Scalable Prediction-based Online Anomaly Detection for Smart Meter Data

Xiufeng Liu, Per Sieverts Nielsen

Technical University of Denmark

Abstract

Today smart meters are widely used in the energy sector to record energy consumption in real time. Large amounts of smart meter data have been accumulated and used for diverse analysis purposes. Anomaly detection raises the big data problem, namely the detection of abnormal events or unusual consumption behaviors. However, there is a lack of appropriate online systems that can handle anomaly detection for large-scale smart meter data effectively and efficiently. This paper proposes a lambda system for detecting anomalous consumption patterns, aiming at assisting decision makings for smart energy management. The proposed system uses a prediction-based detection method, combined with a novel *lambda* architecture for iterative model updates and real-time anomaly detection. This paper evaluates the system using a real-world data set and a large synthetic data set, and compares with three baselines. The results show that the proposed system has good scalability, and has a competitive advantage over others in anomaly detection.

Keywords: Anomaly detection, Lambda architecture, Real-time, Data mining, Scalability

1. Introduction

Anomalies are often defined as the observations that lie outside the overall pattern of distribution [36], also known as outliers. In recent years, due to the wide deployment of smart meters, anomaly detection of energy consumption has received increasing research efforts, such as the works [11, 31, 29, 44]. Smart meters are the electronic devices for recording energy consumption at a regular time interval, usually every 15 minutes [12]. Detailed smart meter data can be used for billing and monitoring purposes. Anomalies may be caused by meter defects, consumer behavior changes, energy leakage and overlighting. Anomaly detection becomes an important part of energy management systems to identify potential risks in order to take timely actions, for example, to prevent energy leakage or energy theft [11]. In addition, anomaly detection can provide reliable data for accurate billing, and for training a model, such as forecasting where anomalous data may result in bias or failure for parameter estimation [41].

Traditional anomaly detection methods are usually performed manually with the help of data visualization tools [21]. With these tools, users identify normal patterns, then single out the samples that deviate from the normal patterns, e.g., [14, 31, 44]. The detection is conducted in an offline fashion and on a sample of data. However, this method is not suitable for the online detection of anomalies in real-time smart meter data. Real-time detection must be able to detect anomalies at the current point in time in order to notify and/or take actions [21]. Real-time detection methods have been found in [5, 11], which use prediction algorithms to predict the next measurement in a data stream, then classify the actual measurement as normal or abnormal by comparing the predicted value. The biggest problem with real-time detection is the update frequency of the detection model, whilst these works use static models in their detection. In order to reflect the latest trends and patterns, online detection models must be trained and updated in a timely manner. Model training is often time-consuming, especially when using large data sets (e.g., entire historical data). The training has to be carried out offline. However, this may lead to using stale models in online detection. This is a prominent issue in detecting abnormal consumption of smart meter data streams.

As mentioned earlier, there are different types of anomalies. In this paper, we limit the scope of work to *pattern anomaly detection*. The reason is based on the observation that residential consumption patterns often show similarly, mainly due to residents' living habits. Figure 1 illustrates an example of daily electricity consumption patterns in four continuous days (synthetic data). As can be seen from the figure, the patterns in the first three days are similar, being peaks in the morning and in the evening respectively, indicating that more energy is used after getting up and after work. However, the fourth day is quite different. The consumption throughout the day is very high and there is no clear morning and evening peak. Compared with historical consumption patterns, this consumption pattern can be classified as an anomaly. According to the duration of this new pattern, the emergence of this pattern can be a good signal for attention. For example, if the next day's pattern is similar to this one, it may be, for example, the residents have begun using high energy consuming appliance (e.g., heating/cooling devices in cold/hot weather); or new residents with a different consumption habit have moved into this apartment.

*Corresponding author

Email address: xiuli@dtu.dk (Xiufeng Liu)

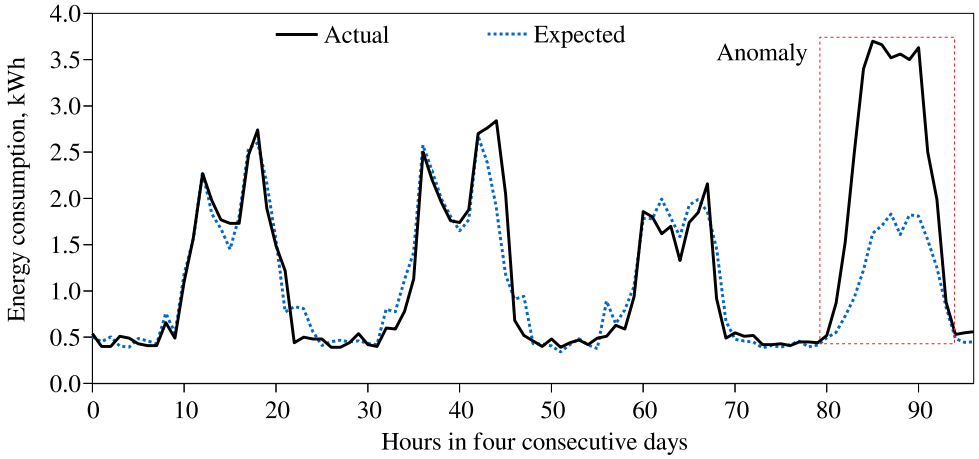


Figure 1: Daily consumption pattern and the anomaly

Utilities can provide personalized energy services and energy-saving suggestions to this household. If this pattern only appears temporarily, it can be due to the reason, for example, network communication errors, forgetting to turning off the stove after cooking, or other unusual events that occur in this apartment (e.g., the residents staying home for a party celebration throughout the day). Therefore, pattern anomaly detection can provide useful information for utilities and households to decide whether to take appropriate actions or not, and real-time detection can be a desirable feature of an energy management system for taking prompt actions.

In this paper, we propose an anomaly detection method that uses machine learning and statistics and combines it with big data technologies to detect abnormal energy consumption patterns in real time. We not only focus on anomaly detection using data analysis, but also emphasize implementing a system that can support real-time data processing and high scalability. We believe this work will raise the research interest in combining data engineering and data-intensive technologies to improve energy management. In order to discover abnormal patterns, we propose a prediction-based detection method; and in order to detect anomalies in real time, we have implemented a lambda system. A lambda system has a three-layer architecture, including batch layer, speed layer, and serving layer [34]. The batch layer runs batch jobs for data-intensive computing, while the speed layer runs real-time jobs for for data stream processing. The serving layer answers queries by merging the results from the speed layer and the batch layer. In our system, we use the batch layer to refresh detection models periodically, and use the speed layer to detect anomalies in real time. The serving layer is used to answer user queries and used for notification purpose. The implementation uses Apache Spark as the batch layer technology, Spark Streaming as the speed layer technology, and PostgreSQL as the serving lay technology. Spark is a distributed computing framework that supports scalable data processing in the cluster (more details will be discussed in Section 3.5). Smart meter data can be streamed into the lambda system directly to detect anomalies in real time. It is worth noting that although the system was developed for the anomaly detection of residential electricity consumption, the method can also be used to the detection of other data, such as water and heating data. This is because of the generic nature of time series, and the prediction-based detection method proposed in this paper. In addition, the proposed method is not limited to daily pattern anomaly detection. It can also be used to detect the abnormal patterns of other window sizes, and point anomalies (i.e., the window size is 1 hour for our example).

The lambda system for anomaly detection was presented in our conference paper [29]. Based on the previous work, this paper refines the detection models using new distance metrics, evaluates the system under different settings, and compares with other detection models. In summary, the contributions of this paper are as follows: 1) A prediction-based method is proposed for detecting unusual consumption patterns; 2) A novel lambda system is implemented to support real-time anomaly detection and batch model refreshment; 3) The lambda system is evaluated comprehensively, and compared with three baseline methods. The results show that the proposed system has good scalability for handling large-scale data sets, and the detection method has a competitive advantage over others.

The remainder of this paper is structured as follows. Section 2 investigates related works. Section 3 discusses the anomaly detection algorithms, the implementation and the baselines for comparison. Section 4 evaluates the system by comparing it with the baselines. Section 5 summarizes the paper and provides research directions for future work.

2. Related Work

Anomaly detection has the applications in the domain of smart meter data analysis. Anomaly detection is crucial to energy management because it helps improve energy services and increase energy savings [29]. Residential consumption is usually estimated based on the days where the patterns differ according to the behaviors of different occupants and different time periods, such as day, night, weekday, weekend and holiday [11]. Therefore, anomaly detection systems for residential

households should detect suspicious consumption based on patterns and seasonality [3]. In this paper, an anomaly is defined as the actual consumption pattern that deviates from a predicted value by calculating the distance. Anomalies are classified according to a calculated Gaussian probability value that is below a given threshold. Chou et al. flagged an anomaly according to two standard deviations above or below the predicted power consumption [11]. However, the anomaly identification in their method is based on the statistical model created using consumption values, while ours is based on the probability model created using distance values.

There is much research work on anomaly detection in the smart energy sector. Chandola et al. conducted a survey about anomaly detection methods in different domains including energy [8], and Zhang et al. compared clustering, regression and entropy methods for detecting anomalous electricity consumption [44]. Normal distribution is an effective statistical method for anomaly classification, which is used in [11, 22, 23, 31] and in this work. However, statistical methods usually have to be combined with other techniques to detect anomalies, such as clustering or regression. Jakkula et al. used clustering along with a statistical method to find outliers in electricity consumption time series from smart homes [22]. Herrerias detected electricity intensity anomalies based on seasonal patterns [20]. These methods are based on pattern similarity, but more works are based on regression methods. The early work [25] detected outliers by a linear regression method with a threshold limit. This approach, however, produced a lot of false positives for large data sets. Adnan et al. later improved it by combining with a clustering method, and obtained a better result [1]. Zhang et al. further considered the impact of outdoor temperature and used a three-piecewise linear regression method to fit the relationship between energy consumption and outdoor temperature. However, linear regression-based methods require well-defined independent variables [32]. Brown et al. predicted electricity demand using K-nearest neighborhood (KNN) in a kernel regression method [5]. Zhang proposed a hybrid model of combining auto-regressive integrated moving average (ARIMA) with adaptive artificial neural network (ANN). These works provide a good accuracy for anomaly detection, but their usefulness for scalable online detection is unclear. In this paper, we propose a prediction-based method for anomaly detection, but incorporate the detection method into a lambda system to support real-time detection, and regular model refreshment. The proposed approach not only has a good accuracy, but also supports parallel detection for large-scale data streams.

Batch and streaming data processing have been receiving much research effort. Liu et al. investigated the technologies for real-time big data processing and proposed the technologies that can be used for the lambda architecture [28]. [37] and [17] developed real-time architectures for processing other domain-specific big data, including healthcare and social media. Cheng et al. suggested that smart city data platforms should have the ability to handle the complexity of smart city data and to process data in real time and in batches [10]. They also suggested that anomaly detection should become a core component of a smart city data platform. Streaming data anomaly detection was also found in [38], but the model used is static. In contrast, the detection models in our work are refreshed iteratively in batch jobs, and the detection is conducted in real-time jobs.

The lambda architecture is becoming increasingly popular for its real-time and batch capability. The works [7, 28] extensively reviewed the technologies that can be used for the lambda architecture. The industrial energy management system [39] used the lambda architecture to process real-time data in a cloud environment. Kroßet al. extended the lambda architecture by adding on-demand capability for processing data streams, and optimized the scheduling system in order to run real-time and batch jobs in the same cluster [24]. Moreover, Martnez-Prieto et al. processed semantic data on a lambda system [33]. Villari et al. proposed AllJoyn lambda to manage smart home devices [40]. Hasani used a lambda system to analyze network log data to discover network anomalies [18], and Liu et al. used it for smart grid complex event processing [27]. In short, the lambda architecture is a generic framework that can be applied to different domains to achieve real-time capability for big data. In this work, we focus its application in the energy field to detect anomalies for scalable energy consumption time series. We study the ability of the lambda architecture to perform regular model updates and real-time anomaly detection, and this study provides a proof of concept for its implementation in the smart energy market.

3. Anomaly Detection System

In this section, we will detail the online anomaly detection system, including system architecture, detection model, and implementation.

3.1. System Overview

The system uses a prediction-based method to detect pattern anomalies for residential electricity consumption. The overall process is depicted conceptually in Figure 2, which is a supervised machine learning method that includes a *training* and a *detection* process. The basis of our system is the prediction model that predicts an observed pattern with the assumption that the pattern will reappear in the future (with slight variations). If the actual assumption or pattern is far from the predicted value in terms of distance, it will be classified as an anomaly according to a predefined threshold value. In the training process, we use historical consumption data to train the prediction model (the model will be described in the next section). At the same time, we calculate Euclidean distances between all actual consumption patterns and the patterns generated by the prediction model. Then, we use normal distribution to fit the distances and compute the mean, μ, as well as the standard deviation, δ. In the end, the prediction model and the statistical parameters, μ and δ, will be used as the input of the detection process. The detection process is relatively straightforward, which uses a probability model to classify anomalies. The distance between an actual consumption pattern and its predicted pattern is first calculated. Then, the probability is computed using the formula, $p(x) = \frac{1}{\delta\sqrt{2\pi}} e^{-(x-\mu)^2/2\delta^2}$, and used to determine if

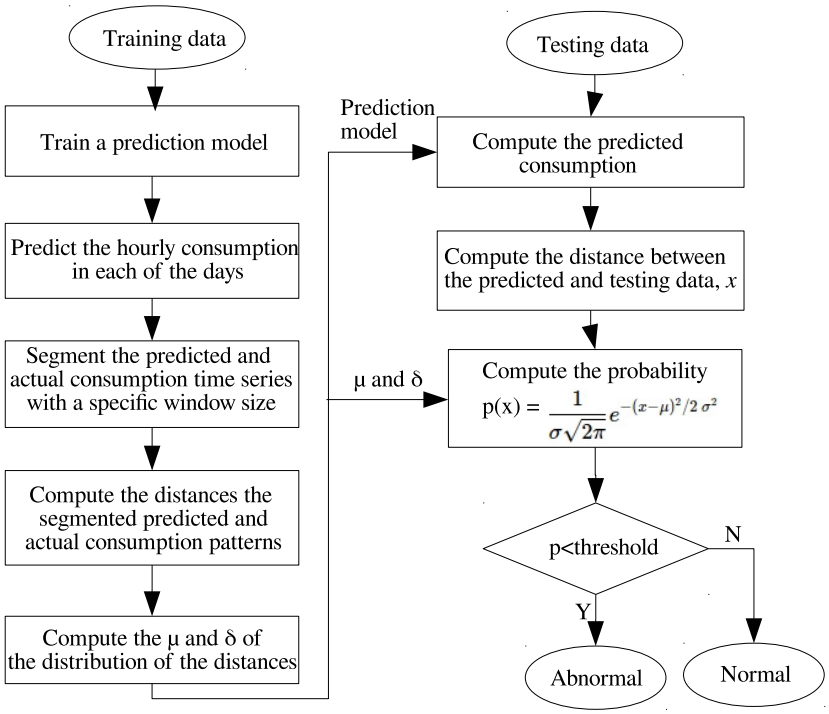


Figure 2: Overview of the anomaly detection processs

the distance is an anomaly or not. The proposed anomaly detection system is built on the lambda architecture with real-time capability for big data. The detection model is refreshed iteratively and used by the detection process to identify anomalies in real time (Section 3.5 will describe further details).

3.2. PARX Prediction Model

We now describe the prediction model. This model is called *periodic auto-regression with exogenous variables (PARX)* [3, 30], which is used for short-term residential electricity consumption prediction. This model takes into account several exogenous variables, including outdoor temperature, the area of the house or apartment, and the size of the family. Due to the difficulty in obtaining socio-economic data, we simplified the model by considering only the impact of outdoor temperature. For example, when the temperature drops in winter, the electricity consumption increases due to the heating demand; in summer when the temperature rises, the consumption also increases due to the cooling demand. The PARX model also considers the impact of residents on how the energy is used, largely due to their living habits. The impact can often be indicated by repeated consumption patterns, such as daily and weekly. For the daily pattern, morning peaks usually appear around 7 AM on weekdays, and evening peaks appear at 5 – 8 PM. On weekends and holidays, morning peaks may be delayed to the time of 9 – 10 AM as the residents get up later. The consumption over days of the week also has a regular weekly pattern. For example, the consumption of weekends is higher than weekdays, due to staying at home.

The PARX model for daily prediction has a seasonality of 24, representing every 24 hours as one period. At the s-th hour of the day, or the s-th season, the consumption values of the same hour of the p previous days are used for the auto-regression in PARX, and the weather temperature is used as an exogenous component. The PARX model of the s-th season and the n-th period can be expressed as

$$y_{s,n} = \sum_{i=1}^{p} \alpha_{s,i} y_{s,n-i} + \beta_{s,1} t' + \beta_{s,2} t'' + \beta_{s,3} t''' + \epsilon_s, \quad s \in [0, 23] \quad (1)$$

where y is the consumption time-series; p is the order of auto-regression; t', t'' and t''' are the exogenous variables accounting for outdoor temperature t, defined in Equation (2) – (4); α and β are coefficients; and ϵ is white noise.

$$t' = \begin{cases} t - 20 & \text{if } t > 20^\circ C \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$t'' = \begin{cases} 16 - t & \text{if } t < 16^\circ C \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$t''' = \begin{cases} 5 - t & \text{if } t < 5^\circ C \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

The temperatures, $5^\circ C$, $16^\circ C$ and $20^\circ C$, represent the critical points of overheating, heating and cooling, respectively. These values may vary from different climate zones.

We choose the PARX model for the online anomaly detection for the following reasons: The model itself has taken into account the exogenous variables that may affect the consumption, including seasonality and ripple effect of energy consumption. PARX is a supervised machine learning method, and has

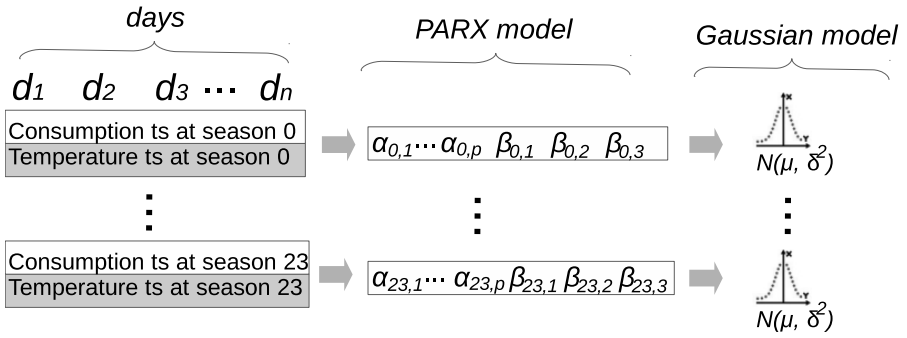


Figure 3: The process of training anomaly detection models

good prediction accuracy (see the experimental results in Section 4.3). PARX can exactly match the lambda architecture that uses batch jobs for model updates, and real-time jobs for anomaly detection in sliding windows.

3.3. Anomaly Detection

In the following, we will describe the anomaly detection of consumption patterns, including daily patterns and the patterns of other window sizes.

3.3.1. Daily Pattern Anomaly Detection

Daily pattern anomalies are defined as the patterns that have not previously appeared. We assume that often-observed daily patterns represent usual consumption behaviors during the day, while rarely-observed patterns represent unusual consumption behaviors. We compute the distances between the actual and the predicted patterns, and use the statistical normal distribution method to determinate whether a distance is anomaly or not. The detection is a supervised learning method consisting of learning and testing processes. In the training process, time series are partitioned into days to train PARX prediction models and Gaussian probability models, respectively (see Figure 3). For each season, two types of the models are generated. The Gaussian model is generated using the distances between the predicted and the actual patterns for all days. The distance metrics used are described in Section 3.4. Figure 3 describes the training process where a raw consumption time series and an outdoor temperature time series are segmented according to days. For each season, a PARX model (or parameters) and a Gaussian model, $N(\mu, \delta^2)$, are computed according to Equation 1. Therefore, a total of 24 models is generated for PARX and Gaussian, respectively.

Algorithm 1 describes the training process which computes the PARX and the Gaussian models that will be used by the detection process. This algorithm will be implemented as a Spark program in order to run in a cluster to compute the models in parallel (we will discuss it further in Section 3.5.1). The details of the algorithm are described in the following. In order to compute the models, the algorithm is given the inputs including a collection of energy consumption time series $\mathcal{TS}$, an outdoor temperature time series ts', and the order of auto-regression p. For each season s, a PARX model is computed using the consumption and the outdoor temperature time series (see line 7–8). According to the data distribution shown in Figure 12, the Euclidean distances between the predicted and the actual consumption at a specific season s satisfy lognormal distribution. Therefore, the Gaussian model can be computed based on the logarithmic values of the Euclidean distances (see line 12-18). The total number of the PARX models is $\|\mathcal{TS}\| \times 24$, as well as the Gaussian models. The generated models are stored in the PostgreSQL database in the serving layer, and will be accessed by the speed layer for online anomaly detection.

Algorithm 1 Anomaly detection model training

```
1: function TRAIN(TimeSeriesCollection TS, TemperatureTimeSeries ts' Order p)
2:     M ← {}                          ▷ Initialize the collection of PARX parameters
3:     N ← {}                          ▷ Initialize the collection of the statistical model parameters
4:     for all ts ∈ TS do
5:         id ← Get the unique identity of ts
6:         for all s ∈ 0...23 do
7:             ts^c, ts^t ← Construct a new consumption time series using ts, and a new
       temperature time series ts^t using ts' at season s
8:             α_1, ..., α_p, β_1, β_2, β_3 ← Compute the PARX model using ts^c and ts^t
9:             Insert (id, s, α_1, ..., α_p, β_1, β_2, β_3) into M
10:            L ← {}
11:            D ← Get the days of ts
12:            for all d ∈ D do
13:                x̂ ← Compute the predicted reading of the season s using PARX
14:                x ← Get the actual hourly reading from ts of the day d
15:                d ← Compute the Euclidean distance between x and x̂
16:                Add d into L
17:            μ, δ ← Compute the mean and standard deviation using the normal distri-
       bution model on L
18:            Insert (id, s, μ, δ) into N
       return M, N
```

In the detection process, unique variate Gaussian distribution is used, which is described in the following. Given a training data set, $X = \{x_1, x_2, ..., x_n\}$ whose data points satisfy the normal distribution with the mean μ and the variance δ^2, the probability density function is defined as

$$p(x; \mu, \delta) = \frac{1}{\delta\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\delta^2}} \quad (5)$$

where $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\delta^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$. The classification is based on the probability value: if p is less than the user-defined threshold, i.e., $p(x) < \epsilon$, the data point will be regarded as an anomaly.

Algorithm 2 describes the detection algorithm where the current daily consumption collection is given as the input. The

algorithm can detect a large number of time series in parallel. For each instance, the latest models from the training process are used to predict new values. The distance and the probability are computed in the algorithm, and they are used for determining an anomaly. This algorithm is self-explanatory, and more details are described in the following. First, it reads meter readings from a data stream, and fetches outdoor temperature time series and the models from PostgreSQL in the serving layer. For a prediction, the algorithm uses the parameters, including the pre-computed PARX models, previous p day's readings at the current season s, and the outdoor temperature at the current hour (see line 4–7). The algorithm then calculates the logarithmic value of the distance between the predicted and the actual readings. Note that the detection process uses the same distance metric as the training process. The distance is used to calculate Gaussian probability (see line 8–10), which is for classifying an anomaly according to the threshold value, ϵ. In the end, an anomaly is written into the PostgreSQL database in the serving layer for user notification purpose (see line 11–12). Note that for a system in production, it is not a good practice to set a single threshold value for all customers. Therefore, the system should support users to define their own threshold values in order to receive alerting message according to their desired purposes or conditions. As today utilities tend to offer portable apps for their customers to interact with smart energy management systems, setting a personalized threshold value will not be a challenge. For example, it can be implemented as one of the features of the app.

Algorithm 2 Online anomaly detection process

1: **function** DETECT(CurrentReadingCollection $\mathcal{V}$, Temperature t, PredictModel $\mathcal{M}$, StatisticalModel $\mathcal{N}$, Threshold ϵ)
2: $\mathcal{R} \leftarrow \{\}$ ▷ Initialize the detection results
3: **for all** $v \in \mathcal{V}$ **do**
4: $id \leftarrow$ Get the unique identity of v
5: $s \leftarrow$ Get the season of v
6: $\alpha_1, ..., \alpha_p, \beta_1, \beta_2, \beta_3 \leftarrow$ Get the parameters from $\mathcal{M}$ by id
7: $\hat{v} \leftarrow$ Compute the predicted reading at s using PARX with the parameters, the p days' readings at s, and temperature t
8: $x \leftarrow$ Compute the distance between the vectors, $\hat{v}$ and v
9: $\mu, \delta \leftarrow$ Get the statistical model parameters from $\mathcal{N}$ by id and s
10: $p \leftarrow$ Compute the probability using the normal distribution function, $\frac{1}{\delta\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\delta^2}}$
11: **if** $p < \epsilon$ **then**
12: Add $(id, s, p, current\_time)$ into $\mathcal{R}$
return $\mathcal{R}$

3.3.2. Anomaly Detection of Other Window Sizes

Although the above discussed anomaly detection for daily patterns, the proposed method can also be applied to the patterns of other window sizes, e.g., a particular time range of the day such as the morning peak at 6 – 9 AM or the evening peak 6 – 9 PM. If the time window is reduced to one hour, it becomes point anomaly detection, which can be seen as a special case of pattern anomaly detection. Again, PARX can be used to predict each point value in a time series and the difference between the predicted and the observed value is computed. The difference is an indicator for the abnormality of the point in a time series. The probability of being an anomaly is computed by using the Gaussian function. The same training and detection process can be applied to detect point anomalies.

3.4. Distance Metrics

In this paper, we use the following two distance metrics in the detection algorithm. The first metric is called Euclidean distance, which is defined as

$$d(x, \hat{x}) = \|x - \hat{x}\|_2 = \sqrt{\sum_{i=0}^{n-1} (x_i - \hat{x}_i)^2} \tag{6}$$

where x is the vector of the actual hourly consumption, while $\hat{x}$ is the vector of predicted consumption by PARX; $n = 24$ for the daily pattern anomaly detection, while $n = 1$ for the point anomaly detection.

The second metric is called *KSC distance* [9], which is defined as

$$\hat{d}(x, \hat{x}) = \min_q \|x_q - \hat{x}\|_2 \tag{7}$$

where x_q is a shifted version of x by the amount q that minimizes the distance. KSC distance can remedy the "double-penalty" problem of the peaks occurring on the time axis with a slight difference [2]. Figure 4 shows an example of the peak value of $1.0kWh$ in two vectors. If the q is shifted to the left for one hour, i.e., -1, the distance $\hat{d}$ becomes 0, otherwise it will be $\sqrt{2}$. This means that the difference between the actual and the predicted pattern with a slight shifting on the time axis may not represent an anomaly. A real-world example is that if one gets up late on some day, the morning peak will appear at 8–9 AM, instead of the usual time at 7–8 AM.

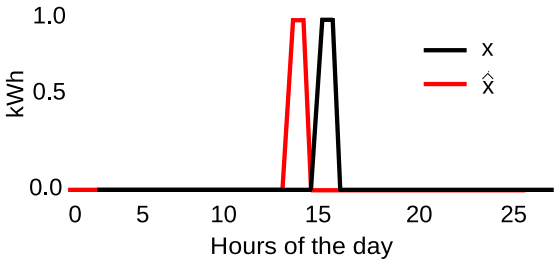


Figure 4: The double-penalty problem

In this paper, the daily pattern is shifted between -1 and 1, i.e., the last hour of the previous day and the first hour of the next day. We compute the corresponding distances for the three shifts and use the minimal one in the anomaly detection.

3.5. Scalable Online Anomaly Detection

In this section, we will introduce the lambda architecture, and describe how to use it to implement an anomaly detection system.

3.5.1. Lambda Architecture

In this system, we use the lambda architecture for online anomaly detection. Lambda architecture was first proposed by Marz for real-time data processing in 2014 [34]. The architecture is shown in Figure 5. As mentioned in Section 1, the

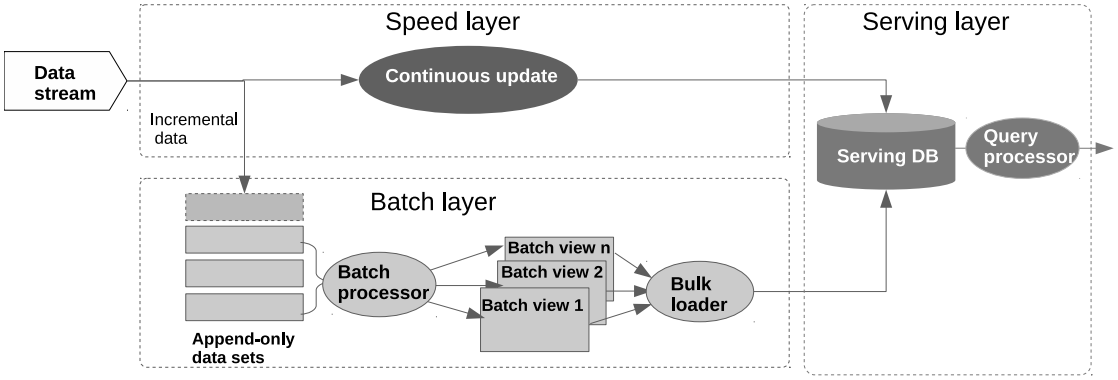


Figure 5: Lambda architecture

lambda architecture consists of a speed layer, a batch layer, and a serving layer. The speed layer obtains data from streaming data sources, processes it, and updates incremental data to the serving layer in real time. Since the size of streaming data is typically small, and the data arrives continuously and fast, the speed layer usually uses memory-based technologies to achieve fast data processing. In contrast, the data for the batch layer is typically much larger, and this layer runs iterative batch jobs to process the full set of the data. The data arriving between the beginning and the end of a batch job will be processed in the next iteration. The generated batch view will replace its predecessor in the serving layer. Batch views are robust to any system failures since they can be regenerated in the next iteration. The serving layer is usually responsible for maintaining the views generated by the speed and batch layers, and answering user queries against the views. The query processor responses to a query by merging the results from real-time views and batch views.

In a lambda system, the data pipeline is broken down into the three layers with clear demarcation of responsibilities. For each layer, there are different technologies that can be used for the implementation. The speed layer performs low-latency computations for incremental data. The streaming technology, such as Spark Streaming, Storm or S4, can be applied to this layer. The batch layer does batch computations for entire data sets, which requires good scalability. The big data processing systems, such as Spark, Hadoop, Pig, and Hive, are the good candidates for this layer. The serving layer needs to respond user queries quickly, which requires a high-performance system. The technologies, including traditional relational data management system (RDBMS), memory-based data stores (Redis or Memcache), are NoSQL database systems (Cassandra, MongoDB, or HBase), are the good options.

3.5.2. Implementation

We now describe the implementation of the system (see Figure 6). We choose the open source distributed computing framework Apache Spark as the lambda system implementation. This framework consists of two components, *Spark* and *Spark Streaming*, which are used as the batch and the speed layer technology, respectively. Spark is responsible for the model updates through batch jobs, while Spark Streaming is responsible for detecting anomalies through real-time jobs. Spark uses all available data to iteratively calculate the PARX and the Gaussian models. The readings from smart meters flow into Hadoop distributed file system (HDFS), and the detection system in the speed layer. The entire lambda system is deployed on a cluster for parallel computing, and the cluster can guarantee scalability when processing large data sets. The cluster can be scaled horizontally by adding more nodes for greater scalability.

The training algorithm is implemented using the application programming interfaces (APIs) of Spark. Internally, Spark uses the data structure, called *resilient distributed data sets (RDDs)*, to achieve fault-tolerant computation on a cluster. Spark has different distributed computing operators, including map, reduce, groupByKey, filter, collect, and others [42]. In this implementation, the consumption and the temperature time series are constructed as RDD objects in Spark. When the PARX model is computed for each season, the *groupByKey* operator is applied to aggregate the consumption time series, using meter ID and season as a composite key. The temperature time series is aggregated by season. Then, the *join* operator is applied to connect the consumption time series and the temperature time series using the season attribute as the join key. The PARX model of each season, in fact, can be seen as a multi-linear regression model, which takes auto-regressors and exogenous variables as independent variables. In the end, the multiple linear regression function offered by the Spark machine learning library, MLib [35], is used to calculate the coefficients.

For the serving layer, we choose PostgreSQL, a relational database management system, to manage the detection models that are updated iteratively by batch jobs. The update is performed at the end of a batch job. The models in the serving layer are fetched by real-time jobs for detections. Therefore, the detection system can always obtain the latest models for detecting anomalies. The detected anomalies are also saved into the PostgreSQL database for user notification purpose.

Real-time anomaly detection is performed in the speed layer. Spark Streaming reads incremental data into the data structure, called *discretized stream (DStream)*, for every time interval [43], which is an hour in our case. Data operations are

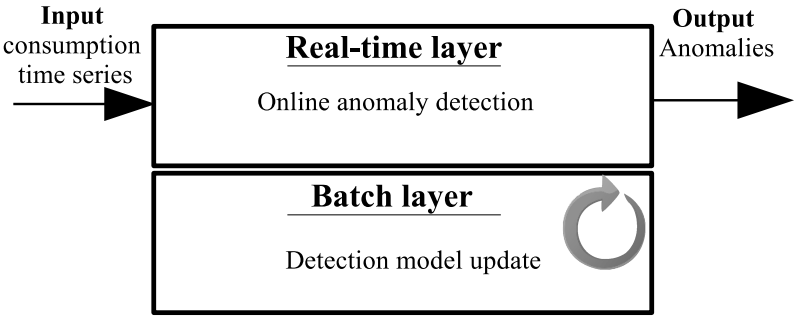


Figure 6: The system implementation

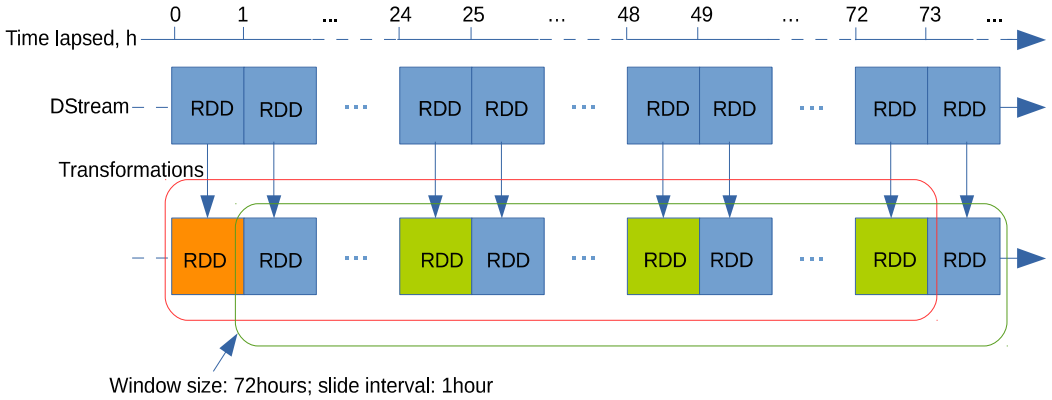


Figure 7: Sliding windows for the online anomaly detection in Spark Streaming

applied to the DStream directly for doing transformations, including removing unnecessary attribute values, filtering records and reformatting timestamp (see Figure 7). The anomaly detection method is prediction-based where a new value is predicted by the auto-regression method of using previous p days' values of a specific season (or hour). For example, Figure 7 shows an example of predicting a new value using the last recent 3 days' readings (i.e., when the order is set to $p = 3$). The window size is set to 72 hours in order to keep the past readings in the same window (see the green-colored RDDs). The window function, *reduceByKeyAndWindow(func, windowLength, slideInterval)*, aggregates the data by a specified key, window length and slide interval (meter ID and season are the composite key, the window length is 72 hours, and the slide interval is 1 hour). In order to keep past values, Spark Streaming uses a data *checkpoint* mechanism to persist past RDDs to the underlying HDFS. For each batch of streams, the detection program first reads the detection models from the serving layer, then broadcasts them to all DStreams for detecting anomalies in real time.

3.6. Baselines for the Comparison

In this section, we will discuss three existing anomaly detection methods as the baselines for comparison. They are the clustering-based detection method proposed by Bellala et al. [4], the weighted averaging method presented by Janetzko et al. [23], and the classical Boxplot method. The three baselines are only used to assess the anomaly detection model as they are not suitable for real-time detection that requires low latency.

The clustering-based detection is an unsupervised learning method based on pattern similarity. Like our approach, a time series is partition into fixed-length fragments according to the day, but the fragments are then transformed into a frequency domain by Fourier transformation. Thus, a segment results in a k-dimensional vector in the frequency domain, k being the parameter of the transformation process. It then uses the dimensionality reduction algorithm MDS (i.e., multi-dimensional scaling) to obtain a low-dimension Euclidean embedding of n observations in a $d << k$ (i.e., $\Re^d$). For each point $y_i \in \Re^d$ in the low dimensional space, a local density is estimated as

$$f(y_i) = \frac{k}{\text{Vol. of smallest hyper-sphere containing } k\text{-NNs of } y_i} \tag{8}$$

The probability of an observation being an anomaly is computed as

$$P_r(y_i) = 1 - \frac{f(y_i)}{\max_{j=1,\ldots,n} f(y_j)} \tag{9}$$

Figure 8 shows an example of scaling a k-dimensional vector into a two-dimension space by dimension reduction (see lower left corner canvas). The density distribution in the reduced MDS is interpreted as an anomaly score. A point with low dense neighbors is assumed to reflect the unusual consumption behavior (e.g., the Day 3 in Figure 8), while the points with many neighbors are assumed to be normal, e.g, Day 1 and 2.

The second baseline is a prediction-based detection method like ours. This method averages the readings at a specific hour of all days, and uses the mean value as the predicted value.

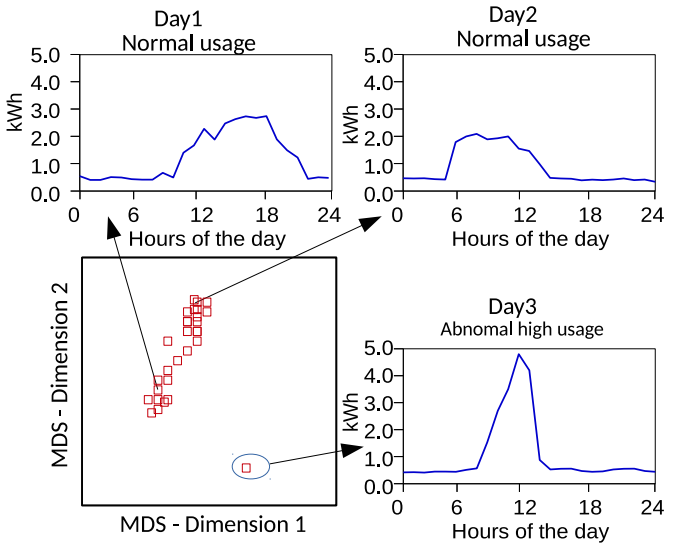


Figure 8: Example of clustering-based anomaly detection (reproduce from [4])

To emphasize recent development of a time series, this method uses weighted averages, which linearly decreases the weights for previous values according to time distances (see [19] for further details). A higher weight is given to the recent values. After predicting the values within a certain time window, it then uses the statistical Gaussian method to determine anomalies.

The third baseline is the Boxplot method, which has been used for examining numeric data outliers for decades. Boxplot uses the following five parameters to describe a data set (see Figure 9): lower fence, lower quartile, median, upper quartile and upper fence. A boxplot is represented by a rectangle for the upper and the lower quartile, and by a solid line for the median. The length between the upper and the lower quartile is the interquartile range, IQR. If a data point lies outside the lower or the upper fence, i.e., $1.5 * IQR$, it will be classified as an outlier. Boxplot has been indicated to be an acceptable approach for identifying outliers in most situations [15].

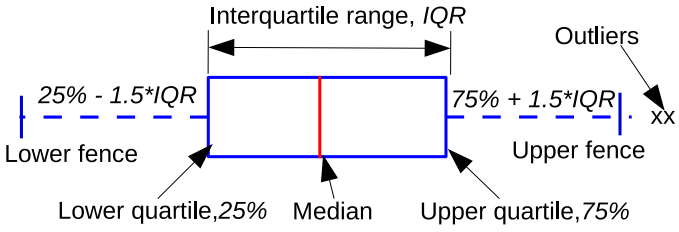


Figure 9: Description of Boxplot

Boxplot is usually applied to the points with a solo dimension. Therefore, in our case, we calculate the distances between the vectors of daily pattern and the mean of the vectors, then apply the Boxplot method. For example, the vector for the ith day is $v_i =< v_0, v_1, ..., v_{23} >$, and the vector of the mean of all days is $\bar{v} =< \bar{v}_0, \bar{v}_1, ..., \bar{v}_{23} >$. The Euclidean distance is

$$d^{(i)}(v_i, \bar{v}) = \|v_i - \bar{v}\|_2 \quad (10)$$

Boxplot detects outliers in the distance values, $d^{(1)}, d^{(2)}, ..., d^{(n)}$, for the n days of a time series.

4. Evaluation

In this section, we will evaluate the proposed anomaly detection system, including the effectiveness of the detection models, and the scalability of the lambda system.

4.1. Datasets

The test data include a real-world residential electricity consumption data set and a synthetic data set. The real-world data set consists of 27,300 time series, two years in length and hourly resolution. The size of the synthetic data set is up to one terabyte, corresponding to over twenty million time series. The data was generated by the time series data generator developed in our previous work [16], which used the real-world data set as the seed. In addition, we use outdoor temperature time series as the exogenous variable in our prediction model. We use the real-world data set to assess anomaly detection accuracy, and use the synthetic data set to assess system scalability.

4.2. Experimental Settings

The experiments were conducted in a cluster of 17 servers. We allocated the servers for the lambda system as follows: 5 servers for the speed layer, 12 servers for the batch layer, and a speed layer server also used for the serving layer to store detection models and notify users. All servers have the same configuration as follows: an Intel(R) i7-4770 CPU (3.40GHz 4 Cores), 16GB RAM, and a Seagate Harddrive (1TB, 6 GB/s, 32 MB Cache and 7200 RPM). The servers run 64bit Ubuntu 12.04. PostgreSQL 9.4 is installed in the serving layer, with the settings: "shared buffers=4096MB, temp buffers=512MB, work mem=1024MB, checkpoint segments=64" and default values for others.

4.3. Accuracy of the Prediction Model

We first evaluate the prediction model. We assume that residential consumption data follow a regular underlying pattern and therefore the model should be able to describe usual customer behaviors well. Otherwise, if the model cannot correctly interpret the observations, the predictions may be far from the observations, which will lead to the wrong classification. Prediction-based anomaly detection methods typically follow this idea and are related to the statistical measurement of the distances between the predicted and the observed values. The prediction method used is crucial for achieving good detection accuracy.

Table 1 shows the coefficient validity of the PARX model generated by a randomly selected time series from the real-world data set. As shown in this table, the coefficient estimates of the last three days' consumption values ($p = 3$) for a day d show very good significance, which is the same as the temperature coefficient estimates (see the p-test values). The number of "*"indicates the level of significance.

Table 1: The validity of the coefficients of the PARX model

Explanatory variable	Coefficient estimate	Std. error	t-value	Two-tailed p-test	Significance
Intercept	0.504	0.0729	6.92	1.33e-11	***
y_{d-1}	0.316	0.0406	7.79	3.58e-14	***
y_{d-2}	0.108	0.0387	3.45	0.001	**
y_{d-3}	0.133	0.0422	2.56	0.0107	*
$XT1$	0.194	0.0189	10.24	1.34e-22	***
$XT2$	-0.029	0.0085	-3.38	0.001	**
$XT3$	0.052	0.0193	2.68	0.008	**

0 '***', 0.001 '**', 0.01 '*', 0.05 '.', 0.1 " Adjusted R^2: 0.6341, n=534

We now use 10% randomly selected households (2,730) from the real-world data set to assess the prediction capability of PARX. For each time series, we use a quarter of the readings as training data to create the model (i.e., 6 months), and the rest are used as test data. In the test, we update the detection models iteratively for each day, and extend the training data set by adding the test data that have been used. We compare PARX with the following methods: 1) *averaging* which uses the mean value at a particular hour of the day as the predicted value for this hour in the next day; 2) *weighted averaging* which predicts a value by averaging weighted values at the same hour in previous days, and the weights linearly decrease according to the distance to the time of the measure [19]; 3) *3-Line* which uses a three-piecewise linear regression algorithm to predict new consumption with outdoor temperature as one of its independent variables [6]; and 4) *typical daily profile (TDP)* [13] uses clustering to compute a typical daily load profile, and uses it for the prediction. For all methods, we compute their root-mean-square error (RMSE) and make the comparison. RMSE is a frequently used measurement to quantify the difference between the predicted values of a model or an estimator and the actual observed values. It is defined by the following equation:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{11}$$

where $\hat{y}_i$ is the predicted value, y_i is the actual value, and n is the number of data points. We obtain the following findings by comparing their RMSE values:

- PARX outperforms the averaging method for 2,586 time series, the weighted averaging for 2,376 time series, the 3-Line for 2,612 time series, and the TDP for 2,508 time series.
- Table 2 summarizes the mean values of RMSE over 2,730 time series for all the methods. PARX is 13.3% lower than the averaging, 7.7% lower than the weighted averaging, 21.7% lower than the 3-Line, and 17.2% lower than the TDP.
- Figure 10 illustrates the mean RMSE values for all the methods for 10 randomly selected households over one year.

Table 2: The mean RMSE values over 2,730 time series

Prediction Method	RMSE
3-Line	0.92
TDP	0.87
Averaging	0.83
Weighted averaging	0.78
PARX	0.72

According to the above results, PARX outperforms the others with the lowest prediction error. The averaging method predicts a new value using the mean of the values at the same hour in previous days. This approach takes into account seasonality, but ignores the impact of outdoor temperature. Compared with the averaging method, the weighted averaging method gives a higher weight to adjacent values, which improves prediction accuracy to some extent. Its RMSE value is lower than the averaging method, which sounds reasonable as energy consumption patterns are often serially correlated [2]. The 3-Line method uses three piecewise linear regression lines to fit the relation between consumption and temperature, but it does not consider seasonality. On the other hand, The TDP method uses cluster centroids to predict new values. However, energy consumption has a variety nature, due to many factors such as the area of building, temperature, and occupancy. 3-Line and TDP both are the rough methods for prediction, which can be evidenced by their high RMSE values. In contrast, the PRAX method considers a variety of factors that can affect energy consumption, including adjacent readings (auto-regression), seasonality, and exogenous variables (weather temperature in our example). The lowest RMSE value confirms that PARX has good prediction accuracy. Nevertheless, we can observe that its mean RMSE value is still relatively high (greater than 0.7). If we explore individual households, we can find that the RMSE values differentiate substantially between households (see Figure 10). The reason is that energy consumption patterns have high variability, which is related to living habits of different residents, weather conditions, occupancy, and many others. It is usually difficult to predict the consumption of the households with high variability, e.g., those with irregular living habits. In this experiment, as we have used the same prediction model for all households, regardless of their variability, this results in a higher RMSE value. For optimization, we can first compute consumption pattern variability using an entropy method [2],

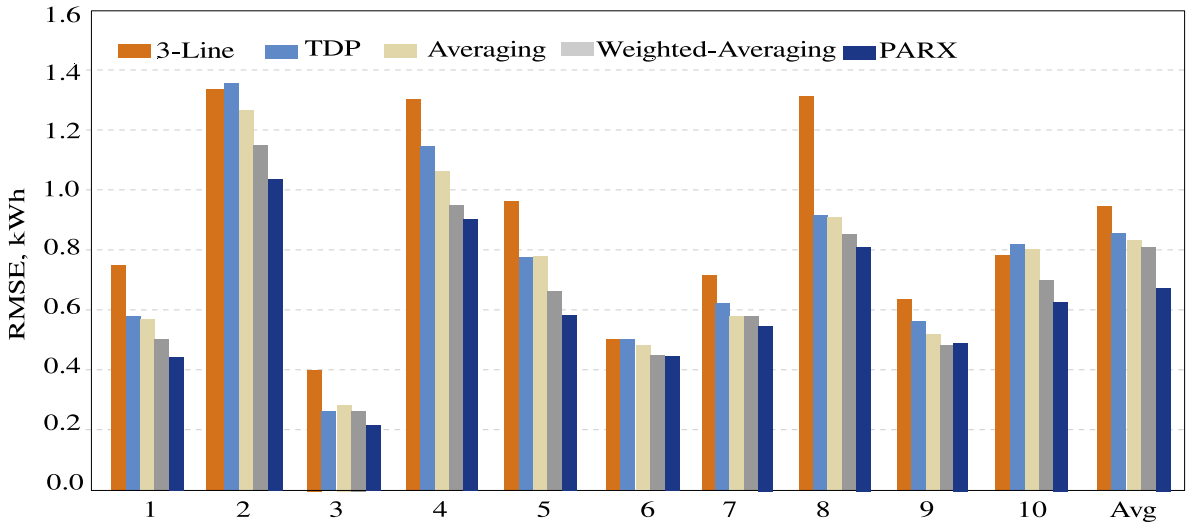


Figure 10: Average RMSE values of the five prediction methods on 10 randomly selected households over one year

and identify the households with low variability, then uses these household data to create the prediction model. This is left to our future work.

4.4. Impact of Model Update Frequency

As discussed in Section 3.5, the benefit of using the lambda architecture is that the models can be iteratively updated at regular time intervals, while the detection performance is highly dependent on prediction accuracy. In the following, we will evaluate the impact of model update frequency on prediction performance. In statistics, the confidence interval is used to estimate the probability of the true value of a parameter in the form: *estimate +/- variance of error*. According to Equation (1), for each season, when the variance of the error term is given, the confidence intervals can be calculated for the prediction, for example, using standard inference techniques. Figure 11 describes time series of observations and predictions, and the 95% confidence intervals of four consecutive days. Figure 11(a) is the result of the model updated every hour, and Figure 11(b) is the result of the model updated every day. The observed time series values (black line) are compared with the predicted values (red line). The 95% confidence intervals are indicated by the dashed lines, which have a different width over the days, due to the change of error variances. Obviously, the confidence intervals become wider when the prediction model is refreshed daily. In other words, frequent model updates can optimize prediction accuracy.

In order to quantify the prediction performance when the model is updated in different frequencies, we compute the Mean Absolute Percentage Error (MAPE) of 2,730 sampling time series over 4 days. Table 3 shows the number of time series contained in each category for different error levels. According to the results, PARX can produce better forecasting results (less than 3% error) for 2,295 time series when the prediction model is refreshed every hour. When the model is refreshed daily, there are 1,601 time series whose errors are below 3%. Clearly, the prediction performance is improved for a higher update frequency.

Table 3: Prediction errors for different model update frequencies

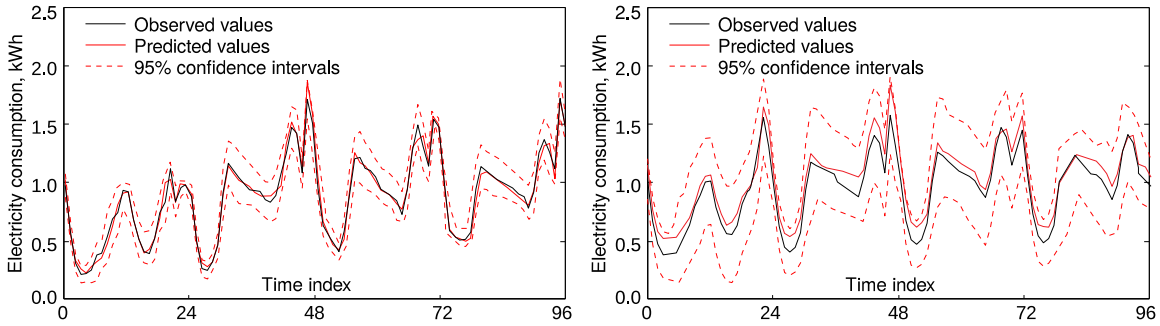
	Update every hour			Update every day		
MAPE	<1%	<3%	<5%	<1%	<3%	<5%
	1.987	2,295	2.324	125	1,601	1,823

4.5. Detection Accuracy

In the following, we will study the performance of the proposed detection method under different settings, and compare it with baseline methods.

4.5.1. Detect Under Various Settings

We now detect the anomalies for a single time series. We first use a histogram to depict the distribution of Euclidean distances for a particular season (see Figure 12). The shape of the histogram shows that the distances satisfy a lognormal distribution for this season. In fact, all 24 seasons have similar shapes where lognormal distribution can fit well. Moreover, residential electricity consumption is closely related to the living habits of residents, e.g., people often get up early for work on weekdays. In the second experiment, we test anomalies according to different day types, including weekday, weekend & holiday. Figure 13 shows the results under different thresholds, ϵ, ranging from 0.05 to 0.15. Obviously, when the days are treated differently, the overall identified anomalies for different day types are less than treating all days as the same type, due to better prediction accuracy. In terms of the threshold values, the number of anomalies is highly correlated, e.g, more are classified as anomalies when the threshold value increases. In real-world applications, the threshold values can be left to residents to decide and set, i.e., what consumption should be regarded as an anomaly, and when to receive an alerting message. We now compare the anomaly detection using Euclidean distance and KSC distance, respectively (see the distance metrics in Section 3.4). Figure 14 shows the number of detected anomalies for the two distance metrics at different threshold values. As shown in the figure, when KSC distance metric is used, the number of anomalies nearly halves. This is because



(a) Update every hour (b) Update every day

Figure 11: Impact of the PARX model update frequency

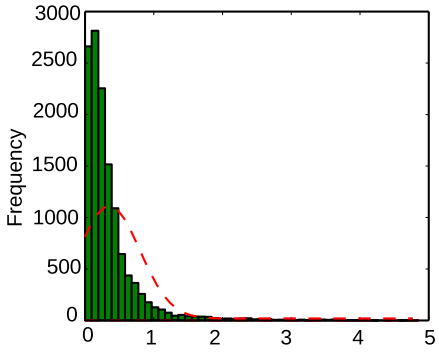


Figure 12: Lognormal distribution of the distances

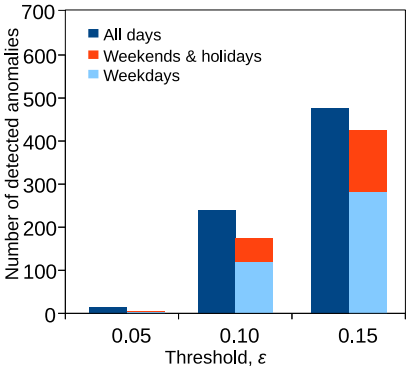


Figure 13: Detection under different day types

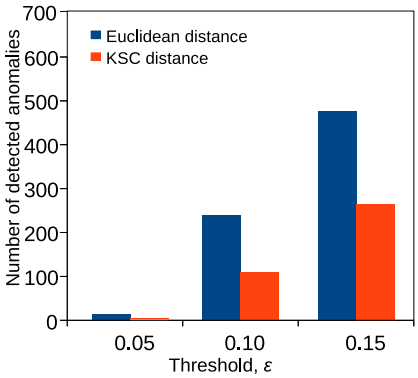


Figure 14: Detection under different distance metrics

the overall consumption pattern has not changed, but has shifted slightly along the time axis. This corresponds to, for example, residents getting up a little later than usual, which should not be classified as an anomaly.

4.5.2. Compare with the Baselines

We now compare the proposed anomaly detection system with the baselines listed in Section 3.6. Since it is hard to obtain labeled data, we use the electricity consumption data in the period with the least impact by outdoor temperature as the training data set, i.e., the time series of our data from spring to early summer, while use the data that are seriously affected by outdoor temperature as test data, i.e., the time series of summer. Figure 15 shows a typical residential consumption time series during this period. As shown in the figure, the time series has many peaks from June 10 to July 31, due to using air conditioners for cooling. We regard these peaks as anomalies if comparing with the historical consumption patterns without using an air conditioner. We now use the proposed prediction-based detection method and the baseline methods to detect anomalies of July 2012. Figure 16 shows the results. As shown, the patterns that are classified as anomalies are fewer by the prediction-based methods (PARX and weighted-averaging) than the clustering-based and the Boxplot method. As a prediction-based method uses historical data for prediction, it is well suited to detect the first occurrence of a new pattern (as an anomaly). As anomaly detection is often associated with contextual factors, detected anomalies can be used to signal people for a further investigation to decide if it is a real anomaly or not. For example, if an air conditioner is used during summer and creates an unusual pattern, this new pattern should not be classified as an anomaly.

As shown in Figure 16, our method classifies the least number of anomalies for reoccurring new patterns. This can be further explained by the following experiment. We zoom in the anomaly detection for a particular time series shown in Figure 15 (This figure only illustrates the time series from Jun 10, 2012 to Jul 31, 2012). Table 4 shows the days of daily pattern anomalies. We can observe that the PARX method only classifies the 5th and 6th days as anomalies. The reason is that these two days have a high consumption pattern that has not appeared before. When these two days' data have been used as the incremental data for training the models, the onward high consumption days are classified to be normal (recall that PARX looks back three days for a prediction). In contrast, the clustering-based method treats all high consumption days as anomalies, and the Boxplot method does the same. The weighted averaging method has not classified the days of anomalies until the consumption data are smoothed for a few days. By the comparison, the proposed PARX method is more suitable for early detection of new patterns ("abnormal") that are different to historical patterns. This can provide the information for utilities or customers to investigate, and decide whether they need to take appropriate actions or not.

4.6. System Scalability

In this section, we will evaluate system scalability on a cluster by using different degrees of parallelism, and different sizes

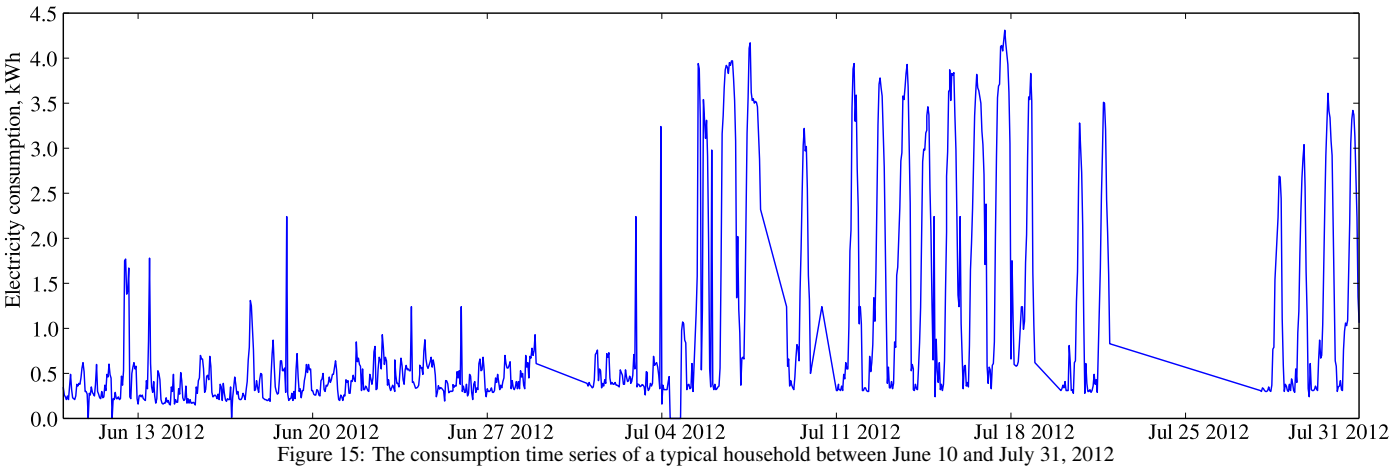


Figure 15: The consumption time series of a typical household between June 10 and July 31, 2012

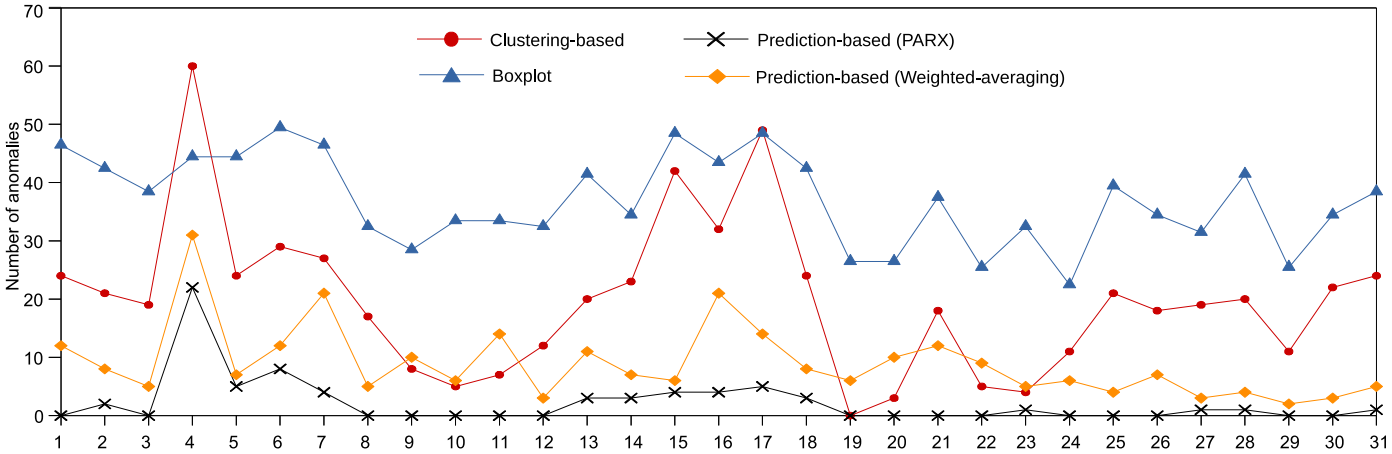


Figure 16: Compared with the baselines, July 2012

Table 4: The anomalies detected from the time series of July 2012

	Days of anomaly
Clustering-based	5, 6, 7, 9, 11, 12, 13, 14, 15, 16, 17, 18, 21, 30, 31
Boxplot	5, 6, 7, 9, 10,11, 12, 13, 14, 15, 16, 17, 18,19, 21, 30, 31
Prediction-based (PARX)	5,6
Prediction-based (Weighted-averaging)	5, 6, 7, 9, 11,12

of data set.

4.6.1. Scale-out Experiment

We now evaluate the proposed system for a large data set. In this experiment, the baseline methods are not evaluated as they are not suitable for real-time anomaly detection. As mentioned earlier, Spark is used to refresh the detection models interatively, and Sparking Streaming is used to detect anomalies in real time. To assess the system scalability, we conduct the experiments by changing the number of executors of Spark, and use a fixed-size synthetic data set, 8 million time series in one year (275 GB). The data were generated by our consumption time series generator [1]. We conduct the experiments using the full set of the physical servers, and scale the number of Spark executors from 8 to 256 to evaluate the capacity of batch jobs for training the models. We repeat each test for ten times, and report the statistical execution times using the Boxplot method (see Figure 17). According to the results shown in the figure, the execution time and the variance decrease when more executors are added. However, when the number of executors reaches 64, the increased parallelism does not further speed up the batch processing performance, which might be due to the overhead of managing a large number of executors in Spark. We now evaluate the performance of real-time anomaly detection on Spark Streaming. Again, we have increased the number of executors from 8 to 256. As we are only interested in the system scalability, we use same detection models for all the tests, i.e., without updating the models residing in the serving layer. Figure 18 in-

[1] Available at https://github.com/xiufengliu/DataGenerator-Cluster-Version

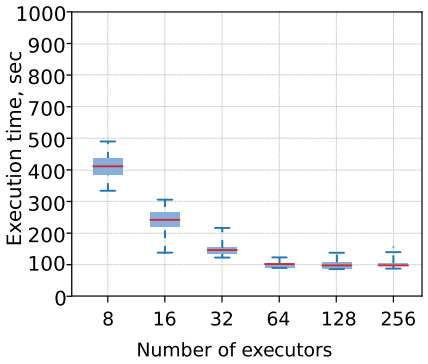


Figure 17: Batch model training

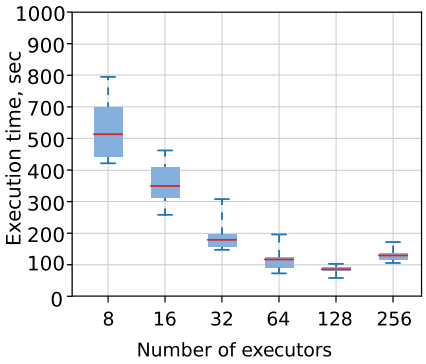


Figure 18: Real-time detection

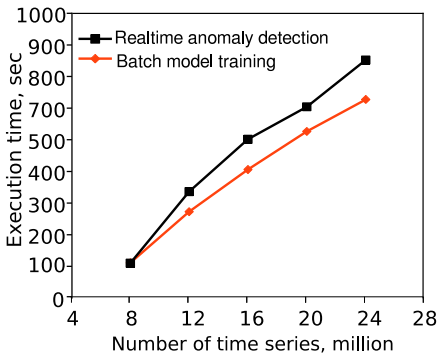


Figure 19: Size-up experiment

dicates that the variance of the execution times is greater than the model training in batch jobs (see Figure 17). This can be explained by the fact that the real-time batch of Spark Streaming has high execution time variability (For clarification, the concept of the real-time batch in Spark Streaming is called *pulse*, which is different to the batch job in the lambda system).

4.6.2. Increased Load Experiment

To evaluate the system scalability for large-scale data sets, we compare different workloads. According to the above experiments, the optimal number of executors for the model training and the detection is 64 and 128, respectively. In this experiment, we choose the best executor number (executor memory is set to 4GB), but increase the number of time series from 8 to 24 million (corresponds to the size from 275 to 825GB). Figure 19 shows the processing times for different workloads. We can observe that the training and the detection process can scale linearly with the number of time series. In this case, the detection time is the total execution time of all time series of one year, i.e., the job takes less than 2 minutes to process 8 million time series under the optimal settings. The average time for each real-time batch is only a few seconds (recall that a batch of Spark Streaming processes the data of one hour). For a real-world deployment, the detection program can be set to run every hour to check hourly readings of smart meters. According to the results, the lambda system has good scalability, and has the capability to handle large-scale smart meter data in parallel.

Unlike the real-time anomaly detection in the speed layer, the training process in the batch layer uses a full set of the data to re-generate the models in each iteration. For our experiments, the real-time detection is performed every hour, and Spark Streaming runs periodic batches (or pulses) to process the data. This results in more total time because of the starting overhead of each batch. The training and the detection program can be deployed in a different cluster or in the same cluster. If deployed in the same cluster, the system requires a reasonable way to allocate computing resources properly. For example, because a batch job takes more time, it can be scheduled to run immediately after a real-time job. A scheduling system for coordinating batch and real-time jobs is needed in order to maximize the usage of computing resources of a cluster, and this will be our future work.

5. Conclusions and Future Work

Analysis and detection of anomalies are crucial in the life cycle of smart meter data management, while the implementation of an online anomaly detection system is challenging. In this paper, we have proposed a novel concept that applies the lambda architecture to the anomaly detection for high frequent energy consumption data. The proposed system supports iterative model updates and real-time anomaly detection. We have proposed a prediction-based detection method that is able to detect daily pattern anomalies, as well as the patterns of other window sizes. We have compared the proposed detection method with three baseline methods, and evaluated the lambda system scalability for processing large-scale data on a cluster. The results have shown that the proposed method has good detection accuracy, and the implemented detection system has good scalability. This paper has validated the effectiveness of using a lambda system to achieve real-time capability for processing big data.

For future work, we will study the anomaly detection for the time series with high variability. We plan to explore how to detect a wide range of anomalies, including missing values, negative energy consumption, device errors and more. In addition, we will provide the support for other energy consumption data, including gas, heating, and water, and implement corresponding detection algorithms.

Acknowledgements

This research is supported by the CITIES project (NO. 1035-00027B) funded by Danish Innovation Fund.

References

[1] Adnan R, Setan H, Mohamad MN. Multiple Outliers Detection Procedures in Linear Regression. Matematika 2003; 19:29–45.

[2] Albert A, Gebru T, Ku J, Kwac J, Leskovec J, Rajagopal R. Drivers of variability in energy consumption. In: ECML-PKDD DARE workshop on energy analytics 2013.

[3] Ardakanian O, Koochakzadeh N, Singh RP, Golab L, Keshav S. Computing Electricity Consumption Profiles from Household Smart Meter Data. In: Proc. of EDBT/ICDT Workshops 2014; 14:140–147.

[4] Bellala G, Marwah M, Arlitt M, Lyon G, Bash C. Following the electrons: methods for power management in commercial buildings. In: Proc. of the 18th ACM SIGKDD international conference on knowledge discovery and data mining 2012; 994–1002.

[5] Brown M, Barrington-Leigh C, Brown Z. Kernel Regression for Real-Time Building Energy Analysis. Building Performance Simulation 2011; 5(4):263–276.

[6] Birt BJ, Newsham GR, Beausoleil-Morrison I, Armstrong MM, Saldanha N, Rowlands IH. Disaggregating categories of electrical energy end-use from whole-house hourly data. Energy Build 2012; 50:93–102

[7] Casado R, Younas M. Emerging trends and technologies in big data processing. Concurrency and Computation: Practice and Experience 2015; 27(8):2078–2091.

[8] Chandola V, Banerjee A, Kumar V. Anomaly Detection: A Survey. ACM Computing Surveys 2009; 41(3):15.

[9] Yang J, Leskovec J. Patterns of temporal variation in online media. In: Proc. of the 4th ACM international conference on Web search and data mining 2011; 177-186.

[10] Cheng B, Longo S, Cirillo F, Bauer M, Kovacs E. Building a Big Data Platform for Smart Cities: Experience and Lessons from Santander. In: IEEE International Congress on Big Data 2015; 592–599.

[11] Chou JS, Telaga AS. Real-Time Detection of Anomalous Power Consumption. Renewable and Sustainable Energy Reviews 2014; 33:400–411.

[12] Depuru SSSR, Wang L, Devabhaktuni V. Smart Meters for Power Grid: Challenges, Issues, Advantages and Status. Renewable Sustainable Energy Reviews 2011; 15(6):2736–2742.

[13] Espinoza M, Joye C, Belmans R, DeMoor B. Short-term load forecasting, profile identification, and customer segmentation: a methodology based on periodic time series. IEEE Trans Power Syst. 2005; 20(3):1622–1630.

[14] De Nadai M, Van Someren M. Short-Term Anomaly Detection in Gas Consumption through ARIMA and Artificial Neural Network forecast. In: IEEE Workshop on Environmental, Energy and Structural Monitoring Systems 2015; 250–255.

[15] Frigge M, Hoaglin DC, Iglewicz B. Some implementations of the boxplot. The American Statistician 1989; 43(1):50–54.

[16] Iftikhar N, Liu X, Nordbjerg FE, Danalachi S. A Prediction-based Smart Meter Data Generator. In: Proc. of the 19th International Conference of Network-based Information Systems (NBIS) 2016.

[17] Gao X. Scalable Architecture for Integrated Batch and Streaming Analysis of Big Data. Doctoral dissertation, Indiana University 2015.

[18] Hasani Z. Implementation of Infrastructure for Streaming Outlier Detection in Big Data. In: Proc. of the World Conference on Information Systems and Technologies, pp. 503–511, 2017.

[19] Hao MC, Janetzko H, Mittelstadt S, Hill W, Dayal U, Keim DA, Sharma RK. A Visual Analytics Approach for PeakPreserving Prediction of Large Seasonal Time Series. Computer Graphics Forum 2011; 30(3):691–700.

[20] Herrerias MJ. Seasonal anomalies in electricity intensity across Chinese regions. Applied Energy 2013; 112:1548-1557.

[21] Hill DJ, Minsker BS. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. Environmental Modelling & Software. 2010 Sep 30;25(9):1014-22.

[22] Jakkula V, Cook D. Outlier Detection in Smart Environment Structured Power Datasets. In: Proc. of the 6th International Conference on Intelligent Environments 2010; 29–33.

[23] Janetzko H, Stoffel F, Mittelstadt S, Keim DA. Anomaly Detection for Visual Analytics of Power Consumption Data. Computers & Graphics 2014; 38:27–37.

[24] KroßJ, Brunnert A, Prehofer C, Runkler TA, Krcmar H. Stream Processing on Demand for Lambda Architectures. EPEW 2015; 9272:243–257.

[25] Lee AH, Fung WK. Confirmation of Multiple Outliers in Generalized Linear and Nonlinear Regressions. Computational Statistics and Data Analysis 1997; 25(1):55–65.

[26] Liu F, Jiang H, Lee YM, Snowdon J, Bobker M. Statistical Modeling for Anomaly Detection, Forecasting and Root Cause Analysis of Energy Consumption for a Portfolio of Buildings. In: Proc. of 12th International Conference of the International Building Performance Simulation Association 2011.

[27] Liu G, Zhu W, Saunders C, Gao F, Yu Y. Real-Time Complex Event Processing and Analytics for Smart Grid. Procedia Computer Science 2015; 61:113–119.

[28] Liu X, Iftikhar N, Xie X. Survey of Real-Time Processing Systems for Big Data. In: Proc. of the 18th International Database Engineering & Applications Symposium 2014; 356–361.

[29] Liu X, Iftikhar N, Nielsen PS, Heller A. Online Anomaly Energy Consumption Detection Using Lambda Architecture. In: Proc. of DaWaK 2016; 193–209.

[30] Liu X, Nielsen PS. An ICT-Solution for Smart Meter Data Analytics. Energy, 2016

[31] Liu X., Thomsen, C., and Pedersen, T. B. 3XL: Supporting efficient operations on very large OWL Lite triple-stores. Information Systems, 36(4):765–781, 2011.

[32] Magld KW. Features Extraction Based on Linear Regression Technique. Computer Science 2012; 8(5):701–704.

[33] Martnez-Prieto MA, Cuesta CE, Arias M, Fernnde JD. The Solid Architecture for Real-Time Management of Big Semantic Data. Future Generation Computer Systems 2015; 47:62–79.

[34] Marz N, Warren J. Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Manning Publications Co. 2013.

[35] Meng X, Bradley J, Yavuz B, Sparks E, Venkataraman S, Liu D, Xin D. MLlib: Machine Learning in Apache Spark. arXiv preprint arXiv:1505.06807 2015.

[36] Moore DS, Mccabe GP. Introduction to the Practice of Statistics Ise. W.H. Freeman & Company, March 1999.

[37] Preuveneers D, Berbers Y, Joosen W. SAMURAI: A Batch and Streaming Context Architecture for Large-Scale Intelligent Applications and Environments. Ambient Intelligence and Smart Environments 2016; 8(1):63–78.

[38] Schneider, M., Ertel, W., Ramos, F.: Expected Similarity Estimation for Large-Scale Batch and Streaming Anomaly Detection. arXiv preprint arXiv:1601.06602 (2016)

[39] Sequeira H, Carreira P, Goldschmidt T, Vorst P. Energy Cloud: Real-Time Cloud-Native Energy Management System to Monitor and Analyze Energy Consumption in Multiple Industrial Sites. In: Proc. of the 7th IEEE/ACM International Conference on Utility and Cloud Computing 2014; 529–534.

[40] Villari M, Celesti A, Fazio M, Puliafito A. Alljoyn Lambda: An Architecture for the Management of Smart Environments in IOT. In: Proc. of IEEE International Conference on Smart Computing Workshops 2014; 9–14.

[41] Wang Y, Chen Q, Hong T, Kang, C. Review of Smart Meter Data Analytics: Applications, Methodologies, and Challenges. arXiv preprint arXiv:1802.04117, 2018.

[42] Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Stoica I. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Dluster Computing. In: Proc. of the 9th USENIX conference on Networked Systems Design and Implementation 2012; 2–2.

[43] Zaharia M, Das T, Li H, Shenker S, Stoica I. Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters. In: Proc. of the 4th USENIX conference on Hot Topics in Cloud Computing 2012; 10–10.

[44] Zhang Y, Chen W, Black J. Anomaly Detection in Premise Energy Consumption Data. In: Proc. of Power and Energy Society General Meeting 2011; 1–8.