Taylor & Francis
Taylor & Francis Group

Applied Artificial Intelligence
An International Journal

ISSN: 0883-9514 (Print) 1087-6545 (Online) Journal homepage: www.tandfonline.com/journals/uaai20

Multivariate Time Series Anomaly Detection Using Directed Hypergraph Neural Networks

Tae Wook Ha & Myoung Ho Kim

To cite this article: Tae Wook Ha & Myoung Ho Kim (2025) Multivariate Time Series Anomaly Detection Using Directed Hypergraph Neural Networks, Applied Artificial Intelligence, 39:1, 2538519, DOI: 10.1080/08839514.2025.2538519

To link to this article: https://doi.org/10.1080/08839514.2025.2538519

Published online: 29 Jul 2025.

APPLIED ARTIFICIAL INTELLIGENCE
2025, VOL. 39, NO. 1, e2538519 (17 pages)
https://doi.org/10.1080/08839514.2025.2538519

Multivariate Time Series Anomaly Detection Using Directed Hypergraph Neural Networks

Tae Wook Ha and Myoung Ho Kim

School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea

ABSTRACT

Multivariate time series anomaly detection is a challenging problem because there can be a number of complex relationships between variables in multivariate time series. Although graph neural networks have been shown to be effective in capturing *variable-variable relationships* (i.e. relationships between two variables), they are hard to capture *variable-group relationships* (i.e. relationships between variables and groups of variables). To overcome this limitation, we propose a novel method called DHG-AD for multivariate time series anomaly detection. DHG-AD employs directed hypergraphs to model variable-group relationships within multivariate time series. For each time window, DHG-AD constructs two different directed hypergraphs to represent relationships between variables and groups of positively and negatively correlated variables, enabling the model to capture both types of relationships effectively. The directed hypergraph neural networks learn node representations from these hypergraphs, allowing comprehensive multivariate interaction modeling for anomaly detection. We show through experiments using various evaluation metrics that our proposed method achieves the best scores among the compared methods on two real-world datasets.

ARTICLE HISTORY
Received 11 November 2024
Revised 27 May 2025
Accepted 18 July 2025

Introduction

Multivariate time series are often generated from complex real-world systems. For instance, a water treatment plant employs diverse sensors to measure flow rates, water levels, valve statuses, etc. The sequence of values from these sensors can be considered as a multivariate time series. Similarly, server machines in data centers generate multivariate time series consisting of various metrics such as CPU load, network traffic, memory usage, and so on. These multivariate time series contain valuable information for numerous tasks, e.g., forecasting, classification, recognition, and anomaly detection.

Anomaly detection in multivariate time series has emerged as an important task, such as receiving alerts about critical issues to provide suitable solutions for a healthy state of a system. Domain experts may be able to utilize their knowledge to detect anomalous states in a single time series or a multivariate time series with a few variables. However, as the number of variables (e.g. sensors and metrics mentioned above) increases, detecting anomalous states in multivariate time series becomes overwhelmingly complex. To address this challenge, various machine learning-based methods have been proposed. Most of recent machine learning-based anomaly detection methods are based on unsupervised learning. This is because it is difficult in general to obtain data about anomalous states in advance, except for clearly extreme cases, such as threshold values of variables. Unsupervised methods generally utilize machine learning models to learn complex patterns of time series obtained from normal states, and identify anomalies that significantly deviate from the learned patterns. For example, several methods (Sanayha and Vateekul 2017; Yu, Jibin, and Jiang 2016) use linear models such as auto-regressive integrated moving average. Deep learning-based methods have also been proposed using various types of neural network models such as an LSTM-based variational autoencoder (Park, Hoshi, and Kemp 2018), a stochastic recurrent neural network (Su et al. 2019), a GRU-based Gaussian mixture model (Guo et al. 2018), and a Transformer-based model (Xu et al. 2022). However,

CONTACT Tae Wook Ha rooterby@kaist.ac.kr School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea

these methods do not provide an appropriate mechanism for learning relationships between variables that are also important in multivariate time series.

To address this challenge, existing methods focus primarily on learning *variable-variable relationships* (i.e., relationships between two variables) within multivariate time series. For example, MSCRED (Zhang et al. 2019) uses a signature matrix where each element represents a pairwise correlation, thus capturing the relationship between two variables. Recently, several approaches that leverage graph neural networks (GNNs) have been introduced to model variable-variable relationships in multivariate time series. Techniques such as MTAD-GAT (Zhao et al. 2020), GDN (Deng and Hooi 2021), and STGAT-MAD (Zhan et al. 2022) use GNNs to represent variable-variable relationships by assigning variables to graph nodes and representing the relationships as edges of the graph. However, many real-world systems often exhibit relationships among more than two variables. In such systems, the behavior of a variable may be influenced by simultaneous interactions with several other variables, not just one or two variables but groups of its related variables. For instance, in networked systems, the load on a server might be influenced by groups of other related metrics including CPU usage, memory consumption, and network traffic. Limiting the model to variable-variable relationships may not adequately capture these interactions. Therefore, to accurately detect anomalies in multivariate time series, it is essential to model *variable-group relationships* (i.e., relationships between variables and groups of their related variables) rather than relying on variable-variable relationships alone.

In this paper, we propose a novel method called the **D**irected **H**yper**G**raph neural networks for multivariate time series **A**nomaly **D**etection (DHG-AD), which is designed to capture variable-group relationships. Unlike graphs that model variable-variable relationships, DHG-AD constructs directed hypergraphs that serve as an effective model to represent variable-group relationships, making them well suited for modeling the simultaneous effects of multiple variables. DHG-AD uses two types of directed hypergraphs as follows: *P-DHG* to capture the relationships between variables and groups of positively correlated variables, and *N-DHG* to capture those between variables and groups of negatively correlated variables. This dual hypergraph approach allows DHG-AD to model the relationships of both positively and negatively correlated variables effectively. After that, directed hypergraph neural networks produce node representations from both P-DHG and N-DHG, providing a comprehensive model of multivariate interactions for anomaly detection.

The remainder of this paper is organized as follows. Section 2 provides related work. In Section 3, we describe the preliminaries of our proposed method. Section 4 presents the proposed method DHG-AD. In Section 5, we show through extensive performance experiments that DHG-AD attains higher scores than the existing methods on all evaluation metrics. Section 7 concludes our paper.

Related Work

Anomaly Detection

The goal of anomaly detection is to detect anomalous states by identifying data points that are significantly different from those observed in normal states. Early methods for anomaly detection utilize statistical analysis such as density (Breunig et al. 2000), k-nearest neighbor (Hautamäki, Kärkkäinen, and Fränti 2004), clustering (He, Xu, and Deng 2003), one-class support vector machine (Manevitz and Yousef 2001), and support vector data description (Tax and Duin 2004). Most machine learning-based anomaly detection methods use unsupervised learning approaches. Note that although there are a few anomaly detection methods (Goernitz et al. 2013; Park et al. 2017; Rodriguez et al. 2010) that use supervised learning, they are hard to be applicable to complex real-world applications in practice because labeled data about anomalous states in such complex applications are difficult to collect.

Deep learning approaches have been widely used for anomaly detection because they can capture complex characteristics of data. Aggarwal (2013) utilizes autoencoders to detect anomalies based on reconstruction errors. Zong et al. (2018) integrate a deep autoencoder with Gaussian mixture modeling to jointly learn low-dimensional representations and distributions of data. Sohn et al. (2021) suggest a two-stage framework that combines self-supervised deep learning with a shallow one-class classifier. Ruff et al. (2018) uses a neural network transformation that attempts to map most

of the data to a hypersphere and identifies anomalies that are mapped outside the hypersphere. Ruff et al. (2021, 2020) use semi-supervised learning (that is, some labeled data) to improve (Ruff et al. 2018).

Multivariate Time Series Anomaly Detection

Different from anomaly detection methods mentioned above, this paper aims to detect anomalous states in multivariate time series. Early methods for multivariate time series anomaly detection utilize linear time series models such as auto-regressive integrated moving average (Sanayha and Vateekul 2017; Yu, Jibin, and Jiang 2016) or vector auto-regressive models (Melnyk et al. 2016). In addition, to model nonlinearities in multivariate time series, deep learning models have also been used in anomaly detection methods. LSTM-VAE (Park, Hoshi, and Kemp 2018) uses an LSTM-based variational autoencoder to estimate the data distribution at each time point. OmniAnomaly (Su et al. 2019) uses an auto-encoder composed of stochastic recurrent neural networks, and GMM-GRU-VAE (Guo et al. 2018) integrates a Gaussian mixture model with a gated recurrent unit (GRU)-based encoder and decoder. AnomalyTransformer (Xu et al. 2022) proposes a Transformer-based model with an anomaly-attention mechanism. Although these methods demonstrate effectiveness in modeling multivariate time series, they do not explicitly address relationships among variables that are also important in multivariate time series. To capture the relationships, MSCRED (Zhang et al. 2019) proposes a multiscale convolutional recurrent neural network-based encoder-decoder model that learns correlations between two variables.

Recent methodologies, including MTAD-GAT (Zhao et al. 2020), GDN (Deng and Hooi 2021), and STGAT-MAD (Zhan et al. 2022) apply GNNs to graphs where each node represents each variable and each edge indicates pairwise relationships between two variables (in (Zhao et al. 2020), time points are also used as nodes). These methods perform the following common processes. For a given window of a multivariate time series, i) build a graph and calculate embedding vectors for the nodes. ii) Use GNNs to embed vectors of nodes to learn relationships among variables. Unlike these GNN-based methods, our DHG-AD aims to effectively capture variable-group relationships that involve groups of variables. DHG-AD specifically considers these relationships that exhibit positive and negative correlations simultaneously.

Preliminaries of Our Proposed Method

Problem Definition

A multivariate time series $\mathcal{X} = \{x_1, \ldots, x_\mathcal{T}\}$ is a sequence of data points indexed in time order, where $\mathcal{T}$ is a length of the time series and each data point $x_t \in \mathbb{R}^n$ at time point t is an n dimensional vector derived from n variables. Note that t is simply an ordered index value and does not need to be an actual time. $\mathcal{X}$ is reduced to a univariate time series when $n = 1$. In this paper, we will focus on a multivariate time series, i.e. $n \geq 2$. The problem of anomaly detection aims to predict a sequence of labels $\mathcal{Y} = \{y_1, \ldots, y_\mathcal{T}\}$ where each label $y_t \in \{0, 1\}$ indicates if an anomaly occurs at time point t.

Unsupervised Anomaly Detection

Except for simple cases, such as threshold values for variables, labeled data about anomalous states are often unavailable in complex real-world applications. Thus, in this paper, we will use an unsupervised learning approach for multivariate time series anomaly detection. Let $\mathcal{X}_{train}$ and $\mathcal{X}_{test}$ denote two time series for training and testing, respectively. We also assume that all data points in $\mathcal{X}_{train}$ are collected from normal states, that is, collected when a system is working correctly. In the training step, our proposed DHG-AD learns the relationships among variables using data points in $\mathcal{X}_{train}$. In the testing step, we compute an anomaly score α_t for each data point at time point t in $\mathcal{X}_{test}$ where α_t is the amount that x_t deviates from the learned relationships of the time series $\mathcal{X}_{train}$. Each label y_t in $\mathcal{Y}$ is 1, if α_t exceeds a certain threshold δ. y_t is 0, otherwise.

Data Preprocessing

Data normalization. To ensure robust learning of the model, we normalize all data points in $\mathcal{X}_{train}$ before input to the model. Among many normalization techniques, we will use min-max normalization in the experiments in Section 5.

Model input. Consider a multivariate time series $\mathcal{X} = \{x_1, \ldots, x_\mathcal{T}\}$. $\mathcal{X}$ can be either $\mathcal{X}_{train}$ or $\mathcal{X}_{test}$. For each time point in $\{1, 2, \ldots, \mathcal{T}\}$, a sliding window $W_t = \{x_{t-w+1}, \ldots, x_t\}$ of size w is defined and becomes an input of our DHG-AD. That is, W_t is a sliding window of size w over $\mathcal{X}$. When t is less than w, W_t has only data points between x_1 and x_t.

Directed Hypergraph

Graphs that are commonly employed to model relationships between two objects cannot, in general, correctly express relationships involving more than two objects. Directed hypergraphs (Gallo et al. 1993), composed of nodes and hyperarcs, can be used to describe variable-group relationships that involve more than two objects. In this paper, we use a directed hypergraph to model various types of variable-group relationships in multivariate time series, where node v_i represents variables i (i.e. time series i) in multivariate time series for $1 \leq i \leq n$. A directed hypergraph $G = (V, E)$ consists of a finite set of nodes V and a set of hyperarcs E. Specifically, for each node v_i, we construct hyperarcs where each hyperarc $e_j = (e_j^{tail}, e_j^{head}), 1 \leq j \leq m$ consists of the tail e_j^{tail} that is a set of nodes correlated with node v_i, and the head $e_j^{head} = \{v_i\}$ contains only node v_i. m is the total number of hyperarcs. Two incidence matrices $H^{tail} \in \{0, 1\}^{n \times m}$ and $H^{head} \in \{0, 1\}^{n \times m}$ indicate the relations between nodes and hyperarcs. The elements of H^{tail} and H^{head}, $h^{tail}(i, j)$ and $h^{head}(i, j)$, become 1 if v_i is incident with e_j^{tail} and e_j^{head}, respectively. Otherwise, $h^{tail}(i, j)$ and $h^{head}(i, j)$ become 0. $\mathcal{W}_{arc} \in \mathbb{R}^{m \times m}$ denotes the diagonal matrix which contains the weights of hyperarcs.

Notations used in our method are summarized in Table 1.

Proposed Method

Overview of the Architecture

The overall architecture of DHG-AD is shown in Figure 1. It consists of three major components as follows: directed hypergraph construction, directed hypergraph convolution, and reconstruction. (1) **Directed hypergraph construction**: Given an input window W_t, DHG-AD constructs directed hypergraphs that model variable-group relationships in W_t. In this paper, we use two different types of relationships, i.e., relationships between variables and groups of positively or negatively correlated variables, and construct a directed hypergraph for each of them. The directed hypergraphs that model these two types of variable-group relationships are called P-DHG and N-DHG, respectively. Each variable within the window is represented as a node in the hypergraph, with the corresponding node feature derived from the window W_t. P-DHG and N-DHG are constructed using correlation values between node features to capture two types of variable-group relationships,

Table 1. Notations.

Symbols	Description
$\mathcal{X}$	A multivariate time series
W_t	An input window
n	The number of variables, i.e. the number of time series, in the multivariate time series
d	A dimension of node representations
F	Node feature matrix for a directed hypergraph
H^{tail}	Incidence matrix for the tail of a directed hypergraph
H^{head}	Incidence matrix for the head of a directed hypergraph
Z_P	Node representations produced by a directed neural network on P-DHG
Z_N	Node representations produced by a directed neural network on N-DHG
U	Unified node representations
$\hat{W}_t$	A reconstructed window

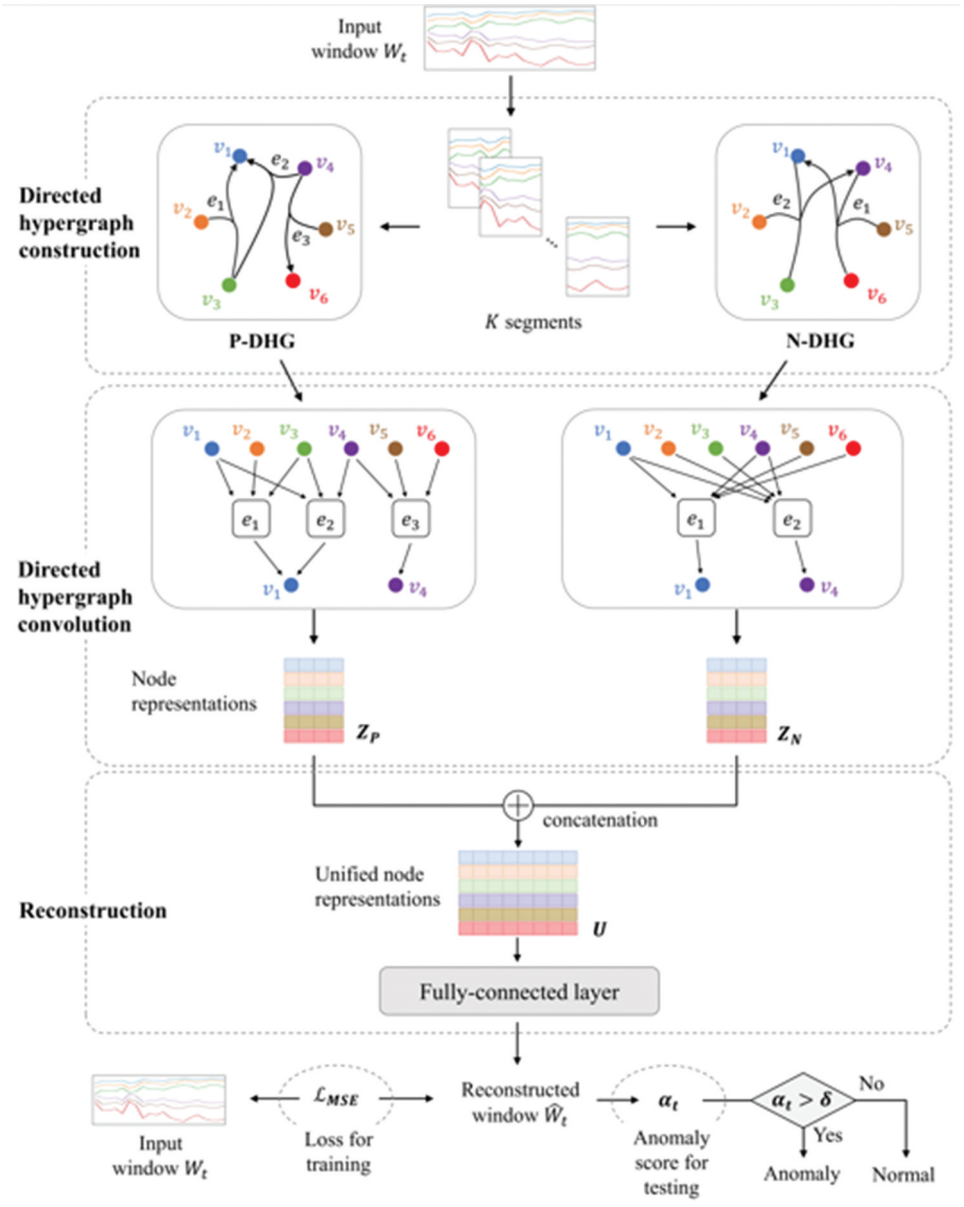

Figure 1. The overall architecture of DHG-AD. Given an input window W_t of size w, we split it into K overlapping segments of length l_s and build two directed hypergraphs: P-DHG and N-DHG. Each node feature is constructed from a concatenation of segment values in K segments, with a dimension $(K \times l_s)$. Then we apply a directed hypergraph convolution for each directed hypergraph. Each directed hypergraph convolution produces a node representation of dimension d (i.e., $(K \times l_s) \rightarrow d$). The unified node representation is a concatenation of node representations with $2d$ dimension, and used to generate the reconstructed window $\hat{W}_t$ of size.

respectively. (2) **Directed hypergraph convolution**: The constructed P-DHG and N-DHG become inputs to the directed hypergraph neural networks. Each network aggregates the node features that are connected by hyperarcs to generate the node representations. (3) **Reconstruction**: The outputs of the two directed hypergraph neural networks are concatenated to produce unified node representations that capture both types of variable-group relationships. The unified node representations are then passed through fully connected layers to reconstruct the input window $\hat{W}_t$. This reconstruction component allows the model to learn the underlying relationships and identify deviations from the normal patterns.

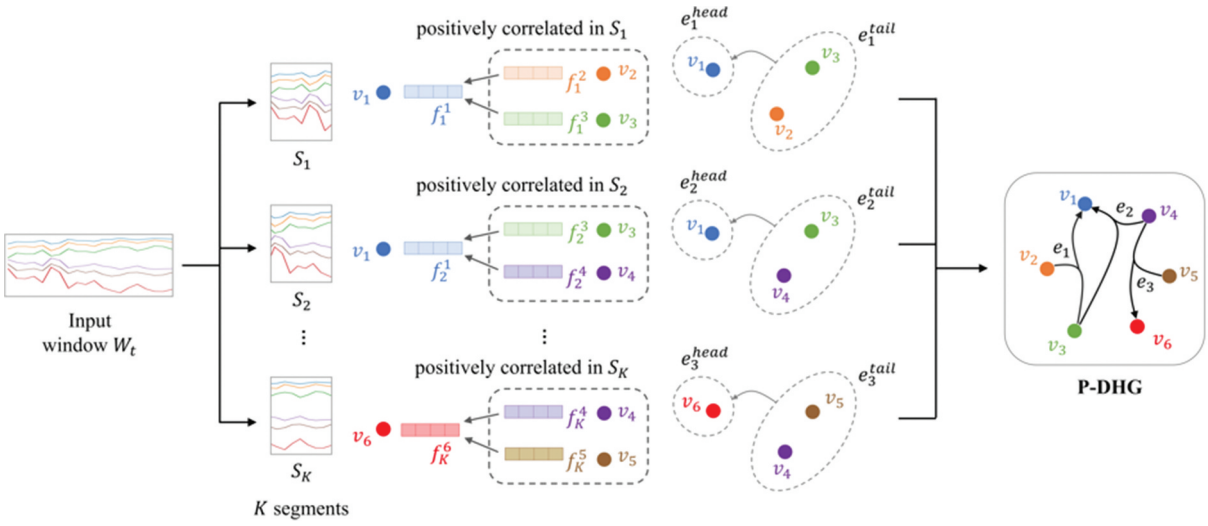

Figure 2. An example of the directed hypergraph construction using multiple overlapping segments.

Directed Hypergraph Construction

Given an input window W_t, we divide W_t into K multiple overlapping segments, denoted by $\{S_1, \ldots, S_K\}$. The use of overlapping segments allows the model to capture variable-group relationships where groups of correlated variables are dynamically changing over time. As shown in Figure 2, we construct multiple hyperarcs derived from the overlapping segments.

For each segment S_k, where k is the index of the segment, each variable i is represented by its raw segment values s_k^i that consists of values of variable i in segment S_k. Specifically, the segment values $s_k^i = \{x_{t_k}^i, x_{t_k+1}^i, \ldots, x_{t_k+l_s-1}^i\}$ are constructed from the values of variable i over the length l_s of segment S_k. The starting time point of each segment is determined by a fixed stride η:

$$t_k = t_{k-1} + \eta, \qquad t_1 = t - w + 1, \eta \leq l_s$$

where w is the size of W_t and t denotes the ending time point of W_t. When $\eta \leq l_s$, successive segments overlap, so every time point is covered by at least one segment. Note that this redundancy mitigates boundary effects by ensuring any temporal pattern crossing a segment edge is fully captured in at least one segment. Within each segment S_k, we then calculate the Pearson correlation matrix $\rho_k \in \mathbb{R}^{n \times n}$ that captures the correlations between two variables based on their behaviors in that segment. For any two variables i and j, the correlation coefficient $\rho_k(i,j)$ is given by:

$$\rho_k(i,j) = \frac{\sum_{t=t_k}^{t_k+l_s-1} (x_t^i - \mu_k^i)(x_t^j - \mu_k^j)}{\sqrt{\sum_{t=t_k}^{t_k+l_s-1} (x_t^i - \mu_k^i)^2} \sqrt{\sum_{k=t_k}^{t_k+l_s-1} (x_t^j - \mu_k^j)^2}} \tag{1}$$

where μ_k^i and μ_k^j represent the mean values of variables i and j within segment S_k.

p-Dhg

Given each segment S_k, we construct hyperarcs based on the correlation matrix ρ_k calculated from the segment. For each node v_i, we define a hyperarc $e_i = (e_i^{tail}, e_i^{head})$ as follows:

$$e_i^{tail} = \{v_i\} \cup \{v_j : 1 \leq j \leq n, j \neq i | \rho_k(i,j) > \tau_P\} \tag{2}$$

$$e_i^{head} = \{v_i\} \tag{3}$$

Here, e_i^{tail} consists of node v_i itself and all nodes v_j whose correlations with v_i in ρ_k exceed a threshold τ_P. Note that we include v_i in e_i^{tail} for a self-loop needed in a directed hypergraph convolution in the next component (Ma, Zhao, and Yang 2024). e_i^{head} is simply defined by $\{v_i\}$.

Incidence matrices. After constructing hyperarcs for all segments, we identify the unique set of hyperarcs by removing duplicates (i.e., hyperarcs with identical tail and head sets across segments). The incidence matrices H^{tail} and H^{head} for P-DHG are defined as follows:

$$H^{tail}(i,j) = \begin{cases} 1, & \text{if } v_i \in e_j^{tail}, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

$$H^{head}(i,j) = \begin{cases} 1, & \text{if } v_i \in e_j^{head}, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Here, j indexes the unique hyperarcs after removing duplicates.

Hyperarc weights. The weight $w(e_j)$ of a hyperarc e_j in $\mathcal{W}_{arc}$ is defined by the ratio of its occurrences across segments to the total number of segments K. Specifically:

$$w(e_j) = \frac{\text{occurrences of } e_j \text{in all segments}}{K} \tag{6}$$

The occurrences of a hyperarc e_j are counted by comparing it with the hyperarcs generated from each segment. This weight reflects how consistently the variable-group relationships, represented by e_j, across the segments. The higher weights are assigned to hyperarcs that occur more frequently.

n-Dhg

Similarly, we construct hyperarcs for N-DHG using the correlation matrix ρ_k computed for that segment. For each node v_i, we construct a hyperarc $e_i = (e_i^{tail}, e_i^{head})$ where e_i^{tail} consists of nodes whose correlations with v_i fall below a threshold τ_N. $e^{head}{}_i$ is simply $\{v_i\}$. Therefore, e_i^{tail} and e_i^{head} are calculated as follows.

$$e_i^{tail} = \{v_i\} \cup \{v_j : 1 \le j \le n, j \neq i | \rho_k(i,j) < \tau_N\} \tag{7}$$

$$e_i^{head} = \{v_i\} \tag{8}$$

Duplicate hyperarcs across segments are also removed to produce a unique set of hyperarcs. The incidence matrices H^{tail} and H^{head}, as well as the hyperarcs weights $w(e_j)$, are calculated in the same manner as described for P-DHG. The weight $w(e_j)$ reflects the number of segments in which e_j occurs, normalized by the total number of segments K.

Directed Hypergraph Convolution

Based on the constructed directed hypergraphs, we capture variable-group relationships by aggregating information through convolutions on both directed hypergraphs and generating more informative representations of the variables. Specifically, we apply a directed hypergraph convolution (Ma, Zhao, and Yang 2024) to each directed hypergraph, i.e. P-DHG and N-DHG. Each hypergraph is associated with a node feature matrix $F \in \mathbb{R}^{n \times (K \times l_s)}$, where each row i represents the node features of variable i aggregated across all segments. For each variable i, node feature f_i in F is constructed as the concatenation of raw segment values from all segments, $f_i = [s_1^i \parallel s_2^i \parallel \ldots \parallel s_K^i]$. This allows f_i to comprehensively represent variable i's behavior across the entire window W_t.

The directed hypergraph convolution produces a node representation matrix $Z \in \mathbb{R}^{n \times d}$ where each row i is the node representation of variable i as follows:

$$Z = \Delta F \Theta \tag{9}$$

where Δ is the Laplacian matrix derived from the directed hypergraph and is defined by:

$$\Delta = \mathrm{I} - \frac{\Pi^{1/2} \mathrm{Q} \Pi^{-1/2} + \Pi^{1/2} \mathrm{Q}^{\mathrm{T}} \Pi^{-1/2}}{2} \tag{10}$$

Here, $\Pi \in \mathbb{R}^{\mathrm{n} \times \mathrm{n}}$ is a diagonal matrix with elements π, representing the stationary distribution of the random walk on the directed hypergraph (Ma, Zhao, and Yang 2024). $\Theta \in \mathbb{R}^{(K \times l_s) \times d}$ is a trainable weight parameter for directed hypergraph convolution and I denotes the identity matrix. The transition probability matrix $Q \in \mathbb{R}^{n \times n}$ for the random walk is calculated as follows.

$$Q = (D_v^{tail})^{-1} H^{tail} \mathcal{W}_{arc} (D_e^{head})^{-1} (H^{head})^T \tag{11}$$

Here, D_v^{tail} is a diagonal matrix whose element $d_i^{tail} = \sum_{j=1}^{m} H^{tail}(i,j)$ represents out-degree of node v_i. D_e^{head} denotes a diagonal matrix where each element $d_j^{head} = \sum_{i=1}^{n} H^{head}(i,j)$ represents in-degree of hyperarc e_j. $\mathcal{W}_{arc}$ is a diagonal matrix where each element has a positive value calculated in Section 4.2.1 or Section 4.2.2.

Reconstruction

After applying directed hypergraph convolution to both P-DHG and N-DHG, we obtain node representations Z_P for P-DHG and Z_N for N-DHG, respectively. These representations capture the variable-group relationships between variables and their correlated variables. To simultaneously capture both variable-group relationships, we concatenate Z_P and Z_N to produce unified node representations $U = [Z_P \parallel Z_N] \in \mathbb{R}^{n \times 2d}$.

Given the unified representations U obtained from the directed hypergraph convolution component, we then use U as inputs of a fully connected layer to generate a reconstructed window $\hat{W}_t = \{\hat{x}_{t-w+1}, \ldots, \hat{x}_t\}$, where $\hat{x}_t$ is a reconstructed data point of x_t. $\hat{W}_t$ is calculated as follows:

$$\hat{W}_t = \sigma\left(U \mathcal{W}_{fc}\right) \tag{12}$$

where $\mathcal{W}_{fc} \in \mathbb{R}^{2d \times w}$ is a two-dimensional weight matrix in the fully connected layer and $\sigma(\cdot)$ is a sigmoid activation function.

Training and Testing

Training. Consider a window W_t at each time point t in the training time series $\mathcal{X}_{train}$. DHG-AD attempts to minimize the mean squared error (MSE) between the window W_t and its reconstruction $\hat{W}_t$. The loss function defined by $\mathcal{L}_{\mathrm{MSE}}$ is computed as follows:

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{w} \sum_{u=t-w+1}^{t} \parallel x_u - \hat{x}_u \parallel_2^2 \tag{13}$$

where x_u and $\hat{x}_u$ are data points in W_t and $\hat{W}_t$, respectively.

Testing. Consider a window W_t from testing time series $\mathcal{X}_{test}$. Testing is performed at each time point t by a trained DHG-AD. The anomaly score α_t is designed to capture the average deviations between the data points of W_t and $\hat{W}_t$ as follows.

$$\alpha_t = \frac{1}{l_a} \sum_{u=t-l_a+1}^{t} \parallel x_u - \hat{x}_u \parallel_2^2 \tag{14}$$

Here, l_a refers to the length of the average deviations that allow a more robust detection of anomalies that exhibit over short-term patterns rather than a single anomaly point. We report that an anomaly occurs at time point t if α_t exceeds an anomaly threshold δ. The overall training and testing procedures for DHG-AD are summarized in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 Training Procedure of DHG-AD

Input: Training time series $\mathcal{X}_{train}$, input window size w, the number of segments K and segment length l_s, correlation thresholds for P-DHG and N-DHG, τ_P and τ_N.
Output: The trained model parameters $\mathcal{W}_{model}$

Randomly initialize model parameter $\mathcal{W}_{model}$ that includes all learnable parameters in DHG-AD.
for each epoch **do**
  for each time point t in $\mathcal{X}_{train}$ **do**
    $W_t = \{x_{t-w+1}, ..., x_t\}$
    Divide W_t into multiple overlapping segments $\{S_1, ..., S_K\}$, each of which has a length l_s.
    Construct incidence matrices of hyperarcs H^{tail} and H^{head} for P-DHG, using Eq.(2) – Eq.(6) with τ_P.
    Construct incidence matrices of hyperarcs H^{tail} and H^{head} for N-DHG, using Eq.(7) – Eq.(8) with τ_N.
    Construct feature matrix F for both P-DHG and N-DHG
    Calculate the node representations Z_P and Z_N using Eq.(9) – Eq.(11).
    Calculate the unified node representations $U = [Z_P \parallel Z_N]$.
    Calculate a reconstructed window $\hat{W}_t$ using Eq.(12).
    Calculate the loss function $\mathcal{L}_{\text{MSE}}$ using Eq. (13)
    Update the model parameter $\mathcal{W}_{model}$ to minimize $\mathcal{L}_{\text{MSE}}$
  end for
end for
return The trained model parameter $\mathcal{W}_{model}$

Algorithm 2 Training Procedure of DHG-AD

Input: Testing time series $\mathcal{X}_{test}$, trained model parameter $\mathcal{W}_{model}$, model hyperparameters w, K, l_s, τ_P, and τ_N, the length of average deviations l_a.
Output: A sequence of predicted labels $\mathcal{Y}_{test}$

for each time point t in $\mathcal{X}_{test}$ **do**
  $W_t = \{x_{t-w+1}, \ldots, x_t\}$
  Divide W_t into multiple overlapping segments $\{S_1, ..., S_K\}$ with K and l_s.
  Calculate a reconstructed window $\hat{W}_t$ using $\mathcal{W}_{model}$ with hyperparameters τ_P and τ_N.
  Calculate anomaly score α_t using Eq. (14) with l_a.
  if $\alpha_t > \delta$ **then**
    $y_t \leftarrow 1$ // "An anomaly occurs at time point t"
  else
    $y_t \leftarrow 0$ // "An anomaly does not occur at time point t"
  end if
end for
return The predicted labels $\mathcal{Y}_{test}$

Experiments

To demonstrate the effectiveness of our DHG-AD, we compare it with existing state-of-the-art methods for multivariate time series anomaly detection, including LSTM-VAE (Park, Hoshi, and Kemp 2018), GMM-GRU-VAE (Guo et al. 2018), OmniAnomaly (Su et al. 2019), AnomalyTransformer (Xu et al. 2022), MSCRED (Zhang et al. 2019), MTAD-GAT (Zhao et al. 2020), GDN (Deng and Hooi 2021), and STGAT-MAD (Zhan et al. 2022).

Datasets

We evaluate our method using two real-world datasets: **Exathlon** (Jacob et al. 2021) and **SMD** (Server Machine Dataset) (Su et al. 2019).

Exathlon: Exathlon dataset consists of eight multivariate time series, each of which contains data traces from stream processing applications on an Apache Spark cluster. For each application, intentional disturbances were made in the cluster and data traces about anomalous states were collected. Undisturbed data traces form a training time series $\mathcal{X}_{train}$, while series containing disturbed traces are used as a testing time series $\mathcal{X}_{test}$.

Table 2. Statistics of two datasets.

Dataset	Exathlon	SMD
# of multivariate time series	8	28
# of total data points for training	1,058,072	566,713
# of total data points for validation	264,537	141,692
# of total data points for testing	792,193	708,420
# of variables	19	38
# of anomalies in testing	100,530	29,444
Ratio of anomalies (%)	12.69	4.16

SMD: SMD dataset consists of 28 multivariate time series, each of which contains data from different machines of a large Internet company over five weeks. For each time series $\mathcal{X}$, the first half has data points from normal states, while the other half has a certain number of data points from anomalous states. Thus, for each multivariate time series $\mathcal{X}$, we use the first half of $\mathcal{X}$ as $\mathcal{X}_{train}$ and use the rest half of $\mathcal{X}$ as $\mathcal{X}_{test}$.

For both datasets, we train and test our method for each multivariate time series independently and measure the average scores of evaluation metrics. During the training step, 80% of $\mathcal{X}_{train}$ are used purely for training, and the rest 20% of $\mathcal{X}_{train}$, which we call $\mathcal{X}_{valid}$, are used for validation. The statistics for these datasets are given in Table 2.

Evaluation Metrics

To comprehensively evaluate the performance of our DHG-AD and other state-of-the-art methods, we employ several metrics described below.

Area Under Curve (AUC) of F1 scores with PA%K (Kim et al. 2022). Evaluation metrics such as precision, recall, and F1 score assess anomalies at each time point separately. However, in real-world applications, anomalies often occur over a range of consecutive time points, known as an anomaly range. According to Xu et al. 2018), it is acceptable for an anomaly detection method to trigger an alert at any time point within this range. Based on this notion (Xu et al. 2018), proposes using point adjustment (PA) before calculating the metrics. PA considers an anomaly detected if any point within the ground-truth anomaly range is identified. Although PA has been widely adopted in recent anomaly detection methods (Deng and Hooi 2021; Su et al. 2019; Xu et al. 2022; Zhan et al. 2022; Zhao et al. 2020), it has been shown to potentially overestimate performance (Kim et al. 2022).

To mitigate this overestimation, PA%K is proposed in (Kim et al. 2022) where we consider that an anomaly is correctly predicted if the proportion of detected anomalies in the given range exceeds a threshold K%. We measure F1 scores with varying K values, denoted as $\mathrm{F1}_{\mathrm{PA\%K}}$, ranged from 0 to 100, and calculate the area under curve (AUC) of the graph defined by these K values, denoted as $\mathrm{AUC\text{-}F1}_{\mathrm{PA\%K}}$. The K values that we used are 0, 1, 3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100. The $\mathrm{F1}_{\mathrm{PA\%K}}$ score is significantly affected by the anomaly threshold δ. As in previous anomaly detection methods (Kim et al. 2022; Su et al. 2019; Zhan et al. 2022), we use the highest $\mathrm{F1}_{\mathrm{PA\%K}}$ score obtained from all possible values of δ.

Area Under the Receiver Operating Characteristics curve and Precision-Recall curve. To further mitigate the dependency on the anomaly threshold δ, we also measure area under the receiver operating characteristics curve (AUROC) and area under the precision-recall curve (AUPRC) scores. These metrics provide a comprehensive assessment of the overall performance of anomaly detection methods considering all possible values of δ.

Implementation Details

We implement DHG-AD using a Pytorch Framework version 1.13.1 with CUDA 11.7 and Pytorch Geometric Library (Fey and Lenssen 2019) version 2.3.1, accelerated by a server with Nvidia RTX 3090 GPU. We optimize our model with AdamW (Loshchilov and Hutter 2019) optimizer. The detailed hyperparameter values in our DHG-AD are shown in Table 3.

Table 3. Hyperparameter values used in DHG-AD.

Datasets	Exathlon	SMD
Size of a window, w	100	100
The number of segments, K	3	3
The length of a segment, l_s	60	60
The size of a stride, η	20	20
Positive correlation threshold, τ_P	0.5	0.5
Negative correlation threshold, τ_N	−0.5	−0.5
Dimension of node representations, d	32	64
Length of the average deviation, l_a	10	10

Results of Performance Comparison

We evaluate the performances of our DHG-AD and other state-of-the-art methods using two real-world datasets with three evaluation metrics. In Table 4, we highlight the highest scores in boldface and the second-highest scores with underlines. As presented in Table 4, the performance evaluation results demonstrate that our proposed DHG-AD achieves superior performance compared with the existing methods in various metrics on the Exathlon and SMD datasets. Specifically, in the Exathlon dataset, AUC-$F1_{PA\%K}$ score of DHG-AD is 0.7761, while the next highest score is 0.6896 achieved by MTAD-GAT. The AUPRC and AUROC scores of DHG-AD are 0.6827 and 0.8538, respectively, both of which are the highest among all methods. On SMD, DHG-AD attains the highest scores, though the performance improvement of DHG-AD is relatively modest (AUC-$F1_{PA\%K}$: 0.5685 vs 0.5594; AUPRC: 0.4103 vs. 0.3843; AUROC: 0.8127 vs. 0.8077). This limited margin can be attributed to several properties of the SMD dataset. First, SMD consists of 28 independent machines – each trained and tested separately – making it is difficult for model to effectively detect anomalies on all machines. Second, the average length of anomalies in SMD is merely 90 time points, far shorter than those in the other datasets (hundreds to thousands of time points), reducing the effectiveness of temporal segmentation in our model. Finally, SMD's normal data exhibits high stochasticity that can be shown as anomalies, so subtle deviations are easily masked as anomalies. Under these dataset constraints, even our DHG-AD yields only incremental improvements on SMD. Overall, these results demonstrate the effectiveness of DHG-AD in accurately detecting anomalies in multivariate time series.

Figures 3 and 4 show the $F1_{PA\%K}$ scores of all the methods with varying K values on the Exathlon and SMD datasets, respectively. As shown in Figures 3 and 4, DHG-AD provides the best performance (i.e. the highest $F1_{PA\%K}$ value) in almost all K values ($K \geq 10$ in the Exathlon dataset and $K \geq 30$ in the SMD dataset). Unlike other methods whose $F1_{PA\%K}$ scores decrease dramatically when the K value increases, DHG-AD provides relatively stable $F1_{PA\%K}$ scores even if the K value is close to 100. These observations show that DHG-AD can correctly detect more anomalies that occur in the ground truth anomaly ranges than the other existing methods.

In summary, DHG-AD consistently obtains the best scores across all evaluation metrics on both real-world datasets investigated in this study. These results demonstrate the effectiveness of directed hypergraph neural networks in capturing the variable-group relationships present in multivariate time series. DHG-AD can offer an effective solution for anomaly detection in real-world applications.

Table 4. Performance comparison on two datasets.

	Datasets					
	Exathlon			SMD		
	AUC-$F1_{PA\%K}$	AUPRC	AUROC	AUC-$F1_{PA\%K}$	AUPRC	AUROC
LSTM-VAE	0.5534	0.3054	0.7061	0.4834	0.2902	0.7880
GMM-GRU-VAE	0.5498	0.3647	0.6998	0.5180	0.3425	0.7944
OmniAnomaly	0.4985	0.2975	0.6699	0.5488	0.3833	0.8077
AnomalyTransformer	0.4547	0.2644	0.4827	0.3809	0.1733	0.5705
MSCRED	0.5790	0.6406	0.7250	0.4501	0.3824	0.6979
MTAD-GAT	0.6896	0.5626	0.8014	0.5592	0.3728	0.7923
GDN	0.6132	0.4924	0.7411	0.4705	0.2849	0.7917
STGAT-MAD	0.6648	0.5317	0.7984	0.5594	0.3843	0.7940
DHG-AD	**0.7761**	**0.6827**	**0.8538**	**0.5685**	**0.4103**	**0.8127**

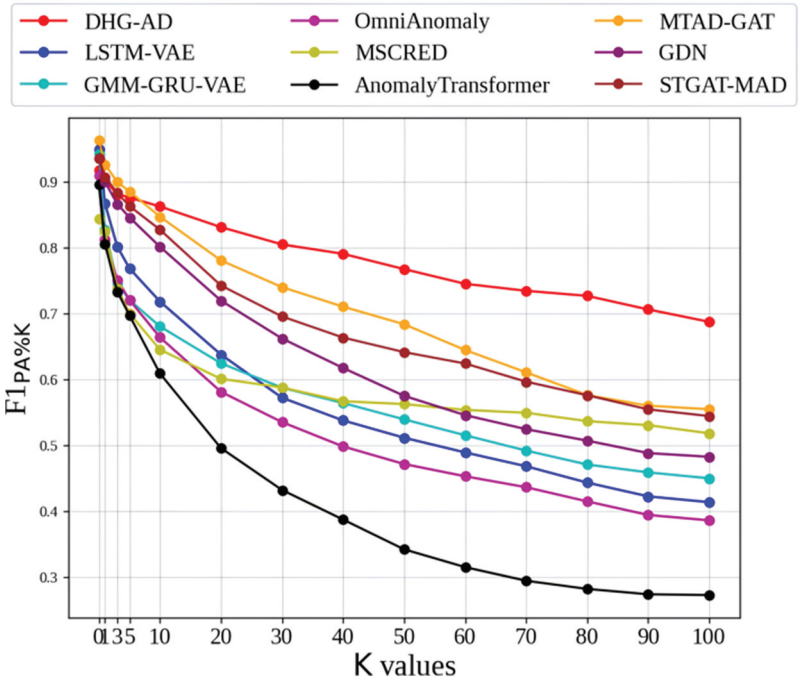

Figure 3. $F1_{PA\%K}$ scores with varying thirteen K values on exathlon dataset.

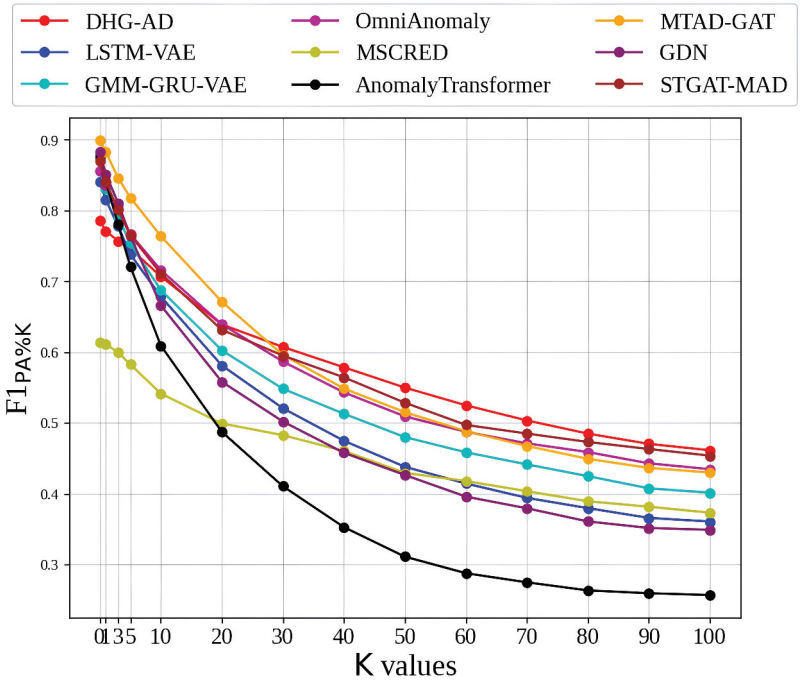

Figure 4. $F1_{PA\%K}$ scores with varying thirteen K values on SMD dataset.

Ablation Study

To further analyze the effectiveness of using two types of directed hypergraphs, P-DHG and N-DHG, we conduct an ablation study to assess the contribution of each hypergraph. Specifically, we evaluate two variations of DHG-AD: **DHG-AD-positive** which only constructs P-DHG and applies a directed hypergraph neural network for P-DHG, and **DHG-AD-negative** which only constructs N-DHG and uses a directed hypergraph neural network for N-DHG. By comparing these variations with the full DHG-AD

Table 5. Ablation study on two datasets.

	Exathlon			SMD		
	AUC-F1$_{PA\%K}$	AUPRC	AUROC	AUC-F1$_{PA\%K}$	AUPRC	AUROC
DHG-AD	**0.7761**	**0.6827**	**0.8538**	**0.5685**	**0.4103**	**0.8127**
DHG-AD-positive	0.7726	0.6677	0.8417	0.5656	0.4024	0.7825
DHG-AD-negative	0.6614	0.5496	0.7387	0.5499	0.3956	0.7898

model, we aim to demonstrate the advantages of incorporating both types of variable-group relationships for anomaly detection in multivariate time series.

The results of the ablation study, presented in Table 5, highlight the performance of each model on the Exathlon and SMD datasets across three evaluation metrics: AUC-F1$_{PA\%K}$, AUPRC, and AUROC. We can observe that DHG-AD consistently outperforms both variations across two datasets. Although DHG-AD-positive achieves competitive results, it is still slightly worse compared to the full model, indicating that positive correlations alone are insufficient to capture variable-group relationships in multivariate time series. On the other hand, DHG-AD-negative performs worse overall across both datasets. These observations show the importance of modeling both types of variable-group relationships to effectively capture the dynamics of multivariate time series and enhance the anomaly detection performance.

Parameter Sensitivity

To evaluate the impact of hyperparameters on the performance of DHG-AD, we conduct a parameter sensitivity analysis. We vary the following parameters: positive correlation threshold τ_P and negative correlation threshold τ_N, which are the key hyperparameters to construct P-DHG and N-DHG, respectively. The other parameters have their values shown in Table 3. All experiments are conducted using the Exathlon dataset, and the performance of DHG-AD is assessed using three evaluation metrics: AUC-F1$_{PA\%K}$, AUPRC and AUROC. The results of these experiments are presented in Figure 5.

For the positive correlation threshold τ_P, as shown in Figure 5(a), we observe that higher-values of τ_P show better performance in almost metrics. The best results are shown at the value $\tau_P = 0.9$. In contrast, when analyzing the effect of the negative correlation threshold τ_N as presented in Figure 5(b), the model performs well in $\tau_N = -0.5$, but extreme values such as $\tau_N = -0.9$ decrease the performance. To better understand

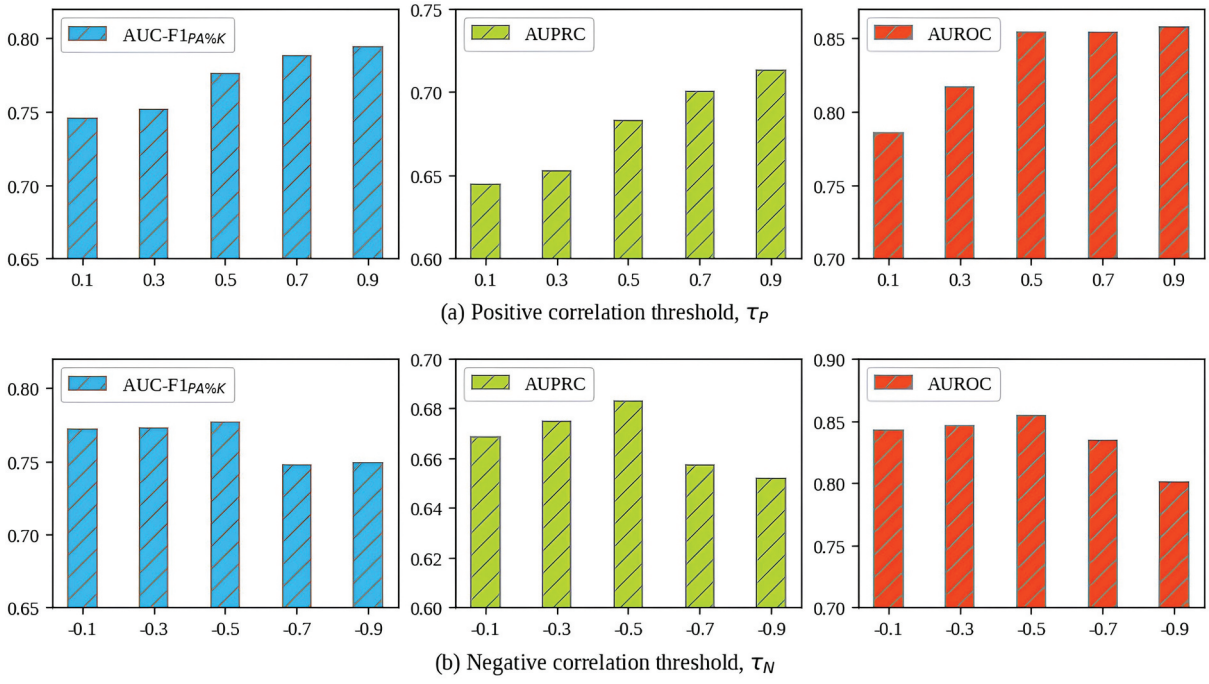

Figure 5. Parameter sensitivity analysis.

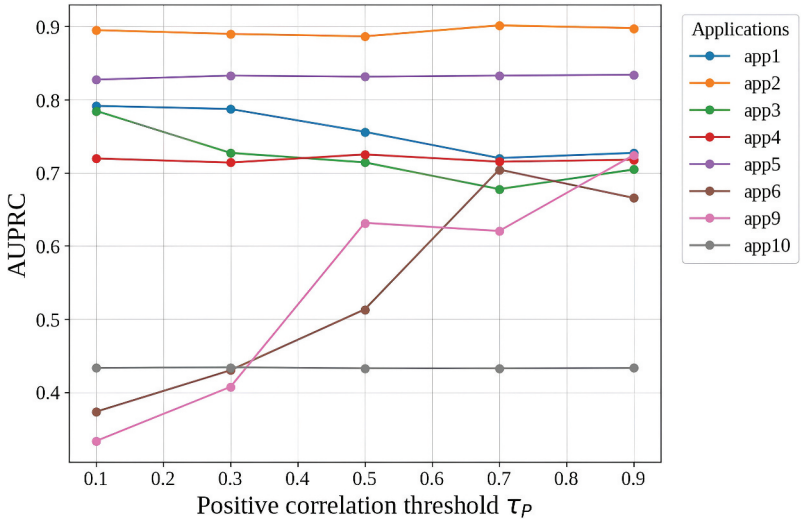

Figure 6. AUPRC scores of each multivariate time series with varying the values of τ_P in the exathlon dataset.

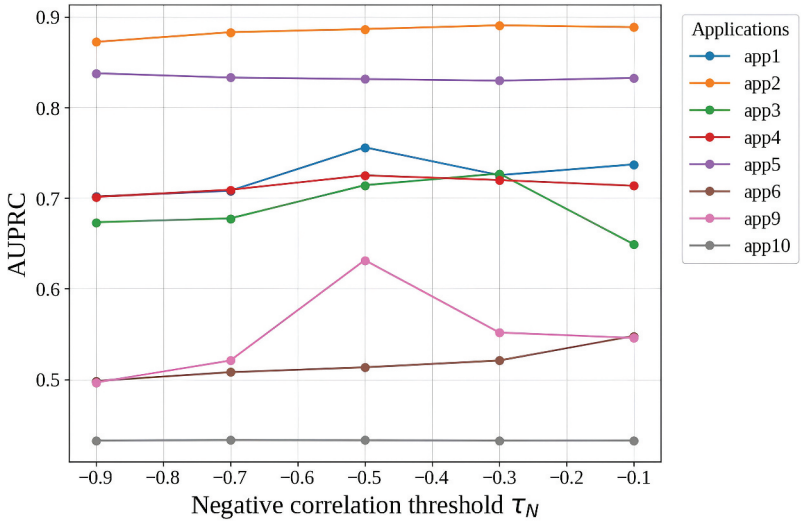

Figure 7. AUPRC scores of each multivariate time series with varying the values of τ_N in the exathlon dataset.

these results, as shown in Figures 6 and 7, we also plot the AUPRC scores of each multivariate time series called "Application" in the Exathlon dataset. We can see that some time series such as "app6" and "app9" are sensitive to the values of τ_P. The other time series have consistent scores with varying the values of τ_P. In other words, almost time series show consistent scores with varying the values of τ_N. Therefore, in general, these results suggest that values in the range [0.7, 0.9] for τ_P, and [−0.7, −0.5] for τ_N are preferable. This is because a high correlation threshold (or a low correlation threshold) helps the model focus on significant correlations, filtering out weaker connections that may introduce noise and impact detection accuracy.

Discussion and Future Work

In this work, we have demonstrated that constructing two directed hypergraphs – P-DHG for positively correlated variable-group relationships and N-DHG for negatively correlated variable-group relationships – yields robust anomaly detection performance. Recent advances in hypergraph neural networks suggest several promising directions to further strengthen DHG-AD.

First, hyperedge-level augmentation strategies, as proposed in (Wei et al. 2022), could be applied to P-DHG and N-DHG during training. By randomly dropping, perturbing, or synthesizing hyperarcs (i.e.,

variable-group relationships) in each view, the model would be encouraged to learn representations that are invariant to minor structural noise, improving generalization and robustness. Importantly, these augmentations can be implemented with minimal changes – simply by stochastically modifying the incidence matrices before each directed hypergraph convolution.

Second, cross-view consistency regularization, inspired by (Xia et al. 2022), offers a way to align the representations produced by two directed hypergraph convolutions for P-DHG and N-DHG. Adding a lightweight contrastive loss that pulls together the representation of each variable in the two hypergraph views would encourage the model to integrate positive and negative variable-group relationships more coherently. This multi-view alignment can sharpen the unified node representations and make deviations more apparent.

These extensions require no fundamental architectural overhaul: hyperarc augmentation simply augments the existing directed hypergraphs, and a small contrastive term can be added on top of the current reconstruction loss. We plan to explore both strategies in future work to enhance DHG-AD's ability to capture complex interactions in multivariate time series.

Conclusion

In this paper, we propose a novel method for multivariate time series anomaly detection called DHG-AD that utilizes directed hypergraph neural networks to capture variable-group relationships. Unlike graph-based methods that focus on variable-variable relationships, DHG-AD uses two directed hypergraphs to model variable-group relationships that exist between variables and their positively and negatively correlated variables, respectively. By aggregating information through directed hypergraph convolutions, DHG-AD effectively learns comprehensive node representations that account for both positive and negative correlations simultaneously. To demonstrate the effectiveness of our DHG-AD, we conduct experiments using multiple evaluation metrics and demonstrates enhanced empirical performance relative to the compared methods on both real-world datasets. Future work can incorporate hyperedge-level augmentations and cross-view contrastive regularization – building on recent hypergraph contrastive loss advances – to further enhance DHG-AD's robustness and its ability to capture complex interactions in multivariate time series.

Acknowledgments

All authors have read and approved the final version of the manuscript.

Author contributions

CRediT: **Tae Wook Ha:** Conceptualization, Formal analysis, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing; **Myoung Ho Kim:** Funding acquisition, Project administration, Resources, Supervision, Writing – review & editing.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the IITP□ITRC grant by the Korea government MSIT (IITP□2025□RS□2020□II201795), and Samsung Electronics (IO201209□07901□01).

ORCID

Tae Wook Ha http://orcid.org/0009-0000-3278-1962
Myoung Ho Kim http://orcid.org/0000-0002-4553-6904

Data Availability Statement

The data that support the findings of this study are openly available. The Exathlon dataset is available at https://github.com/exathlonbenchmark/exathlon., reference number (Jacob et al. 2021), and the SMD dataset at https://github.com/NetManAIOps/OmniAnomaly., reference number (Su et al. 2019). Data related to the figures and tables in this study are available on Figshare at https://doi.org/10.6084/m9.figshare.27643182.

References

Aggarwal, C. C. 2013. *Outlier analysis*. New York, NY: Springer.

Breunig, M. M., H. Kriegel, R. T. Ng, and J. Sander. 2000. LOF: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ed. W. Chen, J. F. Naughton, and P. A. Bernstein, 93–104. Dallas, TX, USA: ACM.

Deng, A., and B. Hooi. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-ThirdConference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021* , 4027–35, AAAI Press.

Fey, M., and J. E. Lenssen. 2019. Fast graph representation learning with pytorch geometric. In *Proceedings of the ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds*, New Orleans, LA.

Gallo, G., G. Longo, S. Pallottino, and S. Nguyen. 1993. Directed hypergraphs and applications. *Discrete Applied Mathematics* 42 (2–3):177–201. doi: 10.1016/0166-218X(93)90045-P.

Goernitz, N., M. Kloft, K. Rieck, and U. Brefeld. 2013. Toward supervised anomaly detection. *The Journal of Artificial Intelligence Research* 46:235–62. doi: 10.1613/jair.3623.

Guo, Y., W. Liao, Q. Wang, L. Yu, T. Ji, and P. Li. 2018. Multidimensional time series anomaly detection: A gru-based gaussian mixture variational autoencoder approach. In *Proceedings of The 10th Asian Conference on Machine Learning, ACML 2018, November 14-16, 2018, volume 95 of Proceedings of Machine Learning Research*, ed. J. Zhu and I. Takeuchi, 97–112. Beijing, China: PMLR.

Hautamäki, V., I. Kärkkäinen, and P. Fränti. 2004. Outlier detection using k-nearest neighbour graph. In *17th International Conference on Pattern Recognition, ICPR 2004*, 430–33. Cambridge, UK: IEEE Computer Society.

He, Z., X. Xu, and S. Deng. 2003. Discovering cluster-based local outliers. *Pattern Recognition Letters* 24 (9–10):1641–-50. doi: 10.1016/S0167-8655(03)00003-5.

Jacob, V., F. Song, A. Stiegler, B. Rad, Y. Diao, and N. Tatbul. 2021. Exathlon: A benchmark for explainable anomaly detection over time series. *Proceedings of the VLDB Endowment* 14 (11):2613–26. doi: 10.14778/3476249.3476307.

Kim, S., K. Choi, H. Choi, B. Lee, and S. Yoon. 2022. Towards a rigorous evaluation of time-series anomaly detection. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022* , AAAI Press.

Loshchilov, I., and F. Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019*, New Orleans, LA, USA, OpenReview.net.

Ma, Z., W. Zhao, and Z. Yang. 2024. Directed hypergraph representation learning for link prediction. In *International Conference on Artificial Intelligence and Statistics*, *2-4 May* 2024, Palau de Congressos, volume 238 of *Proceedings of Machine Learning Research*, ed. S. Dasgupta, S. Mandt, and Y. Li, 3268–76. Valencia, Spain: PMLR.

Manevitz, L. M., and M. Yousef. 2001. One-class svms for document classification. *Journal of Machine Learning Research: JMLR* 2:139–54.

Melnyk, I., B. L. Matthews, H. Valizadegan, A. Banerjee, and N. C. Oza. 2016. Vector autoregressive model-based anomaly detection in aviation systems. *Journal of Aerospace Information Systems* 13 (4):161–73. doi: 10.2514/1.I010394.

Park, D., Y. Hoshi, and C. C. Kemp. 2018. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters* 3 (3):1544–51. doi: 10.1109/LRA.2018.2801475.

Park, D., H. Kim, Y. Hoshi, Z. Erickson, A. Kapusta, and C. C. Kemp. 2017. A multimodal execution monitor with anomaly classification for robot-assisted feeding. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017*, 5406–13. Vancouver, BC, Canada: IEEE.

Rodriguez, A., D. A. Bourne, M. T. Mason, G. F. Rossano, and J. Wang. 2010. Failure detection in assembly: Force signature analysis. In *IEEE Conference on Automation Science and Engineering, CASE 2010*, 210–15. Toronto, ON, Canada: IEEE.

Ruff, L., N. Görnitz, L. Deecke, S. A. Siddiqui, R. A. Vandermeulen, A. Binder, E. Müller, and M. Kloft. 2018. Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research*, ed. J. G. Dy and A. Krause, 4390–99. Stockholmsmässan, Stockholm, Sweden: PMLR.

Ruff, L., R. A. Vandermeulen, B. J. Franks, K. Müller, and M. Kloft. 2021. Rethinking assumptions in deep anomaly detection. In *Proceedings of the ICML 2021 Workshop on Uncertainty & Robustness in Deep Learning*.

Ruff, L., R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K. Müller, and M. Kloft. 2020. Deep semi-supervised anomaly detection. In *8th International Conference on Learning Representations, ICLR 2020*, Addis Ababa, Ethiopia, OpenReview.net.

Sanayha, M., and P. Vateekul. 2017. Fault detection for circulating water pump using time series forecasting and outlier detection. In *9th International Conference on Knowledge and Smart Technology, KST 2017*, 193–98. Chonburi, Thailand: IEEE.

Sohn, K., C. Li, J. Yoon, M. Jin, and T. Pfister. 2021. Learning and evaluating representations for deep one-class classification. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event*, Austria, OpenReview.net.

Su, Y., Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei. 2019. 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*, ed. A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, 2828–37. Anchorage, AK, USA: ACM.

Tax, D. M. J., and R. P. W. Duin. 2004. Support vector data description. *Machine Learning* 54 (1):45–66. doi: 10.1023/B:MACH.0000008084.60811.49.

Wei, T., Y. You, T. Chen, Y. Shen, J. He, and Z. Wang. 2022. Augmentations in hypergraph contrastive learning: Fabricated and generative. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022*, ed. S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. New Orleans, LA, USA: NeurIPS.

Xia, L., C. Huang, Y. Xu, J. Zhao, D. Yin, and J. X. Huang. 2022. Hypergraph contrastive collaborative filtering. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ed. E. Amigó, P. Castells, J. Gonzalo, B. Carterette, J. S. Culpepper, and G. Kazai, 70–79. Madrid, Spain: ACM.

Xu, H., W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng. 2018. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018*, ed. P. Champin, F. Gandon, M. Lalmas, and P. G. Ipeirotis, 187–96. Lyon, France: ACM.

Xu, J., H. Wu, J. Wang, and M. Long. 2022. Anomaly transformer: Time series anomaly detection with association discrepancy. In *The Tenth International Conference on Learning Representations, ICLR 2022,*OpenReview.net.

Yu, Q., L. Jibin, and L. Jiang. 2016. An improved arima-based traffic anomaly detection algorithm for wireless sensor networks. *International Journal of Distributed Sensor Networks* 12 (1):9653230:1–9653230. doi: 10.1155/2016/9653230.

Zhan, J., S. Wang, X. Ma, C. Wu, C. Yang, D. Zeng, and S. Wang. 2022. Stgat-mad: Spatial-temporal graph attention network for multivariate time series anomaly detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore*, Sands Expo & Convention Centre, 3568–72. IEEE.

Zhang, C., D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla. 2019. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, 1409–16. Honolulu, HI, USA: AAAI Press.

Zhao, H., Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang. 2020. Multivariate time-series anomaly detection via graph attention network. In *20th IEEE International Conference on Data Mining, ICDM 2020*, ed. C. Plant, H. Wang, A. Cuzzocrea, C. Zaniolo, and X. Wu, 841–50. Sorrento, Italy: IEEE.

Zong, B., Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. *6th International Conference on Learning Representations, ICLR 2018, April 30 - May 3, 2018, Conference Track Proceedings*, Vancouver, BC, Canada, OpenReview.net.