Patch-based Time-Series Anomaly Detection with Cross-Variable Attention

Jeena Son
Department of Industrial and Management Engineering
Korea University
Seoul, South Korea
a-jeen0722s@korea.ac.kr

Seunghwan Song
Department of Industrial and Management Engineering
Korea University
Seoul, South Korea
ss-hwan@korea.ac.kr

Jun-Geol Baek*
Department of Industrial and Management Engineering
Korea University
Seoul, South Korea
jungeol@korea.ac.kr

***Abstract*— Modern manufacturing processes are influenced by smart factories, which collect data from multiple sensors in real time. However, detecting anomalies is challenging due to their irregular and complex patterns, often resulting in unreliable labeling that depends on the engineer's expertise. Moreover, in these processes, where sensors are interconnected, fluctuations in one variable can potentially influence subsequent variables. The proposed method uses cross-variable attention to reflect the relationship between variables at different time points. It also utilizes contrastive learning to extract a representation of only normal data that can be distinguished from anomalies without labels. Experimental results on multivariate time series data demonstrate that the proposed method outperforms existing models in anomaly detection.**

Keywords— Anomaly Detection, Cross-Variable Attention, Multivariate Time Series, Contrastive Learning

I. INTRODUCTION

Anomaly is a deviation from the normal state or an unexpected change. To maintain the stability and reliability of a system, anomaly detection techniques are necessary. Anomaly detection is used in a variety of fields, including manufacturing, energy, finance, healthcare, and cloud computing. In particular, anomaly detection in manufacturing processes is closely related to product defects and the final product of the manufacturing process [1]. Therefore, accurate anomaly detection is necessary to improve quality and productivity.

Due to the influence of smart factories, data from multiple sensors are being collected in manufacturing processes. These are stored at regular time intervals in the form of multivariate time series. However, multivariate time series anomaly detection in manufacturing processes is challenging due to the following reasons: (1) labeling problems dependent on the engineer's expertise, (2) the absence of a representation for the relationship between variable at different time points.

First, the amount of sensors collected is vast, and the anomaly patterns are irregular and complex. Since data labeling is done manually by engineers, the reliability of labeling can vary depending on the experience level of the engineer. Therefore, an unsupervised learning model is necessary for detecting anomalies without labels.

Second, it is crucial to consider both temporal dependency and correlation between variables simultaneously. With the development of deep learning, various approaches have been explored to capture the temporal relationships, including Long Short Term Memory (LSTM), Transformer, and Temporal Convolution Network (TCN) [2], [3], [4]. These studies mainly consider only the dependencies between variables within the same time point. However, in the manufacturing process, multivariate sensors are organically connected, so small fluctuations in one variable can affect the later time points of other variables [5]. Hence, there is a need for research that reflects the relationship between variables at different time points.

In this study, we propose a patch-based time-series anomaly detection method that reflects the dependency between variables at different time points, which we define as cross-variable dependency. The proposed method utilizes contrastive learning to extract a representation of only normal data that is distinct from anomalies without labels. In addition, patch based temporal attention is used to learn inter-patch and intra-patch relationships. Finally, cross-variable attention captures not only the relationships between variables at the same point in time, but also the relationships between variables at different points in time.

To summarize, the contributions of this study are as follows.

- Utilizes the patch module and contrastive learning to learn a consistent representation of normal without labels.
- Proposes cross-variable attention to capture correlations between variables at different time points.
- Demonstrates high anomaly detection performance on a multivariate time series anomaly detection benchmark dataset.

The composition of this paper is as follows. First, in Section 2, we review prior work on unsupervised time series anomaly detection and contrastive learning. Section 3 describes the method we propose in this study, and then evaluates the performance of the proposed method in Section 4. Section 5 presents conclusions and future works.

II. RELATED WORKS

A. Unsupervised Time Series Anomaly Detection

Time series anomaly detection is broadly divided into supervised and unsupervised learning based on the presence or absence of labels during training. However, since it is difficult to acquire labeled data in actual industrial sites, unsupervised time series anomaly detection models are mainly studied.

*Corresponding author—Tel: +82-2-3290-3396; Fax: +82-2-3290-4550

The existing unsupervised time series anomaly detection models are shown in Fig. 1 (a), they only consider the dependencies between variables within the same time point [6], [7], [8]. However, in a real manufacturing process, it is necessary to reflect the relationship between variables at different time points, because there are time delay situations where fluctuations in a certain variable affect other variables several time points later. Therefore, an intuitive time series anomaly detection model is proposed as shown in Fig. 1 (b).

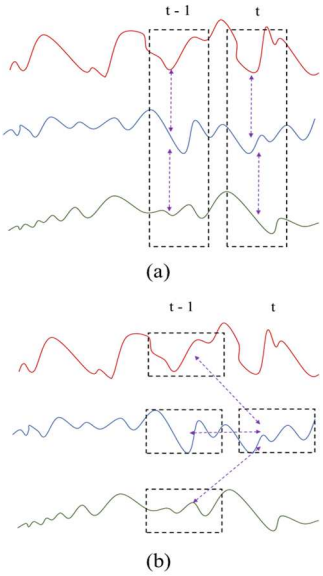

Fig. 2. (a) Considers dependencies between variables at the same time point, (b) relationships between variables at different time points.

B. *Contrastive Learning in Time Series*

Representation learning is an approach that seeks to obtain useful representations from unlabeled data to improve the performance of various downstream tasks [9]. It extracts information based on the inherent structure and patterns in the data.

Contrastive learning using data augmentation is a typical representation learning technique [10]. The goal of contrastive learning is to learn an embedding space where similar data samples are close together and dissimilar data samples are far apart. A classical contrastive learning model is trained by generating pairs of samples, where positive pairs represent similar data and negative pairs represent different data.

However, unlike the positive pairs generated by the data augmentation, it is difficult to define the negative pairs. This is because in unsupervised learning, similar samples can be selected as negative pairs due to the lack of labels. Therefore, models that do not use negative pairs have recently emerged [11], [12]. Typically, they use the structure of a siamese network, which is a network with shared weights for two or more inputs. However, this structure can suffer from mode collapsing, where all output values converge to a single constant. One of the methods used to solve this problem is the stop-gradient method [12].

Most contrastive learning models are based on image data. However, recently, methods for applying contrastive learning to time series data have appeared [9], [13]. However, the data augmentation used by these methods do not take advantage of the inherent characteristics of time series data [14]. Therefore, unlike existing methods that use data augmentation to generate positive pairs, this study considers the characteristics of time series data. Specifically, it reflects the characteristic that normal and abnormal have different representations from different perspectives [7].

III. PROPOSED METHOD

In this study, we propose a multivariate time series anomaly detection method that combines the patch module and contrastive learning to learn the representation of normal data and reflect the relationship of variables with different time points. The overall structure of the proposed method is shown in Fig. 2. The proposed method consists of the following four steps (1) Patch module: divide the whole time series into patches. (2) Attention module: Learn each representation through inter-patch attention, intra-patch attention, and cross-variable attention. (3) Encoder: Equalize the dimensionality of the representations derived from each attention. (4) Training: The similarity between the final output, the inter-patch representation and the intra-patch representation, is measured by the Kullback-Leibler (KL) divergence and trained as a loss function. Finally, the anomaly score is used to determine whether the data point is normal or abnormal.

A. *Patch module*

The patch module converts an original time series into patches. An input time series $X \in \mathbb{R}^{P \times N \times d}$ is passed through the patch module to $X \in \mathbb{R}^{P \times N \times d}$. T is the length of the time series, d is the number of variables, P is the patch size, and N is the number of patches. The product of P and N is equal to

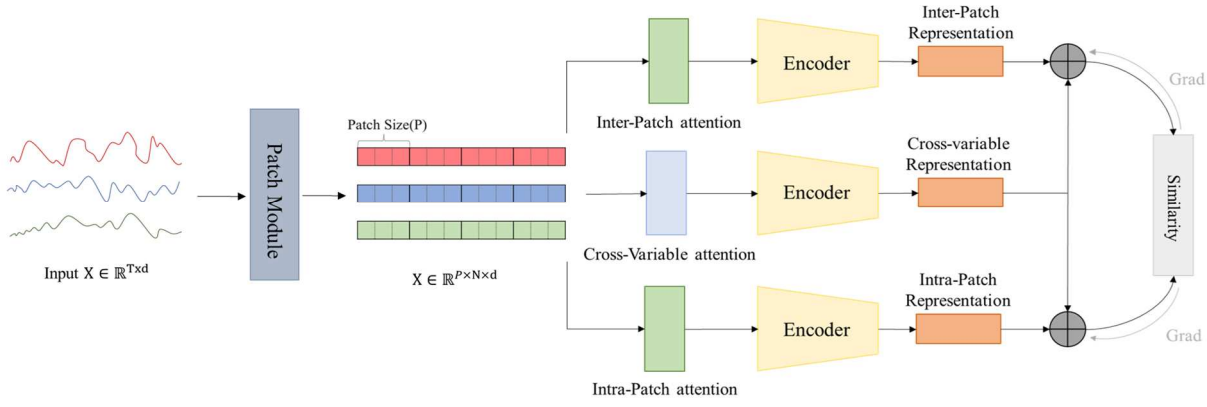

Fig. 2. Architecture of Proposed Methods

T. When input to the inter-patch and intra-patch attention modules, the batch size and the variable dimension are combined to form a univariate time series $X \in \mathbb{R}^{P \times N}$. Cross-variable attention, on the other hand, reflects the relationship between variables and is therefore input as $X \in \mathbb{R}^{N \times d}$. This has the advantage of reducing the number of input tokens, which reduces computation and memory usage [15]. It can also reflect more local semantic information than a point-wise time series [16].

B. Attention module

The proposed method consists of three types of attention, as shown in Fig. 3: Inter-patch attention to reflect the relationship between patches, intra-patch attention to check the interaction within patches, and cross-variable attention to consider the relationship between variables at different time points. Normal data will share similar feature patterns because they are strongly correlated with each other, so the difference between representations from different perspectives will be small, while anomalies will be large. Therefore, inter-patch and intra-patch are used to view the representation of the time series from different perspectives. In addition, cross-variable attention is used to reflect the relationship between variables at different time points, i.e., not only the relationship between variables at the same time point, but also the relationship with variables at previous time points.

1) Inter-patch attention

Fig. 3 (a) illustrates inter-patch attention. $X \in \mathbb{R}^{P \times N}$ converted into patch units by the patch module is input. After that, it is converted into an embedding vector of $X_N \in \mathbb{R}^{N \times d_{model}}$ through the embedding module. Here, d_{model} is the embedding vector size. To understand the relationship between the embedding vectors, we apply inter-patch attention. The initial query and key are as shown in Equation (1).

$$Q_{N_i}, K_{N_i} = W_{Q_i} X_{N_i}, W_{K_i} X_{N_i}, 1 \le i \le \mathrm{H} \qquad (1)$$

$Q_{N_i}, K_{N_i} \in \mathbb{R}^{N \times \frac{d_{model}}{H}}$ denotes the query and key, and $W_{Q_i}, W_{K_i} \in \mathbb{R}^{\frac{d_{model}}{H} \times \frac{d_{model}}{H}}$ denotes the learnable parameter matrix. H is the number of heads. Next, the attention weight is equal to Equation (2).

$$Attn_{N_i} = \mathrm{Softmax}\left(\frac{Q_{N_i} {K_{N_i}}^T}{\sqrt{d_{model}}}\right) \qquad (2)$$

It computes the dot product of query and key, divides by $\sqrt{d_{model}}$ to scale it. It then converts it to a probability distribution using Softmax. As a result, we calculate how relevant a particular patch is to other patches. Then, concatenate each representation of the multi-head and derive the inter-patch representation $Attn_N$ as shown in Equation (3).

$$Attn_N = Concat(\, Attn_{N_1}, \cdots, Attn_{N_H}) W_N^o \qquad (3)$$

$W_N^o \in \mathbb{R}^{d_{model} \times d_{model}}$ is the learnable parameter matrix.

2) Inter-patch attention

The intra-patch attention in Fig. 3 (b) learns the dependencies between points within the same patch. It shares weights with the inter-patch attention network. Same as inter-patch attention, attention is applied to compute the relationship between each embedding vector. The initial query and key are as shown in Equation (4).

$$Q_{P_i}, K_{P_i} = W_{Q_i} X_{P_i}, W_{K_i} X_{P_i}, 1 \le i \le \mathrm{H} \qquad (4)$$

$Q_{P_i}, K_{P_i} \in \mathbb{R}^{P \times \frac{d_{model}}{H}}$ denotes the query and key, and $W_{Q_i}, W_{K_i} \in \mathbb{R}^{\frac{d_{model}}{H} \times \frac{d_{model}}{H}}$ denotes the learnable parameter matrix. The attention weights $Attn_{P_i}$ are computed in a similar way to (2).

However, unlike (2), which reflects the relationship between patches, it calculates how relevant a particular point in a patch is to other points. Finally, the intra-patch representation $Attn_P$ is derived in the same way as equation (3).

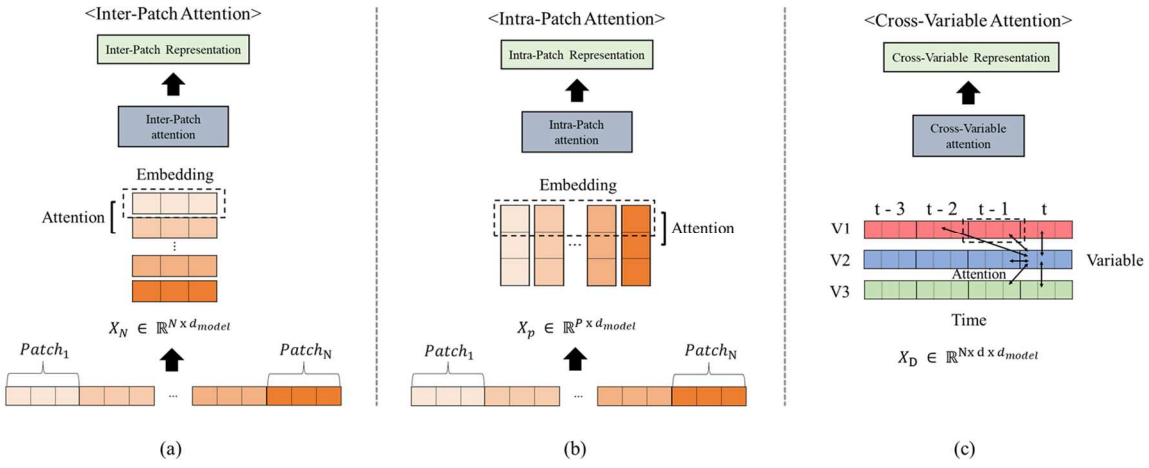

Fig. 3. Attention module: It consists of three attentions. (a) Inter-patch attention, which looks at relationships between patches; (b) Intra-patch attention, which learns relationships within patches; and (c) Cross-variable attention, which learns relationships between variables at different time points.

3) Cross-variable attention

Inter-patch attention and intra-patch attention consider temporal dependency. At the same time, we apply cross-variable attention as shown in Fig. 3 (c) to reflect the relationship between variables with different time points. The input data $X \in \mathbb{R}^{N \times d}$ are transformed into an embedding vector of $X_D \in \mathbb{R}^{N \times d \times d_{model}}$ by the embedding module, i.e., there are d time series, and each time series consists of N patches. We apply cross-variable attention to compute the relationship between patches at one point in time and patches at another point in time. The initial query and key are as shown in Equation (5).

$$Q_{D_i}{}^{t}, K_{D_i}{}^{t-F} = W_{q_i} X_{D_i}{}^{t}, W_{k_i} X_{D_i}{}^{t-F}, \quad 1 \le i \le \mathrm{H}, \ 1 \le t \le N \tag{5}$$

$Q_{D_i}{}^{t}$, $K_{D_i}{}^{t-F} \in \mathbb{R}^{d \times \frac{d_{model}}{H}}$ denotes the query and key, and $W_{q_i}, W_{k_i} \in \mathbb{R}^{\frac{d_{model}}{H} \times \frac{d_{model}}{H}}$ denotes the learnable parameter matrix. At this time, as shown in Fig. 3 (c), $Q_{D_i}{}^{t}$ is formed by the variables at a certain time, i.e., all the patches existing at t, and $K_{D_i}{}^{t-F}$is formed by the patches at the previous time, $t - F$ ($F \ge 0$). Here, F is a hyper parameter, which means that it reflects the relationship between the patches at time t and the patches at time $t - F$. The attention weight of a query at time t with all keys up to time $t - F$ is given by Equation (6).

$$Attn_{D_i}{}^{t-F} = \mathrm{Softmax}\left(\frac{Q_{D_i}{}^{t} K_{D_i}{}^{t-F\,T}}{\sqrt{d_{model}}}\right) \tag{6}$$

Furthermore, the farther away the point is, the lower the influence will be. To accomplish this, we apply an exponential smoothing weighting method. This gives more weight to recent observations and an exponentially decreasing weight to past observations. The weight at time $t - F$ is given by equation (7).

$$Weight_{t-F} = e^{-\beta \times F} \tag{7}$$

β is a parameter that controls the rate of weight decay. The larger β leads to a more rapid decrease in weight, while a smaller β results in a slower decrease in weight. The final attention weight, which reflects the relationship between all patches from time t to time $t - F$, is equal to (8).

$$\begin{aligned} Attn_{D_i} = {} & Weight_{t-F} \times Attn_{D_i}{}^{t-F} \\ & + Weight_{t-F-1} \times Attn_{D_i}{}^{t-F+1} \\ & + \cdots + Weight_{t} \times Attn_{D_i}{}^{t} \end{aligned} \tag{8}$$

Add up all the relationships between all the variables at a given time t, as well as the relationships between the variables at time $t - F$ and the variables at time t. Finally, concatenating each of the multi-head representations and deriving the cross-variable representation $Attn_D$ is shown in Equation (9).

$$Attn_D = Concat(Attn_{D_1}, \cdots, Attn_{D_H}) W_D^O \tag{9}$$

$W_D^O \in \mathbb{R}^{d_{model} \times d_{model}}$ is the learnable parameter matrix.

C. Encoder

The three representations ($Attn_N$, $Attn_P$, $Attn_D$) generated by the Attention module need to calculate similarity when training. Therefore, we need to equalize the dimensions through an encoder. We used a fully connected layer as the encoder. The encoders for each representation all have the same structure, but they are trained independently. Finally, three representations with the same dimensionality are derived as shown in Equation (10), (11), and (12).

$$R_{Inter-patch} = f(Attn_N) \tag{10}$$

$$R_{Intra-patch} = f(Attn_P) \tag{11}$$

$$R_{cross-variable} = f(Attn_D) \tag{12}$$

Finally, the cross-variable representation is added to the inter-patch representation and the intra-patch representation to derive the final inter-patch representation N and the final intra-patch representation P. N and P are equal to (13), (14).

$$N = R_{Inter-patch} + R_{cross-variable} \tag{13}$$

$$P = R_{Intra-patch} + R_{cross-variable} \tag{14}$$

D. Training

For the same input, we derive two representations, N and P, from different perspectives (inter-patch and intra-patch). We use KL-divergence to measure the similarity of the two representations. According to the previous assumption, representations of the same normal input should be similar to each other because there are few anomalies, and normal data share latent patterns. The similarity metric for two representations N and P is defined as $D(N, P) = KL(N||P)$. where $KL(\cdot \,||\, \cdot)$is the KL divergence. The final loss function is shown in Equation (15):

$$\begin{aligned} L\{N, P; X\} = {} & \frac{1}{2} D\big(N, Stopgrad(P)\big) \\ & + \frac{1}{2} D(P, Stopgrad(N)) \end{aligned} \tag{15}$$

where X is the input time series, and N and P are the inter-patch representation and intra-representation. Stopgrad means stop-gradient, which prevents the gradient from updating when backpropagation, so that N and P are alternately updated and become similar. The anomaly score for a time series $X \in \mathbb{R}^{T \times d}$is given by Equation (16):

$$AnomalyScore(X) = \frac{1}{2} D(N, P) + \frac{1}{2} D(P, N) \tag{16}$$

Based on a threshold δ, which is a hyper-parameter, anomaly score is judged as abnormal (1) if it exceeds δ, and normal (0) otherwise.

IV. EXPERIMENTS

A. Dataset

To evaluate the proposed model, the Soil Moisture Active Passive (SMAP) dataset, Mars Science Laboratory (MSL) dataset, and Secure Water Treatment (SWaT) dataset were used. SMAP provides soil samples and remote sensing information measured by NASA's Mars rovers. MSL is NASA's Mars rover sensor and actuator data. SWaT is data from real industrial water treatment plants that produce purified water. The details of the datasets are summarized in Table I.

TABEL I. DETAILS OF BENCHMARK DATASETS

Data	Dimension	# Train	# Test	Anomaly Ratio (%)
SMAP	25	135,183	427,617	12.8
MSL	55	58,317	73,729	10.5
SWaT	51	496,800	449,919	11.98

B. Evaluation Metrics

To evaluate the performance, we used confusion matrix as shown in Table II.

TABEL II. CONFUSTION MATRIX

		Actual	
		True (Abnormal)	False (Normal)
Predicted	True (Abnormal)	TP (True Positive)	FP (False Positive)
	False (Normal)	FN (False Negative)	TN (True Negative)

According to Table II, Precision, Recall, and F1 score were used as evaluation metrics. Each metric is shown in Equations (17), (18), and (19).

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{19}$$

C. Experimental Design

The GPU specification used in the experiment is NVIDIA GeForce GTX 1080 Ti 11GB. We used 1 head for training and set the d_{model} to 256. The initial learning rate was set to 0.001 and the optimizer was Adam. Patch size was set to 5 and window size to 100. Threshold δ is specified as the top α% percentile, where α is measured on the validation set. For performance evaluation, we use the popular point adjustment technique [6], [7], [17]. In this method, if some point is detected in a certain consecutive anomaly interval, all anomalies in that interval are considered to be correctly detected.

D. Experiment Results

To compare the performance in multivariate time series anomaly detection, we set Omnianomaly, USAD, TranAD, Anomaly Transformer, and DCdetector as the comparison models [6], [7], [17], [18], [19].

Table III presents a performance comparison between the proposed method and other time series anomaly detection models. The experiment was repeated five times, and the results are presented as the average performance. When comparing the performance based on F1 score, we can see that the proposed method outperforms the comparison models.

In particular, the proposed method outperforms the attention-based models Anomaly Transformer and DCdetector. This means that the proposed method captures the interactions between variables that occur across time points more precisely through cross-variable attention. It can also be interpreted as a stable model in that the standard deviation is the smallest in the SMAP dataset and the second smallest in the MSL dataset.

The visualization of false positive case and true positive case in SWaT dataset is shown in Fig. 5. We compare the proposed method with DCdetector, which is the second best performer based on F1 score in Table III, through visualization. The blue section is the true positive case and the red section is the false positive case. The proposed method in Fig. 4 (b) has

TABEL III. PERFORMANCE COMPARISION TABLE

Dataset	SMAP			MSL			SWaT		
Metric	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
Omnianomaly	0.9125 (0.0372)	0.7219 (0.0214)	0.8060 (0.0262)	0.9183 (0.0473)	0.7151 (0.1399)	0.7934 (0.0924)	0.9416 (0.0502)	0.7008 (0.0073)	0.8027 (0.0140)
USAD	0.9253 (0.0255)	0.7112 (0.1871)	0.7905 (0.1258)	0.9234 (0.0114)	0.6577 (0.2040)	0.7506 (0.1418)	0.9529 (0.0585)	0.7121 (0.0344)	0.8127 (0.0011)
TranAD	0.8646 (0.0665)	0.8195 (0.1110)	0.8350 (0.0578)	0.8500 (0.0592)	0.9679 (0.1742)	0.8558 (0.0788)	0.9763 (0.0151)	0.6951 (0.0051)	0.8120 (0.0016)
Anomaly Transformer	0.7690 (0.0461)	0.9964 (0.0009)	0.8879 (0.0136)	0.8500 (0.0066)	0.9679 (0.0095)	0.9051 (0.0061)	0.9042 (0.0569)	0.9362 (0.0033)	0.9045 (0.0319)
DCdetector	0.8926 (0.0055)	0.9879 (0.0060)	<u>0.9378</u> (0.0014)	0.8634 (0.0014)	0.9905 (0.0016)	<u>0.9226</u> (0.0003)	0.8944 (0.0152)	0.9737 (0.0222)	<u>0.9265</u> (0.0102)
Proposed Method	0.9364 (0.0000)	0.9916 (0.0009)	**0.9632 (0.0004)**	0.9216 (0.0005)	0.9823 (0.0059)	**0.9509 (0.0030)**	0.9371 (0.0061)	0.9818 (0.0256)	**0.9587 (0.0092)**

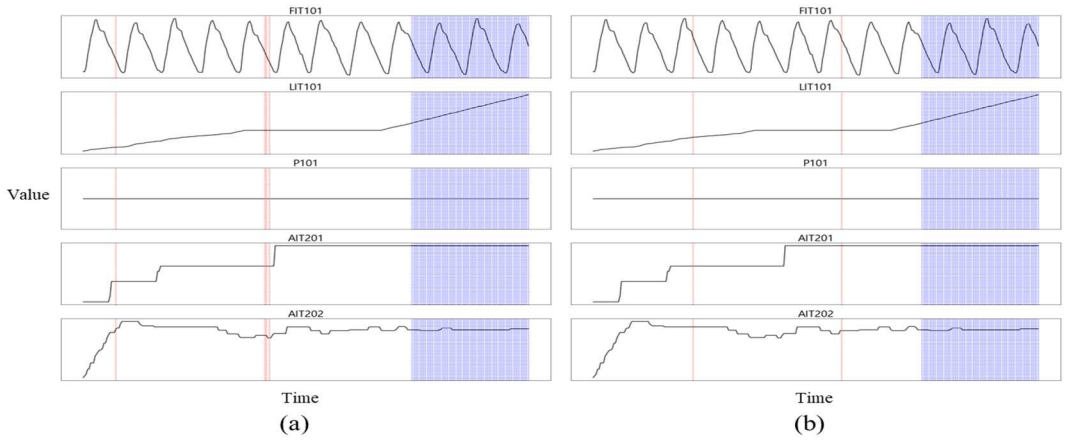

Fig. 4. Red segments are the false positive case, and blue segments are the true positive case. (a) DCdetector, (b) Proposed method.

fewer false positive cases than DCdetector in Fig. 4 (a), which means that the proposed method has learned the relationship between variables well, so the frequency of false alarms is lower in the normal pattern similar to the actual abnormal pattern.

V. Conclusion

In this study, we proposed a patch-based contrastive learning anomaly detection model that reflects the dependencies between variables in different time points. The proposed method considers cross-variable dependency and temporal dependency, and improves the anomaly detection performance by extracting representations of the normal by inter-patch attention and intra-patch attention.

For the three benchmark datasets, we obtained superior results compared to the F1 score baseline comparison models. Therefore, it is expected that the proposed method can efficiently learn the relationship between variables to detect anomalies in process situations with time delays.

In this study, the hyper-parameter F was set to 1, i.e., only the relationship between variables before one patch was considered. However, this has the limitation that it does not reflect the case of time delay for multiple time points. In the future, we plan to increase the number of F.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (NRF-2022R1A2C2004457). This work was also supported by Samsung Electronics Co., Ltd. (IO201210-07929-01) and Brain Korea 21 FOUR.

References

[1] K. Choi, J. Yi, C. Park, and S. Yoon, “Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines,” IEEE Access, vol. 9, pp. 120043–120065, 2021

[2] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[3] Vaswani, A., et al. "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.

[4] S. Bai, J. Kolter, V. Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," arXiv preprint arXiv:1803.01271, 2018.

[5] Y. Zhang, J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," The Eleventh International Conference on Learning Representations, 2022.

[6] Audibert, J., Michiardi, P., Guyard, F., Marti, S., & Zuluaga, M. A., "Usad: Unsupervised anomaly detection on multivariate time series," Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020, pp. 3395–3404.

[7] Xu, J., Wu, H., Wang, J., & Long, M, "Anomaly transformer: Time series anomaly detection with association discrepancy," In International Conference on Learning Representations, 2021.

[8] A. Deng, B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," Proceedings of the AAAI conference on artificial intelligence, 2021, pp. 4027–4035.

[9] Eldele, E., et al. "Time-series representation learning via temporal and contextual contrasting," arXiv preprint arXiv:2106.14112, 2021.

[10] Zhang, X., Zhao, Z., Tsiligkaridis, T., & Zitnik, M., "Self-supervised contrastive pre-training for time series via time-frequency consistency," Advances in Neural Information Processing Systems, vol. 35, pp. 3988–4003, 2022.

[11] Grill, J., et al. "Bootstrap your own latent-a new approach to self-supervised learning," Advances in neural information processing systems, vol. 33, pp. 21271–21284, 2020.

[12] X. Chen, K. He, "Exploring simple siamese representation learning," Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 15750–15758.

[13] Woo, G., Liu, C., Sahoo, D., Kumar, A., & Hoi, S., "CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting," arXiv preprint arXiv:2202.01575, 2022.

[14] Wen, Q., et al. "Time series data augmentation for deep learning: A survey," arXiv preprint arXiv:2002.12478, 2020.

[15] Cirstea, R., et al. "Triformer: Triangular, Variable-Specific Attentions for Long Sequence Multivariate Time Series Forecasting–Full Version," arXiv preprint arXiv:2204.13767, 2022.

[16] Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J., "A Time Series is Worth 64 Words: Long-term Forecasting with Transformers," The Eleventh International Conference on Learning Representations, 2022.

[17] Su, Y., et al, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 2828–2837.

[18] S. Tuli, G. Casale, N. Jennings. "TranAD: deep transformer networks for anomaly detection in multivariate time series data," Proceedings of the VLDB Endowment, vol. 15, no. 6, pp. 1201–1214, 2022.

[19] Yang, Y., Zhang, C., Zhou, T., Wen, Q., & Sun, L., "DCdetector: Dual Attention Contrastive Representation Learning for Time Series Anomaly Detection," Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023.