FluxEV: A Fast and Effective Unsupervised Framework for Time-Series Anomaly Detection

Jia Li
The Hong Kong University of Science and Technology
Hong Kong SAR, China
jlidw@cse.ust.hk

Shimin Di
The Hong Kong University of Science and Technology
Hong Kong SAR, China
sdiaa@cse.ust.hk

Yanyan Shen*
Shanghai Jiao Tong University
Shanghai, China
shenyy@sjtu.edu.cn

Lei Chen
The Hong Kong University of Science and Technology
Hong Kong SAR, China
leichen@cse.ust.hk

ABSTRACT

Anomaly detection in time series is a research area of increasing importance. In order to safeguard the availability and stability of services, large companies need to monitor various time-series data to detect anomalies in real time for troubleshooting, thereby reducing potential economic losses. However, in many practical applications, time-series anomaly detection is still an intractable problem due to the huge amount of data, complex data patterns, and limited computational resources. SPOT is an efficient streaming algorithm for anomaly detection, but it is only sensitive to extreme values in the whole data distribution. In this paper, we propose *FluxEV*, a fast and effective unsupervised anomaly detection framework. By converting the non-extreme anomalies to extreme values, our framework addresses the limitation of SPOT and achieves a huge improvement in the detection accuracy. Moreover, Method of Moments is adopted to speed up the parameter estimation in the automatic thresholding. Extensive experiments show that *FluxEV* greatly outperforms the state-of-the-art baselines on two large public datasets while ensuring high efficiency.

CCS CONCEPTS

• **Computing methodologies** → **Anomaly detection**; • **Mathematics of computing** → *Time series analysis.*

KEYWORDS

Anomaly detection; time series; unsupervised learning

ACM Reference Format:
Jia Li, Shimin Di, Yanyan Shen, and Lei Chen. 2021. FluxEV: A Fast and Effective Unsupervised Framework for Time-Series Anomaly Detection. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM '21), March 8–12, 2021, Virtual Event, Israel.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3437963.3441823

*Corresponding author.

WSDM '21, March 8–12, 2021, Virtual Event, Israel

ACM ISBN 978-1-4503-8297-7/21/03…$15.00
https://doi.org/10.1145/3437963.3441823

1 INTRODUCTION

Time series data, a series of data points in temporal order, is very common and plays an important role in many real-world applications [14, 15]. Detecting anomalies over time series data is critical in ensuring the availability, reliability, and security in practical scenes, such as industry machines [25], spacecraft [16, 19], financial transactions [17, 30, 34], key performance indicators (KPIs) [12, 27, 33].

With the development of the Internet, online services have become more and more popular. To ensure the stability and availability of online services, large companies usually need to monitor a large amount of time-series metrics (e.g. page views, transaction volume, success rate, system delay) in real time. Anomaly detection aims to identify unexpected items or events in these data streams, in order to reduce the chance of economic loss. However, providing anomaly detection service on online platforms is a non-trivial task due to the following two challenging factors: large data volume and timely response. First, hundreds of thousands or even millions of business-related time series are generated every minute. Extracting features from such large amounts of data can be very difficult for any complex anomaly detection models. Needless to say, such an amount of data has not been labeled and is not likely to be labeled manually. Moreover, online time-series anomaly detection requires a detection algorithm to respond fast since every second of delay may cause severe consequences. Hence the models with high time complexity [1–3] are insufficient, though they are shown to provide outstanding accuracy. Over the past years, a variety of works [8, 10, 11, 21–24, 26, 27, 29, 30, 32, 33, 35] were proposed for time-series anomaly detection task. However, these algorithms suffer from one or more of the following problems: lack of labels, unsatisfying performance, reliance on empirical parameter tuning, time-consuming, retraining need, and cold-start issue (will be further discussed in Section 2).

Among the existing works, SPOT [30] achieves outstanding performance in detecting extreme values without any label-related information, which has the potential to solve the anomaly detection problem in large-scale time series. Meanwhile, it can be used as an automatic thresholding block that can provide strong statistical guarantees and adapt to the changes in data stream. However, the effectiveness and efficiency of SPOT are still unsatisfactory. On

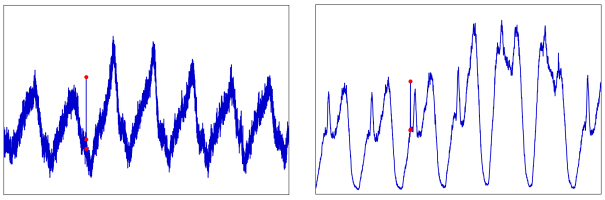


Figure 1: Examples of non-extreme anomalies. Anomalies are marked by red dots.

the one hand, SPOT is only sensitive to the extreme values in the entire data distribution. Unfortunately, in many real-life cases, the anomaly detection task is not only to discover extreme values in the entire data distribution, but also the fluctuations that do not fit the normal pattern. As shown in Figure 1, the marked anomalies in red color are within the normal range of values but are fluctuations that do not meet the normal periodic patterns. In reality, such non-extreme abnormal fluctuations are very common, and SPOT fails to handle such cases. On the other hand, the efficiency of SPOT is not as good as expected because it performs complex calculations using Maximum Likelihood Estimation (MLE) [9], which brings high computational overhead.

In this paper, we aim to develop a more efficient and effective solution to handle time-series anomalies in real-world online services. Inspired by the superiority of SPOT in detecting extreme values, we propose an intuitive idea: if we can extract appropriate features to indicate the degree of abnormality (in a stationary distribution), and make the features of anomalies as extreme as possible, then the problem of detecting abnormal fluctuations can be solved well. Besides, we hope to use a faster estimation method to replace MLE, so as to improve the efficiency of SPOT. Thus, in this work, we develop *FluxEV*, a simple, fast, and effective unsupervised anomaly detection framework. Specifically, we first utilize a predictor to process time series and extract fluctuations. Then we propose a simple but effective two-step smoothing method to eliminate the noises of fluctuations and the effect of periodic patterns. Finally, we leverage SPOT to set the thresholds automatically to make a decision. Furthermore, under the premise of ensuring accuracy, we adopt Method of Moments (MOM) [9] as the parameter estimation method to improve the efficiency of automatic thresholding.

The contributions of our work are summarized as follows:

- We propose a fast and efficient anomaly detection scheme focusing on fluctuation features. It is an unsupervised anomaly detection scheme for time-series data, which can handle non-extreme fluctuation anomalies involved in periodic patterns.
- We use fluctuations as the breakthrough point. For non-extreme abnormal fluctuations involving periodicity, we design a simple and effective two-step smoothing to eliminate the noises of fluctuations and the effect of periodic patterns, so that to retain potential abnormal fluctuations.
- With the help of MOM, we improve the efficiency of our anomaly detection framework by 4-6 times.
- We conduct extensive experiments on two popular public datasets. The experiment results show that our method has high efficiency, and outperforms the current state-of-the-art methods.

The rest of this paper is organized as follows. We introduce related works about time-series anomaly detection in Section 2. The problem analysis is described in Section 3. The methodology is introduced in Section 4. We analyze the experimental results in Section 5. Finally, we conclude our work in Section 6.

2 RELATED WORK

Basically, time-series anomaly detection algorithms can be classified into three categories: supervised, unsupervised, and statistical approaches. Supervised approaches usually integrate multiple detectors or detection algorithms, relying on manual labels to learn how to distinguish anomalies, such as Opprentice [23] and EGADS [21]. Despite showing promising results, in practice, such supervised approaches are not feasible due to a lack of sufficient labels for training. In recent years, some deep generative models have aroused more attention for unsupervised anomaly detection, such as variational auto-encoders (VAEs) [8], DONUT [33], and SeqVL [11]. However, such complex models are relatively time-consuming and often rely on empirical parameter tuning. Moreover, data patterns may vary over time due to some realistic reasons (e.g. seasonality, product upgrading, policy changes), which make existing models no longer applicable and lead to high false-positive rates or false-negative rates [31]. In order to maintain satisfactory accuracy, deep learning networks must be regularly retrained to fit new data patterns [37], which is quite time-consuming and not economical. Taking into account the efficiency issue and computing resources, some traditional statistical models [10, 22, 24, 26, 29, 32, 35] are still used in practical applications. Although statistical approaches are simple to implement and not require on labels, they may not perform well without expert efforts.

In recent years, some novel methods have been proposed for anomaly detection. In 2017, two advanced approaches SPOT and DSPOT [30] are proposed to detect outliers in streaming data, which is efficient and only needs a few data points (according to [30], around 1000 points) for initialization. It can set thresholds automatically with strong statistical guarantees. However, SPOT is only sensitive to extreme values in data distribution, while DSPOT can handle simple drifting cases by a moving average.

Microsoft proposed SR and SR-CNN [27] for industrial anomaly detection services. The authors borrow Spectral Residual (SR) model from the visual domain to solve the time-series anomaly detection task. Compared to SR, SR-CNN achieves better accuracy with a latency increase. Besides, although SR-CNN performs well, it requires a large amount of anomaly-free data for training (according to [27], extra 65 million points are used). However, in real-world scenarios, it is not easy to warm start the SR-CNN since it is quite challenging for finding proper data to pre-train the model.

3 PRELIMINARIES

In this section, we first give the problem definition. Then we analyze the anomaly detection task in online service scenarios and highlight

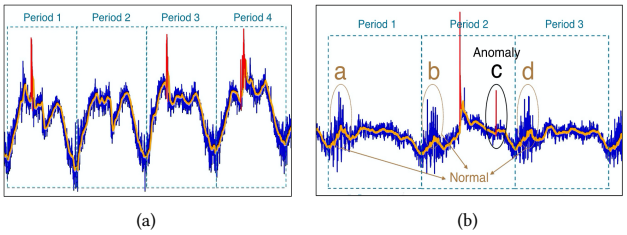


Figure 2: Examples of local fluctuations and periodic pattern.

two key points, i.e. *Local Fluctuation* and *Periodic Pattern*, from the perspective of anomaly labeling.

3.1 Problem Statement

For the sake of brevity, we use the notations as follows. Given arbitrary time-series data $\boldsymbol{X} = [X_1, \cdots, X_n]$, where $X_i \in \mathbb{R}$, X_i is the data point at time i, $\boldsymbol{X}$ denotes this whole data array, and $\boldsymbol{X}_{i,j} = [X_i, X_{i+1}, .., X_{j-1}, X_j]$ represents a window slice from time i to j.

Then, the time-series anomaly detection task can be defined as follows:

Problem 1. *Given a real-valued time series $\boldsymbol{X} = [X_1, X_2, .., X_n]$, the task of anomaly detection is to predict the label of each point $\boldsymbol{R} = [R_1, R_2, .., R_n], R_i \in \{0, 1\}$, where "1" indicates an abnormal point. At timestamp t, only $[X_1, X_2, .., X_{t-1}, X_t]$ can be used to predict R_t, since subsequent points are not visible.*

3.2 Motivations

Large companies often need to monitor tens of thousands or even millions of metric data in real time to prevent huge economic losses from potential anomalies. This kind of time-series data is often related to user operations and roughly shows regular patterns at certain intervals (e.g. daily or weekly). However, this repetitive pattern is not strictly the same, which means that there are local fluctuations based on rough patterns. As shown in Figure 2(a) and 2(b), the orange line shows the rough repetitive patterns, the blue line is raw data which fluctuates up and down, and the red line denotes anomaly cases whose anomaly degree is significantly larger than the normal fluctuation range.

In order to implement an effective anomaly detection algorithm, we first consider how human operators accurately find anomalies. As described in [36], given unlabeled time-series data, manual labeling usually involves two steps:

- **Observation.** The operator first scans data to obtain the information of normal patterns.
- **Comparison.** For candidate points, the operator needs to make a comprehensive comparison with adjacent points (local fluctuation) and adjacent periods (periodic pattern) to draw conclusions.

When a point enters the operator's line of sight and is considered a potential anomaly, its value must be significantly different from the average of the surrounding points. From the data changes over time, we think that it has caused a large fluctuation locally when compared with previous points. However, such relatively large local fluctuations may occur frequently at the same time slots in different periods and are part of the normal periodic pattern. So in order to reach a final decision, the operators must compare this local fluctuation with the corresponding behaviors of the adjacent periods. Figure 2(b) gives such an example. We can observe that a, b, d are located in the same time slot of adjacent periods, which generally shows large local fluctuations in the periods and is only part of the normal pattern. Hence, although the fluctuations of a, b, c, and d are very close, only c will be regarded as abnormal. It is not difficult to understand in real scenarios. In the peak hours of online services, data usually fluctuates greatly, but it does not represent any abnormality. In off-peak time, the data usually remains stable, then a large fluctuation will be regarded as abnormal.

From the perspective of anomaly labeling, there are typically two key points for the anomaly detection task of such data: *Local Fluctuation* and *Periodic Pattern*. To be specific, the anomalies in time series data are usually local fluctuations that do not meet the periodic pattern. However, most of the existing methods only focus on the raw value itself (e.g. SPOT [30]) or the simple statistics of raw values (e.g. traditional statistical models) or implicitly handle the data distribution (e.g. DONUT [33]). We argue that periodicity of fluctuations must be considered for anomaly detection in online service scenarios, otherwise both false positives and false negatives will increase. Specifically, in relatively stable time slots, a small fluctuation may be abnormal; but in unstable time slots, a large fluctuation can also be normal.

As mentioned earlier, SPOT is only sensitive to extreme values. If we can extract appropriate features to indicate the abnormal degree of local fluctuations involved in periodical patterns, and make it a stationary distribution, then SPOT can be directly applied. In this paper, we take fluctuations as the starting point, and consider two main problems to be resolved: (i) how to extract local fluctuations? (ii) how to eliminate the noises and periodic effects, so as to help SPOT achieve better results?

4 METHODOLOGY

Figure 3 depicts an overview of our proposed *FluxEV* framework. It consists of four components: data preprocessing, fluctuation extraction, two-step smoothing, and automatic thresholding. In this section, we will introduce these components separately and then summarize the overall streaming detection framework.

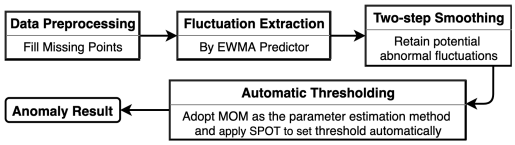


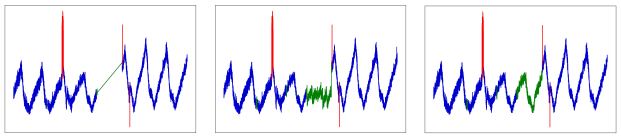
Figure 3: FluxEV framework overview.

4.1 Data Preprocessing

Time series data in real applications often suffer from missing values due to server downtime or network crashes. Simple filling methods (i.e. filled with zeros or average values) severely affect the

normal data patterns, especially when the missing interval is really long. Similarly, the widely used linear interpolation does not handle long missing segments well. Based on Deep Generative Models, an MCMC-based missing data imputation technique is proposed by [28]. In [33], authors adopted such an imputation method for time-series data with a trained VAE model. The idea is to iteratively reconstruct the data so that the missing parts gradually conform to the normal pattern. However, for the long missing segments, in implementation, we found that after iterative reconstructions, the generated values still apparently differ from normal patterns.

In order to maintain the normal patterns and introduce as few noises as possible, we adopt a simple strategy to fill the missing parts. The filling method is divided into two cases based on the missing segment lengths: (1) when the missing part is shorter than 5 points, it is filled with the first-order linear interpolation on adjacent points; (2) otherwise, the values of the same time slot from the previous period plus a bias (i.e. half of the difference between the means of the two periods, $\frac{\mu_i - \mu_{i-1}}{2}$) are used to fill the missing segment. Figure 4 shows the effects of three filling methods for a long missing segment. Our strategy can better maintain the data patterns, thereby reducing the negative impact on subsequent fluctuation extraction.



(a) Linear (b) MCMC-based (c) Our Strategy

Figure 4: Effect contrast of three filling methods. Blue, green, and red curves denote normal data, missing parts with filled values, and anomalies, respectively. For the MCMC-based method, we utilized DONUT [33] as the trained model and performed 100 reconstructions iteratively.

4.2 Fluctuation Extraction

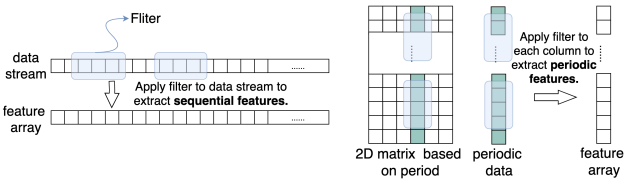
Different from directly extracting simple statistics features in other works [1, 2], e.g. mean, min, max, standard deviation, we focus on fluctuations in data, which is more direct and useful in indicating anomalies.

In order to achieve a simpler and faster calculation, Exponentially Weighted Moving Average (EWMA) [20] is picked as a simple predictor instead of any deep network (e.g. LSTM [18], GRU [13]), then the error between the current point value and the predicted value is calculated. Here, we assume $\boldsymbol{E} = [E_1, ..., E_n]$ is an array with the same length as $\boldsymbol{X} = [X_1, ..., X_n]$, and E_i stores the prediction error at time i. EWMA value and prediction error are calculated as below:

$$\text{EWMA}(\boldsymbol{X}_{i-s,i-1}) = \frac{X_{i-1} + (1-\alpha)X_{i-2} + \cdots + (1-\alpha)^{s-1}X_{i-s}}{1 + (1-\alpha) + \cdots + (1-\alpha)^{s-1}} \tag{1}$$

$$E_i = X_i - \text{EWMA}(\boldsymbol{X}_{i-s,i-1}) \tag{2}$$

where α denotes the smoothing factor, and s is the window size which means only previous s points are used to calculate the predicted value of the current point.



(a) Sequential Processing (b) Periodic Processing

Figure 5: Two types of feature processing methods.

Actually, this predicted value can also be regarded as the expected value, then we can call the prediction error in Equation (2) as local fluctuation at the point i. Below we will introduce how to use two-step smoothing to make normal fluctuations close to zero and retain potential abnormal fluctuations.

4.3 Two-step Smoothing Processing

Recall the two key points (i.e. *Local Fluctuation* and *Periodic Pattern*) mentioned in Section 3.2. We introduce two feature processing methods for two-step smoothing operation: *sequential* and *periodic*, as shown in Figure 5. Here, the filter denotes an arbitrary sliding-window function.

In general, most of the originally extracted fluctuation values (i.e. E_i) are in the normal range. They are slightly different and look like noises (we refer to them as local noises). Besides, at different time slots in one period, the degree of these local fluctuations also varies, and the same time slots in different periods usually have similar fluctuations (we think this is part of periodic patterns). The smoothing is mainly to eliminate these local noises and the effect of periodic patterns, try to make normal fluctuations close to zero and only retain potential abnormal fluctuations.

First-step Smoothing is used to eliminate local noises by processing extracted fluctuation values (i.e. E_i) sequentially, as described in Equation (3) - (4):

$$\Delta\sigma = \sigma(\boldsymbol{E}_{i-s,i}) - \sigma(\boldsymbol{E}_{i-s,i-1}) \tag{3}$$

$$F_i = \max(\Delta\sigma, 0) \tag{4}$$

where F_i denotes the fluctuation value at time i after the first smoothing, $\Delta\sigma$ indicates the change of standard deviation when E_i is added to the current window $\boldsymbol{E}_{i-s,i-1}$. The rationality behind is: a fluctuation value will be retained that, if it is added to the current local window, it would cause a great increase in the standard deviation; otherwise, those noises with similar amplitude will approach zero after the difference operation. Here, the *max* operation means that, if the addition of a point reduces the current standard deviation, we consider it a normal point and set its fluctuation to 0.

Second-step Smoothing aims to eliminate periodic noises (i.e. the effect of periodic patterns) by periodic processing. Before performing an action on periodic data, one issue we need to consider is data drift. That is, although the same time slots in different periods usually have similar fluctuations, it is not strictly one-to-one correspondence. Such fluctuations in different periods may shift left or right in time, especially in unstable curves. Here we adopt a simple method to deal with this kind of data drift. Suppose the current point is X_t, the period length is l, for the p-th period prior to the current

time, we use $max(X_{t-pl-d}, .., X_{t-pl-1}, X_{t-pl}, X_{t-pl+1}, .., X_{t-pl+d})$ to construct the sliding window of periodic processing, instead of X_{t-pl} itself. The second smoothing is defined as Equation (5) - (7).

$$M_{i-d} = \max(\boldsymbol{F}_{i-2d,i}) \tag{5}$$

$$\Delta F_i = F_i - \max\left(M_{i-l(p-1)}, .., M_{i-2l}, M_{i-l}\right) \tag{6}$$

$$S_i = \max(\Delta F_i, 0) \tag{7}$$

First, we use an array $\boldsymbol{M}$ to store the local maximum of $\boldsymbol{F}$ (i.e. the fluctuation values after the first-step smoothing), where d denotes half of the window size to handle data drift, e.g. M_{i-d} denotes the max value of $[F_{i-2d}, F_{i-2d+1}, .., F_{i-d}, .., F_{i-1}, F_i]$. For Equation (6), we can regard the maximum value of the local maximum of the past p periods as the upper limit of the normal fluctuation at the current timestamp, and then use the difference ΔF_i to express the abnormal degree of the current fluctuation. If the $\Delta F_i < 0$, we think it is a normal point and just set it to zero. Finally, we can get the fluctuation value S_i after the second-step smoothing.

The whole process of fluctuation extraction and smoothing is summarized in Algorithm 1. There are mainly three sliding window parameters: s, p, d. s is used to extract fluctuations and the first sequential smoothing, p is used for the second periodic smoothing, d is used to handle the data drift issue. l indicates the number of points contained in one period. In particular, our fluctuation extraction and smoothing processing use past points as references, so at the very beginning of the algorithm, we will "waste" some points as a start-up. As shown in Algorithm 1, we can get E_i from s-th point (line 3-4), F_i from $2s$-th point (line 5-7), M_i from $(2s + d)$-th point (line 8-9), and S_i from $(2s + d + l(p - 1))$-th point (line 10-12). Finally, we use S_i to represent the abnormal degree of the data point and use it for subsequent detection. Therefore, we will waste a total of $2s + d + l(p - 1)$ points, and starting from the $(2s + d + l(p - 1))$-th point, the corresponding feature values E_i, F_i, M_{i-d}, S_i can be calculated in real time.

Figure 6 gives an example of the smoothing effects. We can see that the data trend is eliminated after extracting the fluctuations and we get a stationary distribution. After the smoothing operations, noise is significantly suppressed, while potential abnormal fluctuations are retained. Essentially, this smoothing operation widens the gap between normal and abnormal values, and makes abnormal fluctuations more extreme in a stationary distribution, thereby improving the performance of anomaly detection algorithms.

4.4 MOM-based Automatic Thresholding

After the smoothing operation, potential abnormal fluctuations are retained and most of the normal fluctuations are eliminated. We can apply a single threshold to detect anomalies, i.e. if the fluctuation feature S_i is larger than the threshold, then this data point will be classified as abnormal. However, such a simple rule is too naïve to guarantee a satisfactory performance, and a more robust decision rule is needed. Instead, we leverage SPOT [30] to provide a strong statistical guarantee and automatically select thresholds.

Although it is prohibitive to obtain enough labels for training supervised models, large-scale unlabeled historical data is usually readily available (usually minute-level data is monitored in online services). Sufficient sample points in real applications encourage us to leverage Method of Moments (MOM) [9] as the parameter

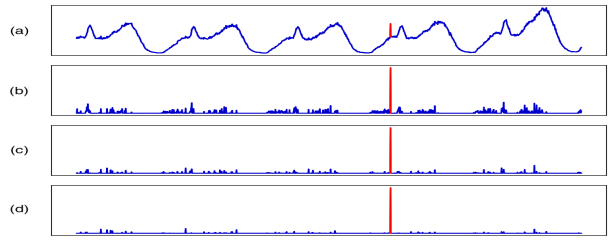


Figure 6: Smoothing. The first row (a) shows the raw time-series curves and anomalies are highlighted in red; (b) stands for originally extracted fluctuations; (c)-(d) represent the fluctuation features after the first and second smoothing respectively.

Algorithm 1: ExtAndSmooth

Input: input data $\boldsymbol{X} = [X_1, .., X_n]$, window sizes s, p, d, period l

Output: $\boldsymbol{E} = [E_1, .., E_n]$, $\boldsymbol{F} = [F_1, .., F_n]$, $\boldsymbol{S} = [S_1, .., S_n]$, $\boldsymbol{M} = [M_1, .., M_{n-d}]$

1 **for** $i = 1$ *to* n **do**
2 $\quad E_i = F_i = M_i = S_i = None$;
3 $\quad$ **if** $i > s$ **then**
4 $\quad\quad E_i = X_i - EWMA(\boldsymbol{X}_{i-s,i-1})$;
5 $\quad$ **if** $i > 2s$ **then**
6 $\quad\quad \Delta\sigma = \sigma(\boldsymbol{E}_{i-s,i}) - \sigma(\boldsymbol{E}_{i-s,i-1})$;
7 $\quad\quad F_i = \max(\Delta\sigma, 0)$;
8 $\quad$ **if** $i > 2s + 2d$ **then**
9 $\quad\quad M_{i-d} = \max(\boldsymbol{F}_{i-2d,i})$;
10 $\quad$ **if** $i > 2s + d + l(p - 1)$ **then**
11 $\quad\quad \Delta F_i = F_i - \max\left(M_{i-l(p-1)}, .., M_{i-2l}, M_{i-l}\right)$;
12 $\quad\quad S_i = \max(\Delta F_i, 0)$;

estimation method to further improve the efficiency of SPOT. In Section 5.5.2, our experiments show that, in *FluxEV*, MOM can achieve competitive results with MLE, while improving efficiency by about 4 to 6 times.

4.4.1 SPOT Algorithm. SPOT [30] is a streaming version of Peaks-Over-Threshold (POT) which is the second theorem in Extreme Value Theory (EVT) [9]. The POT approach does not assume the distribution of the monitoring data, instead, it only relies on the distribution of extreme values, which is almost independent of the data distribution according to EVT. The philosophy is to fit the tail distribution by Generalized Pareto Distribution (GPD):

$$\bar{F}_t(x) = P(X - t > x | X > t) \sim \left(1 + \frac{\gamma x}{\sigma}\right)^{-\frac{1}{\gamma}} \tag{8}$$

where t is the initial threshold to retrieve the peaks; γ and σ are the shape and scale parameters of GPD; X denotes the data point and $X - t$ indicates the portion over a threshold t. Hence, Equation (8) means the portion over a threshold t, i.e. $X - t$, are likely to follow a GPD with parameters γ, σ.

Unlike the original SPOT, we estimate the parameters $\hat{\sigma}$ and $\hat{\gamma}$ by MOM, then the final threshold th_F can be calculated as follows:

$$th_F = t + \frac{\hat{\sigma}}{\hat{\gamma}}\left(\left(\frac{qn}{N_t}\right)^{-\hat{\gamma}} - 1\right) \tag{9}$$

where t is the initial threshold; $\hat{\sigma}$ and $\hat{\gamma}$ are shape and scale parameters of GPD, which will be estimated by MOM; q is the risk coefficient to determine anomalies; n is the number of current observations; N_t is the number of X_i (peaks), s.t. $X_i > t$.

During the streaming pipeline, such POT operations will be updated once at each timestamp. The original SPOT is performed directly on raw data values. In our framework, we employ automatic thresholding of SPOT on S_i (i.e. fluctuation values after two-step smoothing) to make a decision.

4.4.2 Method of Moments. The basic idea of MOM is to use sample moments to replace population moments, and then derive unknown parameters of the distribution from the expressions for the population moments. For GPD, the mean and the variance can be expressed as: $E(Y) = \frac{\sigma}{1-\gamma}$, $var(Y) = \frac{\sigma^2}{(1-\gamma)^2(1-2\gamma)}$.

Replacing $E(Y)$ by $\mu = \sum_{i=1}^{N_t} \frac{Y_i}{N_t}$ and $var(Y)$ by $S^2 = \sum_{i=1}^{N_t} \frac{(Y_i-\mu)^2}{N_t-1}$, where Y_i is the excesses of peaks ($Y_i = X_i - t$ for $X_i > t$, X_i is the sample point and t is initial threshold) and N_t is the number of peaks, then estimates for $\hat{\sigma}$ and $\hat{\gamma}$ can be computed via Equation (10) - (11):

$$\hat{\sigma} = \frac{\mu}{2}(1 + \frac{\mu^2}{S^2}) \tag{10}$$

$$\hat{\gamma} = \frac{1}{2}(1 - \frac{\mu^2}{S^2}) \tag{11}$$

After introducing MOM, we can summarize POT as Algorithm 2. Also, it is the initialization step of SPOT. We set t to the 98% quantile; $\mathbf{Y_t}$ indicates the peaks set, i.e. the portion over t; GPD parameters γ, σ are estimated by MOM, and the final threshold th_F is calculated via Equation (9).

Algorithm 2: POT (Peaks-over-Threshold)

Input: input data $[X_1, .., X_n]$, risk q
Output: initial threshold t, final threshold th_F

1 $t \leftarrow SetInitialThreshold([X_1, .., X_n])$;
2 $\mathbf{Y_t} \leftarrow \{X_i - t \mid X_i > t\}$;
3 $\hat{\sigma}, \hat{\gamma} \leftarrow MOM(\mathbf{Y_t})$;
4 $th_F \leftarrow CalcThreshold(q, \hat{\sigma}, \hat{\gamma}, n, N_t, t)$;

4.5 Overall Streaming Detection

Algorithm 3 shows our streaming detection framework. First, we need $a = 2s+d+l(p-1)$ points to start up the fluctuation extraction and smoothing operations (line 1-2), then we can calculate the fluctuation features in real time. After that, extra k points are used to initialize SPOT (line 3). From the $(a+k+1)$-th point, we can perform anomaly detection in a streaming fashion (i.e. perform detection and update the threshold th_F at each timestamp). It is worth mentioning that, for the real-world datasets, after an abnormal point is detected, we should update its F_i (i.e. the fluctuation values after the first-step smoothing) and M_{i-d} (i.e. the local maximum of F_i), so as to avoid using this abnormal value as a wrong reference for subsequent points. Lines 17-19 show this update process. Here, *None* indicates the abnormal fluctuation is removed from the feature array and will not be used as a reference for the succeeding points.

In the implementation of SPOT, the efficiency is mainly limited by the complex calculations of MLE. In our framework, we use MOM to replace MLE to break through the computational bottleneck, which only needs to calculate the mean and variance of the sample to estimate the population. Moreover, compared with the original SPOT, at each timestamp of the streaming detection, *FluxEV* only performs a small amount of additional calculations to extract fluctuations and eliminate local noise and periodic effects through smoothing operations, which is shown in line 6 (i.e. *CalcFeats*) of Algorithm 3 and defined in detail as Equation (1) - (7). From predictor to parameter estimation method, *FluxEV* simplifies the calculations as much as possible to achieve higher efficiency.

Algorithm 3: Streaming Detection in FluxEV

Input: input data $\boldsymbol{X} = [X_1, .., X_n]$; window sizes s, p, d; period l; point num to initialize SPOT k; risk q
Output: detection result R_i

1 $a = 2s + d + l(p-1)$;
2 $\boldsymbol{E} = [E_1, .., E_{a+k}], \boldsymbol{F} = [F_1, .., F_{a+k}], \boldsymbol{S} = [S_1, .., S_{a+k}], \boldsymbol{M} = [M_1, .., M_{a+k-d}] \leftarrow ExtAndSmooth(\boldsymbol{X}_{1,a+k}, s, p, d, l)$;
3 $th_F, t \leftarrow POT([S_{a+1}, .., S_{a+k}], q)$;
4 **for** $i > a + k$ **do**
5 $R_i \leftarrow 0$;
6 $E_i, F_i, M_{i-d}, S_i \leftarrow CalcFeats(\boldsymbol{X}_{i-s,i}, \boldsymbol{E}, \boldsymbol{F}, \boldsymbol{M})$;
7 **if** $S_i > th_F$ **then**
8 $R_i \leftarrow 1$;
9 **else if** $S_i > t$ **then**
10 $Y_i \leftarrow S_i - t$;
11 Add Y_i in $\mathbf{Y_t}$;
12 $N_t \leftarrow N_t + 1; k \leftarrow k + 1$;
13 $\hat{\sigma}, \hat{\gamma} \leftarrow MOM(\mathrm{Y_t})$;
14 $th_F \leftarrow CalcThreshold(q, \hat{\sigma}, \hat{\gamma}, k, N_t, t)$;
15 **else**
16 $k \leftarrow k + 1$;
17 **if** $R_i = 1$ **then**
18 $F_i \leftarrow None$;
19 $M_{i-d} \leftarrow \max(\boldsymbol{F}_{i-2d,i})$;

5 EXPERIMENTS

5.1 Datasets

Two popular public datasets are used in our experiments: **KPI** [5] and **Yahoo** [7]. Their statistics are shown in Table 1.

KPI dataset is provided by AIOps Challenge [5, 6]. It is a set of desensitized KPI curves with anomaly labels from real-world scenarios of many Internet companies, such as Tencent, eBay, Alibaba. Most KPIs have an interval of 1 minute, while seven KPI curves have an interval of 5 minutes. This dataset contains some missing points or long missing segments.

Table 1: Detailed information of the datasets.

Dataset	#Curves	#Timestamps	#Anomalies	Missing
KPI	29	5922913	134114/2.26%	224210/3.79%
Yahoo	367	572966	3896/0.68%	0/0%

Yahoo is a benchmark dataset for time series anomaly detection from Yahoo Labs, of which data interval is one hour. Yahoo contains four groups, where A1 is the real production traffic data and A2/A3/A4 are the synthetic time series. For three synthetic groups, the seasonalities (i.e. periodicities) are different. A3 and A4 have three seasonal components (i.e. 12-hour, daily, weekly), while A2 has random seasonality. The anomalies in the real traffic data are labeled by humans, while the anomaly points of synthetic data are algorithmically generated and inserted at random positions. All these time-series data do not contain any missing points.

5.2 Baselines and Metrics

We compare our unsupervised framework with six state-of-the-art algorithms using the two datasets mentioned before. The baseline algorithms include SPOT and DSPOT [30], DONUT [33], SR and SR-CNN [27] and SeqVL [11]. By utilizing Saliency Map (SR) as the feature extractor and CNN network as the discriminator, SR-CNN is trained with a large amount of synthetic data by injecting anomalies to anomaly-free data. Among all the algorithms compared, only SR-CNN is supervised.

We measure different anomaly detection methods from three aspects: **accuracy**, **efficiency**, and **generality**. For **accuracy**, we use three metrics (i.e. precision, recall, F_1-score) to evaluate the performance. In practice, the human operators generally only care about whether a continuous abnormal interval can be detected, rather than every anomaly in the abnormal interval. And, it is acceptable if a continuous abnormal interval is successfully detected within a small delay.

Here, we follow the adjustment strategy of previous works [27, 33]. For a labeled continuous anomaly segment, if the algorithm detects any anomaly within m points after the start of the segment, we think this segment is detected correctly, so every point in the abnormal segment is counted as a true positive (TP); otherwise, every point in the segment is counted as a false negative (FN). The points outside the abnormal segments are not adjusted. This adjustment strategy is illustrated in Figure 7. Specifically, we set $m = 7$ for minute-level time series and $m = 3$ for hour-level time series, which is consistent with [6, 27].

The explosion of online business volume puts forward higher requirements for detection **efficiency**. Also, higher detection efficiency means that more data can be processed with the same computational resources. In our experiments, we use CPU running times on test data to compare the efficiency of different methods.

As discussed in [27], the time series in online business scenarios can be divided into three typical categories: seasonal, stable, and unstable. Similarly, we manually classify the time series in the KPI dataset into three categories, then calculate the $\boldsymbol{F_1}$**-score** of different classes to evaluate the **generality**.

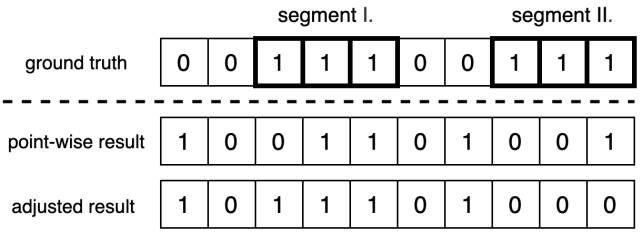


Figure 7: Illustration of the adjustment strategy for performance metrics. The first row is the ground truth with two anomaly segments. The second row is the prediction results of the algorithm. The adjusted results are shown in the last row. Here, we assume that the allowed delay is 1 point (i.e. $m = 1$). Then, all points in segment I are adjusted to TPs, while all points in segment II are counted as FNs.

5.3 Experiment Setup

During unsupervised experiments, each time series is equally divided into two halves according to the temporal order, the first half is utilized for training unsupervised models and the second half is used for evaluation (i.e. as test data). Since SR does not need additional data to start, we directly apply it to the test data. In our work, we mainly need to consider two types of parameters, the risk coefficient q for automatic thresholding and window sizes s, p, d. Besides, KPI curves have the same periodicity (i.e. daily), but for Yahoo, different groups have different periods and even contain multiple seasonal components (i.e. 12-hour, daily, weekly). Therefore, these periodicities need to be taken into account during the second-step smoothing operation.

Risk Coefficient q is a key parameter as a false-positive regulator. It determines to what extent a peak value will be considered abnormal. In [30], authors discussed the impact of the risk q and gave a recommended value range: values of q between 10^{-3} and 10^{-5} allow to have a high True Positive rate (TPr) while keeping a low False Positive rate (FPr). Specifically, we set $q = 3 \cdot 10^{-3}$ and $q = 10^{-3}$ for KPI and Yahoo, respectively.

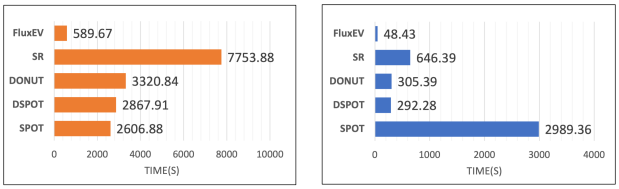
Window Sizes. Empirically, we tend to choose relatively small window sizes, since oversized windows will contain irrelevant fluctuation characteristics and periodic patterns. Here, we set $s = 10$, $p = 5$, $d = 2$ for both datasets.

Periods of Yahoo Dataset. For the real production traffic data (A1), similar to KPI data, it also has a daily periodicity. We take the number of data points in one day as the period for the second-step smoothing. Since A2 has a random seasonality, only the first smoothing is applied after the fluctuation extraction. Among three seasonal components of A3 and A4, "daily" is the dominant component by our observation, so we use "daily" as its period for the second-step smoothing.

As the code or training data is not available, the accuracy results of SR-CNN and SeqVl are copied from [11, 27] respectively. Except for SR-CNN and SeqVl, all experiments are conducted in a streaming fashion, that is, we only detect if the latest point is abnormal or not at each timestamp. All algorithms are written in Python and run on an Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz. For the efficiency comparison, all the results are averaged over three runs. Specifically, for DONUT, the number of z dimensions is set to 5,

Table 2: Accuracy comparison results of various algorithms on test Data. The supervised method is marked with "*", others are unsupervised.

	KPI			Yahoo		
Algorithm	F_1-score	Precision	Recall	F_1-score	Precision	Recall
SPOT	0.181	0.957	0.100	0.338	0.269	0.454
DSPOT	0.488	0.542	0.444	0.316	0.241	0.458
DONUT	0.729	0.876	0.624	0.058	0.031	0.567
SR	0.654	0.636	0.673	0.576	0.466	0.752
SR-CNN*	**0.771**	0.797	0.747	0.652	0.816	0.542
SeqVL	0.664	0.716	0.619	**0.661**	0.891	0.526
FluxEV	**0.790**	0.858	0.732	**0.666**	0.707	0.630



(a) KPI Dataset (b) Yahoo Dataset

Figure 8: CPU running time on test data.

the window size K is set to 120/30 on KPI/Yahoo dataset. The risk q is set to 10^{-3} for SPOT and DSPOT. And the window size d for DSPOT is set to 10. The rest of the configurations is kept the same as original works.

5.4 Overall Performance

We compare *FluxEV* with six baselines, i.e. SPOT and DSPOT [30], DONUT [33], SR and SR-CNN [27] and SeqVL [11]. As mentioned earlier, since the code or training data is not available, the accuracy results of SR-CNN and SeqVl are copied from [11, 27]. For efficiency and generality, only other four baseline methods are implemented to construct the comparison experiments.

The **accuracy** comparison results of different algorithms are demonstrated in Table 2. We can see that the improvement of our method over the original SPOT is huge: *FluxEv* improves F_1-score by 336% on KPI and 97% on Yahoo, which shows the limitation of the original SPOT. The results further illustrate that extreme anomalies in data including periodic patterns are only a small part, and by correctly processing fluctuation information, our framework can handle more anomaly cases. Compared to the best results of unsupervised baseline solutions, our approach achieves 8.4% and 0.8% improvement in F_1-score on KPI and Yahoo respectively. Even when compared with supervised SR-CNN, we still get competitive F_1-score results: 0.790 vs 0.771 on KPI, 0.666 vs 0.652 on Yahoo.

From the CPU running time in Figure 8, we can see *FluxEV* is the most **efficient** method. Moreover, we conduct **generality** comparison experiments on the test data of KPI dataset. The results in Table 3 show *FluxEV* improves the performance a lot in seasonal and unstable data while getting a relatively poor effect on stable data. In fact, since stable data does not contain periodicity, SPOT can be directly applied to detect extreme values and get a satisfying result.

Table 3: Generality comparison on KPI dataset. *Std* indicates the standard deviation of the overall F1-scores for the three classes.

	Seasonal	Stable	Unstable	Overall	***Std***
SPOT	0.150	**0.762**	0.181	0.181	0.336
DSPOT	0.379	0.529	0.497	0.488	**0.067**
DONUT	0.700	0.051	0.740	0.729	0.392
SR	0.706	0.035	0.688	0.654	0.359
FluxEV	**0.931**	0.368	**0.788**	0.790	0.257

5.5 Impact of Different Components

5.5.1 Smoothing Operation. Although [4] also used EWMA and MOM to improve the original SPOT algorithm, it lacks further analysis and improvement for data involving periodicity. Here, we emphasize the difference, that is, the effect of two-step smoothing on improving accuracy. In this section, we construct comparative experiments of different smoothing cases on Yahoo Dataset. Here, the s, p, and d are set to 10, 5, 2. Table 4 gives the comparison results.

Three smoothing cases are considered: (1) without any smoothing; (2) only with the first-step smoothing; (3) with two-step smoothing. We notice that the smoothing operation improves performance significantly. Specifically, the first and second smoothing operations increase the F_1-score by 78.4% and 9.18%, respectively. Compared to the case without any smoothing, a total of 94.7% F_1-score improvement is achieved by the two-step smoothing.

Table 4: Effects of smoothing on Yahoo dataset.

	F_1-score	Precision	Recall
No smoothing	0.342	0.248	0.550
First-step smoothing	0.610	0.628	0.593
Two-step smoothing	**0.666**	**0.707**	**0.630**

5.5.2 Estimation Method. In this section, we compare the effectiveness and efficiency of MOM and MLE under our framework. Four metrics are reported: (1) F_1-score; (2) Recall; (3) Precision; and (4) CPU running times on test data. To get a fair comparison, we uniformly set $s = 10$, $p = 5$ and $d = 2$. Table 5 presents the comparison results of MOM and MLE on KPI and Yahoo.

From the results, we can see that the MOM increases the detection speed by around 4 to 6 times when obtaining almost the same results as MLE. In industrial scenarios, faster detection speed means that under the same computational resources, more time-series data can be covered per unit time. Moreover, MOM achieves better results than MLE on KPI dataset, which means that a sufficiently large sample size can provide a certain guarantee for the performance of MOM.

Table 5: MOM vs MLE on two datasets.

Dataset	Algorithm	F_1-score	Precision	Recall	Time(s)
KPI	FluxEV-MLE	0.788	0.885	0.710	2525.79
	FluxEV-MOM	**0.790**	0.858	0.732	589.67
Yahoo	FluxEV-MLE	**0.671**	0.720	0.629	278.98
	FluxEV-MOM	0.666	0.707	0.630	48.43

5.6 Discussion on Cold-start Issue

As mentioned in Section 2, for SR-CNN [27], extra 65 million anomaly-free points are required to generate the training data. The acquisition of such a large amount of clean data is challenging in reality, which is about 10 times larger than the sum of KPI and Yahoo. Instead, our framework only needs a few periods of data during the preparation: $2s + d + l(p-1)$ points to start up the calculation of features and k points to initialize the automatic thresholding. According to [30], the error curves between the computed threshold th_F by SPOT and the theoretical threshold can converge when $k \simeq 1000$. Hence, *FluxEV* needs a much smaller set of training data than SR-CNN. Also, at the very beginning of the framework, we can appropriately relax the demand for periodic data to start earlier. Then the cold-start issue will no longer be a big problem.

6 CONCLUSION

In this paper, we proposed *FluxEV*, a simple, fast, and effective unsupervised anomaly detection framework. We take fluctuations as a breakthrough point, and from the perspective of anomaly labeling, we have emphasized two key points of anomaly detection tasks in online services: local fluctuation and periodic pattern. By converting data with periodic patterns into a stationary distribution and expanding the gap between normal and abnormal values, *FluxEV* greatly improves the performance of the original SPOT. Experimentally, we compare *FluxEV* with six state-of-the-art algorithms on two public datasets. The results show that *FluxEV* outperforms the best-performed of all baseline solutions, even when compared with one supervised model. Moreover, *FluxEV* only needs a small amount of training data in the startup and initialization stage, its simplicity provides high efficiency for real-time detection.

ACKNOWLEDGMENTS

This work is partially supported by the Hong Kong RGC GRF Project 16207617, CRF Project C6030-18G, C1031-18G, C5026-18G, AOE Project AoE/E-603/18, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, Didi-HKUST joint research lab project, and Wechat and Webank Research Grants.

REFERENCES

[1] [n.d.]. https://github.com/lizeyan/workshop.aiops.org/raw/master/files/2018/logicmonitor2018.pdf.
[2] [n.d.]. https://github.com/lizeyan/workshop.aiops.org/raw/master/files/2018/d.i.2018.pdf.
[3] [n.d.]. https://github.com/lizeyan/workshop.aiops.org/raw/master/files/2018/ica1282018.pdf.
[4] [n.d.]. https://github.com/DawnsonLi/EVT.
[5] [n.d.]. AIOps Challenge, Final Dataset. http://iops.ai/dataset_detail/?id=10.
[6] [n.d.]. AIOps Challenge, KPI Anomaly Detection Competition. http://iops.ai/competition_detail/?competition_id=5&flag=1.
[7] [n.d.]. Yahoo! Webscope Dataset. A Labeled Anomaly Detection Dataset, version 1.0. https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70.
[8] Jinwon An and Sungzoon Cho. 2015. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE* 2, 1 (2015).
[9] Jan Beirlant, Yuri Goegebeur, Johan Segers, and Jozef L Teugels. 2006. *Statistics of extremes: theory and applications*. John Wiley & Sons.
[10] Chris Chatfield. 1978. The Holt-winters forecasting procedure. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 27, 3 (1978), 264–279.
[11] Run-Qing Chen, Guang-Hui Shi, Wan-Lei Zhao, and Chang-Hui Liang. 2019. Sequential VAE-LSTM for Anomaly Detection on Time Series. *arXiv preprint arXiv:1910.03818* (2019).
[12] Wenxiao Chen, Haowen Xu, Zeyan Li, Dan Peiy, Jie Chen, Honglin Qiao, Yang Feng, and Zhaogang Wang. 2019. Unsupervised anomaly detection for intricate kpis via adversarial training of vae. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 1891–1899.
[13] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[14] Philippe Esling and Carlos Agon. 2012. Time-series data mining. *ACM Computing Surveys (CSUR)* 45, 1 (2012), 1–34.
[15] Tak-chung Fu. 2011. A review on time series data mining. *Engineering Applications of Artificial Intelligence* 24, 1 (2011), 164–181.
[16] Sylvain Fuertes, Gilles Picart, Jean-Yves Tourneret, Lotfi Chaari, André Ferrari, and Cédric Richard. 2016. Improving spacecraft health monitoring with automatic anomaly detection techniques. In *14th International Conference on Space Operations*. 2430.
[17] Aurea Grané and Helena Veiga. 2010. Wavelet-based detection of outliers in financial time series. *Computational Statistics & Data Analysis* 54, 11 (2010), 2580–2593.
[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[19] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 387–395.
[20] J Stuart Hunter. 1986. The exponentially weighted moving average. *Journal of quality technology* 18, 4 (1986), 203–210.
[21] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. 2015. Generic and scalable framework for automated time-series anomaly detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1939–1947.
[22] Suk-Bok Lee, Dan Pei, MohammadTaghi Hajiaghayi, Ioannis Pefkianakis, Songwu Lu, He Yan, Zihui Ge, Jennifer Yates, and Mario Kosseifi. 2012. Threshold compression for 3g scalable monitoring. In *2012 Proceedings IEEE INFOCOM*. IEEE, 1350–1358.
[23] Dapeng Liu, Youjian Zhao, Haowen Xu, Yongqian Sun, Dan Pei, Jiao Luo, Xiaowei Jing, and Mei Feng. 2015. Opprentice: Towards practical and automatic anomaly detection through machine learning. In *Proceedings of the 2015 Internet Measurement Conference*. 211–224.
[24] Wei Lu and Ali A Ghorbani. 2008. Network anomaly detection based on wavelet analysis. *EURASIP Journal on Advances in Signal Processing* 2009 (2008), 1–16.
[25] Dong Yul Oh and Il Dong Yun. 2018. Residual error based anomaly detection using auto-encoder in SMD machine sound. *Sensors* 18, 5 (2018), 1308.
[26] Brandon Pincombe. 2005. Anomaly detection in time series of graphs using arma processes. *Asor Bulletin* 24, 4 (2005), 2.
[27] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. 2019. Time-Series Anomaly Detection Service at Microsoft. *arXiv preprint arXiv:1906.03821* (2019).
[28] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).
[29] Bernard Rosner. 1983. Percentage points for a generalized ESD many-outlier procedure. *Technometrics* 25, 2 (1983), 165–172.
[30] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. 2017. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1067–1075.
[31] Alexey Tsymbal. 2004. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin* 106, 2 (2004), 58.
[32] Owen Vallis, Jordan Hochenbaum, and Arun Kejariwal. 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] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. 2018. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 187–196.
[34] Zhao Xu, Kristian Kersting, and Lorenzo von Ritter. 2017. Stochastic Online Anomaly Analysis for Streaming Time Series.. In *IJCAI*. 3189–3195.
[35] Asrul H Yaacob, Ian KT Tan, Su Fong Chien, and Hon Khi Tan. 2010. Arima based network anomaly detection. In *2010 Second International Conference on Communication Software and Networks*. IEEE, 205–209.
[36] Nengwen Zhao, Jing Zhu, Rong Liu, Dapeng Liu, Ming Zhang, and Dan Pei. 2019. Label-Less: A Semi-Automatic Labelling Tool for KPI Anomalies. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 1882–1890.
[37] Indrė Žliobaitė. 2010. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784* (2010).