Multivariate Time Series Anomaly Detection Based on Reconstructed Differences Using Graph Attention Networks

Jung Mo Kang and Myoung Ho Kim(✉)

Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea
{qazec19,mhkim}@kaist.ac.kr

Abstract. Today, many real-world applications generate the amount of multivariate time series data. Monitoring those data and detecting some meaningful events early is important. As one of those tasks, interest in anomaly detection has grown. In recent research, some authors conducted anomaly detection in multivariate time series data by using graph attention networks to capture relationships among series and timestamps respectively. And another author suggested some connections between timestamps called Spatio-temporal connections. In this paper, we combine two ideas jointly and propose another multivariate time series anomaly detection method using series differences between adjacent timestamps. By using the proposed method, we conduct anomaly detection on two public datasets and compare the performance with other models. Also, to check for the possibility of operation on the edge environment, we measure the throughput of our proposed method in the IoT edge gateway that has restricted resources.

Keywords: Multivariate time series data · Anomaly detection · Graph attention networks · IoT edge gateway

1 Introduction

Today, several real-world applications or systems, e.g., sensors installed in the smart factory, generate the amount of data. As the size of generated data has grown more and more, it is one of the important interests how we utilize and analyze this huge data. Although there are several ways to utilize collected data, one of the important tasks is event detection. Event detection is a process of finding the particular event or pattern of event that we have been interested in among the numerous flow of the data. Meanwhile, the type of generated data usually becomes time series data, i.e., multivariate time series data, because the data is generated, collected, or observed periodically. Consequently, event detection is essentially relevant to the time series data domain. Although there exist many types of valuable events in the time series data domain, one of the important events is an anomaly. An anomaly is an event shown differently from the general or expected event pattern. If we can detect anomalies early, we prevent some disasters that will occur in the future and get some benefits. For example, in terms of industrial, detecting anomalies in systems early can save money and time. Moreover, these monitoring and detection can have more benefits in the edge environment because of lower latency. On the other hand, apart from the edge environment, research about anomaly detection in multivariate time series data has been conducted because of the importance of anomaly detection tasks. Especially, after the graph attention mechanism [10] was proposed, some researchers used the graph attention mechanism to detect anomalies in the multivariate time series data. As one of those research, the authors of MTAD-GAT [13] generated graph structure data in terms of series and timestamps respectively, adapted the graph attention mechanism to each graph structure data, and utilized the reconstruction-based method and forecasting-based method together. Consequently, MTAD-GAT showed good performance on two public datasets. In the forecasting field, the author of [14] suggested the Spatio-temporal connection between adjacent timestamps. The author showed the connection can improve the forecasting performance. So, in this paper, we combine these two ideas jointly and suggest another multivariate time series anomaly detection method using series differences between adjacent timestamps. Also, as we mentioned, because monitoring anomaly is one of the valuable tasks in the edge environment, we measured the possibility of the proposed method being executed in restricted hardware resources of the edge environment.

2 Related Works

In this section, we describe some related works about anomaly detection in the multivariate time series domain. At first, the outlier detection algorithm like [1] had been used because the anomaly detection was started from the outlier detection problem. But those outlier detection algorithms couldn't consider both the temporal character of the time series and the complex inner relationship between series. As the result, the forecasting-based method had been used based on the statistical model such as ARIMA to consider the temporal dependency. In recent, however, the deep learning-based method has been widely proposed because of the good computation and inference performance as the universal function approximator. One of the first deep learning-based methods was [5]. This method suggested the LSTM autoencoder architecture and compared the forecasting-based method and reconstruction-based method for the multivariate time series anomaly detection. Next, LSTM-NDT [3] used the forecasting-based method with the LSTM autoencoder and suggested a non-parametric dynamic threshold technique. OmniAnomaly [9] used the GRU autoencoder and variational inference and exploited a planar normalization flow to build a more complex latent distribution from the simple normal distribution. Also, the author adapted the extreme value theory introduced in [8] to set the threshold automatically as possible. MSCRED [12] used the correlation to capture the relationship between series. The author also used the convolution LSTM to reflect temporal property. However, the relationship between series couldn't be perfectly

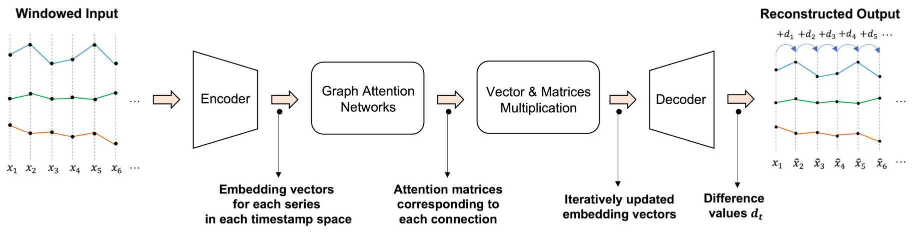

Windowed Input
x_1 x_2 x_3 x_4 x_5 x_6 ⋯
Encoder
Embedding vectors for each series in each timestamp space
Graph Attention Networks
Attention matrices corresponding to each connection
Vector & Matrices Multiplication
Iteratively updated embedding vectors
Decoder
Difference values d_t
Reconstructed Output
$+d_1$ $+d_2$ $+d_3$ $+d_4$ $+d_5$ ⋯
x_1 $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_4$ $\hat{x}_5$ $\hat{x}_6$ ⋯

Fig. 1. An overview of the detection process conducted in our neural net model

represented by the linear correlation. So, to represent the relationship between series more correctly, the graph structure data has been used. GDN [2] and MTAD-GAT [13] used the graph attention network (GAT) [10] to consider the relationship between series. Moreover, [13] showed good performance by using the reconstruction-based method and the forecasting-based method together. Inspired by [2,13], our proposed method also uses graph attention networks. But differing from them, we combine the Spatio-temporal connection concept introduced in STJGCN [14] as the temporal connection for improved performance. We propose another reconstruction-based anomaly detection method using series differences between adjacent timestamps. We also measure the execution possibility in the restricted hardware resource assuming the edge environment, unlike previous studies (Fig. 1).

3 Methods

3.1 Problem Definition

In this paper, we address the anomaly detection problem in multivariate time series data. We assume the overlapping sliding windowed input $X_{t+1:t+w} = \{x_{t+1}, \dots, x_{t+w}\}$ with a fixed length of w. Each $x_{t+n,n=1,\dots,w}$ is a k dimension vector and we denote the *i-th* scalar element of the vector x_{t+n} as x^i_{t+n}. Our purpose is to determine whether the given input $X_{t+1:t+w}$ contains some anomalous parts or not based on the anomaly score $S(X_{t+1:t+w})$. We will describe how we compute the anomaly score in Sect. 3.6.

3.2 Encoder Architecture

Some research [6,13] used the entire series to generate graph structure data, but we build the series relationship graph from the bottom, i.e., timestamp unit. To generate *i-th* series' embedding vector f^i_{t+n} from x^i_{t+n} scalar value, we use the transpose convolution 1D 4 layers with a kernel size of 7 as an encoder. Also, to capture the diverse range of series information, we use three additional sub-encoders. Specifically, these three sub-encoders consist of convolution 1D [6, 3, 2] layers with the same kernel size of 7, and dilation size of [2, 2, 1] respectively. The first sub-encoder can capture the long-range of series information because 6 layers have large reception fields. The second and third sub-encoders capture the

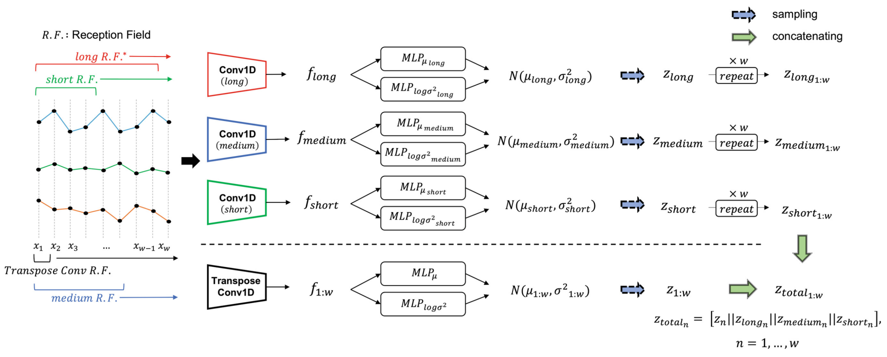


Fig. 2. The process of encoding embedding vectors (The case of $t = 0$)

medium and short range of series information respectively in proportion to the number of layers and dilation sizes. After being generated, embedding vectors are given to MLP layers to compute mean and log variance because we adapt the VAE [4] architecture (Fig. 2).

$$\mathrm{TransConv1D}(x_{t+n}) = f_{t+n},\ \mathrm{Conv1D}_{type}(X_{t+1:t+w}) = f_{type} \tag{1}$$

$$\mu_{t+n} = \mathrm{MLP}_{\mu}(f_{t+n}),\ \log\sigma^2_{t+n} = \mathrm{MLP}_{\log\sigma^2}(f_{t+n}) \tag{2}$$

$$\mu_{type} = \mathrm{MLP}_{\mu_{type}}(f_{type}),\ \log\sigma^2_{type} = \mathrm{MLP}_{\log\sigma^2_{type}}(f_{type}) \tag{3}$$

$$z_{t+n} = sample(N(\mu_{t+n}, \sigma^2_{t+n})),\ z_{type} = sample(N(\mu_{type}, \sigma^2_{type})) \tag{4}$$

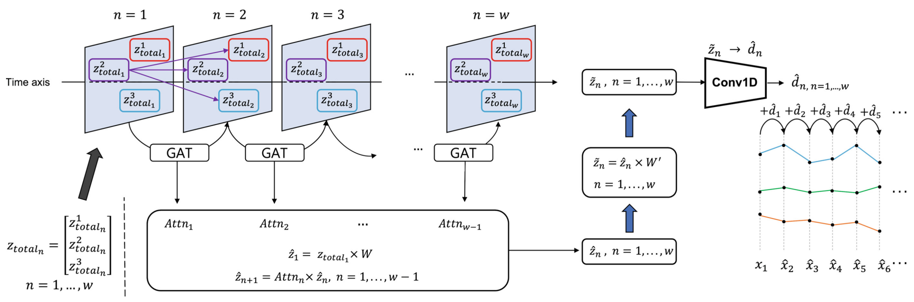
$$z_{type_{1:w}} = repeat(z_{type}, w) \tag{5}$$

$$z_{total_{t+n}} = [z_{t+n} \mid\mid z_{long_n} \mid\mid z_{medium_n} \mid\mid z_{short_n}],\ n = 1, ..., w \tag{6}$$

In (1), the $f_{t+n} \in \mathbb{R}^{k\times d_f}$ and the $f_{type} \in \mathbb{R}^{k\times d_{type}}$ are embedding vectors. The *type* means each element of $\{long,\ medium,\ short\}$. The d_f and d_{type} mean each series' embedding vector dimensions respectively. The *sample* means sampling and the *repeat* replicates z_{type} latent vector w times. The reason why we need *repeat* is to match the number of z_{type} to the number of z_{t+n}. Because $\mathrm{Conv1D}_{type}$ sub-encoders use the windowed input $X_{t+1:t+w}$, f_{type} and z_{type} vectors are generated only once for each *type*. In contrast, the TransConv1D uses each timestamp input and z_{t+n} feature vectors are generated as the number of timestamps, w. Consequently, we use *repeat* to resolve the inconsistency of the length. Next, we use a concatenate operate [...||...] to combine z_{t+n} and z_{type_n} along the series dimension axis. The $z_{total_{t+1:t+w}} \in \mathbb{R}^{w\times k\times d_{total}}$ is a concatenated feature and the value of $d_{total} = d_f + d_{long} + d_{medium} + d_{short}$.

3.3 Graph Attention Mechanism Between Timestamps

As the previous research [2,13] showed, the graph attention network (GAT) can capture the relation between time series. MTAD-GAT [13] generated two graph



$n=1$
$n=2$
$n=3$
$n=w$
Time axis
GAT
$Attn_1$
$Attn_2$
$Attn_{w-1}$
$\hat{z}_1 = z_{total_1} \times W$
$\hat{z}_{n+1} = Attn_n \times \hat{z}_n,\ n=1,\dots,w-1$
$z_{total_n} = [z^1_{total_n}; z^2_{total_n}; z^3_{total_n}]$, $n=1,\dots,w$
$\hat{z}_n,\ n=1,\dots,w$
$\tilde{z}_n = \hat{z}_n \times W'$, $n=1,\dots,w$
$\tilde{z}_n,\ n=1,\dots,w$
$\tilde{z}_n \to \hat{d}_n$
Conv1D
$\hat{d}_{n,\,n=1,\dots,w}$
$+\hat{d}_1\ +\hat{d}_2\ +\hat{d}_3\ +\hat{d}_4\ +\hat{d}_5$
$x_1\ \hat{x}_2\ \hat{x}_3\ \hat{x}_4\ \hat{x}_5\ \hat{x}_6$

Fig. 3. The process of GATs, vector & matrices multiplication, and differences decoding

structures to capture the series and timestamps relationship, but we combine them as a single graph attention mechanism by using the concept of Spatio-temporal connection [14]. If you see Fig. 3, you can see connections between $z^2_{total_1}$, i.e., the $2nd$ series' embedding vector of $n = 1$ timestamp space, and the entire embedding vectors of $n = 2$ timestamp space. They were introduced as Spatio-temporal connections in [14], but in our proposed method, we use those connections as temporal connections between adjacent timestamps. After assuming the temporal connection exists in every pair of vectors between adjacent timestamp space, we compute attention scores by (7) and (8).

$$e_{p,q,t+n} = \text{LeakyReLU}([z^p_{total_{t+n}} W \mid\mid z^q_{total_{t+n+1}} W]a) \tag{7}$$

$$\alpha_{p,q,t+n} = \frac{exp(e_{p,q,t+n})}{\sum_{r=1}^{k}(exp(e_{p,r,t+n}))} \tag{8}$$

In (7), $W \in \mathbb{R}^{d_{total} \times l}$ is a weight matrix for the linear transform and $a \in \mathbb{R}^{2l \times 1}$ is a shared attention mechanism. l is a linear transform dimension and exp means a natural exponential function. $\alpha_{p,q,t+n}$ is an attention coefficient corresponding to the edge from the latent vector $z^p_{total_{t+n}}$ to the latent vector $z^q_{total_{t+n+1}}$. Also, the $z^p_{total_{t+n}} \in \mathbb{R}^{1 \times d_{total}}$ means a p-th series' latent vector at $(t+n)$-th timestamp space and $z^q_{total_{t+n+1}} \in \mathbb{R}^{1 \times d_{total}}$ indicates a q-th series' latent vector at $(t+n+1)$-th timestamp space. In this process, the graph attention layer may learn the temporal property and series relationship simultaneously.

3.4 Vector and Matrices Multiplication

After getting attention score matrices for all connections between adjacent timestamps, we compute the linear transformed latent feature $\hat{z}_{t+1:t+w}$ by multiplying with attention matrices $Attn_{t+1:t+w-1}$ iteratively. This process can be understood as the consecutive implicit forecasting process because we start from the first latent vector and compute the remaining vectors by considering the relationship between each series, i.e., multiplication with attention matrices. Also, if anomalous data exist in some timestamps, the relationship between normal preceding timestamps and trailing abnormal timestamps may cause incorrect graph attention scores in the previous step. As proceeding with this step, the wrong attention scores may gradually amplify errors and the amplified error can make the model detect an anomaly easier.

$$\begin{aligned}\hat{z}_{t+n+1} = Attn_{t+n} \times \hat{z}_{t+n},\ \hat{z}_{t+1} = z_{total_{t+1}} \times W,\ n = 1, ..., w-1 \\ Attn_{t+n} = \{\alpha_{p,q,t+n} \mid p = 1, ..., k, q = 1, ..., k\},\ Attn_{t+n} \in \mathbb{R}^{k \times k}\end{aligned} \tag{9}$$

3.5 Decoder Architecture

Before we pass the computed latent features to the decoder layer, we first change the dimension of each latent feature $\hat{z}_{t+n}$ from $\mathbb{R}^{k \times l}$ to $\mathbb{R}^{k \times d_{total}}$. To do this, we use a weight matrix $W^{'} \in \mathbb{R}^{l \times d_{total}}$. After each latent feature's dimension comes back, we pass the latent feature to the convolution decoder. Because this process is a reverse version of Sect. 3.2, the convolution 1D 4 layers with kernel size 7 are used instead of transpose convolution. As one important thing in this phase, we don't reconstruct the original input time series data directly. Otherwise, the model may memorize the data rather than learn it. To avoid that, we make the model decode series differences between adjacent timestamps. After finishing the reconstruction of differences, the model finally reconstructs input time series values by adding differences to the first timestamp input value iteratively. And then, we compute the loss function value following (12) and train our model to reduce the loss value. The first term of (12) is a mean squared error between input data and reconstructed data. The second and third terms reduce a gap between the encoder's recognition latent distribution and normal distribution.

$$\tilde{z}_{t+n} = \hat{z}_{t+n} \times W',\ \hat{d}_{t+n} = \text{Conv1D}_{dec}(\tilde{z}_{t+n}) \tag{10}$$

$$\hat{x}_{t+n+1} = \hat{x}_{t+n} + \hat{d}_{t+n},\ \hat{x}_{t+1} = x_{t+1},\ n = 1, ..., w-1 \tag{11}$$

$$\begin{aligned}Loss = {} & MSE(X_{t+1:t+w}, \hat{X}_{t+1:t+w}) \\ & - \tfrac{1}{2} \textstyle\sum_{types}(1 + \log \sigma_{type}^2 - \mu_{type}^2 - \sigma_{type}^2) \\ & - \tfrac{1}{2} \sum_{n=1}^{w} (1 + \log \sigma_{t+n}^2 - \mu_{t+n}^2 - \sigma_{t+n}^2)\end{aligned} \tag{12}$$

3.6 Criterion for Detection

We check whether the given windowed input $X_{t+1:t+w}$ contains an anomaly or not based on the last time series value x_{t+w} because we reconstruct the input data by adding reconstructed differences consecutively to the first input value x_{t+1}. If only normal time series data exist in the given windowed input, our neural net model may reconstruct differences between adjacent timestamps with a small error. In contrast, if some anomalous data exist in the given window, our model cannot reconstruct differences well. So, the error will be accumulated and the last time series value may show a huge error compared with the ground truth value. To quantify how the windowed data $X_{t+1:t+w}$ is anomalous, we compute the anomaly score $S(X_{t+1:t+w})$ based on the reconstruction error of the last time series value x_{t+w} by using (13). If the computed anomaly score exceeds the fixed threshold value τ, we regard the windowed data $X_{t+1:t+w}$ to have some anomalies. Otherwise, the windowed data $X_{t+1:t+w}$ is regarded as the normal data.

$$S(X_{t+1:t+w}) = \sqrt{\tfrac{1}{k}\textstyle\sum_{i=1}^{k}(x_{t+w}^{i} \; - \; \hat{x}_{t+w}^{i})^2} \tag{13}$$

3.7 Strategy for Setting Threshold Automatically

In this section, we describe how to select the threshold automatically. In the time series domain, the threshold is a key criterion to decide the model's performance. However, the model can't know future inputs. To address this issue, [8] proposed a method to select threshold values automatically based on the Extreme Value Theory (EVT). A summary of the EVT is that the probability distribution of extreme values is similar to each other regardless of the original probability distribution where extreme values are extracted. It means the probability distribution of the extreme score value corresponding anomaly is similar to each other in the training dataset and testing dataset. Consequently, we can adapt the extreme score distribution computed by the training dataset to the test dataset. To use this method, we first set the criterion for peak values among known data, i.e., training data. The author suggested τ_{init} in (14) as a value of 0.98 quantiles [8]. Based on this initial threshold τ_{init}, we can get peak values from the known dataset. Next, we conduct the maximum likelihood estimation over the Generalized Pareto Distribution (GPD) by using observed peak values. The GPD is a generalized probability distribution of tail distributions and we can get a real threshold value z_q by setting the *risk* value q as desired value, e.g., 10^{-4}. The Eq. (14) is a result of maximum likelihood estimation. $\hat{\sigma}$ and $\hat{\gamma}$ are results from maximum likelihood estimation and N_t means the number of observed peak values. n is the number of total observed values, and q is a *risk* value. z_q means the quantile value satisfying $P(X > z_q) < q$. This z_q will be used as a fixed threshold τ in the test phase.

$$z_q \; \simeq \tau_{init} + \tfrac{\hat{\sigma}}{\hat{\gamma}}\left(\left(\tfrac{qn}{N_t}\right)^{-\hat{\gamma}} - 1\right) \tag{14}$$

4 Experiments

4.1 Datasets

In experiments, we used two public datasets in the multivariate time series anomaly detection domain. One is the MSL (Mars Science Laboratory) dataset and the other is the SMAP (Soil Moisture Active Passive) dataset. Both datasets were used in [3] and collected by NASA. Please refer to Table 1 if you want to know the details of these two datasets. Because original datasets don't have validation datasets, we used 30% of training datasets as validation datasets for

Table 1. The details of datasets.

Name	# of items	Dimensions	# of train timestamps	# of valid timestamps	# of test timestamps	Ratio of anomalies
MSL	27	55	40,822	17,495	73,728	10.72%
SMAP	55	25	94,629	40,554	427,617	13.13%

early stopping. We also used both training and validation datasets to select the threshold τ automatically based on the EVT [8].

4.2 Experiment Settings

We implemented the proposed model by using PyTorch and used Adam optimizer with a learning rate of 0.0003 for training. We trained our model during 20 epochs and adopted early stopping when the validation loss exceeds the average of the previous 5 validation losses. We also set the window size $w = 100$ following previous research [9,13] and batch size as 1 following [4]. All training processes were conducted under the Ubuntu 20.04 LTS OS with Intel(R) Core(TM) i7-10700 CPU 2.90 GHz and NVIDIA RTX 3090 with 24 GB VRAM. We also measured the possibility of whether our model can be executed on some devices having restricted hardware resources. We tested our model in the IoT gateway device based on the Odroid N2 model. The IoT gateway has the ARM Cortex-A73 1.8 GHz and Dual-core Cortex-A53 1.9 GHz with 4 GB RAM. Lastly, We empirically used the risk $q = 0.005$ and $\tau_{init} = 0.95$ quantiles for the SMAP dataset and the risk $q = 0.025$ and $\tau_{init} = 0.95$ quantiles for the MSL dataset.

5 Results

5.1 The Performance Comparison

In this section, we compare the proposed method's performance with other multivariate time series anomaly detection models. The metric is an f1-score used widely in the anomaly detection domain. We also quote the [13]'s model performance table because we used the same threshold selection method based on the EVT [8]. The performance comparison result is like Table 2. As you can see, our proposed method shows the best precision and the f1-score performance on the SMAP dataset. However, the proposed method shows the third performance on the MSL dataset. So, we should find the reason why the performance is lower in the MSL dataset in future work. Although we're not sure, we may infer two reasons. According to [14], the author said the Spatio-temporal connection grows up the forecasting performance. And [13] said the forecasting-based anomaly detection methods such as LSTM-NDT and DAGMM show better performance on the SMAP dataset than the MSL dataset. In Sect. 3.4, we may understand that the model implicitly forecasts differences between adjacent timestamps. Consequently, our model may show a better performance on the SMAP data compared

Table 2. The performance comparison for each dataset.

Model	SMAP			MSL		
	Precision	Recall	F1-score	Precision	Recall	F1-score
OmniAnomaly	0.7416	**0.9776**	0.8434	**0.8867**	0.9117	0.8989
KitNet	0.7725	0.8327	0.8014	0.6312	0.7936	0.7031
GAN-Li	0.6710	0.8706	0.7579	0.7102	0.8706	0.7823
MAD-GAN	0.8049	0.8214	0.8131	0.8517	0.8991	0.8747
LSTM-VAE	0.8551	0.6366	0.7298	0.5257	0.9546	0.6780
LSTM-NDT	0.8965	0.8846	0.8905	0.5934	0.5374	0.5640
DAGMM	0.5845	0.9058	0.7105	0.5412	**0.9934**	0.7007
MTAD-GAT	0.8906	0.9123	0.9013	0.8754	0.9440	**0.9084**
Proposed	**0.9379**	0.9257	**0.9318**	0.8316	0.9225	0.8747

with the MSL data because of the Spatio-temporal connection and the property of the SMAP dataset. The other reason may be a property of differences. In the time series data analysis, getting the difference between adjacent timestamps is one way of transforming non-stationary time series into stationary time series. In the stationary time series case, statistical indices such as mean and variance don't change over time. That makes sometimes the stationary time series be looked like a random signal such as white noise. It makes also the stationary time series hard to be forecasted in long term. So, we have a plan to conduct a test such as KPSS to check for the stationary property of reconstructed differences. Additionally, there may exist some other reasons that lower the model performance in the MSL data set, e.g., incorrect graph structure generation method, or absence of the neural net model to capture the temporal property.

5.2 The Throughput Measurement of the Proposed Model

As we mentioned in the introduction and abstract, in the monitoring system, utilizing the edge environment may be useful. However, one of some problems in the edge environment is the restricted device performance. So we measured the throughput to check for the execution possibility of the proposed method in the restricted hardware resources. Specifically, we checked the throughput of our proposed model by using the `tqdm` library displaying batch iterations per second or seconds per batch iteration. We used a single batch in both training and testing and the single batch corresponds to a single-window data. Therefore the iterations per second can be understood as the throughput of our proposed method. As a result, in the IoT edge gateway described in Sect. 4.2, our proposed method shows 2.24 iterations per second on the SMAP dataset. It means our proposed method can conduct detection for 2.24 windows with a window length w=100 per second. In the MSL dataset case, the `tqdm` library shows 1.07 iterations per second. Because the MSL dataset consists of 55 series compared with

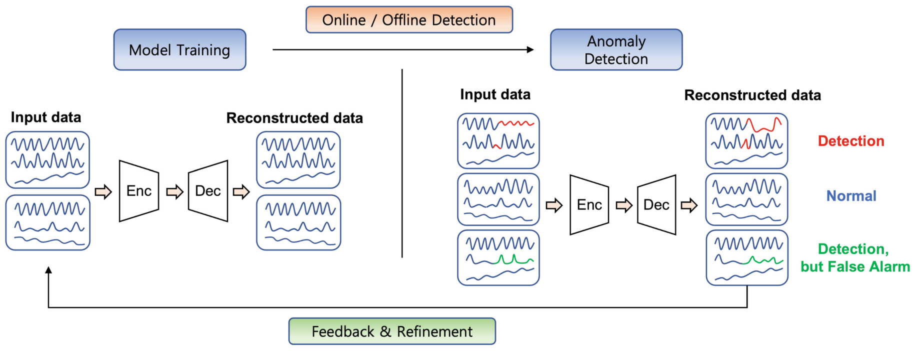


Fig. 4. The entire process of training, detection, feedback, and refinements.

25 series of the SMAP data, the MSL data requires more computation and this causes the lower throughput performance. So, if we deploy our anomaly detection model in the small IoT edge gateway, the proposed model can be operated normally when the gap of taking raw data input is larger than 1 s without considering other overhead. That condition seems hard to be satisfied, but one solution is to use a non-overlapping sliding window mechanism. When we suppose the gap between every new raw input data is 1 s, the overlapping sliding window mechanism has to process windowed input within every 1 s. However, in the non-overlapping sliding window mechanism, the proposed model doesn't need to produce the detection output within every 1 s because the model can have spare time about 100 s before getting the next windowed input. Fortunately, because some recent research [7,11] also shows good performance in the non-overlapping sliding window mechanism, we will consider the non-overlapping sliding window mechanism in future work.

5.3 Feedback and Refinement Process

Although we used the reconstruction-based method to detect anomalies in this paper, in fact, it is a drawback of the reconstruction-based method that the model cannot correctly distinguish the input data not seen in the training phase. In other words, in the test or detection phase, if the model encounters the data that doesn't appear during the training, the model may regard the data as an anomaly. But the data also can be normal. To avoid this situation, we should periodically collect those false alarm cases and retrain the model. This feedback and refinement process can be represented in Fig. 4. First, the model is trained on the collected multivariate time series data regarding as normal. Second, the model conducts detection in the online or offline environment. The online environment means real-time, and the offline means not real-time but collected data during some periods. Third, whenever the model detects anomaly candidates, the human expert may confirm whether candidates are real anomalies or not. In Fig. 4, the model correctly detects the anomaly in the first input data but makes a false alarm on the third input data. Lastly, we aggregate data that are turned out as false alarms and give them to the model to improve the model performance by retraining them. In this step, the model can learn the new normal data patterns and recognize them as normal. By repeating these feedback and refinement steps, we can improve the model's performance gradually.

6 Conclusion

In this paper, we proposed another multivariate time series anomaly detection method using the difference between adjacent timestamps. We jointly combined the graph attention concept and Spatio-temporal connection concept. We conducted experiments on the two public datasets in the multivariate time series anomaly detection domain and the proposed method showed better performance than the baseline model [13] in one dataset. We also measured the proposed method's possibility that can be executed in the small restricted hardware resources device such as the edge environment. And we suggested another direction to improve the throughput of the proposed method. Lastly, we suggested the feedback and refinement process to improve and maintain the model's performance. However, as you have seen, there are some remaining issues in this research. We will find the specific reason why the model's performance is lower in the MSL dataset, and we will measure the throughput of the proposed method in the restricted hardware environment again after retraining the model in a non-overlapping sliding window mechanism.

Acknowledgements. This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-0-01795) supervised by the IITP (Institute of Information & Communications Technology Planning & Evaluation). This work was supported by the Bio-Synergy Research Project (2013M3A9C4078137) of the MSIT (Ministry of Science and ICT), Korea through the NRF. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1A2C1004032).

References

1. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. SIGMOD Rec. **29**(2), 93–104 (2000). https://doi.org/10.1145/335191.335388
2. Deng, A., Hooi, B.: Graph neural network-based anomaly detection in multivariate time series. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 5, pp. 4027–4035 (2021). https://ojs.aaai.org/index.php/AAAI/article/view/16523
3. Hundman, K., Constantinou, V., Laporte, C., Colwell, I., Soderstrom, T.: Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining. ACM (2018). https://doi.org/10.1145/3219819.3219845

4. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2013). https://doi.org/10.48550/ARXIV.1312.6114
5. Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., Shroff, G.: LSTM-based encoder-decoder for multi-sensor anomaly detection (2016). https://doi.org/10.48550/ARXIV.1607.00148
6. Shang, C., Chen, J., Bi, J.: Discrete graph structure learning for forecasting multiple time series (2021). https://doi.org/10.48550/ARXIV.2101.06861
7. Shen, L., Li, Z., Kwok, J.: Timeseries anomaly detection using temporal hierarchical one-class network. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 13016–13026. Curran Associates, Inc. (2020). https://proceedings.neurips.cc/paper/2020/file/97e401a02082021fd24957f852e0e475-Paper.pdf
8. Siffer, A., Fouque, P.A., Termier, A., Largouet, C.: Anomaly detection in streams with extreme value theory. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2017, pp. 1067–1075. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3097983.3098144
9. Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., Pei, D.: Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, KDD 2019, p. 2828–2837. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3292500.3330672
10. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks (2017). https://doi.org/10.48550/ARXIV.1710.10903
11. Xu, J., Wu, H., Wang, J., Long, M.: Anomaly transformer: time series anomaly detection with association discrepancy (2021). https://doi.org/10.48550/ARXIV.2110.02642
12. Zhang, C., et al.: A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, pp. 1409–1416 (2019). https://doi.org/10.1609/aaai.v33i01.33011409. https://ojs.aaai.org/index.php/AAAI/article/view/3942
13. Zhao, H., et al.: Multivariate time-series anomaly detection via graph attention network (2020). https://doi.org/10.48550/ARXIV.2009.02040
14. Zheng, C., Fan, X., Pan, S., Wu, Z., Wang, C., Yu, P.S.: Spatio-temporal joint graph convolutional networks for traffic forecasting (2021). https://doi.org/10.48550/ARXIV.2111.13684