electronics

Article

Binary-Convolution Data-Reduction Network for Edge–Cloud IIoT Anomaly Detection

Cheng Xie [1,*,†], Wenbiao Tao [1,†], Zuoying Zeng [2] and Yuran Dong [1]

[1] School of Software, Yunnan University, Kunming 650504, China; taowenbiao@mail.ynu.edu.cn (W.T.); dongyuran@mail.ynu.edu.cn (Y.D.)
[2] Broadvision Engineering Consultants Co., Ltd., Kunming 650041, China; zzuoying@163.com
* Correspondence: xiecheng@ynu.edu.cn
† These authors contributed equally to this work.

Abstract: Industrial anomaly detection, which relies on the analysis of industrial internet of things (IIoT) sensor data, is a critical element for guaranteeing the quality and safety of industrial manufacturing. Current solutions normally apply edge–cloud IIoT architecture. The edge side collects sensor data in the field, while the cloud side receives sensor data and analyzes anomalies to accomplish it. The more complete the data sent to the cloud side, the higher the anomaly-detection accuracy that can be achieved. However, it will be extremely expensive to collect all sensor data and transmit them to the cloud side due to the massive amounts and distributed deployments of IIoT sensors requiring expensive network traffics and computational capacities. Thus, it becomes a trade-off problem: *"How to reduce data transmission under the premise of ensuring the accuracy of anomaly detection?"*. To this end, the paper proposes a binary-convolution data-reduction network for edge–cloud IIoT anomaly detection. It collects raw sensor data and extracts their features at the edge side, and receives data features to discover anomalies at the cloud side. To implement this, a time-scalar binary feature encoder is proposed and deployed on the edge side, encoding raw data into time-series binary vectors. Then, a binary-convolution data-reduction network is presented at the edge side to extract data features that significantly reduce the data size without losing critical information. At last, a real-time anomaly detector based on hierarchical temporal memory (HTM) is established on the cloud side to identify anomalies. The proposed model is validated on the NAB dataset, and achieves 70.0, 64.6 and 74.0 on the three evaluation metrics of SP, RLFP and RLFN, while obtaining a reduction rate of 96.19%. Extensive experimental results demonstrate that the proposed method achieves new state-of-the-art results in anomaly detection with data reduction. The proposed method is also deployed on a real-world industrial project as a case study to prove the feasibility and effectiveness of the proposed method.

Keywords: anomaly detection; data reduction; edge–cloud; industrial internet of things (IIoT); deep learning; neural networks; case study

Citation: Xie, C.; Tao, W.; Zeng, Z.; Dong, Y. Binary-Convolution Data-Reduction Network for Edge–Cloud IIoT Anomaly Detection. *Electronics* **2023**, *12*, 3229. https://doi.org/10.3390/electronics12153229

Academic Editor: Carlo Mastroianni

Received: 21 June 2023
Revised: 22 July 2023
Accepted: 24 July 2023
Published: 26 July 2023

1. Introduction

Industrial internet of things (IIoT) sensors have been widely deployed and applied to continuously carry real-time anomaly detection [1] since it is one of the key ingredients for guaranteeing the quality and safety of industrial production [2]. Typically, data collected by IIoT sensors, such as servo motor power, filling pressure, ambient temperature, etc., can be further analyzed by an online time-series detector to identify anomalies before system failures really happen [3]. This requires massive sensor data to be obtained in time. In order to keep the low latency and network traffic during anomaly detection, current works apply edge–cloud IIoT architecture to connect these sensors [4]. The architecture is roughly divided by the edge side and the cloud side. The edge side is deployed very close to IIoT sensors but with a low computational capacity, while the cloud side is deployed far away from IIoT sensors but has a much higher computational capacity. In such architecture, the

edge side is normally responsible for collecting sensor data in time while the cloud side is responsible for accurate anomaly detection [5–9].

The more complete the data sent to the cloud side, the higher the anomaly-detection accuracy that can be achieved [10,11]. It thus becomes a trade-off problem, as higher anomaly-detection accuracy requires more complete sensor data to be sent to the cloud side. It inevitably increases overheads on network traffic and latency from the edge side to the cloud side. IIoT sensors collect large amounts of data in a very short period and continuously increase. Researchers estimate that a typical industrial manufacturer's production line generates several gigabytes of sensor data per day, which is often multiplied by a large number of sensors on the production line [12]. An extreme case is that a smart city with 1 million residents is estimated to generate 180 petabytes of sensor data per day [13]. Obviously, almost all these raw sensor data do not contain anomaly information, especially in a mature industrial environment. It is difficult and inefficient for the cloud side, instead of the edge–cloud architecture, to independently conduct the complete anomaly-detection task.

There are many works focusing on online time-series anomaly-detection approaches for edge–cloud IIoT environments. Cloud-oriented approaches focus on analyzing sensor data on the cloud-side, which requires continuous raw sensor data collection [14,15]. Elaborated data collectors are thus designed to reduce raw sensor data with data-filtering methods, such as down-sampling, rotation sampling, eigenvalue calculation, and representative data extraction. These approaches are feasible and effective when the amount of sensors is not very large, and the real-time performance is not essential for anomaly detection. Edge-oriented approaches conduct anomaly detection at the edge side, and only transmit the resulting data to the cloud side [5–7]. The edge side can obtain continuously and complete raw sensor data in the field with very low latency. But a high computational capacity and complex analyzing models are required at the edge side for time-series analysis. Edge-oriented approaches are useful in smart city and smart home scenarios since edge-side IoT devices, such as smart refrigerators, sweeping robots, smart charging stations, etc., are intelligent devices that have enough abilities to analyze anomalies by themselves. But edge-oriented approaches are infeasible for the industrial production environment because almost all IIoT sensors, such as turbidimeters, manometers, thermometers, etc., are non-smart devices that cannot actively analyze the data. Edge–cloud hybrid approaches discover data features from raw sensor data at the edge side, then detect anomalies based on these data features at the cloud side [8,9,16]. The anomaly-detection model proposed in this paper is also an edge–cloud hybrid approach, which significantly reduces the amount of data transmission and to some extent, maintains the data integrity. However, a trade-off problem still exists.

Anomaly Detection with Data Reduction. It is quite easy for online time-series anomaly detection to generate large amounts of real-time data. Increasingly with the edge data, to relieve the computing pressure in the cloud, edge–cloud hybrid approaches offload some tasks to the edge. However, the above approaches inevitably suffer from the problem that is "*the model should achieve a higher accuracy rate on anomaly detection but requires less amount of data transmission*". Therefore, how to balance the amount of raw sensor data transmission and the accuracy of anomaly detection becomes the research focus.

To this end, in the paper, a binary-convolution data-reduction network for edge–cloud IIoT anomaly detection is proposed. It collects raw sensor data and extracts their features at the edge side, while on the cloud side, only data features are received to discover anomalies. To implement this, a time-scalar binary feature encoder is proposed and deployed on the edge side that encodes raw data into time-series binary vectors. Then, a binary-convolution data-reduction network is presented at the edge side to extract data features that significantly reduce the data size without losing critical information. At last, a real-time anomaly detector based on hierarchical temporal memory (HTM) is established on the cloud side to identify anomalies. Extensive experimental results demonstrate the proposed method achieves new state-of-the-art results on anomaly detection with data

reduction. We also deploy the proposed method on a real-world industrial project as a case study to prove its feasibility and effectiveness.

In summary, the main contributions of this paper are as follows:

- We propose a new time-series data-encoding method called the time-scalar binary feature encoder that can significantly extract sensor data features with binary representations.
- We propose a binary-convolution data-reduction network that can extract binary data features without losing critical data information. It is useful for the edge side to pre-process the raw data and then transmit them to the cloud-side for further analysis.
- We propose a new hierarchical temporal memory-based detection model that achieves new state-of-the-art anomaly detection with data reduction. The model is also deployed on a real-world industrial project to present a representative case study.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the proposed online anomaly-detection method for time series. Section 4 analyzes the experimental results. Section 5 presents a real-world case study based on the proposed method. Section 6 concludes the work and discusses the future research lines.

2. Related Work

2.1. Online Time-Series Anomaly Detection

Online time-series anomaly detection is an emerging and interesting research field in recent years. However, due to its stringent requirements, very few of the research studies proposed in the anomaly-detection field can directly meet online time-series anomaly detection.

- EXPoSE (expected similarity estimation) [17]: an algorithm that determines anomalies by computing the deviation between input observations and the estimated distribution of past input values.
- Bayesian changepoint [18]: a Bayesian-based algorithm that detects a sudden changepoint.
- Skyline (https://github.com/etsy/skyline (accessed on 20 June 2023), https://github.com/earthgecko/skyline (accessed on 20 June 2023)): an ensemble learning algorithm that identifies anomalies when most detections are confirmed.
- Windowed Gaussian (https://github.com/numenta/NAB/blob/master/nab/detectors/gaussian/windowedGaussian_detector.py (accessed on 20 June 2023)): an anomaly-detection algorithm that determines anomalies based on the probability calculated by the new observation on the Gaussian distribution.
- Twitter ADVec (https://github.com/twitter/AnomalyDetection (accessed on 20 June 2023)): a method based on the seasonal hybrid ESD (S-H-ESD) algorithm. Extreme student deviations from the given time-series values are calculated for anomaly detection.
- Random cut forest [19]: an anomaly-detection algorithm published by Amazon, an improvement on the isolated forest.
- Relative Entropye [20]: a method that uses Kullback–Leibler divergence of two data distributions to decide whether the data are an anomaly.
- KNN CAD [21]: an algorithm based on K-nearest neighbors classification, which compares the observed values with reference values on the non-conformity measure that are calculated using the created caterpillar matrix to identify anomalies.
- CAD OSE (https://github.com/smirmik/CAD (accessed on 20 June 2023)): a method that determines anomalies if the contexts of the recent subsequence are significantly different from the past subsequences.

These methods judge anomalies based on previous data patterns but do not take full advantage of the impact of the current data on the future. From the perspective of prediction-based time-series methods, online anomaly detection has been improved [22]. Several representative proposals based on time-series prediction in recent years are as follows:

- HTM (hierarchical temporal memory) [23]: This is a representative unsupervised prediction-based method for online time-series anomaly detection. It implements a working mechanism similar to that of the cerebral cortex.
- OLAD (online non-parametric Bayesian method) [24]: This is a predictive algorithm with HTM as the base anomaly detector. It identifies anomalies when observations deviate from the modeled normality.
- OeSNN-UAD (the online evolving spiking NNs for unsupervised anomaly-detection framework) [25]: This is an online anomaly-detection algorithm based on OeSNN architecture but that works in an unsupervised way. With eSNNs, input values are labeled as inliers or outliers.
- EORELM-AD [26]: This is an anomaly-detection framework that contains streaming data normalization and online anomaly scoring and identification, enabling predictive algorithms to adapt to online time-series anomaly detection.

Compared to other methods, the HTM algorithm has the capabilities of continuous learning to handle concept drift, detecting subtle temporal anomalies, making fewer assumptions and adapting to diverse datasets [23]. Methods like Skyline are flawed in concept drift and non-parametric. EXPoSE meets the first three points, but depends on the size of the dataset and is more suitable for large-scale datasets with high-dimensional features. OeSNN-UAD relies on a large window to capture data features, causing a slow startup speed. EORELM-AD is an ensemble framework with a complex structure and a number of parameters. Also, due to the advanced encoding format of sparse distributed representations (SDRs), the HTM algorithm is faster, more robust, and more energy efficient than conventional neural networks [27]. In addition, for anomaly detection at a single point, the HTM algorithm can achieve millisecond-level analysis speed, which makes it achieve timely online real-time anomaly detection [23,28]. So far, HTM has been used by several enterprises, such as Grok (https://grokstream.com/anomaly-detection/ (accessed on 20 June 2023)), for anomaly detection.

The above properties enable HTM to be more suitable for IIoT anomaly detection, compared with other algorithms. In IIoT, a real-time anomaly detector is very important to ensure production safety. The fine-grained detection capability of HTM can meet the requirements of real-time monitoring in actual production, and its high robustness accommodates noise data in complex production environments [27]. So, in this paper, we improve HTM to make it more suitable for anomaly-detection tasks in the edge–cloud architecture.

2.2. Data Reduction for Edge–Cloud

Data-reduction methods for edge–cloud can be broadly divided into two categories, representative data sampling, and data features extraction.

2.2.1. Representative Data Sampling

These approaches calculate important values or results at the edge to minimize the data transferred to the cloud. Ref. [14] gave weights to data points to find important values but changed those values as a result. They are not suitable for applications that require actual values or ranges. Ref. [29] reduced the amount of data to be stored by studying problem approximations of the data stream. Ref. [15] proposed an adaptive moving average window sampling (AWBS) algorithm to reduce data, where the window size changes based on the changes in the incoming data. Ref. [30] passed the collected sensor data to the intelligent gateway to reduce them according to a defined pattern and then transmit them to the cloud. However, without complete information, the cloud can no longer perform further analysis.

2.2.2. Data Features Extraction

These approaches put part of the entire model at the edge to extract data features and reduce them. Tang et al. [8] proposed a hierarchical fog computing architecture for connected devices in smart city scenarios. They implemented feature extraction at the edge

to complete the corresponding machine-learning tasks. However, they only analyzed the mean and variance of the signal, so they lack the generalization of the application scenario. Refs. [9,16] place the encoder and decoder of the autoencoder in the edge and in the cloud, respectively, to enable feature extraction and data reduction at the edge. The former is used for image classification, while the latter transmits sensor data to the cloud to complete the classification of human activities. Such methods relieve computing pressure in the cloud but require computing power at the edge.

3. Binary-Convolution Data-Reduction Network for Edge–Cloud IIoT Anomaly Detection

As shown in Figure 1, our proposed method is mainly divided into three modules. The first module is time-scalar binary feature encoding, which collects production data at the edge and encodes them as time-scalar binary feature vectors for anomaly detection in the cloud. The second module is the binary-convolution data-reduction network, which contains feature smoothing for enhancing semantic expressiveness and binary feature reduction for feature compression. The last module is binary feature sequence anomaly detection, which detects anomalies in the current data based on the HTM model.

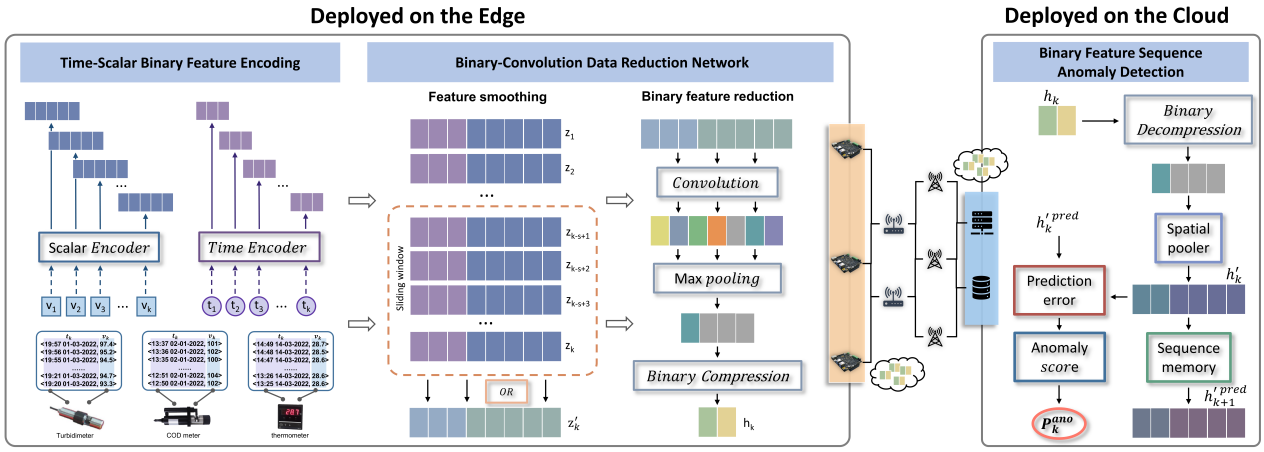


Figure 1. The overview of the proposed method. It can be divided into three modules. The first module, time-scalar binary feature encoding, is introduced in Section 3.1; the second module, binary-convolution data-reduction network, including feature smoothing and binary feature reduction, is detailed in Section 3.2; the last module, binary feature sequence anomaly detection, is explained in Section 3.3.

Throughout the whole model, the three modules play different roles and are closely linked. As shown in Figure 2, the middle module reduces the raw sensor data encoded by the first module in the edge for the last one to detect anomalies in the cloud with fewer data features.

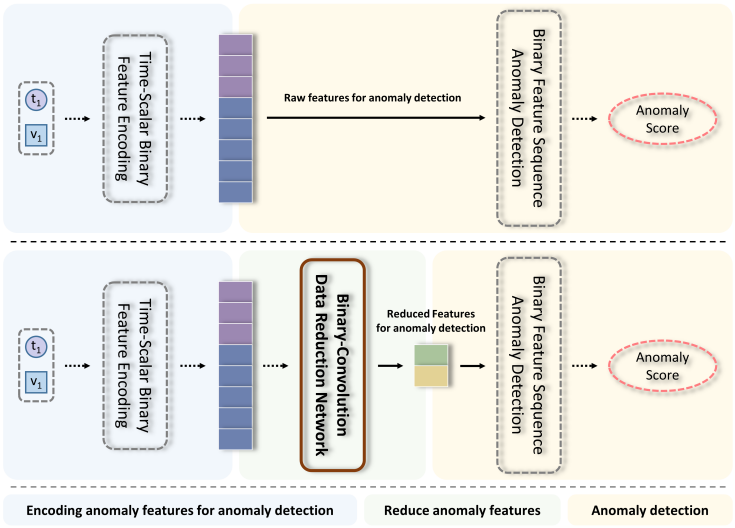


Figure 2. The relationship between the modules.

3.1. Time-Scalar Binary Feature Encoding

The raw data collected by IIoT sensors, consisting of "time" (t) and "value" (v), cannot be directly used for anomaly detection. This is because (1) there is no clear semantics that can be described among "t,v" records. (2) "t,v" records cannot be identified and used by the downstream machine learning models for anomaly detection and data reduction. A semantic-rich and learning-acceptable encoding is required. In this work, a famous scalar encoding format called sparse distributed representations (SDRs) [31] is referred to for "time"(t) and "value"(v) encoding. An SDR consists of a large array of bits, of which most are zeros and a few are ones. Each bit carries some semantic meaning, so if two SDRs have more than a few overlapping one bits, then those two SDRs have similar meanings.

In detail, we encode the time data by using the cyclic categories encoder ($\mathcal{E}_t$) and value data by the cochlea encoder ($\mathcal{E}_v$) [32], while $\mathcal{E}_t$ and $\mathcal{E}_v$ are the implementations of SDR. With $\mathcal{E}_t$, "time" (t) is converted to an SDR like "000111...11000...00", where the relative position of consecutive 1s reflects a certain time, just like the Monday of the week. Unlike time, which has cyclic properties, "value" (v) is better suited for discrete representations. $\mathcal{E}_v$ encodes "value" (v) into an intensively 1-distributed SDR and converts it to a discrete distribution like "00100...0100...0010" using a hash function. Then, the encoded time and value are concatenated into a time-scalar merged binary vector z_k, defined as the following equation:

$$z_k = \mathcal{E}_t(t_k) \oplus \mathcal{E}_v(v_k), \tag{1}$$

where k denotes the k-th record collected by an IIoT sensor. t_k and v_k represent the time and value of this record, respectively.

The process of encoding is illustrated in Figure 3. A turbidimeter produces turbidity values with time series. Then, the turbidity values are encoded into an SDR by $\mathcal{E}_v$, while time series are encoded by $\mathcal{E}_t$. At last, the time SDR and value SDR are concatenated into a time-scalar merged binary vector z_k. This will be the input data for the downstream learning tasks.

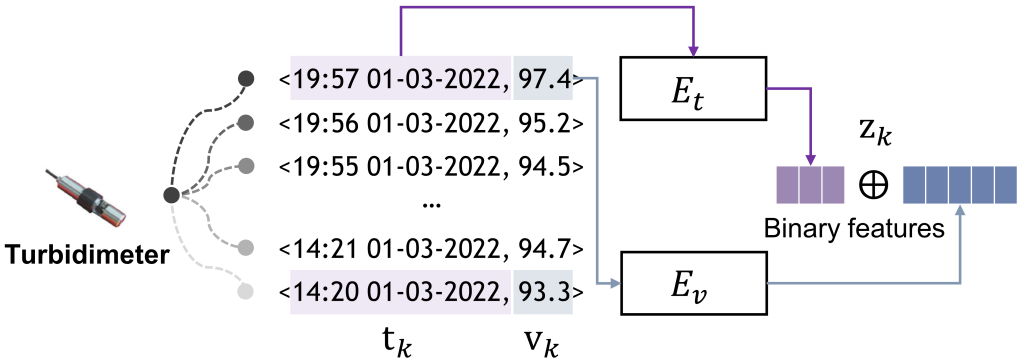


Figure 3. The process of time-scalar binary feature encoding.

3.2. Binary-Convolution Data-Reduction Network

As introduced in Section 3.1, the raw sensor data are encoded into a binary representation z_k on the edge side. Massive sensors continuously produce a large amount of z_k that is unable to be completely sent back to the cloud side in a timely manner. In this section, the binary-convolution data-reduction network (deployed on the edge side) is introduced to reduce z_k without losing significant information before sending it back to the cloud side.

3.2.1. Feature Smoothing

The encoded data feature z_k follows the SDR structure, according to which similar data (e.g., z_1 and z_2) hold more than a few overlapping one bits. The purpose of feature smoothing is to gather as many semantics as possible among previous data records to enhance semantic expressiveness, compensating for the lack of semantic information caused by the subsequent binary feature reduction. The idea of feature smoothing is to apply a sliding window (whose size is ρ) to combine features from z_k to $z_{k+\rho}$ by an "or" operation. The "or" operation combines corresponding bits to a one bit if a one bit exists, else to a zero bit. Since the overlapping of one bits holds semantics among data features, the sliding window, indeed, propagates semantics from the current feature to the next feature. The following equation defines the feature smoothing with a sliding window:

$$z'_k = z_{k-\rho+1} \text{ or } z_{k-\rho+2} \text{ or } \ldots z_k, \tag{2}$$

where z'_k is the data feature after smoothing. ρ is the size of the sliding window. An example of a feature-smoothing process is presented in Figure 4, with $\rho = 2$. The current feature is enhanced by the semantic representation of the previous time step, which enriches the current information.

3.2.2. Binary Feature Reduction

The feature smoothing, indeed, focuses on extracting data features from other adjacent data records. However, in this part, the proposed binary feature-reduction module focuses on extracting data features only from its own data record. It is designed to extract highly concentrated features but with a rather small data size. The input of the reduction module is the smoothed feature z'_k, and the output is the reduced data feature h_k. In detail, the three operations, a convolution network $Conv_\lambda(\cdot)$, a max pooling $Mp_\mu(\cdot)$ and a binary compression $BC(\cdot)$ are used. Both the convolution core and the max pooling core are one-dimensional vectors. In the experiments, the convolution core's size is $1 \times \lambda$, and the size of the max pooling core is $1 \times \mu$, with its stride also being μ. The binary compression is to merge all continuous zero bits into one bit, whose value is the number of zero bits. The reduction process is defined as the following equation:

$$h_k = BC(Mp_\mu(Conv_\lambda(z'_k))). \tag{3}$$

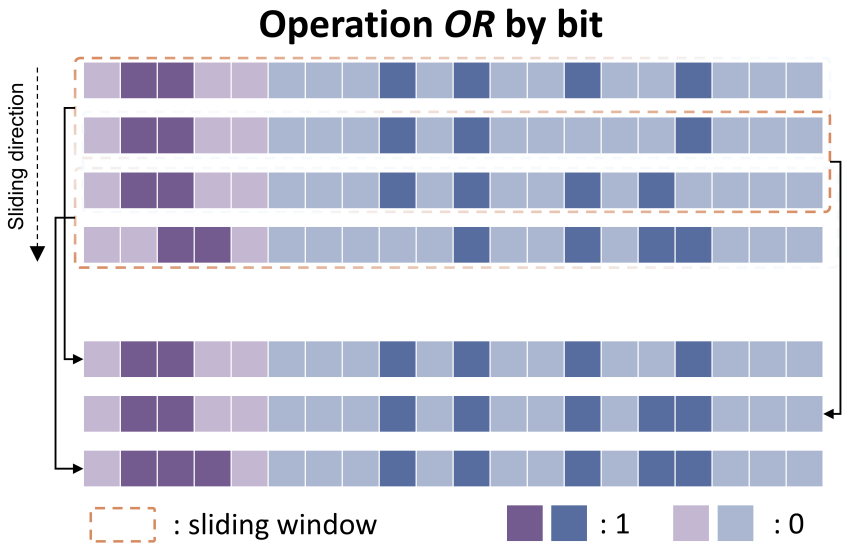


Figure 4. The process of feature smoothing.

As shown in Figure 5, the value part of each z' is calculated by the convolution core, from the first dimension to the last with step = 3. Then, the data feature is calculated by the max pooling core, also from the first dimension to the last with step = 3. After max pooling, the size of the data feature will be $1/\mu$ of the original feature. At last, the data feature is further compressed by a binary compression operation $BC(\cdot)$. It compresses all continuous zero bits into a single bit with the value representing the number of zero bits. Thus, the compression rate depends on the sparsity of the data feature and is marked as θ. The size of the final data feature will be close to θ/μ of the original feature size.

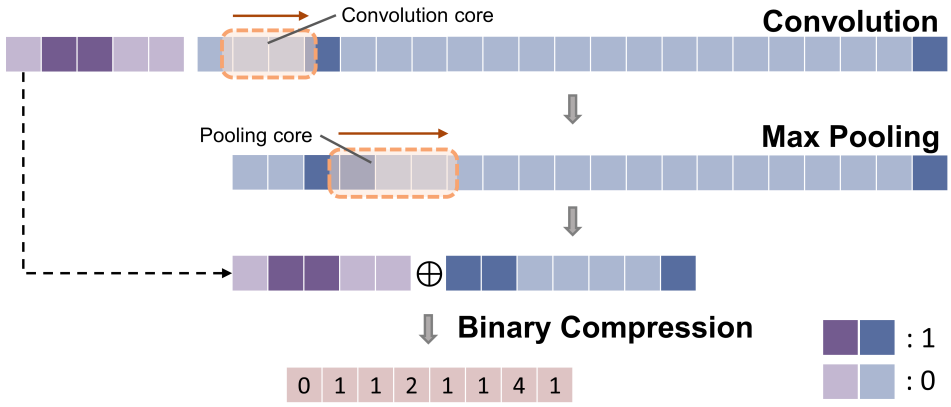


Figure 5. An example of binary feature reduction.

After convolution and max pooling, the value part feature is concatenated with its time part feature, and then becomes the reduced data feature h_k with binary compression. It is used for anomaly detection, which is described in the next section.

3.3. Binary Feature Sequence Anomaly Detection

The reduced data feature h_k is a sparse vector with zero bits and one bits. Based on h_k, the idea of prediction is to use a spatial network to activate h_k by the spatial relationships among the vector's one bits. We call this activated feature h'_k, which is a highly concentrated data feature. Then, based on h'_k, we push the time sequence to $k+1$ by using a sequence network to predict the next data feature ($k+1$ data feature). We call this predicted data

feature $h_{k+1}^{'pred}$. In the experiment and the case study, the spatial pooler [33] is used for feature activation, while sequence memory [28] is used for feature prediction, given as

$$\begin{aligned} h_k' &= SpatialPooler(h_k), \\ h_{k+1}^{'pred} &= SequenceMemory(h_k'). \end{aligned} \tag{4}$$

In detail, during the spatial pooler process, an SDR vector h_k from the binary-convolution data-reduction network is activated as a sparser SDR o_k according to Equation (5). The top 2% bits in o_k is converted to 1, and the rest is 0:

$$\begin{aligned} o_{ki} &= b_i \sum_j W_{ij} h_{kj}, \\ h_{ki}^{'} &= \begin{cases} 1 & \text{if } o_{ki} \text{ in top 2\% of } o_k \\ 0 & \text{otherwise} \end{cases}, \end{aligned} \tag{5}$$

where o_{ki} is the i-th element in o_k, and h_{kj} is the j-th element in h_k. W is a transformation matrix connecting h_k and $h_k^{'}$. b is the weight.

In sequence memory, we use a set of the matrix, $D = \left\{ D^d \mid d = 1, 2, ..., 128 \right\}$, to denote the permanence of the predictive state. If there is a D^d making the predictive state match the activated state beyond the threshold, the position of the predicted feature becomes one bit; otherwise, it is zero bit:

$$h_{k+1,i}^{'pred} = \begin{cases} 1 & \text{if } \exists_d \left\| D_i^d \odot h_k' \right\|_1 > \eta \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

where $h_{k+1,i}^{'pred}$ denotes the i-th element in $h_{k+1}^{'pred}$, D_i^d is the i-th vector in the matrix D^d, and $\odot$ represents element-wise multiplication.

Based on the prediction, we can calculate the errors between the predicted data feature and the actually happened data feature. An anomaly might happen if the errors continuously grow. In detail, the errors are divided into three categories that are single errors, short-term errors, and long-term errors. The single error $\mathcal{E}_k^{single}$ is measured between h_k' and $h_k^{'pred}$ as defined in Equation (7):

$$\mathcal{E}_k^{single} = 1 - \frac{h_k^{'pred} \cdot h_k'}{|h_k'|}, \tag{7}$$

where $h_k^{'pred}$ is the predicted data feature based on h_{k-1}'. And h_k' is the actually happened data feature.

Figure 6 presents the calculation process of $\mathcal{E}_k^{single}$. The long-term errors are the accumulation of single errors during a long-term window $\mathcal{W}_l$, while the short-term errors are accumulated within a relatively shorter-term window $\mathcal{W}_s$. Equation (8) provides the definition:

$$\begin{aligned} \mathcal{E}_k^{long} &= \frac{\sum_0^{|\mathcal{W}_l|} \mathcal{E}_{k-i}^{single}}{|\mathcal{W}_l|}, \\ \mathcal{E}_k^{short} &= \frac{\sum_0^{|\mathcal{W}_s|} \mathcal{E}_{k-i}^{single}}{|\mathcal{W}_s|}, \end{aligned} \tag{8}$$

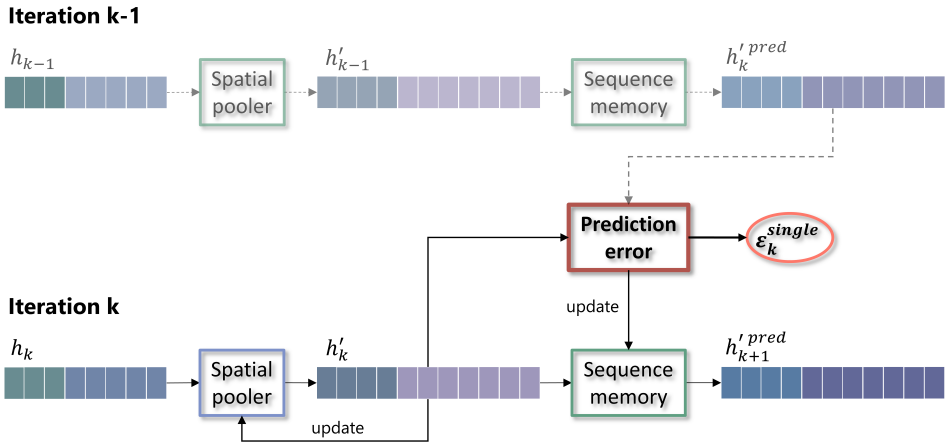


Figure 6. The calculation process of single error $\mathcal{E}_k^{single}$.

After, we calculate the distance between $\mathcal{E}_k^{short}$ and $\mathcal{E}_k^{long}$, and put the result into a Gaussian tail probability function $\mathcal{Q}(\cdot)$ [34] to measure the anomaly probability as defined in Equation (9):

$$\begin{aligned} \sigma_k &= \sqrt{\frac{\sum_0^{\mathcal{W}_l}\left(\mathcal{E}_{k-i}^{single} - \mathcal{E}_{k-i}^{long}\right)^2}{|\mathcal{W}_l|}}, \\ \mathcal{P}_k^{ano} &= 1 - \mathcal{Q}\left(\frac{|\mathcal{E}_k^{short} - \mathcal{E}_k^{long}|}{\sigma_k}\right), \end{aligned} \tag{9}$$

where σ_k is the standard deviation of error prediction in the long-term window. $\mathcal{P}_k^{ano}$ is the probability of anomaly detection. In the experiment, ϵ is a user-defined threshold. When $\mathcal{P}_k^{ano} \geq \epsilon$, an anomaly can be reported.

The whole anomaly-detection process can be seen in Algorithm 1.

Algorithm 1 Anomaly-detection process.

Input: $dataset$, ρ, λ, μ
Output: $\mathcal{P}_k^{ano}$

$Result_{set} = [\,]$
for (t_k, v_k) in dataset **do**
  $z_k \longleftarrow Encode(t_k, v_k)$
  $z'_k \longleftarrow FeatureSmoothing(z_k)$
  $h_k \longleftarrow DataReduction(z'_k)$
  $h'_k \longleftarrow SpatialPooler(h_k)$
  $h'^{pred}_{k+1} \longleftarrow SequenceMemory(h'_k)$
  $\mathcal{E}_k^{single} \longleftarrow CalcSingleError(h'_k, h'^{pred}_{k+1})$
  $\mathcal{E}_k^{long}, \mathcal{E}_k^{short} \longleftarrow CalcError(\mathcal{E}_k^{single})$
  $\mathcal{P}_k^{ano} \longleftarrow CalcAnomaliy(\mathcal{E}_k^{long}, \mathcal{E}_k^{short})$
  $Result_{set}.append(t_k, 1\ if\ \mathcal{P}_k^{ano} \geq \epsilon\ else\ 0)$
end for
return $Result_{set}$

4. Experimental Evaluation

4.1. Preparations

4.1.1. Datasets

The Numenta anomaly benchmark (NAB) [35] is used as the benchmark of the experiment. NAB consists of 58 time-series data files that are divided into seven categories:

"realAWSCloudwatch", "realAdExchange", "realKnownCause", "realTraffic", "realTweets", "artificialNoAnomaly" and "artificialWithAnomaly":

- realAWSCloudwatch: AWS server metrics including CPU utilization, network bytes in, and disk read bytes.
- realAdExchange: online advertisement clicking rates, where the metrics are cost per click (CPC) and cost per thousand impressions (CPM).
- realKnownCause: the data where we know the anomaly causes, with no hand labeling.
- realTraffic: real-time traffic data from the Twin Cities Metro area in Minnesota, including occupancy, speed, and travel time from specific sensors.
- realTweets: a collection of Twitter mentions of large publicly traded companies, such as Google and IBM.
- artificialNoAnomaly: artificially generated data without any anomalies.
- artificialWithAnomaly: artificially generated data with varying types of anomalies.

In the dataset, every record has two data items, time and value, for example (2014/4/1 5:40:00, 20). And every subdataset has thousands of pieces of data, totaling 329,270. More detailed information about NAB is summarized in Table 1.

Table 1. NAB datasets structure: 7 categories consisting of 58 subdatasets.

Categories	Subdatasets	Anomaly Num	Total	Examples
artificialNoAnomaly	art daily no noise	0	4032	(2014/4/1 5:40:00, 20)
	art daily perfect square wave			(2014/4/2 11:50:00, 80)
	art daily small noise			(2014/4/1 0:00:00, 18.3249185392)
	art flatline			(2014/4/1 1:50:00, 45)
	art noisy			(2014/4/1 0:00:00, 18.6221849224)
artificialWithAnomaly	art daily flatmiddle	403	4032	(2014/4/1 0:00:00, -21.0483826823)
	art daily jumpsdown			(2014/4/1 1:30:00, 18.1821019992)
	art daily jumpsup			(2014/4/1 0:00:00, 19.761251903)
	art daily nojump			(2014/4/1 0:00:00, 21.5980110405)
	art increase spike density			(2014/4/1 0:00:00, 20)
	art load balancer spikes			(2014/4/1 5:45:00, 0.1465598018)
realAdExchange	exchange-2 cpc results	163	1624	(2011/7/1 0:00:01, 0.0819647355164)
	exchange-2 cpm results	162	1624	(2011/7/1 0:00:01, 0.401048098657)
	exchange-3 cpc results	153	1538	(2011/7/1 0:15:01, 0.102708933718)
	exchange-3 cpm results	153	1538	(2011/7/1 0:15:01, 0.405422534525)
	exchange-4 cpc results	165	1643	(2011/7/1 0:15:01, 0.0917952281677)
	exchange-4 cpm results	164	1643	(2011/7/1 0:15:01, 0.61822635122)
realAWSCloudwatch	ec2 cpu utilization 5f5533	402	4032	(2014/2/14 14:27:00, 51.846)
	ec2 cpu utilization 24ae8d	402	4032	(2014/2/14 14:30:00, 0.132)
	ec2 cpu utilization 53ea38	402	4032	(2014/2/14 14:30:00, 1.732)
	ec2 cpu utilization 77c1ca	403	4032	(2014/4/2 14:25:00, 0.068)
	ec2 cpu utilization 825cc2	343	4032	(2014/4/10 0:04:00, 91.958)
	ec2 cpu utilization ac20cd	403	4032	(2014/4/2 14:29:00, 42.652)
	ec2 cpu utilization c6585a	0	4032	(2014/4/2 14:29:00, 0.066)
	ec2 cpu utilization fe7f93	405	4032	(2014/2/14 14:27:00, 2.296)
	ec2 disk write bytes 1ef3de	473	4730	(2014/3/3 7:59:00, 2423190)
	ec2 disk write bytes c0d644	405	4032	(2014/4/2 15:00:00, 19949200)
	ec2 network in 5abac7	474	4730	(2014/3/1 17:36:00, 42)
	ec2 network in 257a54	403	4032	(2014/4/10 0:04:00, 251643)
	elb request count 8c0756	402	4032	(2014/4/10 0:04:00, 94)
	grok asg anomaly	465	4621	(2014/1/16 0:00:00, 33.5573)
	iio us-east-1 i-a2eb1cd9 NetworkIn	126	1243	(2013/10/9 16:25:00, 9926554)
	rds cpu utilization cc0c53	402	4032	(2014/2/14 14:30:00, 6.456)
	rds cpu utilization e47b3b	402	4032	(2014/4/10 0:02:00, 14.012)

Table 1. *Cont.*

Categories	Subdatasets	Anomaly Num	Total	Examples
realKnownCause	ambient temperature system failure	726	7267	(2013/7/4 0:00:00, 69.88083514)
	cpu utilization asg misconfiguration	1499	18,050	(2014/5/14 1:14:00, 85.835)
	ec2 request latency system failure	346	4032	(2014/3/7 3:41:00, 45.868)
	machine temperature system failure	2268	22,695	(2013/12/2 21:15:00, 73.96732207)
	nyc taxi	1035	10,320	(2014/7/1 0:00, 10844)
	rogue agent key hold	190	1882	(2014/7/6 20:10:00, 0.064534524)
	rogue agent key updown	530	5315	(2014/7/6 20:10:00, 1.04725631)
realTraffic	occupancy 6005	239	2380	(2015/9/1 13:45:00, 3.06)
	occupancy t4013	250	2500	(2015/9/1 11:30:00, 13.56)
	speed 6005	239	2500	(2015/8/31 18:22:00, 90)
	speed 7578	116	1127	(2015/9/8 11:39:00, 73)
	speed t4013	250	2495	(2015/9/1 11:25:00, 58)
	TravelTime 387	249	2500	(2015/7/10 14:24:00, 564)
	TravelTime 451	217	2162	(2015/7/28 11:56:00, 248)
realTweets	Twitter volume AAPL	1588	15,902	(2015/2/26 21:42:53, 104)
	Twitter volume AMZN	1580	15,831	(2015/2/26 21:42:00, 57)
	Twitter volume CRM	1593	15,902	(2015/2/26 21:42:53, 11)
	Twitter volume CVS	1526	15,853	(2015/2/26 21:42:53, 0)
	Twitter volume FB	1582	15,833	(2015/2/26 21:42:53, 53)
	Twitter volume GOOG	1432	15,842	(2015/2/26 21:42:53, 35)
	Twitter volume IBM	1590	15,893	(2015/2/26 21:42:53, 7)
	Twitter volume KO	1587	15,851	(2015/2/26 21:42:53, 8)
	Twitter volume PFE	1588	15,858	(2015/2/26 21:42:53, 3)
	Twitter volume UPS	1585	15,866	(2015/2/26 21:42:53, 2)

4.1.2. Hyperparameters

Three hyperparameters need to be considered in the proposed method. The first hyperparameter is the sliding window size (ρ) used in Equation (2), which is proposed for feature smoothing. With the sliding window, features from the latest ρ timestamps are fused into one, enhancing the semantic expressiveness of the current data. The second one is the size of the convolution core (λ) used in Equation (3), which is a hyperparameter for binary feature reduction. Convolution makes semantic information in features more concentrated. The last one is the size of the pooling core (μ) also used in Equation (3), which is another hyperparameter for binary feature reduction. Max pooling removes redundant 1s and 0s in SDR to achieve data reduction. In the experiments, all the settings of these hyperparameters are summarized in Table 2.

Table 2. The settings of hyperparameters.

Symbol	Description	Settings
ρ	size of sliding window	1/2/3/4/5/6/7/8/9/10
λ	size of convolution core	2/3/4/5/6/7/8/9/10
μ	size and stride of pooling	2/3/4/5/6/7/8/9/10

4.1.3. Evaluation Metrics

In the experiment, the NAB scoring mechanism [35] is applied for evaluating the performance of anomaly detection. It is a novel scoring mechanism that has been widely used for performance evaluation in industrial anomaly detection. It contains three metrics that are standard profile (SP), reward low FP (RLFP) and reward low FN (RLFN), whose definitions are shown in Equation (10):

$$Score = Norm(\alpha \cdot \sum \mathcal{D}(TP) + \beta \cdot \sum \mathcal{D}(FP) - \gamma \cdot \sum FN), \quad (10)$$

where TP is the true positive anomaly detection. FP and FN are false positive and false negative detections, respectively. $\mathcal{D}(TP/FP)$ is a sigmoid function that lets TP/FP be distributed from -1.0 to 1.0. $Norm(\cdot)$ is a normalization function that lets the result score be distributed from $-\infty$ to 100.

Then, to set different values for α, β and γ, we can obtain SP, RLFP and RLFN metrics as presented in Table 3.

Table 3. Scoring weights for three metrics.

Metrics	α	β	γ
SP	1.0	1.0	0.11
RLFP	1.0	1.0	0.22
RLFN	1.0	2.0	0.11

In addition, to evaluate the performance of data reduction during the data transmission from the edge side to the cloud side, a metric called the data-reduction rate ($\mathcal{DRR}$) is provided:

$$\mathcal{DRR} = 1 - \frac{(l_t + \frac{l_v+\mu-1}{\mu}) \cdot \theta}{l_t + l_v}, \tag{11}$$

where l_t and l_v are the lengths of the time part and value part of the raw data feature, respectively. μ is the size of the max pooling core. θ is the sparsity of one bits in the raw data feature.

4.2. State-of-the-Art Comparisons

In the experiment, twelve well-known anomaly-detection methods are selected as the competitors. They are EORELM-AD (EA) [26], ARTime (AR) https://github.com/markNZed/ARTimeNAB.jl (accessed on 20 June 2023), Numenta HTM (HTM) [23], CAD OSE (CO) https://github.com/smirmik/CAD (accessed on 20 June 2023), earthgecko Skyline (EGS) https://github.com/earthgecko/skyline (accessed on 20 June 2023), KNN CAD (KC) [21], relative entropy (RE) [20], random cut forest (RCF) [19], Twitter ADVec (TA) https://github.com/twitter/AnomalyDetection (accessed on 20 June 2023), Windowed Gaussian (WG) https://github.com/numenta/NAB/blob/master/nab/detectors/gaussian/windowedGaussian_detector.py (accessed on 20 June 2023), Etsy Skyline (ES) https://github.com/etsy/skyline (accessed on 20 June 2023), Bayesian changepoint (BC) [18], and EXPoSE (EX) [17]. In these methods, the latest online time-series anomaly-detection method is EORELM-AD (EA), the state-of-the-art methods are ARTime (AR) and Numenta HTM (HTM), and the baseline method is EXPoSE (EX). Also, a random detection method (RD) https://github.com/numenta/NAB/blob/master/nab/detectors/random/random_detector.py (accessed on 20 June 2023) is referred to as the bottom bound.

In the comparisons, the hyperparameters are set as $\rho = 2$, $\lambda = 2$, and $\mu = 3$. As shown in Table 4, the proposed method surpasses almost other methods on evaluation metrics of SP, RLFP, and RLFN, and achieves a comparable performance with the HTM method. It has better performance results than HTM on realKnownCause, realTraffic, and realTweets datasets, and has similar performance results on artificialWithAnomaly and realAdExchange. When compared with the latest method EORELM-AD (EA), it still holds the full advantage. Moreover, the effect of the proposed model is quite close to that of SOTA models AR and HTM, and even better on some subdatasets, such as realAdExchange and realAWSCloudwatch. It is indicated that the data-reduction module is more suitable for data that change slowly and smoothly, because HTM-based methods have the ability to detect subtle temporal anomalies. The model smooths the current record with historical data to synthesize time-series information, and fuses numerical features to reduce the amount of data, which further accurately establishes the pattern of similar data. In addition, thanks to feature smoothing and binary feature reduction, the proposed method can

achieve data reduction and maintain competitive anomaly-detection performance. It means, compared with HTM, the proposed method only requires 3.81% of data, which can achieve comparable anomaly-detection performance. It is an interesting and feasible way to apply the method to the edge–cloud IIoT environment, especially when a large amount and distributed IIoT sensors are deployed.

Table 4. State-of-the-art comparisons.

Dataset		Ours	EA	AR	HTM	CO	EGS	KC	RE	RCF	TA	WG	ES	BC	EX	RD
Total	SP	70.0	46.4	**74.9**	70.1	69.9	58.2	58.0	54.6	51.7	47.1	39.6	35.7	17.7	16.4	11.0
	RLFP	64.6	32.1	65.2	63.1	**67.0**	46.2	43.4	47.6	38.4	33.6	20.9	27.1	3.2	3.2	1.2
	RLFN	74.0	52.5	**80.4**	74.3	73.2	63.9	64.8	58.8	59.7	53.5	47.4	44.5	32.2	26.9	19.5
	DRR(%)	**96.19**	-	-	-	-	-	-	-	-	-	-	-	-	-	-
artificialWithAnomaly	SP	68.9	0.2	70.3	70.2	73.0	40.2	45.3	56.6	15.5	**76.7**	2.6	30.9	-5.8	40.3	56.9
	RLFP	62.6	−32.2	60.6	**68.4**	67.1	33.2	26.9	50.2	15.5	57.4	−11.2	0.0	−1.7	0.0	51.4
	RLFN	73.7	11.2	74.7	74.6	74.7	43.5	52.4	60.0	31.1	**84.5**	7.3	31.7	25.9	33.1	71.2
realAdExchange	SP	72.4	58.8	75.1	**76.9**	72.0	56.8	71.0	36.8	62.8	49.6	56.9	18.8	42.9	13.0	-
	RLFP	**71.6**	50.9	70.8	67.7	70.4	53.7	62.6	29.9	45.9	48.8	53.8	0.0	−0.8	6.3	-
	RLFN	74.4	63.0	78.7	**79.8**	72.9	59.3	75.9	41.2	65.4	52.1	59.4	19.7	60.8	13.1	-
realAWSCloudwatch	SP	73.3	50.0	69.5	**73.4**	71.5	58.1	60.7	50.0	57.6	38.5	31.2	49.3	39.7	2.3	16.4
	RLFP	**69.7**	36.2	59.0	65.8	67.7	45.2	53.1	44.0	44.9	25.5	4.9	42.2	24.1	2.8	7.0
	RLFN	76.6	56.6	75.2	**76.7**	75.8	64.3	64.9	53.3	64.7	45.7	41.9	56.2	51.0	15.0	25.4
realKnownCause	SP	56.3	28.0	**63.9**	55.5	41.8	32.9	45.2	49.9	43.1	26.7	13.4	10.1	−1.2	13.5	44.4
	RLFP	49.4	15.8	**52.2**	49.3	38.0	27.7	27.9	39.3	16.0	21.5	9.9	6.8	−38.7	5.1	25.1
	RLFN	62.1	32.7	**70.7**	61.5	49.4	36.0	54.7	56.1	47.8	30.1	15.9	12.0	22.9	21.8	57.7
realTraffic	SP	84.1	51.0	86.9	82.5	**91.2**	76.5	49.9	78.6	63.8	57.7	64.3	74.9	44.1	36.1	-
	RLFP	81.7	41.6	81.5	75.7	**88.4**	73.5	40.0	71.3	36.8	55.7	61.5	49.6	25.1	0.0	-
	RLFN	87.0	55.4	91.2	86.0	**92.7**	79.6	54.7	83.4	66.1	59.9	66.7	78.5	51.8	55.9	-
realTweets	SP	68.4	54.8	**81.8**	68.0	74.5	68.9	64.3	59.8	48.1	59.6	51.4	29.4	2.7	19.8	-
	RLFP	59.6	37.4	70.1	61.1	**73.1**	45.3	42.2	55.0	46.9	35.8	16.3	31.8	1.7	3.0	-
	RLFN	72.9	61.8	**87.9**	72.6	76.5	78.3	73.2	63.1	62.2	69.0	64.6	50.9	9.4	33.5	-

The bold indicates best performance, and the underline indicates secondary performance. ArtificialNoAnomaly is missing because there are no anomalies in this type of dataset and cannot be calculated separately. EORELM-AD (EA) is reported using global optimal hyperparameter values for all datasets to ensure fairness.

4.3. Hyperparameters Learning

4.3.1. Smoothing Window Size Tuning

In the tuning process, the window size ρ is increased from 1 to 10. Three metrics, SP, RLFP, and RLFN, are used for the evaluation. It can be observed in Figure 7 that the proposed model achieves relatively good results when ρ equals 2 or 3. In more detail, the model achieves the best performance when ρ equals 2 since it has better SP and RLFP than ρ equals 3. Thus, in this work, $\rho = 2$ is selected.

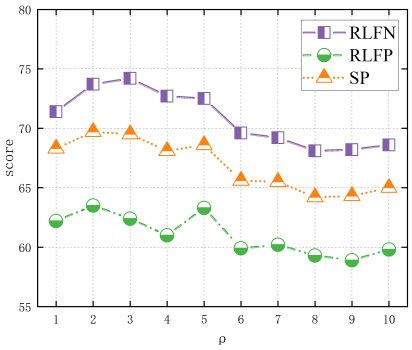


Figure 7. Smoothing window size tuning.

4.3.2. Binary-Convolution Hyperparameters Tuning

There are two hyperparameters, λ and μ, in the binary-convolution network. In the tuning process, λ increases from 2 to 10, combined with μ increased from 2 to 10. It can be observed from Figure 8 that the model achieves the best performance when $\lambda = 2$ and $\mu = 3$.

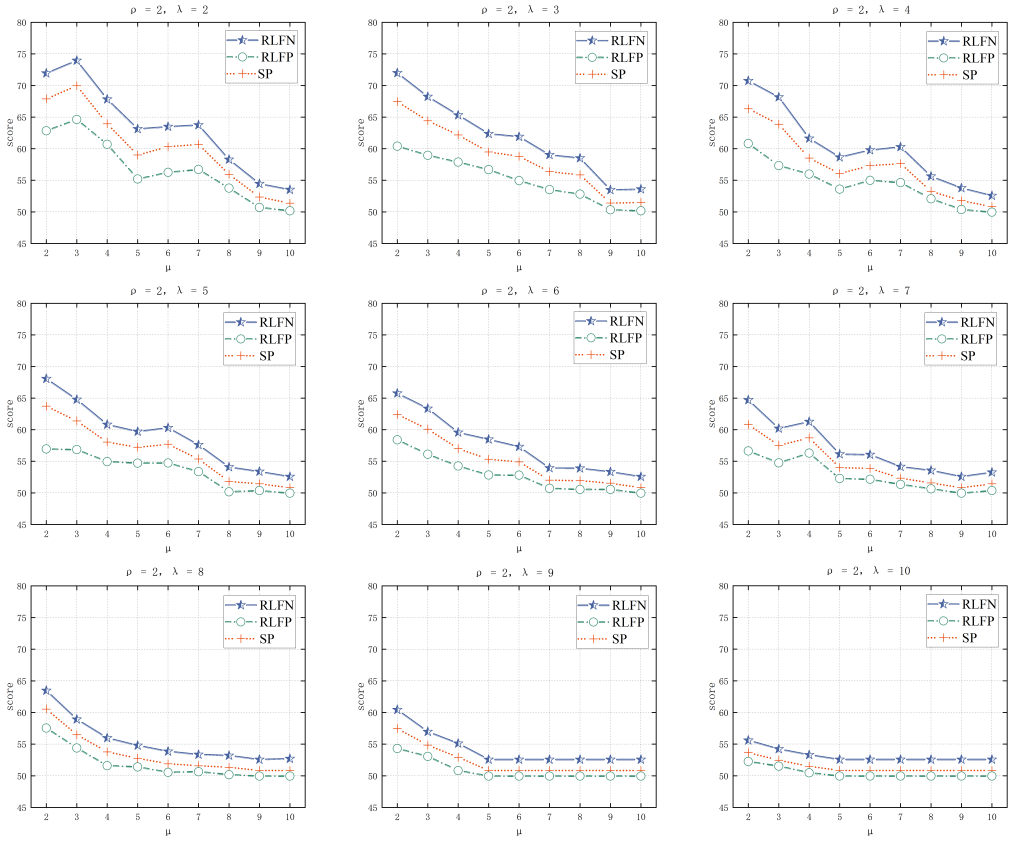


Figure 8. Binary-Convolution Hyperparameters Tuning.

4.4. Data-Reduction Analysis

From the definition of the data-reduction rate ($\mathcal{DRR}$), we can know that $\mathcal{DRR}$ depends on two parameters, μ and θ. Actually, θ is a fixed value decided by the sparsity of data features. In the practice, the sparsity is 0.092. Thus, we conduct a series of experiments to evaluate the data reduction by tuning μ. The result is shown in Table 5.

It can be observed from the result that the profits from data reduction become less than the loss of the performance with the increase in μ. However, when $\mu = 3$, the model achieves the best performance and a rather good data-reduction rate.

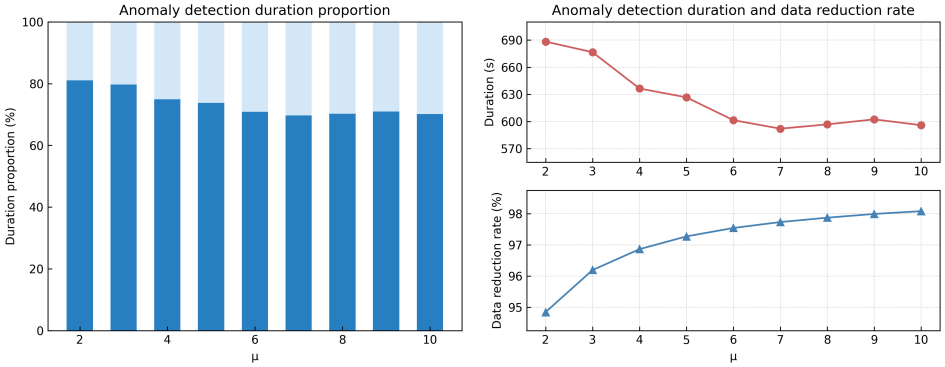
Table 5. Data-reduction analysis.

	SP	RLFP	RLFN	APD *	DRR (%)
HTM	70.1	62.2	74.7	baseline	none
ours, $\mu = 2$	67.9	62.8	71.9	−1.94%	94.84%
ours, $\mu = 3$	70.0	64.6	74.0	+0.91%	96.19%
ours, $\mu = 4$	64.0	60.7	67.8	−6.79%	96.86%
ours, $\mu = 5$	59.5	56.7	62.4	−13.51%	97.27%
ours, $\mu = 6$	60.3	56.3	63.5	−12.84%	97.54%
ours, $\mu = 7$	60.7	56.7	63.7	−12.30%	97.73%
ours, $\mu = 8$	55.9	53.8	58.3	−18.58%	97.87%
ours, $\mu = 9$	52.4	50.7	54.4	−23.63%	97.99%
ours, $\mu = 10$	51.5	50.2	53.6	−24.71%	98.08%

* The average performance difference (APD) of our method compared to Numenta HTM on SP, RLFP and RLFN.

4.5. Anomaly Detection-Duration Analysis

With the edge–cloud architecture, the duration of anomaly detection in the cloud is dropped. In Figure 9, we analyze the anomaly detection-duration reduction and the relationship between the anomaly-detection duration and the data-reduction rate. In Figure 9a, every bar shows the original anomaly-detection duration in the cloud. The dark blue bars represent the duration using the proposed reduction method with different pooling sizes (μ), while the light blue ones denote the reduction after using it. Apparently, less duration is needed when using a larger μ. In Figure 9b, with the size and stride of pooling (μ) increasing, the data-reduction rate is increased simultaneously. As fewer data features are analyzed in the cloud, the anomaly-detection duration is reduced. Thus, the anomaly-detection duration and the data-reduction rate are in reverse proportions.



(**a**) (**b**)

Figure 9. Anomaly detection-duration analysis. (**a**) Duration proportion. (**b**) Duration and data-reduction rate.

4.6. Ablation Studies

The core part of the proposed method is the binary-convolution data-reduction network, which consists of a data feature-smoothing module and a binary convolution module. To reveal the effects of each module, three sub-experiments, "binary-convolution only", "smoothing only" and "binary-convolution + smoothing", are conducted for ablation. In the ablation experiments, the hyperparameters are the same with the best performance settings of the method, which are $\rho = 2$, $\lambda = 2$, and $\mu = 3$.

It can be observed from Table 6 that the performance of the model increased with the modules applied. The model achieves the best performance when all modules are added.

Table 6. Ablation studies.

Method	SP	RLFP	RLFN
Smoothing + Binary Convolution	70.0	64.6	74.0
Smoothing	69.7	63.5	73.7
Binary Convolution	58.2	50.6	63.5

4.7. Discussion

The experimental results demonstrated that the proposed method is able to balance the performance of anomaly detection and the data-reduction rate in the edge–cloud IIoT environment. Moreover, we found that the method performs better, while the hyperparameters, ρ, λ, and μ, are relatively in a small value range (2 to 4). When ρ, λ, and μ are increasing, the data-reduction rate is increasing. But meanwhile, the performance of anomaly detection is significantly declining. We consider that this is because feature smoothing and binary convolution will lose lots of critical information when ρ, λ, and μ are increasing.

The work proposed a data-reduction module to significantly reduce the amount of data before sending them to the cloud-side for anomaly detection. But interestingly, we find that this reduction module has no negative impact on the performance of anomaly detection. In the contrary, the proposed method has even better performance results than the state-of-the-art method on some sub-datasets. We speculate that this is because the feature smoothing and binary convolution in the data-reduction model are useful for the detection model to gather more significant feature information. Since the data-reduction module fuses previous data into the current record, it smooths multiple records to extract information. It can more easily obtain slow-changing data patterns, but it is difficult to cope with data that are stable but change abruptly because feature smoothing tends to blur the trend of change. For example, when analyzing load data in the artificialWithAnomaly subdataset, the proposed model misjudges sudden changes in data that are always 0 as anomalies and often lags. For subdatasets like realAWSCloudwatch, whose data are always changing slightly, the model can have good anomaly-detection performance.

Since we only apply the binary convolution on the value part features, the time part features are not reduced by the binary convolution. We also try to apply binary convolution on the time part features, but the effect is not ideal. This is probably because the time-part feature is a cyclic-categories encoding of SDR, whose one bits are continuous. The more concentrated one-bits data features are not suitable for convolution.

5. Case Study

5.1. The Scenario

There is a copper alloy rod material production line in Chinalco Kunming Copper Co., Ltd. It is designed to produce high-strength and high-conductivity copper alloy rods with a capacity of 2000 tons per year. However, it does not have an integrated anomaly-detection system since almost all IIoT sensors are not smart and online. This significantly affects the stability and productivity of the production line. The company thus launched an updating project to reform the production line, and it aims to identify anomalies of all sensor data. In order to control the cost and stability, all the sensors and related devices are not allowed to be replaced. Since almost all the IIoT sensors are non-smart devices that cannot be connected to high-level networks, such as the internet, Wi-Fi, Ethernet, etc., they are unable to directly collect the IIoT sensor data in the anomaly-detection model. Thus, in this scenario, an edge–cloud architecture is required.

The production line consists of four processes—smelting, upward casting, rolling, and extruding—as shown in Figure 10a,b. The IIoT sensors are distributed among these processes to acquire real-time manufacturing data, like temperature, voltage, power, viscosity, etc. As an example, Figure 10c shows a series of crystallizer water temperature sensors for the melting process. However, anomalies that happen during the melting process cannot

be obviously detected because workers cannot always keep their eyes on these sensors to observe anomalies.

In this work, as a case study, we apply the proposed method in the melting process to collect crystallizer water temperature sensor data on the edge side, and predict anomalies on the cloud side. Then, the detected anomalies are reported in the manufacturing digital system of the production line.

(**a**) (**b**) (**c**)

Figure 10. Production line processes and sensors: (**a**) smelting, upward casting and rolling, (**b**) extruding and (**c**) crystallizers.

5.2. The Deployment

The configuration of the deployment is shown in Figure 11, which has three layers, a sensor layer, an edge layer, and a cloud layer. In the sensor layer, we deploy PLCs (Siemens S1200) and use the Modbus protocol to connect all IIoT sensors (crystallizer water temperature sensors). In the edge layer, a gateway (Flex Fbox-4G) is deployed. It connects to PLCs through Ethernet and connects to the cloud layer by using the MQTT protocol. The gateway also supports self-defined edge computing. The proposed time-scalar binary encoder and binary convolution reduction network are deployed in this gateway. In the cloud layer, the detection model is deployed in a flask service framework. In the service, a series of MQTT controllers are used to receive the reduced data features sent from the edge layer. In detail, a Huawei-cloud computation-aware server with 64 cores and 256 GB of main memory holds the anomaly-detection model in the cloud. We also developed a manufacturing digital system for this production line to receive anomaly information and visualize them. The digital system was deployed in another Huawei-cloud ECS server with 12 cores and 64 GB of main memory. The digital system provides a series of HTTP RESTful services for querying, visualizing, and reporting the detected anomalies as shown in Figure 12.

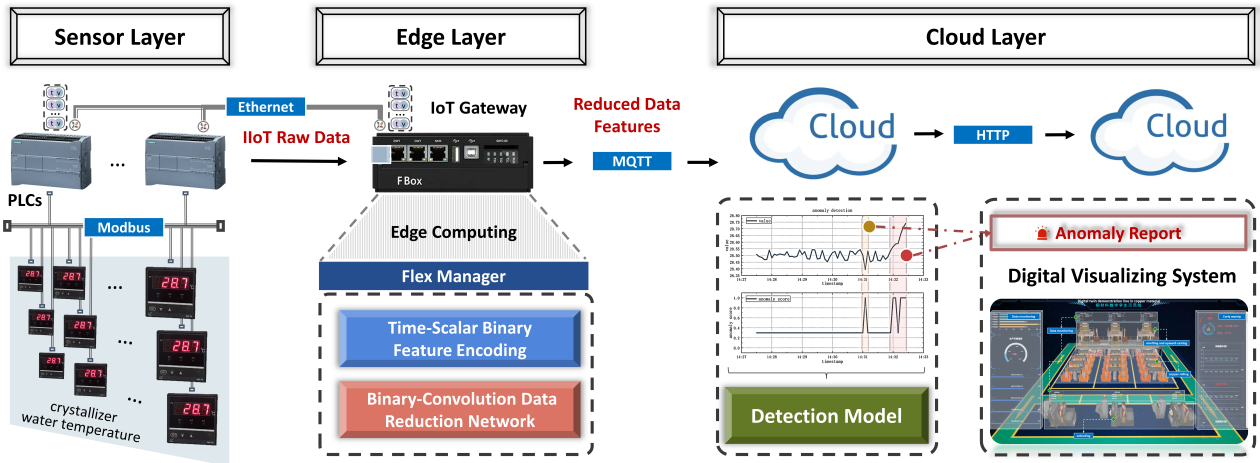


Figure 11. The deployment of the edge–cloud IIoT anomaly detection.

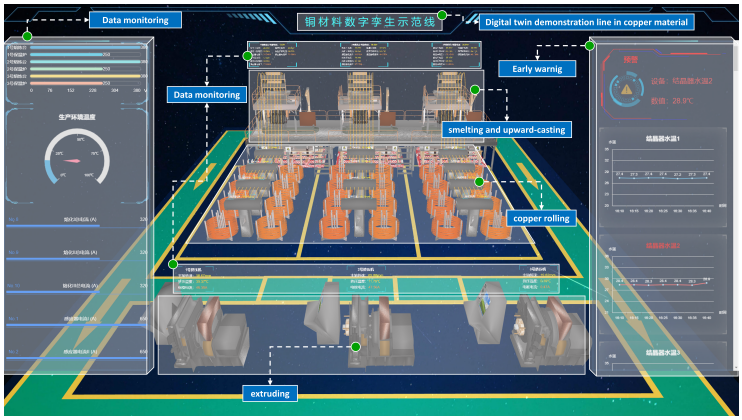


Figure 12. Digital visualizing system for anomaly detection.

5.3. The Evaluations

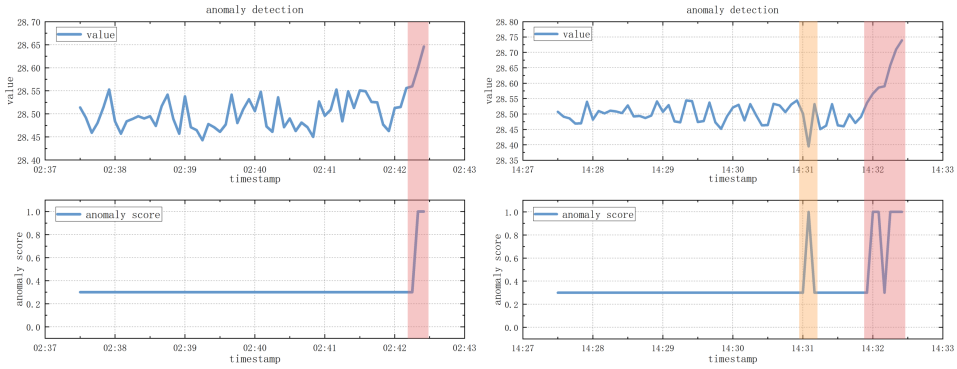
We monitored the deployed anomaly reporting system for 168 h during the smelting process. There were 231,840 records (t,v) of crystallizer water temperature collected and encoded in the edge. After reduction, these encoded data were sent to the cloud side for anomaly detection. For the evaluation, the records were further divided into five groups, each group having about 46,000 records. For all these records, there were 54 anomalies reported by the system, of which 46 reported anomalies are true positive, while 8 are false positive. According to the site workers' reports, 19 anomalies were not found by the system during this time. In summary, the evaluation result is shown in Table 7. Generally, the proposed model can reach about 70 scores on the three evaluation metrics of SP, RLFP and RLFN, and group 2 is even close to 80. With the data-reduction module, the amount of data transferred from the edge to the cloud can be reduced by 96.19%. Since each data bit occupies 4 B, each data feature requires 1816 B. As a result, a total of 401.5 MB of data in 168 hours can be reduced by about 386.2 MB.

Table 7. The statistical performance of the model in the case.

	SP	RLFP	RLFN	DRR
group1	65.3	63.9	65.8	
group2	79.5	78.9	79.6	
group3	66.3	65.9	66.4	96.19%
group4	70.6	69.9	70.9	
group5	71.0	70.6	71.2	

Moreover, a representative anomaly detection is presented in Figure 13. In Figure 13a, the temperature values fluctuate reasonably in a range with no anomalies. Correspondingly, the system continuously reports a stable and relatively low anomaly probability (close to 0.2), which means there are no anomalies that can be reported. However, at 02:42:15, a series of abnormal values is increasing, which causes the system to report high-confidence anomalies. In Figure 13b, at 14:31:05, there are abnormal values that are decreasing, which also lead to reporting an anomaly. But, according to the site working report, this is not a true positive anomaly, which means the system has reported a wrong anomaly.

It can be seen that the proposed model can realize "anomaly detection with data reduction", which solves the problem that "the model should achieve a higher accuracy rate on anomaly detection but requires less amount of data transmission" in the edge–cloud environment.



(**a**) (**b**)

Figure 13. The representative detected anomalies. (**a**) True positive in red. (**b**) False positive in orange and true positive in red.

In practice, all the detected anomalies are pushed to the "digital visualizing system for anomaly detection" (as shown in Figure 12). The system will raise warnings for these anomalies with customized ranking strategies. Managers and workers can easily and quickly receive, analyze, and respond to anomalies.

6. Conclusions

The trade-off between performance and data transmission in IIoT anomaly detection is a challenge in an industrial production environment. In this work, we proposed a binary-convolution data-reduction network for edge–cloud IIoT anomaly detection. The proposed method collects raw sensor data and extracts their features at the edge side, while the cloud side only receives data features to identify anomalies. It is thus able to balance the performance of anomaly detection and the data-reduction rate in the edge–cloud IIoT environment. To implement this, a time-scalar binary feature encoder is proposed and deployed on the edge side that encodes raw data into time-series binary vectors. Then, a binary-convolution data-reduction network is presented at the edge side to extract data features that significantly reduce the data size without losing critical information. At last, a hierarchical temporal memory (HTM)-based detection model is established on the cloud-side to conduct anomaly detection. Extensive experimental results demonstrate that the proposed method achieves a new state-of-the-art anomaly detection with data reduction. We also deployed the method in a real-world industrial project as a case study that proves the feasibility and effectiveness of the proposed model.

The current research is limited to the numerical time-series data collected by IIoT sensors, such as crystallizer water temperature. In the near future, multimedia time-series data, such as audio and video, should be supported to realize multi-modal industrial edge data offloading. Meanwhile, multi-device and multi-modal streaming data in IIoT can be deeply fused to more precisely detect anomalies and further reduce the amount of data transmitted throughout the production line.

Author Contributions: Conceptualization, C.X. and W.T.; Funding acquisition, C.X.; Investigation, C.X., W.T. and Y.D.; Methodology, C.X. and W.T.; Project administration, C.X.; Resources, C.X.; Software, W.T. and Z.Z.; Validation, W.T.; Visualization, C.X. and W.T.; Writing—original draft, W.T.; Writing—review and editing, C.X. and W.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Yunnan Provincial Science and Technology Department, grant number 202102AB080019-2 and 202002AB080001-5.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wu, Y.; Dai, H.N.; Tang, H. Graph neural networks for anomaly detection in industrial internet of things. *IEEE Internet Things J.* **2021**, *9*, 9214–9231. [CrossRef]
2. Siegel, B. Industrial anomaly detection: A comparison of unsupervised neural network architectures. *IEEE Sensors Lett.* **2020**, *4*, 1–4. [CrossRef]
3. Jalali, A.; Heistracher, C.; Schindler, A.; Haslhofer, B.; Nemeth, T.; Glawar, R.; Sihn, W.; De Boer, P. Predicting time-to-failure of plasma etching equipment using machine learning. In Proceedings of the 2019 IEEE International Conference on Prognostics and Health Management (ICPHM), San Francisco, CA, USA, 17–20 June 2019; pp. 1–8.
4. Li, B.; Yu, D.; Wu, J.; Ju, P.; Li, Z. Coordinated Cloud-Edge Anomaly Identification for Active Distribution Networks. *IEEE Trans. Cloud Comput.* **2022**, *11*, 1204–1216. [CrossRef]
5. Bowden, D.; Marguglio, A.; Morabito, L.; Napione, C.; Panicucci, S.; Nikolakis, N.; Makris, S.; Coppo, G.; Andolina, S.; Macii, A. A Cloud-to-edge Architecture for Predictive Analytics. In Proceedings of the EDBT/ICDT Workshops, Lisbon, Portugal, 26 March 2019.
6. Chen, A.; Liu, F.H.; Wang, S.D. Data reduction for real-time bridge vibration data on edge. In Proceedings of the 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Washington, DC, USA, 5–8 October 2019; pp. 602–603.
7. Chen, B.; Wan, J.; Lan, Y.; Imran, M.; Li, D.; Guizani, N. Improving cognitive ability of edge intelligent IIoT through machine learning. *IEEE Netw.* **2019**, *33*, 61–67. [CrossRef]
8. Tang, B.; Chen, Z.; Hefferman, G.; Pei, S.; Wei, T.; He, H.; Yang, Q. Incorporating intelligence in fog computing for big data analysis in smart cities. *IEEE Trans. Ind. Inform.* **2017**, *13*, 2140–2150. [CrossRef]
9. Ghosh, A.M.; Grolinger, K. Edge-Cloud Computing for Internet of Things Data Analytics: Embedding Intelligence in the Edge With Deep Learning. *IEEE Trans. Ind. Inform.* **2021**, *17*, 2191–2200.
10. Pang, G.; Shen, C.; Cao, L.; Hengel, A.V.D. Deep learning for anomaly detection: A review. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–38. [CrossRef]
11. Sadr, A.V.; Bassett, B.A.; Kunz, M. A flexible framework for anomaly Detection via dimensionality reduction. In Proceedings of the 2019 6th International Conference on Soft Computing & Machine Intelligence (ISCMI), Johannesburg, South Africa, 19–20 November 2019; pp. 106–110.
12. Hafeez, T.; Xu, L.; Mcardle, G. Edge intelligence for data handling and predictive maintenance in IIOT. *IEEE Access* **2021**, *9*, 49355–49371. [CrossRef]
13. Shi, W.; Dustdar, S. The promise of edge computing. *Computer* **2016**, *49*, 78–81. [CrossRef]
14. Wen, Z.; Bhatotia, P.; Chen, R.; Lee, M. Approxiot: Approximate analytics for edge computing. In Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2–5 July 2018; pp. 411–421.
15. Hafeez, T.; McArdle, G.; Xu, L. Adaptive window based sampling on the edge for Internet of Things data streams. In Proceedings of the 2020 11th International Conference on Network of the Future (NoF), Bordeaux, France, 12–14 October 2020; pp. 105–109.
16. Ndubuaku, M.U.; Ali, M.K.; Anjum, A.; Liotta, A.; Reiff-Marganiec, S. Edge-enhanced analytics via latent space dimensionality reduction. In Proceedings of the 2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT), Leicester, UK, 7–10 December 2020; pp. 87–95.
17. Schneider, M.; Ertel, W.; Ramos, F. Expected similarity estimation for large-scale batch and streaming anomaly detection. *Mach. Learn.* **2016**, *105*, 305–333. [CrossRef]
18. Adams, R.P.; MacKay, D.J.C. Bayesian Online Changepoint Detection. *arXiv* **2007**, arXiv:0710.3742.
19. Guha, S.; Mishra, N.; Roy, G.; Schrijvers, O. Robust random cut forest based anomaly detection on streams. In Proceedings of the International Conference on Machine Learning (PMLR), New York, NY, USA, 20–22 June 2016; pp. 2712–2721.
20. Wang, C.; Viswanathan, K.; Choudur, L.; Talwar, V.; Satterfield, W.; Schwan, K. Statistical techniques for online anomaly detection in data centers. In Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, Dublin, Ireland, 23–27 May 2011; pp. 385–392.
21. Burnaev, E.; Ishimtsev, V. Conformalized density- and distance-based anomaly detection in time-series data. *arXiv* **2016**, arXiv:1608.04585.
22. Blázquez-García, A.; Conde, A.; Mori, U.; Lozano, J.A. A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Comput. Surv.* **2021**, *54*, 56:1–56:33. [CrossRef]
23. Ahmad, S.; Lavin, A.; Purdy, S.; Agha, Z. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* **2017**, *262*, 134–147. [CrossRef]
24. Xu, Z.; Kersting, K.; von Ritter, L. Stochastic Online Anomaly Analysis for Streaming Time Series. In Proceedings of the Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia, Melbourne, Australia, 19–25 August 2017; pp. 3189–3195.
25. Maciąg, P.S.; Kryszkiewicz, M.; Bembenik, R.; L. Lobo, J.; Del Ser, J. Unsupervised Anomaly Detection in Stream Data with Online Evolving Spiking Neural Networks. *Neural Netw.* **2021**, *139*, 118–139. [CrossRef]
26. Iturria, A.; Labaien, J.; Charramendieta, S.; Lojo, A.; Del Ser, J.; Herrera, F. A framework for adapting online prediction algorithms to outlier detection over time series. *Knowl.-Based Syst.* **2022**, *256*, 109823. [CrossRef]
27. Hole, K.J.; Ahmad, S. A thousand brains: toward biologically constrained AI. *SN Appl. Sci.* **2021**, *3*, 743. [CrossRef]

28. Cui, Y.; Ahmad, S.; Hawkins, J. Continuous online sequence learning with an unsupervised neural network model. *Neural Comput.* **2016**, *28*, 2474–2504. [CrossRef]
29. Alieksieiev, V. One approach of approximation for incoming data stream in iot based monitoring system. In Proceedings of the 2018 IEEE second international conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2018; pp. 94–97.
30. Chang, K.C.; Chiang, M.H. Design of data reduction approach for aiot on embedded edge node. In Proceedings of the 2019 IEEE 8th Global Conference on Consumer Electronics (GCCE), Osaka, Japan, 15–18 October 2019; pp. 899–900.
31. Ahmad, S.; Hawkins, J. How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites. *arXiv* **2016**, arXiv:1601.00720.
32. Purdy, S. Encoding data for HTM systems. *arXiv* **2016**, arXiv:1602.05925.
33. Cui, Y.; Ahmad, S.; Hawkins, J. The HTM Spatial Pooler—A Neocortical Algorithm for Online Sparse Distributed Coding. *Front. Comput. Neurosci.* **2017**, *11*, 111. [CrossRef] [PubMed]
34. Karagiannidis, G.K.; Lioumpas, A.S. An improved approximation for the Gaussian Q-function. *IEEE Commun. Lett.* **2007**, *11*, 644–646. [CrossRef]
35. Lavin, A.; Ahmad, S. Evaluating real-time anomaly detection algorithms—The Numenta anomaly benchmark. In Proceedings of the 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 9–11 December 2015; pp. 38–44.