Learning Sparse Latent Graph Representations for Anomaly Detection in Multivariate Time Series

Siho Han*
siho.han@rtm.ai
RTM
Seoul, South Korea

Simon S. Woo†
swoo@g.skku.edu
Applied Data Science Department
Sungkyunkwan University
Suwon, South Korea

ABSTRACT

Anomaly detection in high-dimensional time series is typically tackled using either reconstruction- or forecasting-based algorithms due to their abilities to learn compressed data representations and model temporal dependencies, respectively. However, most existing methods disregard the relationships between features, information that would be extremely useful when incorporated into a model. In this work, we introduce Fused Sparse Autoencoder and Graph Net (FuSAGNet), which jointly optimizes reconstruction and forecasting while explicitly modeling the relationships within multivariate time series. Our approach combines Sparse Autoencoder and Graph Neural Network, the latter of which predicts future time series behavior from sparse latent representations learned by the former as well as graph structures learned through recurrent feature embedding. Experimenting on three real-world cyber-physical system datasets, we empirically demonstrate that the proposed method enhances the overall anomaly detection performance, outperforming baseline approaches. Moreover, we show that mining sparse latent patterns from high-dimensional time series improves the robustness of the graph-based forecasting model. Lastly, we conduct visual analyses to investigate the interpretability of both recurrent feature embeddings and sparse latent representations.

CCS CONCEPTS

• **Computing methodologies** → Neural networks; Anomaly detection.

KEYWORDS

Anomaly detection; Multivariate time series; Joint optimization; Graph neural network; Sparse representations

ACM Reference Format:
Siho Han and Simon S. Woo. 2022. Learning Sparse Latent Graph Representations for Anomaly Detection in Multivariate Time Series. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539117

*Work done while a student at Sungkyunkwan University
†Corresponding author

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539117

1 INTRODUCTION

An essential technology of Industry 4.0, cyber-physical systems (CPSs) integrate physical and software components to control and monitor mechanisms, such as robotic, automotive or industrial control systems (ICSs) and other critical infrastructure. Industrial sensors (and actuators) organized in distinct processes within such systems continuously and frequently interact with each other, generating a massive amount of time series characterized by high complexity and dimensionality. Both properties complicate human operation and monitoring of complex industrial systems, thus requiring automated security measures to ensure CPS reliability. Challenges related to this issue give rise to a plethora of research problems, one of which is the development of deep learning algorithms to correctly identify anomalies and hopefully, prevent potential risks.

However, CPSs are in general already reliable in nature, resulting in a natural lack of anomalous cases and accordingly, only a few labeled anomalies in the produced time series. Nevertheless, anomaly detection in CPSs remains as an important task because even seldom occurring anomalous events can initiate catastrophic failure, often leading to a cascading one, across the whole system due to the high interconnectivity of sensors within and between processes. In addition to irretrievable damages caused, for instance, by cybersecurity incidents, manageable cases of faults or anomalies in the system may be equally troublesome, in that they necessitate inspections by human experts with inevitable downtime. Moreover, anomalies may appear unexpectedly in diverse, possibly new forms. To address the rarity of anomalies and variability of their patterns in high-dimensional sensor data, previous work have mostly relied on unsupervised learning to formulate novelty detection problems based on one-class approaches [26], which have shown to outperform statistical, rule-based approaches [8].

A prominent method of applying the unsupervised learning paradigm to tackle such problems includes deep learning algorithms, which have seen enormous success in the past few years due to their ability to effectively learn from non-linear patterns in a large amount of data. The most widely employed techniques are based on Convolutional Neural Networks (CNNs) [23] and Recurrent Neural Networks (RNNs) [27] for forecasting-based anomaly detection, and generative models, such as Variational Autoencoders (VAEs) [3] and Generative Adversarial Networks (GANs) [13] for reconstruction-based anomaly detection; vanilla and other variants of Autoencoders are also commonly used for anomaly detection, using reconstruction errors as anomaly scores [1]. Forecasting- and

reconstruction-based models have distinct advantages—such as the former's ability to model temporal dependencies and the latter's ability to learn meaningful data representations—that are worthwhile exploiting simultaneously, rather than independently [38]. Nonetheless, most existing algorithms rely either on forecasting or reconstruction for anomaly detection. Moreover, they cannot learn complex interdependencies among features [10], information that can be extremely useful when incorporated into the model.

In that regard, Graph Neural Networks (GNNs) [39] have recently gained traction among researchers attempting to model relationships (i.e., edges) between features (i.e., nodes or vertices) in non-Euclidean space. While there exist numerous forms of graphical models, the essence of GNNs in geometric deep learning [7] is the message passing (also referred to as diffusion, propagation, or neighborhood aggregation under different guises) function [6]. A key characteristic of graphs, permutation invariance in GNNs is preserved by aggregating information across neighbor nodes through differentiable functions, such as summation, averaging or maximization [6]. However, the message passing scheme, albeit effective in graph representation learning, renders GNNs vulnerable to noises due to, for example, the inherent evolving and highly variable nature of operating conditions in CPSs [32, 33]. Moreover, directly applying GNNs to high-dimensional time series is arduous because the graph structure is initially unknown in most cases [10].

To address the abovementioned challenges, we formulate the following research questions (RQs):

- **RQ1 (Detection performance)**. Can we accurately detect anomalies in high-dimensional sensor data under a realistic scenario where data representing abnormal events are extremely scarce?
- **RQ2 (Ablation)**. Does each of our proposed model's core components have a positive impact on the final anomaly detection performance?
- **RQ3 (Interpretabilty)**. Do the graph structure and sparse latent features learned by our model aptly characterize highly variable sensor data?

In this work, we introduce a time series anomaly detection framework that can capture complex inter-sensor relationships and achieve robust detection performance on real-world CPS datasets by jointly optimizing reconstruction and forecasting by Sparse Autoencoder (SAE) [22] and GNN, respectively. Our proposed model, Fused Sparse Autoencoder and Graph Net (FuSAGNet), combines SAE and GNN, such that the latter predicts future sensor behavior from sparse latent representations of input data learned by the former as well as graph structures learned through recurrent sensor embedding. To effectively characterize constituent sensors in a CPS, our embedding modules are separately applied to sensors pertaining to distinct processes. The final anomaly score is computed by either aggregating the reconstruction and forecasting errors or considering only the latter generated from the graph-based forecasting model enhanced by joint optimization. Our contributions are summarized as follows:

- We propose FuSAGNet, an anomaly detection framework that jointly optimizes reconstruction and forecasting under sparsity constraint while capturing interdependencies among high-dimensional time series generated by sensors.
- We experiment on three real-world CPS datasets and empirically demonstrate our model's effectiveness by comparing its detection performance to those of baseline approaches.
- We conduct visual analyses to provide interpretability of both graph structure and sparse latent representations obtained through recurrent sensor embedding and sparsity-constrained reconstruction, respectively.

2 RELATED WORK

A sizeable body of literature has proposed anomaly detection algorithms for identifying abnormal patterns in multivariate times series. In this section, we provide a high-level overview of methods relevant to and/or involved in our proposed framework.

2.1 Time Series Anomaly Detection

Deep learning models can learn non-linear and/or compressed representations of multivariate time series. RNNs are in particular suitable for modeling and forecasting time series due to their ability to retain knowledge on past observations. Long Short-Term Memory (LSTM) [27] has been proposed to mitigate the vanishing/exploding gradient problem [14] encountered when training RNNs by introducing a memory cell and three activation-regulated gates that selectively remember/forget information on past observations. Simplified variants of LSTMs are Gated Recurrent Units (GRUs) [9], which only consist of two gates, update and reset, leading to fewer model parameters and higher computational efficiency. Applicable to RNNs, a bidirectional variant comprises an additional module that processes the input in the backward direction [29]; the resulting forward- and backward-processed outputs are then combined to better characterize the input time series, containing richer information. Due to their ability to effectively model time series, methods including RNNs often learn normal data patterns during training to identify abnormal patterns during testing.

Reconstruction-based deep learning models form another class of popular time series anomaly detection approaches [8]. For example, Autoencoders (AEs) [1] use reconstruction errors between the original input and the reconstructed output as anomaly scores, based on which abnormal samples are distinguished from normal ones with respect to a threshold. Regularized AEs such as SAEs [22] have been developed to prevent AEs from learning the identity function, thus yielding meaningful latent representations in the hidden space. Unlike most AEs, SAEs have a unique characteristic of allowing, based on user preference, the hidden space size to be at least equal to the input/output space size [22]; this is a useful property that we will leverage to extract sparse latent representations for each of the multivariate time series (described in Section 3.4), that is, for each and every sensor or actuator in a CPS.

However, the above mentioned forecasting- and reconstruction-based models are limited to either capturing temporal dependencies within or learning representations of input features and cannot explicitly learn inter-feature relationships [10].

2.2 Graph Neural Networks

The common limitations of reconstruction- and forecasting-based deep learning models, that is, their inability to capture complex relationships between features, can be addressed by GNNs. Graph

Convolution Networks (GCNs) [17] apply convolution operations in non-Euclidean space to aggregate information from neighboring nodes connected by edges. Graph Attention Networks (GATs) [36] have been developed to account for different weights of these edges using the attention mechanism [35]. However, the above approaches are not suitable for application to high-dimensional sensor data for three main reasons: first, they do not explicitly model varying behaviors of each node, precluding them from reflecting highly variable operating conditions of different sensors; second, even if they do address the first issue, they are prone to be affected by the noisy nature of sensor data; third, the relationships between sensors, that is, the graph structure, are initially unknown, but are required as part of the input.

Recently, Deng and Hooi proposed Graph Deviation Network (GDN) [10], which learns inter-sensor relationships through sensor embedding and predicts future sensor behavior through graph attention-based forecasting. It has shown great promise in accurately detecting anomalies on real-world CPS datasets, but there is still room for improvement: first, GDN does not consider different processes within a CPS when embedding sensors as well as temporal dependencies within the embedded sensor vectors; second, GDN's detection performance significantly varies depending on data representing different operating conditions, implying lack of robustness. Our proposed FuSAGNet integrates GDN to directly address the state-of-the-art GNN's limitations.

Another applicable work related to ours is Multivariate Time-series Anomaly Detection via Graph Attention Network (MTAD-GAT) proposed by Zhao et al. [38], which uses GAT to model the relationships between sensors in addition to reconstruction- and forecasting-based deep learning models jointly optimized to enhance anomaly detection performance. Our proposed FuSAGNet differs from MTAD-GAT in both arrangement of reconstruction- and forecasting-based models (concurrent vs. parallel) and usage of graph attention (forecasting vs. inter-feature relationship learning). Moreover, MTAD-GAT assumes a given set of multivariate time series to form a complete graph [38], which may not necessarily hold in real-world scenarios where sensors in a CPS may be related in an asymmetric fashion. Hence, we assume a directed graph for our approach as Deng and Hooi [10] did to implement a more realistic anomaly detection scenario for high-dimensional sensor data.

3 PROPOSED METHOD

In this section, we explain our proposed methodology in detail. A high-level overview of FuSAGNet is depicted in Fig. 1. To summarize, our framework includes four main phases: 1) intra-process recurrent sensor embedding, 2) graph structure learning, 3) sparsity-constrained joint optimization, and 4) anomaly scoring.

3.1 Problem Formulation

Given a set of multivariate time series generated from N sensors, we experiment under an unsupervised learning setting where our train set consists only of normal data while our test set additionally includes anomalous data, to cope with the class imbalance between normal samples and anomalies; that is, we aim to learn sensor behaviors under normal operating conditions during training and detect abnormal sensor readings during testing. At the end, each test sample at a given time t is assigned a label, $a(t)$, indicating whether it is deemed normal ('0') or anomalous ('1') by our algorithm.

3.2 Intra-Process Recurrent Sensor Embedding

CPS sensors can be categorized within a process based on their functions. In a water treatment plant, an initial process may store the received raw water in a tank, while a subsequent process may pre-treat the water by chemical dosing and pass on to the next processes for further water filtration [12], as depicted in Fig. 2.

As illustrated in this simplified example, there exist clear dependencies between processes, implying correlations (or even causality) among sensors in distinct processes. On the other hand, it is equally plausible that sensors composing the same process ("intra-process") also exhibit notable correlations that are worthwhile examining [10]. Also, given a possible sequential (or even cyclic) arrangement of sensors interacting with each other, there may also exist temporal dependencies among them. This motivates our choice to separately embed sensors in each process followed by recurrent units to capture temporal dependencies within a group of sensors in a process, unlike in GDN [10] where all sensors are embedded as a whole using the same embedding module and without recurrent units.

More formally, let $v_{p,i} \in \mathbb{R}^d$, where $d \in \mathbb{N}^+$ is the embedding dimension, denote a randomly initialized embedding vector for node i pertaining to a process p. Then, the intra-process recurrent embedding of $v_{p,i}$ denoted by $r_{p,i}$ is defined as:

$$r_{p,i} = f_p^m(v_{p,i}), \tag{1}$$

where $f_p^m(\cdot)$ denotes recurrent units for the process p stacked m times. In our framework, we choose to stack $m = 3$ bidirectional GRUs (biGRUs) for all p mainly due to their ability to model richer representations than the unidirectional variants and higher computational efficiency than biLSTMs; other RNN variants, however, either unidirectional or bidirectional, can also be used depending on the data and hardware specifications. Concatenated, these recurrent sensor embeddings are optimized along the training process and utilized for other components of our framework, namely construction of adjacency matrices for graph structure learning and calculation of attention weights for graph attention-based forecasting, analogously to GDN [10] (described in Sections 3.3 and 3.4).

3.3 Graph Structure Learning

Revealing the graph structure or the dependencies between complexly related sensors is crucial for enabling the application of GNNs to multivariate time series. Adopting the methodology by Deng and Hooi [10], we first define an adjacency matrix, A, to represent a directed graph, such that A_{ij} denotes the presence of a directed edge from node i to node j. Next, assuming we have no prior information on the CPS except its constituent processes, we compute the cosine similarity, e_{ij}, between the recurrent sensor embeddings of a node i and those of all its potential neighbor nodes j as follows:

$$e_{ij} = \frac{r_i^T \cdot r_j}{\|r_i\| \cdot \|r_j\|}, \tag{2}$$

where $\|\cdot\|$ denotes magnitude. From these similarity measures, we define an element of our adjacency matrix, A_{ji}, as follows:

$$A_{ji} = \mathbb{1}\{j \in TopK(\{e_{ki} : k \in \{1, 2, ..., N\}\})\}, \tag{3}$$

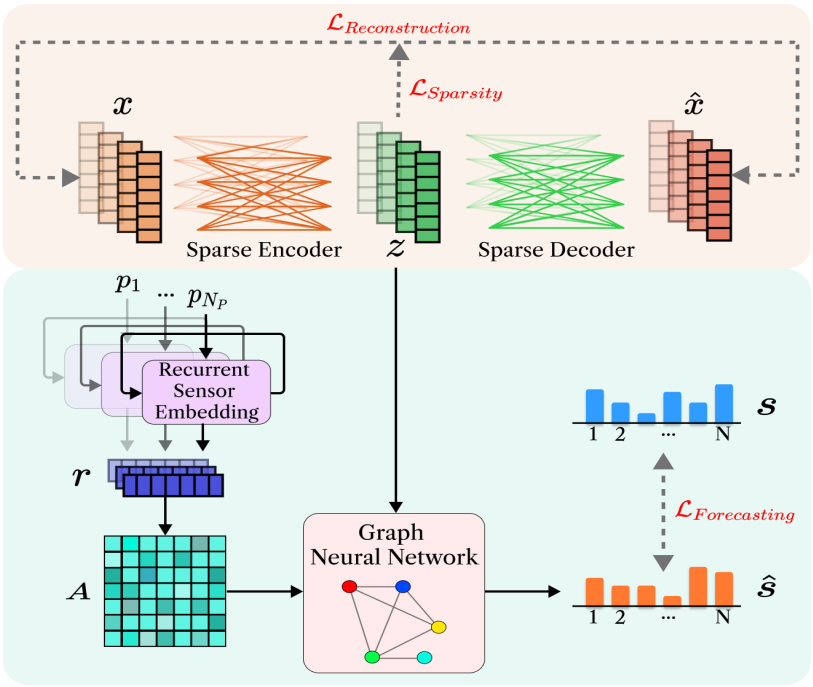

Figure 1: Overview of FuSAGNet where the top branch (orange background) corresponds to reconstruction and the bottom branch (blue background) corresponds to forecasting. x and $\hat{x}$ respectively denote the input and reconstructed output with batch size B, number of features N, and window size w; z denotes the sparse latent representations of x, and $\hat{s}$ and s respectively denote the predicted and actual observations; $p_1, ..., p_{N_p}$ denote groups of sensors (represented by integer labels) within processes where N_p is the number of processes, r denotes recurrent embeddings, and A denotes the adjacency matrix. Loss functions involved in different parts of the model are highlighted in red.

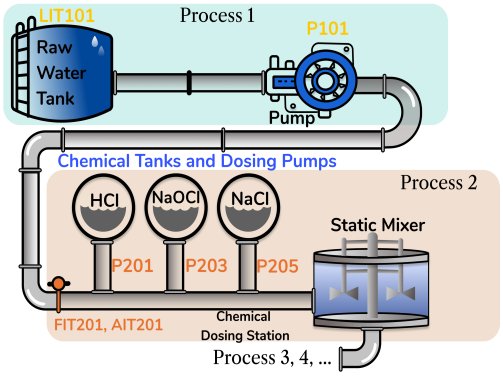

Figure 2: Diagram of the first two Secure Water Treatment (SWaT) [12] test bed processes. Sensors in processes 1 and 2 are highlighted in yellow and orange, respectively.

where $\mathbb{1}$ denotes the indicator function and $TopK(\cdot)$ selects the k node indices with the highest cosine similarities [10]. Note that a small k results in a sparse adjacency matrix.

3.4 Sparsity-Constrained Joint Optimization

We first leverage SAE to extract non-trivial (i.e., sparse and latent) features from high-dimensional time series, which will be fed as input to both the decoder for reconstruction and GNN for forecasting.

Sparse Autoencoder (SAE) for Sparse Latent Representation Learning. Unlike regular AEs and most of their variants with a smaller hidden space than the input space, SAEs allow a hidden space at least as large as the input space without compromising effective representation learning [22]. This is achieved by imposing a sparsity constraint on the hidden units that forces most neurons to be inactive (i.e., close to 0) rather than active (i.e., close to 1).

More formally, an SAE induces sparsity in its latent space by introducing a parameter, ρ, which satisfies the following property:

$$\rho = \hat{\rho}_l = \mathbb{E}[a_l(x)] \approx 0, \tag{4}$$

where $a_l(x)$ denotes the activation of the l^{th} hidden unit for an input x and $\hat{\rho}_l$ denotes the corresponding average across the training set. In other words, most activations in the latent space will be inactive and the remaining active ones will turn out to be the most representative features of the input [22]. The sparsity constraint can be satisfied through several ways; in our framework, we choose to include a Kullback-Leibler (KL) divergence [18] term in the SAE's

objective function as follows:

$$\mathcal{L}_{Reconstruction} = \mathcal{L}_2(x, \hat{x}) + \beta \sum_{l=1}^{L} KL(\rho||\hat{\rho}_l), \tag{5}$$

where $\mathcal{L}_2(x, \hat{x})$ denotes the mean squared error (MSE) between the input x and its reconstructed output $\hat{x}$, L denotes the number of neurons in the hidden layer, and β denotes the weighting factor of the KL divergence term, defined as follows:

$$\sum_{l=1}^{L} KL(\rho||\hat{\rho}_l) = \sum_{l=1}^{L} \rho \log \frac{\rho}{\hat{\rho}_l} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_l}. \tag{6}$$

SAE is trained to minimize both the reconstruction error and the distance between the distributions ρ and $\hat{\rho}_l$, resulting in sparse latent representations of the input. In addition to this sparsity constraint, the preserved hidden space size with respect to the input space allows the model to learn individual latent representations for each node, making them compatible when fed as input to our graph-based forecasting model. This justifies our choice of SAE (over other AE variants such as VAEs) as our reconstruction model for joint optimization with GNN, since GNNs require features of all nodes of interest as input. The complete objective function, combining the reconstruction and forecasting errors, for our whole framework will be introduced next.

Graph Attention-Based Forecasting. We use GDN [10] as our base forecasting model but with three modifications made, which will be described in the remainder of this section.

Our raw input data, x_{train}, is preprocessed using a sliding window of length w, such that $x^{(t)} := [s^{(t-w)}, s^{(t-w+1)}, ..., s^{(t-1)}] \in \mathbb{R}^{N \times w}$, where $s^{(t)}$ denotes normalized sensor readings at time t within the range $[0, 1]$. We are thus aiming to model data patterns in historical observations corresponding to normal sensor behaviors and predict the future behavior of each sensor. However, unlike for GDN where the raw time windows are directly used as input, we feed the sparse latent representations of $x^{(t)}$, denoted by $z^{(t)} \in \mathbb{R}^{N \times w}$, generated from our sparse encoder to minimize the negative impact of noises on the GNN.

Another distinctive aspect of our FuSAGNet is that it incorporates intra-process biGRU sensor embeddings, $r_{p,i}$, during message passing rather than regular sensor embeddings as in GDN. The former embedding can lead to richer (temporal) representations of each node's characteristics by combining the outputs of forward- and backward-processing units of biGRU. Using the sparse latent representations of the original input, $z^{(t)}$, node i's aggregated feature at time t, denoted by $Z_i^{(t)}$, is defined as follows:

$$Z_i^{(t)} = ReLU\left(\alpha_{i,i} W z_i^{(t)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} W z_j^{(t)}\right), \tag{7}$$

where $\mathcal{N}(i) = \{j \mid A_{ji} > 0\}$ denotes the set of neighbor nodes of node i, $W \in \mathbb{R}^{d \times w}$ denotes a trainable weight parameter matrix, and $\alpha_{i,j}$ denotes attention coefficients computed using recurrent sensor embeddings, r_i, as follows:

$$g_i^{(t)} = r_i \oplus W z_i^{(t)}, \tag{8}$$

$$\pi_{i,j} = LeakyReLU\left(a^T \left(g_i^{(t)} \oplus g_j^{(t)}\right)\right), \tag{9}$$

$$\alpha_{i,j} = \frac{e^{\pi_{i,j}}}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} e^{\pi_{i,k}}}, \tag{10}$$

where $\oplus$ denotes concatenation and a denotes learnable parameters. The extracted features, following scaling by recurrent sensor embeddings, are fed to a linear layer, $f_{Linear}(\cdot)$, to make final predictions, $\hat{s}^{(t)} \in \mathbb{R}^N$, at time t as follows:

$$\hat{s}^{(t)} = f_{Linear}\left(\left[r_1 \circ Z_1^{(t)}, r_2 \circ Z_2^{(t)}, ..., r_N \circ Z_N^{(t)}\right]\right), \tag{11}$$

where $\circ$ denotes element-wise multiplication.

Lastly, we push the model to learn better sparse latent representations of the original input over the training process by jointly optimizing our SAE and graph-based forecasting model, unlike in GDN, which only attempts to minimize the forecasting error, that is, $\mathcal{L}_2\left(s^{(t)}, \hat{s}^{(t)}\right)$. To this end, we aim to minimize the following loss function, which combines the respective objective functions for reconstruction and forecasting:

$$\mathcal{L}_{total} = \lambda \mathcal{L}_{Forecasting} + (1-\lambda) \mathcal{L}_{Reconstruction}, \tag{12}$$

where λ denotes a balancing parameter between 0 and 1, and $\mathcal{L}_{Forecasting}$ denotes the root mean squared error (RMSE) between the actual and predicted future sensor behaviors, defined as follows:

$$\mathcal{L}_{Forecasting} = \sqrt{\mathcal{L}_2\left(s^{(t)}, \hat{s}^{(t)}\right)}. \tag{13}$$

3.5 Anomaly Scoring

As a result of our concurrent joint optimization scheme between the reconstruction and forecasting tasks, we obtain two sets of anomaly scores at the end of our training. To compensate erroneous predictions made by either one of the reconstruction or forecasting model, we opt to calculate the weighted harmonic mean (WHM) of the reconstruction and forecasting anomaly scores obtained through graph deviation scoring [10].

Graph deviation scoring has been proposed to compute a robust anomaly statistic less sensitive to severe deviation of a particular sensor's behavior at a given time. The authors [10] achieve this by standardizing each sensor's forecasting error set with its median and interquartile range (IQR) instead of its mean and standard deviation, which are strongly affected by outliers, as follows:

$$a_i(t) = \frac{Err_i(t) - \widetilde{\mu}_i}{\widetilde{\sigma}_i}, \tag{14}$$

where $a_i(t)$ denotes the anomaly score for node i at time t, and $\widetilde{\mu}_i$ and $\widetilde{\sigma}_i$ each denotes the median and IQR of $Err_i(t)$, defined as $Err_i(t) = \left|s_i^{(t)} - \hat{s}_i^{(t)}\right|$. The anomaly scores for individual sensors are aggregated by taking the maximum across all sensors to produce an overall anomaly score at time t. Smoothing by moving average is applied to reduce the effect of abrupt changes in anomaly scores, which may have been caused by a trivial glitch under normal operating conditions [15]. The detection threshold is set as the maximum of these smoothed anomaly scores across the validation set.

For our approach, we compute the graph deviation scores for forecasting errors as explained above but add an intermediate step for reconstruction errors to reduce their dimension and later generate a combined score with the forecasting anomaly scores. To this end, we first take the maximum across the window size w, such that its dimensionality is reduced from $N \times w$ to N, now matching

that of the forecasting errors. To aggregate the graph deviation scores for forecasting and reconstruction, respectively denoted by $A^{(t)}_{Forecasting}$ and $A^{(t)}_{Reconstruction}$, we compute their WHM using the weighting factor λ, which has been used to assign weights in our loss function as shown in Equation 12. We define the WHM of the two anomaly scores at time t, denoted by $A^{(t)}_{WHM}$, as follows:

$$A^{(t)}_{WHM} = \left(\frac{\lambda}{A^{(t)}_{Forecasting}} + \frac{1-\lambda}{A^{(t)}_{Reconstruction}} \right)^{-1}, \quad (15)$$

with the threshold set as its maximum across the validation set.

4 EXPERIMENTS AND RESULTS

In this section, we explain our experiments in detail, including performance comparison against baseline approaches. We also conduct quantitative and qualitative analyses to investigate the effectiveness of our proposed method and gain insights into limitations. Our detailed experimental setup is presented in Appendix A.1.

4.1 Datasets

High-dimensional sensor data generated from real-world industrial systems with anomalous cases are rare; yet there exist two widely used CPS datasets for time series anomaly detection research—Secure Water Treatment (SWaT) [12] and Water Distribution (WADI) [2]—both generated and released[1] by iTrust Center for Research in Cybersecurity, Singapore University of Technology and Design. Another CPS dataset that has been fairly recently proposed is Hardware-In-the-Loop-based Augmented Industrial Control System Security (HAI) [30], generated and released[2] by National Security Research Institute (NSR), South Korea. Dataset description, composition, and preprocessing steps are presented in Appendix A.2.

4.2 Baseline Approaches

We select ten baselines for anomaly detection in multivariate time series to compare the performance of our proposed FuSAGNet:

- **PCA** [31]. Principal Component Analysis projects data onto lower-dimensional space, such that most variations are preserved. Similar to its non-linear variants, Autoencoders, it uses reconstruction errors to identify anomalies.
- **KNN** [4]. K-Nearest Neighbor uses the distance to the nearest data points as a local density estimate to detect anomalies.
- **OCSVM** [28]. One-Class Support Vector Machine transforms data through a kernel function and constructs a decision boundary to distinguish normal data from anomalies.
- **AE** [27]. Autoencoder uses the error between the input and its reconstructed output to detect anomalies.
- **DAGMM** [40]. Combining deep Autoencoders with GMM, Deep Autoencoding Gaussian Mixture Model learns to generate low-dimensional representations and reconstruction errors for each data point.
- **LSTM-VAE** [24]. Long Short-Term Memory Variational Autoencoder uses LSTM layers to compose VAE, which uses the reconstructions to detect anomalies.
- **MAD-GAN** [20]. Multivariate Anomaly Detection with Generative Adversarial Network is trained adversarially, involving an LSTM-RNN discriminator and a GAN generator, to generate reconstruction errors for anomaly detection.
- **USAD** [5]. Unsupervised Anomaly Detection adversarially trains Autoencoders with two decoders that respectively act as a discriminator and a generator for anomaly detection.
- **MTAD-GAT** [38]. Multivariate Time-series Anomaly Detection via Graph Attention Network uses feature- and time-oriented graph attention mechanisms to capture both causal relationships and temporal dependencies among multivariate time series, while jointly optimizing reconstruction and forecasting to detect anomalies in spacecraft telemetry data.
- **GDN** [10]. Recently proposed state-of-the-art graph-based model for anomaly detection in high-dimensional sensor data, Graph Deviation Network uses graph structure learning and graph attention-based forecasting to identify anomalies with behaviors deviating from normal ones.

4.3 Research Question 1: Detection Performance

We present the anomaly detection performance of our model as well as those of baseline approaches in Table 1. The best performance for SWaT is obtained by applying aggregated anomaly scoring, whereas the performances for WADI and HAI are obtained by applying regular graph deviation scoring [10]; that is, for WADI and HAI, we only consider the detection performance derived from forecasting errors after joint optimization.

The best performances of FuSAGNet shown on Table 1 are achieved with relatively low values of λ, namely $\lambda = 0.1$ for SWaT and HAI, and $\lambda = 0.3$ for WADI. Overall, we can observe that on the three CPS datasets, our FuSAGNet outperforms all baseline approaches in terms of f1 score. It also achieves the highest recall in all cases except on WADI, where it ranks second behind OCSVM (47.87% vs. 100%); however, OCSVM has a significantly lower precision (5.77%) than FuSAGNet (82.92%) on the same dataset.

On the other hand, FuSAGNet performs relatively poorly in terms of precision, especially on WADI where the best-performing model GDN's precision exceeds ours by more than 14% (97.50% vs. 82.92%). We assume that this performance difference is attributed to the rather simple architecture of our single-layer SAE. Recall that WADI contains more than twice as many sensors compared to SWaT and almost as twice as many compared to HAI. Since our SAE has a latent space size equal to that of the input space, the number of hidden units is much larger than for SWaT and HAI, rendering the general training process difficult even with the sparsity constraint. Although the SAE in FuSAGNet only consists of a single hidden layer for simplicity, increasing its depth corresponding to the size of input space may help mitigate this problem.

Nevertheless, high recall with slight compromise of precision otherwise, as achieved by FuSAGNet, can be of paramount value for CPS security. Considering most cases where the trade-off between FPs and FNs is inevitable, in practice, maintenance technicians with domain expertise will prefer a high level of sensitivity over specificity to avoid missing any critical incidents that are worthwhile inspecting for future reference [5]. While minimizing downtime

[1] https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/
[2] https://github.com/icsdataset/hai

Table 1: Anomaly detection performance on SWaT, WADI, and HAI in terms of precision (Pr) (%), recall (Re) (%), and f1 score (F1) (%). The best performances for each evaluation metric are highlighted in bold and the second best are underlined to facilitate visual comparison. Results are partly from the work of Li et al. [20], and Deng and Hooi [10].

	SWaT			WADI			HAI		
Method/Metric	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**
PCA	24.92	21.63	23.16	39.53	5.63	9.86	37.44	12.09	18.28
KNN	7.83	7.83	7.83	7.76	7.75	7.75	12.53	11.60	12.05
OCSVM	62.88	67.33	65.03	5.77	**100.0**	10.91	3.76	34.92	6.79
AE	72.63	52.63	61.03	34.35	34.35	34.35	53.55	73.38	61.92
DAGMM	27.46	69.52	39.37	54.44	26.99	36.09	21.38	68.79	32.62
LSTM-VAE	96.24	59.91	73.85	87.79	14.45	24.82	57.83	70.51	63.54
MAD-GAN	98.97	63.74	77.54	41.44	33.92	37.30	47.70	32.83	38.89
USAD	**100.0**	56.00	71.79	43.09	22.51	29.57	9.32	13.35	10.98
MTAD-GAT	21.03	64.46	31.71	11.72	30.55	16.94	13.30	34.29	19.17
GDN	99.35	68.12	80.82	**97.50**	40.19	56.92	**88.37**	60.32	71.70
FuSAGNet (Ours)	98.78	**72.60**	**83.69**	82.92	47.87	**60.70**	86.79	**74.79**	**80.34**

Table 2: Ablation study by evaluating the detection performance in terms of precision (Pr) (%), recall (Re) (%), and f1 score (F1) (%) when core components of FuSAGNet are removed. The best performances, that is, those without removal of any components, are highlighted in bold. *IRSE* denotes intra-process recurrent sensor embedding and *SAE* denotes Sparse Autoencoder. For the variant without SAE ("*w/o SAE*"), we show the detection performance averaged over multiple trials (50 times) on different validation sets; the values in parentheses refer to standard deviations.

	SWaT			WADI			HAI		
Method/Metric	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**	**Pr**	**Re**	**F1**
FuSAGNet (Ours)	**98.78**	**72.60**	**83.69**	**82.92**	**47.87**	**60.70**	**86.79**	**74.79**	**80.34**
w/o IRSE	96.82	68.61	80.31	76.18	41.53	53.76	81.77	73.48	77.40
w/o SAE	84.15 (± 5.76)	65.79 (± 1.62)	76.48 (± 6.35)	65.72 (± 8.11)	39.38 (± 3.82)	48.25 (± 4.91)	78.39 (± 7.99)	65.66 (± 3.19)	72.39 (± 4.34)

caused by FPs is indeed important for CPS efficiency, detecting subtle signs of potential anomalies can lead to improved reliability in the long term because even rarely occurring anomalies may result in the failure of the entire system, as discussed in Section 1.

Apart from GDN [10], MTAD-GAT [38] is another graph-based model highly relevant to our FuSAGNet. However, we can observe in Table 1 that MTAD-GAT performs relatively poorly on SWaT, WADI, and HAI. This may be attributed to how MTAD-GAT treats a given set of sensors generating multivariate time series to compose a complete graph [38] instead of an incomplete, directed graph as is the case for sensors in complex CPSs. Since we already have knowledge that the constituent sensors in the CPS datasets we are dealing with are not necessarily connected to all other sensors (given their sequential arrangements as shown in Fig. 2), we can presume that the inter-sensor relationships modeled through a complete graph actually hindered the learning process of the model, provided with inaccurate information.

4.4 Research Question 2: Ablation

To demonstrate the necessity of FuSAGNet constituents for achieving the best detection performances shown in Table 1, we conduct an ablation study, as shown in Table 2. First, we can observe that the removal of intra-process recurrent sensor embeddings (denoted by IRSE) negatively affects the detection performance. Since these modules characterize each sensor's behavior as well as temporal dependencies between a group of sensors within a process of a CPS, they contribute to the improvement of performance on all three datasets: the f1 score increased from 80.31% to 83.69% for SWaT, from 53.76% to 60.70% for WADI, and from 77.40% to 80.34% for HAI, as shown in Table 2.

Second, the absence of SAE drastically degrades the overall performance. As discussed in Section 2.2, GNNs are sensitive to noise and without SAE to feed sparse latent representations of the input data, the graph-based forecasting model alone trained on the raw input shows lack of robustness across different validation sets. As shown in Table 2, this is illustrated through repeated trials (50 times) of anomaly detection on different train and validation sets representing varying operating conditions. We can observe that the average performance drop also entails significant variance when the sparse latent representations learned by SAE is not used as input data for graph-based forecasting: the average f1 score decreased from 83.69% to 76.48% (± 6.35) for SWaT, from 60.70% to 48.25% (± 4.91) for WADI, and from 80.34% to 72.39% (± 4.34) for HAI.

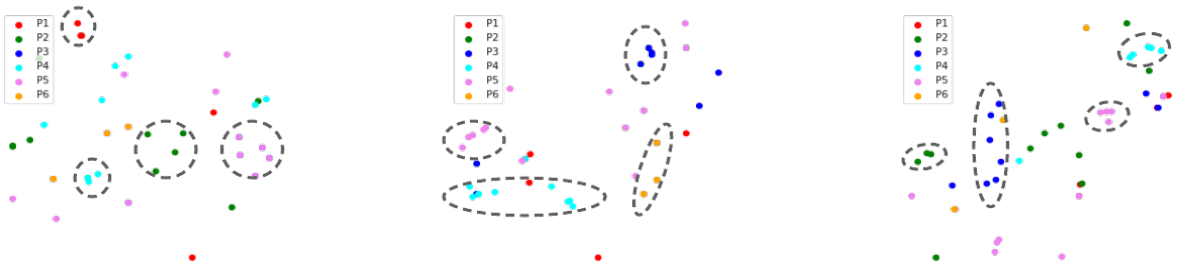

Figure 3: Two-dimensional t-SNE plots of the intra-process recurrent sensor embeddings from our FuSAGNet trained on SWaT. Each color represents a process (P1, P2, ...) within the system. Points of the same process forming clearly visible clusters are enclosed by dashed circles.

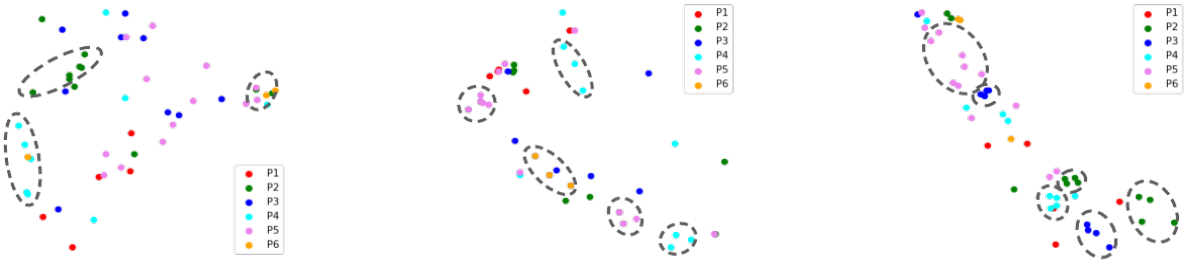

Figure 4: Two-dimensional t-SNE plots of the sparse latent representations extracted by our FuSAGNet trained on SWaT. Each color represents a process (P1, P2, ...) within the system. Points of the same process forming clearly visible clusters are enclosed by dashed circles.

4.5 Research Question 3: Interpretability

Intra-Process Recurrent Sensor Embedding. We further investigate the interpretability of intra-process recurrent sensor embedding through visualization using t-distributed Stochastic Neighbor Embedding (t-SNE) [34]. Since our recurrent sensor embedding modules are applied individually to groups of sensors within different processes, we are interested in whether this is reflected on the t-SNE space through, for example, clustering of those groups; this would imply that similarity between intra-process sensors is effectively characterized [10]. Fig. 3 presents three t-SNE plots (obtained through different trials for exemplification) of the intra-process recurrent sensor embeddings from our FuSAGNet trained on SWaT. In all three plots, we can observe that the embedded points form clusters with other ones pertaining to the same process. This qualitative examination indicates that our intra-process recurrent sensor embedding modules can effectively learn different sensor behaviors while considering the system environment (e.g., distinct functional roles of processes in a CPS).

Sparse Latent Representations. Adequate sparse representation can enhance data interpretability, in addition to model generalizability by revealing the latent structure of the input data [19, 21, 37]. To empirically confirm that the latter statement holds for our case as well, we again conduct visual analysis using t-SNE. Fig. 4 shows three t-SNE plots (obtained through different trials for exemplification) of the sparse latent representations extracted by our FuSAGNet trained on SWaT. Analogously to intra-process recurrent sensor embeddings projected on the t-SNE space, we can observe that sensors, especially those in processes 4 and 5, constantly form clusters in all three plots; this suggests that these sensors may exhibit particularly useful latent features for learning data patterns from the originally noisy sensor readings.

5 CONCLUSION

In this work, we proposed Fused Sparse Autoencoder and Graph Net (FuSAGNet), a time series anomaly detection framework that jointly optimizes reconstruction by Sparse Autoencoder and forecasting by Graph Neural Network to effectively capture interdependencies in high-dimensional sensor data with few anomalies. FuSAGNet outperformed baseline approaches in terms of anomaly detection performance on three real-world cyber-physical system datasets, enhanced the robustness of the graph-based forecasting model, and provided interpretability regarding both inter-sensor relationships and sparse latent representations.

ACKNOWLEDGMENTS

We thank Le Minh Binh for his help in reviewing the draft and producing figures. This work was partially supported by the Basic Science Research Program through National Research Foundation of Korea (NRF) grant funded by the Korean Ministry of Science and ICT (MSIT) under No. 2020R1C1C1006004 and Institute for Information & communication Technology Planning & evaluation (IITP) grants funded by the Korean MSIT: No. 2022-0-01199 (Graduate School of Convergence Security at Sungkyunkwan University), No. 2022-0-01045 (Self-directed Multi-Modal Intelligence for solving unknown, open domain problems), No. 2022-0-00688 (AI Platform to Fully Adapt and Reflect Privacy-Policy Changes), No. 2021-0-02068 (Artificial Intelligence Innovation Hub), and No. 2019-0-00421 (AI Graduate School Support Program at Sungkyunkwan University).

REFERENCES

[1] Charu C Aggarwal. 2015. Outlier analysis. In *Data mining*. Springer, 237–263.
[2] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. 2017. 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*. 25–28.
[3] Jinwon An and Sungzoon Cho. 2015. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE* 2, 1 (2015), 1–18.
[4] Fabrizio Angiulli and Clara Pizzuti. 2002. Fast outlier detection in high dimensional spaces. In *European conference on principles of data mining and knowledge discovery*. Springer, 15–27.
[5] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. 2020. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3395–3404.
[6] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. 2021. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478* (2021).
[7] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.
[8] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 1–58.
[9] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).
[10] Ailin Deng and Bryan Hooi. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada*. 2–9.
[11] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).
[12] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. 2016. A dataset to support research in the design of secure water treatment systems. In *International conference on critical information infrastructures security*. Springer, 88–99.
[13] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661* (2014).
[14] Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6, 02 (1998), 107–116.
[15] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 387–395.
[16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[18] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.
[19] Honglak Lee, Chaitanya Ekanadham, and Andrew Ng. 2007. Sparse deep belief net model for visual area V2. *Advances in neural information processing systems* 20 (2007), 873–880.
[20] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. 2019. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*. Springer, 703–716.
[21] Feng Li, JM Zuraday, and Wei Wu. 2018. SPARSE REPRESENTATION LEARNING OF DATA BY AUTOENCODERS WITH $L_{1/2}$ REGULARIZATION. *Neural Network World* 28, 2 (2018), 133–147.
[22] Andrew Ng et al. 2011. Sparse autoencoder. *CS294A Lecture notes* 72, 2011 (2011), 1–19.
[23] Keiron O'Shea and Ryan Nash. 2015. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* (2015).
[24] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. 2018. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters* 3, 3 (2018), 1544–1551.
[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).
[26] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. 2014. A review of novelty detection. *Signal Processing* 99 (2014), 215–249.
[27] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. *Learning internal representations by error propagation.* Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science.
[28] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13, 7 (2001), 1443–1471.
[29] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing* 45, 11 (1997), 2673–2681.
[30] Hyeok-Ki Shin, Woomyo Lee, Jeong-Han Yun, and HyoungChun Kim. 2020. {HAI} 1.0: HIL-based Augmented {ICS} Security Dataset. In *13th {USENIX} Workshop on Cyber Security Experimentation and Test ({CSET} 20)*.
[31] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. 2003. *A novel anomaly detection scheme based on principal component classifier.* Technical Report. MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING.
[32] Lu-An Tang, Jiawei Han, and Guofei Jiang. 2014. Mining sensor data in cyber-physical systems. *Tsinghua Science and Technology* 19, 3 (2014), 225–234.
[33] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. 2020. Transferring robustness for graph neural network against poisoning attacks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 600–608.
[34] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).
[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[37] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. 2008. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence* 31, 2 (2008), 210–227.
[38] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. 2020. Multivariate time-series anomaly detection via graph attention network. *arXiv preprint arXiv:2009.02040* (2020).
[39] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.
[40] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*.

A APPENDIX

A.1 Experimental Setup

Our proposed FuSAGNet is implemented in PyTorch [25] version 1.7.1 and PyTorch Geometric [11] version 1.5.0 with CUDA version 11.0. We conduct our experiments on a server with Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz and 4 TITAN RTX GPUs.

We train our model with the Adam optimizer [16] at a learning rate of 0.0005 for 30 epochs with early stopping patience of 6 epochs. Our training and validation sets are split into 8:2 ratio from data collected under normal operating conditions, whereas our test set includes all data collected under simulated attacks. Embedding dimensions for the intra-process recurrent sensor embedding as well as hidden layer sizes for the graph-based forecasting model are 16, 32, and 64 for SWaT, HAI, and WADI, respectively. The number of nodes considered for the adjacency matrices (Equation 3) are set to $k = 10$, 15, and 20 for SWaT, HAI, and WADI, respectively. Our SAE consists of a single hidden layer with size equal to that of the input/output space, that is, $N \times w$; the sparsity constraint, ρ, is predefined to be a number close but not equal to 0, e.g., 0.0001, and $\beta = 1.0$ for penalizing the reconstruction loss (Equation 5). We set the sliding window size to be $w = 16$ with batch size $B = 64$ and add a denoising factor to each batch of input to encourage robustness by injecting Gaussian noise. We evaluate the anomaly detection performance based on precision, $Pr = \frac{TP}{TP+FP}$, recall, $Re = \frac{TP}{TP+FN}$, and f1 score, $F1 = \frac{2 \cdot Pr \cdot Re}{Pr+Re}$, where TP, FP, and FN respectively denote true positive, false positive, and false negative.

A.2 Datasets

Table 3: Summary of dataset statistics for SWaT, WADI, and HAI.

Dataset	#Process	#Feature	#Train	#Test	%Anomaly
SWaT	6	51	49,500	44,992	12%
WADI	4	127	120,961	17,281	6%
HAI	4	79	92,161	4,321	2%

Dataset statistics are summarized in Table 3. Comprising six processes, SWaT contains data collected from a scaled-down industrial water treatment plant that has been continuously operated for 11 days, of which 7 days were under normal operating conditions while the remaining 4 days were under simulated attack scenarios. WADI is a more comprehensive dataset representing water distribution systems, extended from SWaT. Comprising four processes, it contains data collected under 16 days of continuous operations, of which 14 days were under normal operating conditions, while the remaining 2 days were under simulated attacks. Comprising four processes, HAI contains data collected from a testbed with a number of physical control systems, combined by an HIL simulator, that has been continuously operated for 15.5 days, of which 10 days were under normal operating conditions while the remaining 5.5 days were under simulated attacks.

We conduct experiments on SWaT, WADI, and HAI to evaluate the anomaly detection performance of our proposed FuSAGNet against competitive baseline approaches. As a common preprocessing step, we down-sample the original datasets to one in every 10 seconds worth of data by taking the median and assign an anomaly label ('0' or '1') by taking the mode [10]. Also, the first 6 hours' worth of data from SWaT and WADI are removed because they happen to be collected while the systems were reaching stabilization following initial operation [12]. Considering that all three datasets have been originally sampled at equal rates (i.e., 1Hz), their sampling rates after the same down-sampling process are equal as well (i.e., 0.1Hz). As an additional preprocessing step, we feature-wise normalize the data within the range $[0, 1]$ to facilitate training. This simple preprocessing step is particularly useful in our framework where SAE learns inactive or active latent space activations that are respectively close to 0 or 1, as discussed in Section 3.4.