DGNN: Dynamic Graph Neural Networks for Anomaly Detection in Multivariate Time Series

Bowen Chen[1], Hancheng Lu[1],Yuang Chen[1], Haoyue Yuan[1], and Minghui Wang[2]
[1] University of Science and Technology of China, Hefei, China
[2] AI Institute of H3C Technologies Co., Ltd.,
{chenbowen, yuangchen21, yhyue}@mail.ustc.edu.cn, hclu@ustc.edu.cn, mhwang@h3c.com

Abstract—**In recent years, there has been significant progress in the importance of anomaly detection for multivariate time series in industrial applications. However, there are still limitations. Although deep learning methods have improved anomaly detection in high-dimensional multivariate time series, they are computationally expensive and do not explicitly learn the relational structure between sequences. In this paper, we propose an unsupervised anomaly detection algorithm called Dynamic Graph Neural Networks (DGNN). Firstly, we propose a data-driven method of generating "subgraphs" to interpret interior correlations between sequences, instead of using the traditional method of a fully connected graph. Secondly, we introduce a novel Graph Attention Networks based on correlation to fuse neighbor sequence features. Experimental results on five public datasets demonstrate that our method consistently achieves state-of-the-art performance compared to other baseline methods, while reducing the edges of the graph by nearly 70%.**

Index Terms—**time series, anomaly detection, graph attention network, correlation coefficient**

I. Introduction

With the advent of the Internet of Things (IoT) era, the recording of time-series data [1]–[3] has experienced a significant increase. This data is collected from various domains, including operations and maintenance, economics, and transportation, and encompasses metrics such as CPU usage, online shopping request numbers, resident purchasing and expenditure indexes, traffic flow, and average vehicle speed. Many time series exhibit periodic behavior, such as human heartbeat, rush-hour traffic, and tides linked to lunar and solar cycles. Anomaly detection in time series data has emerged as a critical topic in data mining, with diverse applications across industries [4]–[6]. Accurate and timely anomaly detection enables industrial systems to continuously monitor key indicators in time series data and issue early warnings for potential events. While detecting anomalies in a single time series is relatively straightforward, with the advancement and expansion of systems, an increasing number of time series data for key indicators (KPIs) are being recorded. As the dimensionality of time series data grows, manual monitoring by humans becomes increasingly challenging. Therefore, there is a need for automatic anomaly detection methods that can efficiently identify anomalies in high-dimensional data and provide explanations to human operators.

Recent techniques based on Graph Neural Networks (GNNs) have improved anomaly detection in high-dimensional datasets. For example, MTAD-GAT (Zhang et al. 2020) [7] learns the relationship between feature dimension and time dimension of multivariate time series through Graph Attention Network (GAT) (Velickovic et al. 2017) [8], and finds potential anomalies in high-dimensional multivariate time series. GDN (Deng et al. 2021) [9] employs Embedding technology to learn the potential relationship of multivariate time series and represent the relationship of each sequence time using embedding, which is convenient for human operators to diagnose and locate anomalies.

However, the existing methods use GAT to directly learn the attention coefficient and aggregate the features of the neighboring time series, which is coarse and inefficient. Because the time series in real-world scenarios have various modes, such as stationary, irregularly fluctuating, or periodic. These existing GAT-based methods all use a fully connected graph, which results in a waste of computing resources for GAT graph learning and may lead to inaccurate aggregation results. There is a wealth of information in multivariate time series, such as the possibility of similar changing trends between series, as well as slow-changing trends within individual series. However, since GAT was not designed for multivariate time series, it cannot fully exploit the information contained within them. Existing anomaly detection methods utilize GAT to fuse multivariate time series features and can only roughly extract some of their internal features.

To overcome these challenges, we propose Dynamic Graph Neural Networks (DGNN), a novel framework for detecting anomalies in multivariate time series. DGNN utilizes dynamic subgraphs that are generated based on correlations, which replace static fully connected graphs, and Adaptive Graph Attention Network(AGAT), a correlation-based alternative to traditional GAT. Compared to previous methods, dynamic subgraphs more accurately depict the proximity between each time series and help to narrow down the potential locations of anomalies. Furthermore, when GNN is employed for feature fusion, fewer edges result in less interference from independent sequences and faster convergence rates of the GNN model. AGAT leverages correlations to calculate attention coefficients and is effective in extracting features from multiple time series compared to GAT.

In summary, our main contributions are as follows:

- We propose DGNN, a dynamic graph neural network approach that enables faster and more accurate learning of temporal relationships between time series than existing methods.

- We introduce Dynamic Subgraph Generation (DSG), a real-time approach for generating graphs between sequences based on input time series, dynamically adapting to changing data.
- We present the Adaptive Graph Attention Network (AGAT), a novel approach for effectively capturing features from multiple time series data by leveraging correlations to compute attention coefficients.
- We conduct extensive experiments on five public real-world datasets. Our results demonstrate the superior performance of DGNN over state-of-the-art baseline methods.

II. Related Work

PCA (Shyu et al. 2003) [10] and IForest (Liu et al. 2008) [11] are machine learning-based methods for anomaly detection. PCA detects anomalies by reducing dimensionality and projecting data back to the original space, and comparing the deviation between original data and reconstructed data. It is fast but may lose information. IForest is an ensemble-based fast anomaly detection algorithm that determines anomalies by constructing randomly generated isolation trees, with shorter path lengths indicating a higher likelihood of anomalies. However, IForest is not suitable for high-dimensional data as randomly selecting dimensions may reduce algorithm reliability.

To address the issue of high-dimensional data loss, deep learning methods have been introduced. Time series anomaly detection methods based on deep learning, such as LSTM (Malhotra et al. 2008) [12], DAGMM (Zong et al. 2018) [13], and USAD (Audibert et al. 2020) [14], are able to incorporate contextual information of time series for prediction. Transformer-based models like TranAD (Tuli et al. 2020) [15] and Anomaly Transformer (Xu et al. 2021) [16] can better learn the contextual information of time series, leading to significant performance improvement. However, they still do not fully utilize the relationships among multiple time series. For example, Anomaly Transformer proposed an anomaly detection model based on the differences between associations, but it only focuses on the contextual information of individual time series without considering the relationships among sequences.

In recent years, GNNs have achieved success in handling graph-structured data. GNNs consider the state of neighboring nodes and utilize the latent relationships among nodes. Graph Convolutional Networks (GCNs) (Kipf and Welling 2016) [17] model node features by aggregating representations of one-step neighbors. GATs extend the approach of GCN by introducing attention functions to compute weights for different neighbors during the aggregation process. GDN (Deng et al. 2017) is a multivariate time series anomaly detection method based on GAT, but it uses a fully connected graph as the input to the GAT network, treating all time series as having the same relationship, which does not align with the actual scenario. Using fully connected graph as input leads to difficulties in model training and slow convergence. Additionally, GAT is not specifically designed for time series, and its effectiveness in feature extraction for time series is limited.

III. Framework of DGNN

A. Problem Statement

We represent the network as a graph $G = (V, E)$, where V is a finite set of nodes with $|V| = N$, corresponding to the observation of N sensors, and E is a set of edges. The observed graph signal $X_G^{(t)} \in R^{N \times d}$ indicates the observation of graph information G at time step t, where each element represents the observation of d sensor features. In this paper, we utilize a forecasting-based model to detect anomalies by comparing the errors between predicted and actual values. The objective of forecasting is to learn a function f from the previous T observations, which can predict the next step from N correlated sensors.

$$[X_G^{t-T+1}, ..., X_G^{t}] \xrightarrow{f} [X_G^{t+1}] \tag{1}$$

Our training dataset comprises sensor data (i.e., multivariate time series) from N sensors across T_{train} time steps. The sensor data is denoted as $s_{train} = [s_{train}^{(1)}, ..., s_{train}^{(T_{train})}]$ and is utilized to train our approach. At each time step t, the sensor values $s_{train}^{(t)} \in \mathbb{R}^{\mathrm{N}}$ form an N-dimensional vector representing the values of our N sensors. Consistent with the standard unsupervised anomaly detection formulation, the training data is assumed to consist solely of normal data.

Our goal is to detect any irregularities in the testing data, which is collected from the same N sensors but covers a distinct set of T_{test} time intervals. The test data is denoted by $s_{test} = [s_{test}^{(1)}, ..., s_{test}^{(T_{test})}]$. The label is a collection of T_{test} binary values that indicate whether each test time interval contains an anomaly or not. Specifically, $a(t) \in \{0, 1\}$, where $a(t) = 1$ indicates that the time interval at t is anomalous.

B. Overview

Our DGNN (Dynamic Graph Neural Network) method aims to cluster sensors with comparable characteristics into subgraphs and subsequently detects and clarifies discrepancies from learned patterns. The method comprises three primary components:

1) **Dynamic Subgraph Generation**: Builds subgraphs consisting of sensors with similar characteristics.
2) **AGAT Features Learning**: Uses Adaptive GAT based on correlation coefficients to learn representations from neighboring nodes.
3) **Forecasting**: Predicts the next step values of each sensor by leveraging a fusion representation generated by AGAT.

Figure 1 provides an overview of our framework.

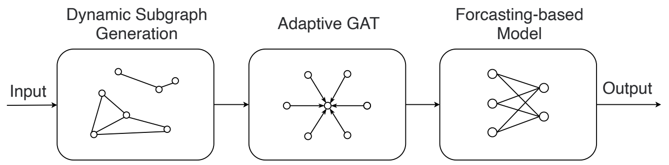

Fig. 1. Overview of our proposed framework.

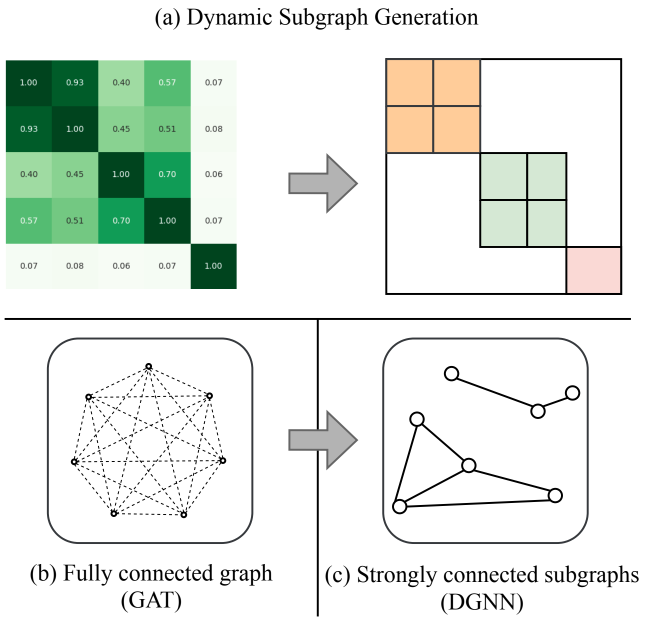

Fig. 2. Dynamic Subgraph Generation.

C. Data Preprocessing

To make our model more robust, we apply data normalization and segment the data into time-series windows for both training and testing. We normalize the time-series data using the following equation:

$$\widetilde{x} = \frac{x - \min(X)}{\max(X) - \min(X)} \tag{2}$$

Here, $max(X)$ and $min(X)$ represent the maximum and minimum values in the training dataset, respectively.

D. Dynamic Subgraph Generation

One of the primary goals of our framework is to establish subgraphs between sensors. To achieve this, we utilize an undirected graph, where the nodes represent sensors and the edges represent dependency relationships between them. An edge connecting two sensors indicates that they can model each other's behavior.

In our approach, we employ the correlation coefficient to establish subgraphs. Specifically, we connect two nodes with an edge if the correlation coefficient of their features exceeds a specified threshold. The correlation coefficient measures the degree of linear correlation between two sets of data variables, denoted as X and Y. It is the ratio of the product of their covariance and their standard deviations. Essentially, it is a normalized measure of covariance, with a range of values between -1 and 1. The correlation coefficient is defined as follows:

$$Corr(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var[X]Var[Y]}} \tag{3}$$

where $Cov(X,Y)$ represents the covariance of X and Y, and $Var[X]$ and $Var[Y]$ represent the variance of X and Y, respectively.

$W_{i,j}$ represents the relationship between the two nodes i and j, and $W_{i,j} \in \{0,1\}$, indicating that the two nodes have an edge ($W_{i,j} = 1$) or no edges ($W_{i,j} = 0$), respectively. It is defined as follows:

$$W_{i,j} = Corr(x_i, x_j) > threshold \tag{4}$$

Where $threshold$ is an experience value, the default is 0.65.

Graph neural networks require the input of the structure of a graph. The lower part of Figure 2 shows a comparison between two methods. As shown in the figure, there are time series generated by seven sensors. The left side represents a model based on GAT that takes a fully connected graph as input, while the right side represents DGNN that uses a graph constructed by DSG as input. The subgraph constructed by DSG has a closer relationship than the fully connected graph.

By DSG, we split a fully connected graph into multiple subgraphs with similar features. Next, we will define our adaptive graph attention network which utilizes those subgraphs.

Algorithm 1: Dynamic Subgraph Generation

```
Input: N time series from $\mathcal{V}(|\mathcal{V}| = N)$
$W$ Initialization, reset to zero matrix;
$E$ Initialization, reset to empty set $E$;
for $i = 1, 2, ..., n$ do
    for $j = i+1, i+2, ..., n$ do
        // CDC: Correlation Distance Calculation
          defined in Equation 3;
        $dist_{i,j} = CDC(V_i, V_j)$; (Equation. 3)
        if $dist_{i,j} > threshold$ then
            $W_{i,j} = 1$; $W_{j,i} = 1$;
        end
    end
end
for $i = 1, 2, ..., n$ do
    for $j = 1, 2, ..., n$ do
        if $W_{i,j} = 1$ then
            $E \leftarrow E \cup \{i, j\}$;
        end
    end
end
return Edge list $E$ of Subgraph $\mathcal{G}$.
```

E. AGAT Features Learning

We propose a graph attention-based feature extractor that captures the relationships between neighboring sensors by fusing a node's information with its neighbors based on subgraphs. Unlike existing graph attention mechanisms, our feature extractor uses the correlation coefficient.

Our AGAT takes a set of node features, denoted by $h = \overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_N}$, where $h_i \in \mathbb{R}^F$ and N is the number of nodes with F features in each node. The AGAT produces a new set of node features, denoted by $h' = \overrightarrow{h'_1}, \overrightarrow{h'_2}, ..., \overrightarrow{h'_N}$, where $h'_i \in \mathbb{R}^{F'}$ may have a potentially different cardinality than F.

To increase the expressive power required to transform input features into higher-level features, at least one learnable linear transformation is necessary. Initially, a shared linear transformation is applied to every node, parametrized by a weight matrix, $W \in \mathbb{R}^{F' \times F}$. Next, self-attention is performed on the nodes using the correlation coefficient, which computes attention coefficients, $e_{i,j}$, indicating the significance of node j's features to node i.

$$e_{i,j} = Corr(W\overrightarrow{h_i}, W\overrightarrow{h_j}) \tag{5}$$

To incorporate graph structure into the mechanism, masked attention is used, computing $e_{i,j}$ only for nodes $j \in \mathcal{N}_i$, where N_i represents the neighborhood of node i in the graph. In all experiments, the neighborhood will include only the first-order neighbors of node i (including i). To make coefficients comparable across different nodes, the softmax function is used to normalize them across all choices of j as follows:

$$a_{i,j} = softmax_j(e_{i,j}) = \frac{exp(e_{i,j})}{\sum_{k \in N_i} exp(e_{i,k})} \tag{6}$$

In our experiments, the attention mechanism is computed as the correlation coefficient and is then subjected to the LeakyReLU nonlinearity with a negative input slope of $\alpha = 0.2$. This allows us to express AGAT as follows:

$$a_{i,j} = \frac{exp(LeakyReLU(Corr(Wh_i, Wh_j)))}{\sum_{k \in N_i} exp(LeakyReLU(Corr(Wh_i, Wh_k)))} \tag{7}$$

Once the attention coefficients have been normalized, they are used to compute a linear combination of the features corresponding to them, which serves as the final output features for each node (with the possibility of applying a nonlinearity σ):

$$\overrightarrow{h}'_i = \sigma(\sum_{j \in N_i} \alpha_{i,j} W\overrightarrow{h_j}) \tag{8}$$

Figure 3 illustrates the attention mechanism based on correlation used in AGAT.

F. Forecasting-based model

The forecasting model predicts the value for the next step by utilizing recurrent neural networks (RNNs) to model temporal dependency. Specifically, we employ Gated Recurrent Units (GRUs) (Chung et al., 2014), a simple yet powerful variant

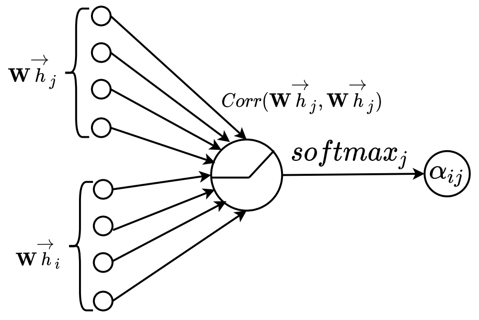
$\mathbf{W}\overrightarrow{h}_j$
$Corr(\mathbf{W}\overrightarrow{h}_j, \mathbf{W}\overrightarrow{h}_j)$
$softmax_j$
α_{ij}
$\mathbf{W}\overrightarrow{h}_i$

Fig. 3. Adaptive Graph Attention Network.

of RNNs. The predicted output of the forecasting model is denoted as $\hat{s}^{(t)}$. To minimize error, we use the Mean Squared Error (MSE) loss function between the predicted output $\hat{s}^{(t)}$ and the observed data $s^{(t)}$:

$$L_{MSE} = \frac{1}{T_{train} - w} \sum_{t=w+1}^{T_{train}} ||\hat{s}^{(t)} - s^{(t)}||_2^2 \tag{9}$$

Here, w represents the sliding window size of the input time series.

G. Anomaly Detection

To ensure a fair comparison, we employ the Peak Over Threshold (POT) method to automatically and dynamically select the threshold. This method is essentially a statistical approach that uses "extreme value theory" to fit the data distribution with a Generalized Pareto Distribution and determine appropriate value at risk for dynamically selecting threshold values. For each dimension i, the anomaly diagnosis label (y_i) and detection result (y) are defined as:

$$y_i = 1(s_i >= POT(s_i)) \tag{10}$$

To compute the overall anomalousness at time tick t, we aggregate over sensors using the max function. We use max as anomalies may affect only a small subset of sensors or even a single sensor:

$$y = \max_i(y_i) \tag{11}$$

Thus, if any of the N dimensions are anomalous, we label the current timestamp as anomalous.

IV. Experiments

A. Datasets

In our experiments, we use several publicly available datasets, and their characteristics are summarized in Table I. For example, the SMAP dataset has 25 dimensions, with a training dataset length of 135,183, a test dataset length of 427,616, and an anomaly ratio of 13.13%.

TABLE I
Dataset Statistics

Datasets	Fearures	Train	Test	Anomalies(%)
SMAP [4]	25	135183	427616	13.13
MSL [4]	55	58317	73729	10.72
SMD [18]	38	708405	708420	4.16
SWaT [19]	51	496800	449919	11.98
WADI [20]	123	1048571	172801	5.99

TABLE II
The F1-score(%) results for anomaly detection on five publicly available datasets.

Method	ASD	MSL	SMAP	SWaT	WADI
PCA[2003]	82.28	91.16	24.67	58.64	31.86
IForest[2008]	75.70	84.56	67.18	73.50	69.64
LSTM-NDT[2018]	97.63	88.23	90.25	57.39	73.31
DAGMM[2018]	71.33	92.54	71.71	75.02	18.58
MTAD-GAT[2018]	97.83	97.33	94.77	77.79	69.75
USAD[2018]	62.30	86.75	76.15	55.37	50.67
GDN[2021]	96.31	96.52	94.33	27.29	73.59
TranAD[2021]	97.17	96.66	93.61	58.35	74.01
AnomalyTransformer[2022]	90.87	95.51	93.45	55.82	64.37
DGNN	**98.21**	**97.94**	**96.34**	**77.80**	**77.19**

B. Baseline

We conduct a comparison between our AGAT and a wide range of state-of-the-art multivariate time-series anomaly detection models, including: 1) Statistics-based models: IForest. 2) Reconstruction-based models: PCA, DAGMM, and USAD. 3) Forecasting-based models: LSTM-NDT, MTAD-GAT, GDN, TranAD, and Anomaly Transformer.

C. Evaluation Metrics

We use F1-Score (F1) over the test dataset and its ground truth values to evaluate the performance of our method and baseline models. The F1-Score is calculated as $F1 = \frac{2\times Prec\times Rec}{Prec+Rec}$, where $Prec = \frac{TP}{TP+FP}$ and $Rec = \frac{TP}{TP+FN}$. Here, TP, TN, FP, FN represent the numbers of true positives, true negatives, false positives, and false negatives, respectively.

D. Results

Our experimental results, presented in Table II, show outstanding performance on all five datasets with the F1 score calculated based on POT. Our model outperforms LSTM-NDT due to its neighbor feature fusion mechanism, while MTAD-GAT falls short due to the fully connected graph causing the GAT model to extract internal connections. The importance of dynamic subgraph generation is emphasized. GDN's poor performance on the SWaT dataset is attributed to the ineffective use of a fully connected layer as a predictive model. Figure 4 shows a significant reduction in the number of edges generated by DSG, with the MSL dataset experiencing a 98% reduction.

E. Ablation

To investigate the necessity of each component in our method, we gradually removed them to observe how the model's performance was affected. Firstly, we evaluated the importance of the learned graph by substituting it with a static complete graph, where every node is connected to all the other nodes. Secondly, to assess the significance of the , we disabled the attention mechanism and aggregated the information from neighboring nodes using equal weights. Lastly, to assess the significance of the Forecasting-based Model, we replaced GRU with a fully connected layer. The outcomes are summarized in Table III, revealing the following findings:

- Replacing the learned graph structure by DSG with a complete graph degrades performance in both datasets. The effect on the WADI dataset is more pronounced. This indicates that the graph structure learner enhances performance, especially for large-scale datasets.
- Removing the attention mechanism degrades the model's performance most in our experiments. Since sensors have very different behaviors, treating all neighbors equally introduces noise and misleads the model. This verifies the importance of the graph attention mechanism.

These findings suggest that the adoption of a learned graph structure, adaptive attention mechanisms, and forecasting-based model all contribute to the accuracy of our DGNN method, which provides an explanation for its superior performance over baseline methods.

TABLE III
Ablation.

Method	ASD	MSL	SMAP	SWaT	WADI
DGNN	**98.21**	**97.94**	**96.34**	**77.80**	**77.19**
w/o DSG	97.75	97.33	95.45	77.45	62.62
w/o AGAT	97.63	88.23	90.25	57.39	73.31
w/o Forecasting	95.31	70.42	92.49	76.86	50.14

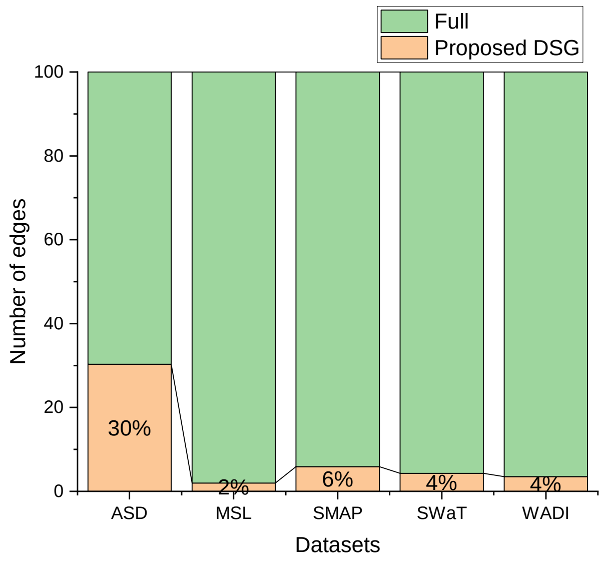

Fig. 4. Comparison of number of edges.

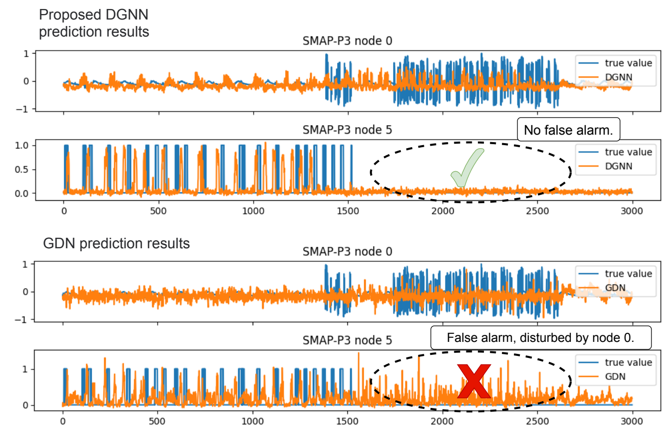

Fig. 5. Comparison of forecasting results of DGNN and GDN in SMAP-P3 dataset.

F. Interpretability of Model

The edges present in our learned graph provide interpretability as they indicate the relationship between different sensors. The attention weights further add to this by indicating the importance of each node's neighbors in modeling the node's behavior.

Figure 5 shows a comparison of the forecasting results between DGNN and GDN on the SMAP dataset. The upper graph in the figure shows the result predicted by DGNN, while the lower graph shows the result predicted by GDN. The orange line represents the predicted value, while the blue line represents the true value. It can be observed that DGNN uses the correlation coefficient to calculate the weight of neighbors, while GDN uses a multi-head attention mechanism to obtain the weight. Due to the low feature correlation between node 0 and node 5, DGNN does not link these two nodes. When node 1 becomes abnormal at x=1500, the predicted value of node 5 does not interfere with node 1. On the other hand, GDN's multi-head attention mechanism fails to build a proper relationship between node 0 and node 5. Therefore, when an abnormality occurs in node 1, the predicted value of node 5 is disturbed by node 0. DGNN can deduce that node 0 is abnormal based on the deviation between the predicted value and the true value. However, the GDN method prompts both node 0 and node 5 to be abnormal simultaneously, leading to false alarms.

V. Conclusion

In this paper, we introduce Dynamic Graph Neural Networks (DGNN) for anomaly detection in multivariate time-series data. Our model uses dynamic subgraph generation (DSG) to identify relationships and generate subgraphs in a data-driven way, instead of fixed fully connected graphs. DSG brings similar time series closer, reduces edges, and speeds up GNN convergence. We propose Adaptive Graph Attention Network (AGAT) for accurate neighbor node feature extraction and fusion, outperforming existing GAT methods. Our algorithm outperforms existing SOTA algorithms on five public datasets. In the future, we plan to apply DSG to more GNN models, replacing fully connected graphs with dynamic subgraphs to enhance their performance.

VI. Acknowledgement

This work was supported by National Key R&D Program of China under Grant 2020YFA0711400, and National Science Foundation of China under Grant U21A20452, U19B2044.

References

[1] G. Kirchgässner, J. Wolters, and U. Hassler, *Introduction to modern time series analysis*. Springer Science & Business Media, 2012.

[2] W. A. Fuller, *Introduction to statistical time series*. John Wiley & Sons, 2009.

[3] H. Madsen, *Time series analysis*. CRC Press, 2007.

[4] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.

[5] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "Deepant: A deep learning approach for unsupervised anomaly detection in time series," *Ieee Access*, vol. 7, pp. 1991–2005, 2018.

[6] R. Wu and E. Keogh, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[7] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Multivariate time-series anomaly detection via graph attention network," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 841–850.

[8] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *stat*, vol. 1050, p. 20, 2017.

[9] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4027–4035.

[10] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," Miami Univ Coral Gables Fl Dept of Electrical and Computer Engineering, Tech. Rep., 2003.

[11] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[12] P. Malhotra, L. Vig, G. Shroff, P. Agarwal *et al.*, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89, 2015, pp. 89–94.

[13] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International conference on learning representations*, 2018.

[14] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3395–3404.

[15] S. Tuli, G. Casale, and N. R. Jennings, "Tranad: Deep transformer networks for anomaly detection in multivariate time series data," *arXiv preprint arXiv:2201.07284*, 2022.

[16] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," *arXiv preprint arXiv:2110.02642*, 2021.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[18] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1067–1075.

[19] A. P. Mathur and N. O. Tippenhauer, "Swat: A water treatment testbed for research and training on ics security," in *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*. IEEE, 2016, pp. 31–36.

[20] C. M. Ahmed, V. R. Palleti, and A. P. Mathur, "Wadi: a water distribution testbed for research in the design of secure cyber physical systems," in *Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks*, 2017, pp. 25–28.