediction-driven approach on the basis of a backbone model combining a series decomposition part and an inference part. We then propose a novel metric, Local Trend Inconsistency (LTI), along with a detection algorithm that efficiently computes LTI chronologically along the series and marks each data point with a score indicating its probability of being anomalous. We experimentally evaluated our algorithm on datasets from UCI public repository and a production environment. The result shows that our scheme outperforms several representative anomaly detection algorithms in Area Under Curve (AUC) metric with decent time efficiency.

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

I. INTRODUCTION

While time series data has been ubiquitous before the coming of big data era, a large number of recently emerging technical scenarios like autonomous driving, edge computing and Internet of Things (IoT) pose new challenges to the detection of anomalies in this type of data. In the meantime, detection techniques that can provide early, reliable reports of anomaly has become crucial for a wide range of systems requiring 24/7 monitoring services. In cloud data centers, for example, a distributed monitoring system usually collects a variety of log data from virtual machine level to cluster level on a regular basis and sends them to a central detection module, which needs to analyze the aggregated time series to detect any anomalous events including hardware breakdown, unavailable services and cyber attacks. This requires an on-line detector capable of making reliable detections (i.e., with strong sensitivity and specificity), otherwise it could bring about unnecessary cost of maintenance.

Several branches of schemes have been applied to the problem of time series anomaly detection, and in certain cases decent results can be achieved by these traditional methods

W. Wu and L. He are with the Department of Computer Science, the University of Warwick.

W. Lin is with the School of Computer Science and Engineering, South China University of Technology.

such as outlier detection [1][5][6][7], pattern(segment) extraction [9][10], sequence mapping [15][17][18], etc. However, we are facing a growing number of new scenarios and applications which produce a large amount of time series data of unprecedented complexity, posing challenges that traditional anomaly detection methods cannot well address. First, more and more time series data are constantly produced without labels since data labelling/annotation is usually very time-consuming and costly. Sometimes it is also unrealistic or impossible to acquire reliable labels with correctness guaranteed. Second, some applications may produce multi-channel series with complex features such as multi-period seasonality (i.e., multiple seasonal 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 to generalize. Third, the process is commonly required to be fast enough to support instant reports or warning once unexpected change appears. On-line detection capability is especially important in event-sensitive scenarios like medical and industrial process control systems.

In this paper, we propose a predictive solution to effectively detecting anomalies in multi-channel (multi-variate) time series with complex seasonality. The basic idea is to continuously collect and analyze the forecasts from previous time points in a time series. 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 model. The accuracy of forecasting model is of great significance. We therefore build a frame-to-sequence Gated Recurrent Unit (GRU) network while extending its input with seasonal terms extracted from channel-wise time series decompositions. Based on the predictions of local sequences (i.e., the forecasting model's output), the proposed detection algorithm computes Local Trend Inconsistency (LTI), a metric we define for measuring the deviation of an actual sequence from expectations, at each data point in an on-line manner and marks them with anomaly scores, which quantify their probability of being anomalous.

The main contributions of our work are as follows:

- We first build a frame-to-sequence forecasting model combining time series decomposition (using Prophet, an additive time series model by Facebook) and GRU network to enable fast, contamination-tolerant training on multi-seasonal time series data without (anomaly) labels.
- We propose a robust metric, namely LTI, and a corre-

sponding detection algorithm for measuring the probability that a data record is anomalous as a score. An iterative procedure is also proposed for fitting the scoring function.

- We also mathematically present the computation of LTI in form of matrix operations and prove the possibility of parallelization for speeding up the detecting procedure.
- We Experimentally evaluated AD-LTI on a public dataset from UCI public repository and a more complex dataset from production environment. The result shows that AD-LTI is fast and outperforms all the baseline algorithms in AUC metric.

The rest of the paper is organized as follows: Section 2 introduces a number of relevant studies on anomaly detection. In Section 3, we define Local Trend Inconsistency as the key metric in our time series anomaly detection scheme. Then we systematically present our solution in Section 4, including the backbone model for prediction and a scoring algorithm for anomaly detection. We detail the experimental evaluation and analysis in Section 5, and finally conclude our work in Section 6.

II. RELATED WORK

The term anomaly refers to a data point that significantly deviates from the rest of the data, which is supposed to follow some distribution or patterns. There are mainly two categories of approaches to anomaly detection: novelty detection and outlier detection. While novelty detection (e.g. classification methods [36][37][38][39]) requires absolutely no pollution in the training data, outlier detection (e.g., clustering, principal component analysis [17] and feature mapping methods [40][41]) does not need a prior knowledge of classes (i.e., labels) and thus is also known as unsupervised anomaly detection. The definitions may vary in different places, for example, we use the same taxonomy as Ahmed et al. [42] whilst in the survey by Hodge and Austin [35] unsupervised detection is classified as a subtype of outlier detection. The focus of our work is on unsupervised anomaly detection because we aim to design a more generic scheme by assuming 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 at the level of a single record or as a subsequence/pattern. Many classical algorithms can be applied to detecting single-record anomaly as an outlier. One Class Support Vector Machine (OCSVM) [1] is a variant of SVM and exploits a hyperplane to separate normal and anomalous data points. Zhang et al. [2] implemented a network performance anomaly detector using OCSVM with Radial Basic Function (RBF), which is a commonly used kernel for SVM. Maglaras and Jiang [3] developed an intrusion detection module for based on K-OCSVM, an algorithm that iteratively performs K-means clustering on anomalies detected by OCSVM to optimize the result. Shang et al. [4] applies Particle Swarm Optimization (PSO) to find the optimal parameters for OCSVM, which is later used to detect abnormal TCP traffic. Several algorithms are developed to discover outliers in the original feature space. Radovanović

et al. [6] investigated the correlation between hub points and outliers, providing a useful guidance on using reverse nearest-neighbor counts to detect anomalies. Liu et al. [5] 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. [7] 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 [8], symbolic sequence matching [15] and pattern extraction [11][12]). For example, Huang et al. [16] proposed a scheme to identify anomalies in VM live migrations by combining the extended Local Outlier Factor (LOF) and Symbolic Aggregate ApproXimation (SAX).

Prediction-driven schemes have been attracting more attention because of the remarkable performance by recurrent neural networks in prediction/forecasting tasks. Filonov et al. [31] proposed a fault detection framework that relies on a Long Short Term Memory (LSTM) network with two stacked (hidden) layers 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 [20] and [22]. Malhotra et al. [21] adopt a different architecture named encoder-decoder, which is built on an idea 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 figured out by Pascanu et al. [23], recurrent neural networks may be struggling to learn the complex seasonal patterns in time series particularly when some channels of the series have long periodicity while the others have weak seasonality. A possible solution to that is decomposing the series before feeding into the network. Shi et al. [32] proposed a wavelet-BP neural network model for predicting wind power. They decompose the input time series into frequency components using wavelet transform and build a prediction network for each of them. To forecast time series with complex seasonality, De Livera et al. [34] adopt an innovations state space modeling framework that incorporates seasonal decomposition methods like Fourier representation. A similar model was implemented by Gould et al. [33] to fit hourly and daily patterns in utility loads and traffic flows data.

Low overhead is essential for real-time anomaly detection. For example, Gu et al. [13] propose an efficient motif (frequently repeated patterns) discovery framework incorporating an improved SAX indexing method as well as a trivial match skipping algorithm. Their experiment result on CPU host load series shows its excellent time efficiency. Zhu et al. [14] propose a new method for locating similar sub-sequences as well as a parallel approach using GPUs to accelerate Dynamic Time Warping (DTW) for time series pattern discovery.

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 - *Local Trend Inconsistency* (*LTI*).

In some systems, more than one data collection device is deployed to simultaneously gather the information of multiple variables, 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.

We also define frames for more convenient references to the records in time series data. The concept of frame we use in this paper is different from but more general than that in video processing as videos can be reckoned as time series of images.

Definition 2: A frame is the data record at a certain point of time in a series. A frame is a vector for multi-channel time series, or a scalar otherwise.

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

Definition 3: A local sequence is a fragment of the target time series, and local sequence at frame x is a fragment of series spanning from a previous frame to frame x.

For clarity, we list all the symbols that may be 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 value of Euclidean Distance largely depends on the dimension, i.e., vector length, whilst it is inefficient to compute DTW Distance concerning time complexity ($O(n^2)$). Therefore, we in this paper propose Dimension-independent Frame Distance ($DFDist$) to measure the distance between two frames **x** and **y**:

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

where m is the number of dimensions (i.e., number of channels), x_i and y_i are the ith component of frame x and frame y, respectively. We do not square root the result for the simplicity when we turn all the computation into matrix operations. For normalized data, the desired scale (i.e., $DFDist \in [0, 1]$) still holds.

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

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

where L is the length of the two local sequences. The definition of $LSDist$ already provides an unified scale of distance but the value of temporal information that naturally exists in time series is neglected. Assume we are detecting anomalous event at time t and 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 $LSDist$, every time point is logically regarded equally important, which violates the rule of time decay. Therefore, we modify $LSDist$ by weighting each term and adding a normalization factor. The Weighted Length-independent Sequence Distance ($WLSDist$) is defined as (3):

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

where d_i is the weight of time decay for frame i and D_L is the normalization factor. Time decay is applied on the basis that the two sequences are chronologically aligned. We in this paper choose an exponential decay model:

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

where i is the temporal distance from current frame plus 1, $d_L = 1$ is the weight for the current frame (as it is the last frame in its local sequence). Hence, the corresponding normalization factor D_L in (4) is the summation of an geometric series of length L:

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

where L is sequence length.

Ideally it is easy to find out anomalies through calculating $WLSDist$ with ground truth, but it is basically impossible to do that when the data has complex patterns and labels are unavailable. A solution is to replace ground truth with expectation, which is usually implemented using time series forecasting methods [20][19]. However, a critical problem for prediction-driven anomaly detection is the reliability of forecasts. On the one hand, prediction error is inevitable. On the other, anomalous frames probably make predictions that are extremely unreliable. To this end, we choose to establish a reliable prediction scheme by taking advantage of multi-source forecasting. Different from previous studies that use frame-to-frame predictor, we base our detection on a series of forecasts at target frame by building a frame-to-sequence prediction network. The resulting forecast collection consists of all the predictions of local sequences at this frame made by previous frames. In other words, we collect a group of expectations.

In order to measure how significantly the time series deviates from the group of "expectations" locally at the current frame, we propose Local Trend Inconsistency (LTI), a robust metric that weights every prediction with the probability that its maker (i.e., a previous frame) is normal. For frame t, $LTI(t)$ is formally defined as:

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

where $S(i+1,t)$ is the actual local sequence from frame $i+1$ to frame t, and $S_i(i+1,t)$ is the predicted one by frame i with the same span. L is the probe length, a parameter that decides both the maximum length and number of local sequence expectations available. $P(i)$ denotes the probability that frame i is normal. Z_t is the normalization factor for frame t defined as the sum of all the probabilistic weights:

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

It is notable that the predicted local sequences ($S_i(i+1,t)$) vary in length depending on the timestamps of their predictors. The rationale behind the expression of LTI is to compute a weighted sum of distances between the latest fragment of the target series and a group of local predictions of sequences stretching to the latest frame. Fig. 1 demonstrates how we calculate LTI in a case where $L = 3$.
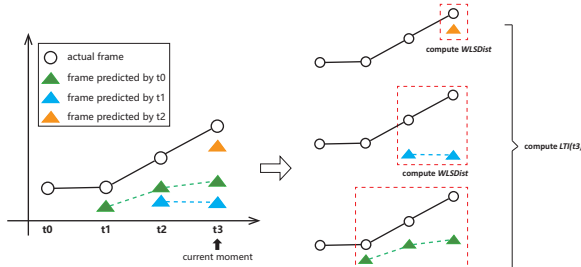

Fig. 1. An example demonstrating the calculation of Local Trend Inconsistency (max probe length equal to 3

Assume we are detecting anomaly at frame $t3$ (Fig. 1), in this case $t0$, $t1$ and $t2$ have made their predictions apiece marked by green, blue and orange triangles, respectively. Then we compute the weighted distance ($WSLDist$) between each predicted local sequence and its corresponding actual fragment of target series (white circles). Finally we get $LTI(t3)$ via (6) by applying probabilistic weights to the three resulting distances.

The whole process can be formulated using matrix operations. Assume we are detecting anomaly at frame t and the max length of prediction is L. For brevity let $df_k(t)$ denote the distance between frame t and a forecast of it by frame 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 brevity 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 (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 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 as below:

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

By using matrices to formulate LTI, we claim that the calculation can be highly parallel and the Degree of Parallelism (DoP) for this process is at least but not limited to L. The reason is that calculating all terms in (6) in parallel can apparently achieve a DoP of L, while at the same time we can further speed up the computation of each term of distance (including $WLSDist$ and $DFDist$) by taking advantage of parallel matrix multiplication if multiple processors are available.

IV. ANOMALY DETECTION WITH LTI

We base our anomaly detection scheme on Local Trend Inconsistency as it well indicates how significantly the series deviates locally from what it is expected to be. From (6), we see two problems still to be resolved in calculating LTI. First, a mechanism is needed to make reliable predictions of local sequences. Second, we need to a concrete algorithm to quantify the probabilistic factors (in matrix $\mathbf{P}$) as it is not a priori.

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 at a low training cost. Then we propose an anomaly scoring algorithm that uses a scoring function to chronologically calculate anomaly probability for each frame based on Local Trend Inconsistency.

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 part and a inference part.

Recurrent Neural Network (RNN) is an ideal implementation of the inference part of our model. RNNs (including mutations like Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), etc.) are usually applied as end-to-end models (e.g., [24] [25]). However, a major limitation of them is the difficulty in learning complex seasonal patterns in multi-seasonal time series. Even though accuracy may be improved by stacking more hidden layers and increasing back propagation distance (through time) during training, yet it could bring about the increase of training cost.

In view of this, we propose to explicitly add seasonal features to neural network's input through conducting time series decomposition beforehand, which is implemented in the decomposition part. The resulting seasonality-related 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* [27], a framework based on decomposable time series model [26], to produce channel-wise seasonal terms extraction. Let X_c denote the cth channel of time series X, and $X_c(t)$ the tth 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 component that models that non-periodic changes, $s_c(t)$ represents the seasonal term that quantifies seasonal effects and is the very term we want. $s_c(t)$ reflects the effects of 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 part of our model. *Prophet* relies on Fourier series to model multi-period seasonality, which enables flexible approximation to any periodic patterns of arbitrary length. The underlying details can be referred to [27].

Separating seasonal terms from original frame values as additional features effectively improves RNN in several aspects. First, Explicit input of seasonal terms helps reduce the difficulty for RNN in learning complex seasonal terms. This is intuitive because seasonal terms numerically quantify seasonal effects. Second, Training cost of time is also expected to decrease as we can apply Truncated Back Propagation Through Time (TBPTT) with a distance much shorter than the length of periodicity. Besides, it is notable that series decomposition itself is very efficient, which will be experimentally proved later. Figure 2 shows an overview of our backbone model, where a stacked GRU network is implemented as the inference part.
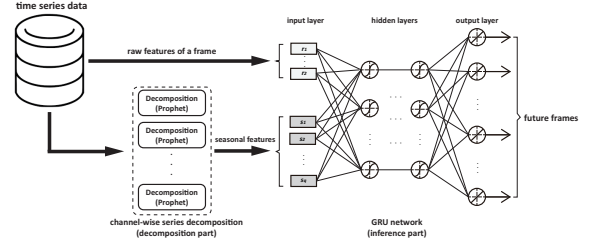

Fig. 2. The proposed prediction model comprised of channel-wise series decompositions and a GRU network

B. Anomaly Scoring

As mentioned, we need appropriate approximations to probabilistic factors $P(t-L), P(t-L+1), ..., P(t-1)$ in order to compute $LTI(t)$ at frame t according to (6). Intuitively, the value of $LTI(t)$ is positively correlated to the likelihood that frame t is an anomaly. Thus what we need is a mapping function that converts $LTI(t)$ to a probabilistic value. We define a logistic mapping function Φ as:

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

where k is the logistic growth rate and x_0 the x-value of the function's midpoint. These two parameters will be fixed automatically using an iterative algorithm introduced later.

We then define the probabilistic anomaly score of frame t as below:

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

Replacing $P(i)$ in (6) with $AS(i)$ where $i = t-L, t-L+1, ..., t-1$, we can re-formulate $LTI(t)$ as:

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

where Z_t here is the normalization factor reformulated as $\sum_{i=t-L}^{t-1} AS(i)$. L is the hyper-parameter of probe length. To unparameterize function $\Phi(\cdot)$, we design an iterative algorithm that runs on a reference time series which is a part of the training data. Here we also assume that our backbone model (Fig. 2) is already trained and thus all the local sequence predictions are available on the reference series. The algorithm is described 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 (12)

        compute $AS(t)$ via (11)

    end

    $k \leftarrow c \cdot \text{stdev}(LTI)$, $x_0 \leftarrow \text{mean}(LTI)$

end

In Algorithm 1, parameters k and x_0 are initially set to 1.0 and 0.5 and we want them to converge to the standard deviation (with a constant multiplier c) and the mean of the LTI values of all the frames in the reference series, respectively. We set a convergence criterion that both k and x_0 change by less than 0.1% since last update. In each loop, the algorithm compute $LTI(t)$ and $AS(t)$ along the whole reference series for each frame t, Parameters k and x_0 remain the same within each loop. After each loop we update them with the recalculated standard deviation (stdev) multiplied by c and the mean of LTI values of all the frames.

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

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 $AS(t)$

The information required for detection at frame t includes frame t itself, anomaly scores of previous frames and, most importantly, the predicted local sequences ending at t as the output of our backbone model. To analyze Algorithm (2)'s time complexity, let m denote the number of dimensions of a frame and L the prediction span, which is a hyper-parameter shared by the backbone model and the detection algorithm. Since it takes $O(m)$ to calculate $DFDist$ between each pair of frames, 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 $O(L^2 m)$, which is caused by our multi-source forecasting mechanism (see (12)). It is also notable that a minimum of $L\times$ speed-up can be achieved because the terms in (12) can be computed in parallel.

V. EXPERIMENTS

In this section, we first evaluate our backbone prediction model for analyzing the impact of seasonal terms as additional features. Then we present empirical results of comparing AD-LTI with several anomaly detection algorithms in sensitivity and speciality (using AUC metric).

We set up our experiments on a PC 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 part of our backbone model is implemented on Pytorch(version: 1.0.1) platform and the decomposition part is realized using Prophet(version: 0.4) released by Facebook. The entire project is coded in Python 3.6. We select two different datasets for evaluation. The first one, CalIt2, is a public dataset published by University of California Irving and available in the UCI machine learning repository. Another dataset we use in our experiment is from the production environment of a cyber security company which collects server logs from a number of clusters (owned by third-party enterprises) on a regular basis. We call it Server Log dataset.

CalIt2 Dataset

CalIt2 (Fig. 3) 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. With timestamp available, the original data spans across 15 weeks (2520 hours) and is half-hourly aggregated. We cut the last 120 hours and conduct a simple processing on the rest to make it hourly-aggregated because a 24-frame-long daily seasonality is more intuitive. The CalIt2 dataset is provided with annotations that label the date, start time as well as end time of events over the whole 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 an multi-channel, hourly-interval time series starting 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. Totally five different processes are recorded including CROND, RSYSLOGD, SESSION, SSHD and SU. Fig. 4 shows all the channels in sub-plots. We pre-process the data by aggregating all the files to form a five-channel time series. Anomalous events like external cyber attacks exist across the whole time series but labels are not available. We acquire manual annotations for the test set from engineers with expertise. Totally 76 frames are labelled as
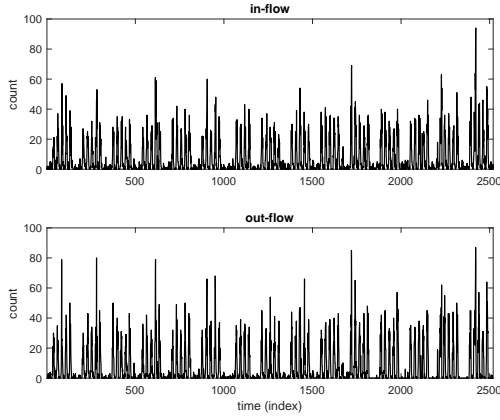
Fig. 3. The CalIt2 time series dataset

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

anomalies in the test set, equivalent to a contamination rate of 14.6%.
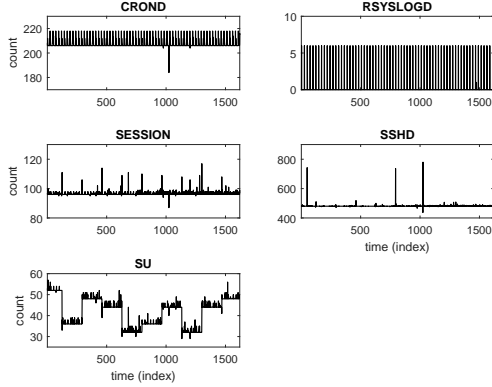

Fig. 4. The server log time series dataset

A. Evaluating Backbone Model

We trained our prediction model on the two datasets separately to evaluate its accuracy as well as the impact of seasonal terms from the decomposition part. We split both datasets into training, validation and test sets. For CalIt2, the first 1900 frames are used for training and the following 500 for testing. For Server Log set, 1100 frames for training and 520 for testing. A same size of 300 is set for validation.

The proposed model use *Prophet* as the decomposition part and a stacked GRU network as the prediction part. Two levels of seasonality (i.e., Daily and weekly) are enabled for *Prophet*, from the output of which we extract daily and weekly terms channel by channel. More specifically, for each channel we obtain two mapping lists from *Prophet* after fitting the data: the first list consists of 24 daily seasonal terms corresponding to the 24 hours in a day, while the second list has 7 terms corresponding to the 7 days in a week. Fig. 5 shows an example of the mapping lists.

The values of seasonal terms are different for CalIt2 and Server Log datasets, but the resulting mapping lists share the same format with the example shown in Fig. 5.
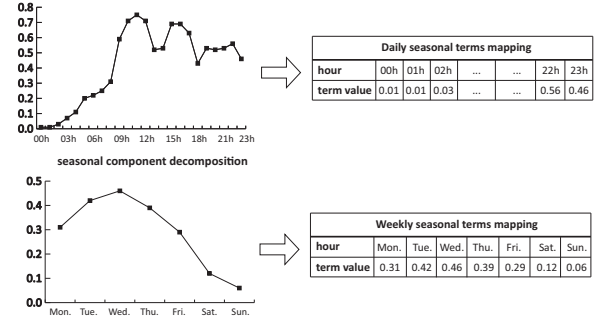
Based on the mapping lists and the timestamp field provided in the data we build our prediction network with seasonal features as additional input. Table II shows the network structures adopted for CalIt2 and Server Log datasets, where L is the maximum length of local sequences as a hyper-parameter. $tanh$ is used as the activation function and Mean Square Error (MSE) loss as the loss function. Dropout is not enabled but we set to a weight decay of $6e-6$ during training to prevent over-fitting. We use Adam [28] as the optimizer with initial learning rate set to 0.001.

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	$[6, 20 \times 2, 2L]$
Server log	GRU	5+10	$[15, 20 \times 3, 5L]$

For comparison we also implemented a baseline GRU network with the same structure and hyper-parameters as our inference part except that 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 more probable that gradients vanish/explode in back propagation, which may result in an unlearnable model. 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. Prediction span L is fixed to 5 (hours). The results are summarized in Table III.

TABLE III
COMPARING ORIGINAL GRU NETWORK TO THE PROPOSED ONE IN
DECOMPOSITION TIME, TRAINING TIME AND TEST LOST UNDER
DIFFERENT TRAINING SETTINGS OF $time\_steps$ (ts)

	Calit2 Dataset		Server Log Dataset	
	GRU+ST	GRU	GRU+ST	GRU
decomp. time	2.7s	-	6.6s	-
training time (ts=24)	173.7s	176.7s	460.4s	464.9s
training time (ts=72)	180.2s	185.6s	468.3s	436.9s
training time (ts=168)	169.8s	174.5s	421.0s	446.5s
MSE loss (ts=24)	0.0068	0.0092	0.0020	0.0039
MSE loss (ts=72)	0.0066	0.0089	0.0013	0.0020
MSE loss (ts=168)	0.0067	0.0085	0.0018	0.0033

In Table III, the decomposition time and training time refer to the fitting/training time cost by the decomposition part and

the inference part, respectively. MSE loss is calculated on normalized test data. From the results we can first observe that our model's decomposition part only needs a few seconds to extract seasonal terms from all the channels. More importantly, we find that our inference network (GRU+ST) outperforms its raw-featured counterpart with a remarkable reduction of MSE loss in each case where *time_steps* is set to 24 (daily seasonality length), 72 or 168 (weekly seasonality length), respectively. The accuracy promotion is above 20 percent on CalIt2 while reaching from 35 to nearly 50 percent on Server Log dataset. These observations demonstrate that the proposed combinatorial model effectively improves prediction accuracy without significantly increasing training cost.

B. Evaluating AD-LTI

In this section we evaluate our unsupervised anomaly detection algorithm AD-LTI. We also implement a number of representative algorithms for the purpose of comparison. These baseline algorithms include One Class Support Vector Machine (OCSVM) [1], Isolation Forest (iForest) [5], Piecewise Median Anomaly Detection [30] and LSTM-based Fault Detection (LSTM-FD) [31].

OCSVM is a mutation of SVM for unsupervised outlier detection. OCSVM shares the same theoretical basis with 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 contamination rate. Both OCSVM and Isolation Forest are embedded in the Scikit-learn package [29]. 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.

We use AUC metric to quantify 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 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 (a anomaly detector in our case). The concept of ROC and AUC can well 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 detection results in the form of probability. Hence metrics like precision and recall cannot be calculated unless we consider threshold as an extra parameter, which violates our generic design purpose.

We evaluate AD-LTI and the baseline algorithms on both datasets. Parameters for OCSVM, iForest, Piecewise Median Anomaly Detection and LSTM-FD are set to default or as per the original papers if suggested. For AD-LTI, *time_steps* is set to 72 (hours) and we fix L at 5.

As shown in Fig. 6 and Fig. 7, we draw two groups of 1-D heatmaps to compare the detection results by each algorithm 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. Fig. 8 and Fig. 9 show their ROC curves on the test series from CalIt2 and the Server Log dataset, respectively.
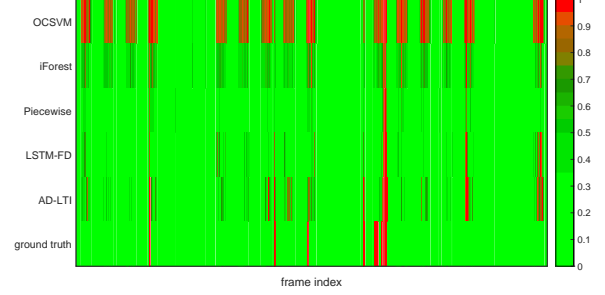

Fig. 6. Heatmaps of decision results by anomaly detection algorithms and the ground truth on CalIt2 dataset
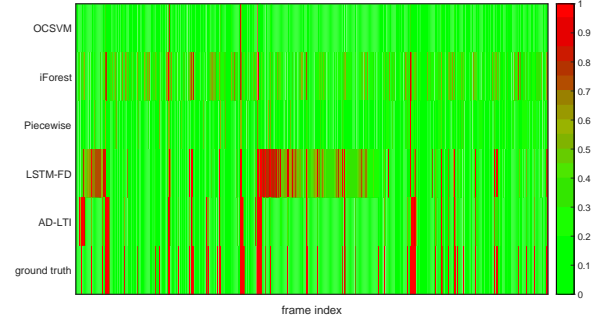

Fig. 7. Heatmaps of decision results by anomaly detection algorithms and the ground truth on Server Log dataset
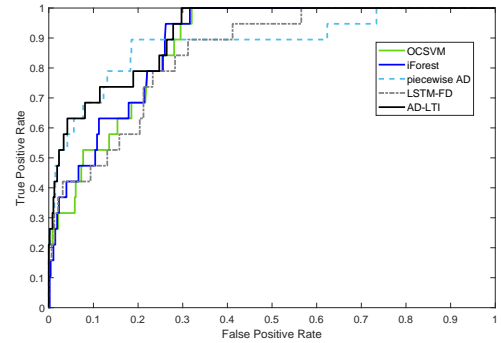

Fig. 8. ROC curves of anomaly detection algorithms on Calit2 dataset

From the ROC curves we can observe that AD-LTI produces the most reliable decisions as its curve is the closest to the top-left corner in Fig. 8. AD-LTI shows a even better characteristic curve in Fig. 9 even though the detection difficulty is higher on the Server Log dataset - none of other algorithms achieve high true positive rate at a low false positive rate. We further
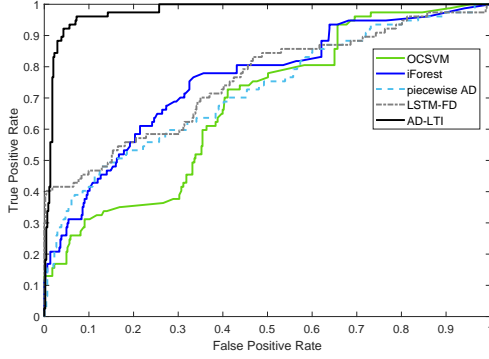
Fig. 9. ROC curves of anomaly detection algorithms on Server Log dataset

calculate the corresponding AUC for each algorithm on both datasets. The resulting AUC values are shown in Table IV.

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

	CalIt2 (CR=0.05)	Server Log (CR=0.15)
OCSVM (default)	0.876	0.677
OCSVM (nu = CR)	0.708	0.672
iForest(default)	0.891	0.756
iForest (cr = CR)	0.877	0.761
Piecewise AD	0.833	0.721
LSTM-FD	0.847	0.755
AD-LTI	**0.915**	**0.975**

As shown in Table IV, AD-LTI achieves the highest AUC values of 0.915 and 0.975 on CalIt2 and the Server Log dataset, respectively. On CalIt2, the AUC values of baseline algorithms are basically 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 of channels, complexity in seasonality and uncertainty (e.g., channel SU is fairly unpredictable). As a result, the AUC values for all the baseline algorithms drop below 0.8 with the best of them, Isolation Forest, reaching 0.761 (with cr set to actual contamination rate at 0.15), which could float as it is a randomized algorithm. However, actual contamination rate is hardly a priori knowledge in practical scenarios. The proposed AD-LTI algorithm makes the most reliable decisions on both series. The main reasons are two-fold: for one thing, the underlying backbone model for AD-LTI is very accurate with the complement of seasonal features. For another, AD-LTI is robust in scoring each frame because we leverage multi-source forecasting as well as weight each prediction based on the confidence on its source.

AD-LTI has an important hyper-parameter L, which decides both the prediction length for the backbone model and the maximum probe length for computing LTI. We reset our algorithm (including re-training of the inference network) with different L values to study its impact on detection reliability

and time efficiency. The result is summarized in Table V.

TABLE V
AUC VALUES AND TIME COST (IN DETECTION) OF AD-LTI UNDER DIFFERENT SETTINGS OF PROBE LENGTH L

	Calit2 Dataset		Server Log Dataset	
	AUC	overhead(ms)	AUC	overhead(ms)
$L = 3$	0.872	0.152	0.967	0.272
$L = 5$	0.911	0.189	0.977	0.282
$L = 10$	0.912	0.353	0.925	0.399
$L = 20$	0.935	0.706	0.845	0.784
$L = 30$	0.912	1.125	0.784	1.461
$L = 50$	0.910	2.655	0.676	3.305
$L = 100$	0.893	8.597	0.643	11.460

L determines the number and length of expectations (from previous frames) we need to consider for detecting the single frame at current moment. From Table V we can see fairly different impacts from this parameter on the two datasets. For CalIt2, its impact on detection reliability (revealed by AUC) is subtle, while on Server Log dataset large L values show an obviously negative effect. The reasons behind are mainly about the decline in prediction accuracy made by the backbone model and the dilution of local information when we set L to 20 or above on Server Log dataset. Besides, a longer probe length leads to increased overhead in detection as expected. Empirically speaking, it is suggested to set L to a value between 5 and 20 considering both detection reliability and efficiency.

VI. ACKNOWLEDGEMENT

VII. 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 multi-channel, multi-seasonal time series. We first present our backbone prediction model composed of a series decomposition part for seasonal feature extraction, and an inference part implemented using GRU network. Then we define Local Trend Inconsistency, a novel metric that measures abnormality by weighing local expectations from previous records. We then use a scoring function along with a detection algorithm to convert LTI value into probability that indicates a record's likelihood of being anomalous. The whole process can leverage matrix operations for parallelization. We empirically evaluated the proposed detection algorithm on CalIt2 (from the public repository of UCI) and Server Log (from a production environment) datasets. The result shows that our algorithm outperforms several representative anomaly detection schemes in accuracy.

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 kinds of systems.

REFERENCES

[1] Schlkopf, 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.

[2] 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).

[3] 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.

[4] 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.

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

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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).

[16] 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.

[17] 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.

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

[19] 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. 17. IEEE, 2015.

[20] Malhotra, Pankaj, Vig, Lovekesh, Shroff, Gautam, and Agarwal, Puneet. Long short term memory networks for anomaly detection in time series. In ESANN, 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2015.

[21] 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.

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

[23] 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).

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

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

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

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

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

[29] 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.

[30] 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).

[31] 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.

[32] 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.

[33] 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.

[34] 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.

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

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] Rajan, J. J., & Rayner, P. J. (1995). Unsupervised time series classification. Signal processing, 46(1), 57-74.

[41] Lngkvist, 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.

[42] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.