Graph Neural Networks With Trainable Adjacency Matrices for Fault Diagnosis on Multivariate Sensor Data

ALEKSANDR KOVALENKO\(^1\), VITALIY POZDNYAKOV\(^1\), AND ILYA MAKAROV\(^1\), (Member, IEEE)

\(^1\)AIRI, 121170 Moscow, Russia
\(^2\)ISP RAS Research Center for Trusted Artificial Intelligence, 109004 Moscow, Russia
\(^3\)Faculty of Computer Science, HSE University, 109028 Moscow, Russia

Corresponding author: Aleksandr Kovalenko (kovalenko@airi.net)

This work was supported by a grant for research centers in the field of artificial intelligence, provided by the Analytical Center in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002) and the agreement with the Ivannikov Institute for System Programming of the Russian Academy of Sciences dated November 2, 2021 No. 70-2021-00142.

ABSTRACT

Timely detection and accurate diagnosis of faults in technological processes can significantly reduce production costs in manufacturing. Modern industrial equipment, equipped with numerous sensors, generates vast amounts of data, providing opportunities for advanced fault detection and diagnosis. While convolutional and recurrent neural networks have achieved state-of-the-art performance, they often overlook the correlations and hidden relationships among sensor signals. To address this, we propose a graph neural network (GNN) architecture that constructs graphs of sensor relationships from data. We evaluated five methods for training different types of adjacency matrices allowing to set certain restrictions on the structure of the graph. The resulting graph structures were analyzed and potential for their use in transfer learning was evaluated. Additionally, we developed an architecture that uses multiple adjacency matrices, which reduces the number of trainable parameters while maintaining high prediction quality. Our models demonstrated state-of-the-art performance on the Tennessee Eastman Process dataset, showcasing their potential for fault diagnosis on multivariate sensor data.

INDEX TERMS

Adjacency matrix, fault diagnosis, graph neural networks, sensor data analysis, Tennessee Eastman process.

I. INTRODUCTION

During production, equipment often stops due to various faults. Finding the root cause of a fault can take a significant amount of time. Deviations in equipment parameters can lead to defective products, resulting in lost time and wasted raw materials. All these factors lead to financial losses for companies. To reduce production costs, equipment must be maintained in good condition, and any deviations in parameters must be corrected as soon as possible. Sometimes the causes of faults are completely non-obvious and have hidden dependencies, creating additional diagnostic difficulties even for highly qualified experts. For decades, scientists and specialists have been developing methods to detect faulty equipment conditions and determine the types of faults. In the literature, these problems are usually referred to as fault detection and diagnosis (FDD) \([1], [2], [3]\).

Recent scientific discoveries and developments in electronics and computer science provide new opportunities in many areas of our life, including industry \([4]\). New types of automation sensors are now available, and computing capacity has increased, enabling the practical application of machine learning methods \([5]\). The concept of Industry 4.0 focuses on interconnectivity, automation, machine learning, and real-time data in manufacturing \([6]\). Modern and modernized production equipment consists of hundreds of sensors, whose data can be used to increase the quality and productivity of production. By applying Artificial Intelligence techniques
various patterns corresponding to certain events in the production process, including faults can be extracted.

Multivariate sensor data are usually presented in the form of time series. Machine learning approaches and classical statistical [7] FDD methods work only with numerical values of time series and do not take into account the information about possible correlations between them. Deep learning techniques such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have achieved state-of-the-art results in FDD applications, leveraging their ability to extract features from temporal and spatial data. However, these methods also treat sensor data independently, failing to capture the intricate correlations and dependencies among different sensors.

Graph neural networks (GNNs) offer a powerful alternative by representing sensor data as graphs, where nodes correspond to sensors and edges represent relationships between them. This approach allows for a more holistic analysis of the data, capturing both local and global patterns. Despite their potential, the application of GNNs in FDD remains relatively unexplored. Over the past couple of years, several papers have been published on the use of GNNs in fault diagnosis problems [8], [9], [10]. In these papers, the adjacency matrices are known in advance or obtained by different methods before the start of the GNN training process.

In our study, we propose a novel GNN architecture specifically designed for FDD in multivariate sensor data. The peculiarity is that the adjacency matrices describing the graph of relationships between sensor signals are obtained using trainable parameters. We explored various ways to obtain such matrices, allowing us to set certain restrictions on the structure of the graph. We compared the resulting graph structure with a process diagram describing the training dataset. Additionally, we evaluate the possibility of using resulted adjacency matrices for transfer learning. Finally, a new GNN architecture was proposed that uses multiple adjacency matrices simultaneously.

Our key contributions are as follows:

- We introduce a method to construct graphs from sensor data, enabling the modeling of relationships between sensors.
- We explore five different methods for training adjacency matrices that define the graph structure.
- We propose an architecture using multiple adjacency matrices, which reduces the number of trainable parameters while maintaining high prediction accuracy.
- We demonstrate the effectiveness of our approach on the Tennessee Eastman Process (TEP) dataset, achieving state-of-the-art performance in fault diagnosis.

The remainder of this paper is organized as follows: Section II reviews related work in FDD and GNN applications. Section III details our proposed GNN architectures. Section IV presents experimental setup, followed by experimental results in Section V. Finally, Section VI concludes the paper and outlines future research directions.

II. RELATED WORKS

Fault detection and diagnosis methods significantly reduce production costs and improve quality and productivity. Traditional FDD methods, including statistical and machine learning techniques, have been extensively explored. Recently, the rapid development of graph neural networks has opened new ways for modeling complex relationships in multivariate sensor data. This section reviews key developments in FDD methods and the application of GNNs to fault diagnosis, highlighting the evolution and current state of research in this field.

A. FAULT DETECTION AND DIAGNOSIS

Fault detection involves identifying that equipment or processes are in an abnormal state, while fault diagnosis determines the root cause of the fault (Fig. 1). Various FDD techniques have been developed over the decades [11]. These methods can be categorized into data-driven [12], [13], [14], model-based [15], [16], [17], [18] and knowledge-based groups [19], [20], [21]. The latter two require expert knowledge and understanding of the technological process, which does not allow to create unique FDD tools applicable to various industries. At the same time, data-driven approaches depend on the analytical models used and on the quality of historical data and could be scaled to different production systems.

Numerous statistical and machine learning methods have been proposed for data-driven FDD. Dimensional reduction methods, such as principal component analysis (PCA), t-SNE, canonical variate analysis (CVA), partial least squares (PLS), represent high-dimensional sensor data in low-dimensional feature space making FDD tractable in cases of a large number of sensors [22], [23], [24]. Hotelling’s T², squared prediction error (SPE), Kullback-Leibler divergence, and other statistics are calculated in feature space in order to detect faults in a process. Ji and Sun reviewed statistical methods of root cause diagnosis [25], that are divided into three groups: contribution plot-based methods [26], probability reasoning-based methods [27] and causal reasoning-based methods [28]. Clustering methods [29], e.g. K-means and DBSCAN, are used as unsupervised FDD to separate sensor data into distinct groups, each of them represents some state of a process [30]. Such clusters can be manually labeled by experts for configuring a process monitoring system based on a clustering algorithm. On the other hand, supervised FDD...
methods, such as random forest [31], support vector machine (SVM) [32], k-nearest neighbors (KNN) [33], are trained to detect faults using labeled sensor data. Usually supervised methods are more accurate than unsupervised methods, but require manual labeling of each data point, which can be difficult and expensive in real industrial cases [34], [35].

The great advances in deep learning over the past decade have affected many areas, including the field of fault detection and diagnosis. Convolutional neural networks (CNN) have become widely used in FDD tasks [36], [37], [38], [39]. Lomov et al. [40] investigated the application of various architectures of recurrent and convolutional neural networks for fault diagnosis in a chemical process. Deep learning approaches show high efficiency but require a large amount of training data.

A graph can be represented by adjacency matrix \( A \in \mathbb{R}^{N \times N} \) where \( A_{ij} = c > 0 \) if \( (u_i, u_j) \in E \) and \( A_{ij} = 0 \) if \( (u_i, u_j) \notin E \) (Fig. 2).

For a long time, signed directed graphs (SDG) have been widely used for failure path analysis, however, they require expert knowledge of equipment and processes [41]. In contrast, data-driven methods rely solely on historical data. An example is representing relationships between sensors by clustering sensors into the groups and finding the correlation between them [42]. Another way to represent relationships between sensors is an adjacency matrix. Graph neural networks model the dependencies between nodes on historical data using predefined adjacency matrix. The first concept of GNN was proposed in 2009 [43]. It can be considered as a generalization of convolutional neural networks. GNNs began to develop rapidly [44], [45], [46] following the publication of Kipf and Welling [47], who described graph convolutional networks (GCNs) from a mathematical perspective.

The main idea behind GCNs is the aggregation of information in a node from its neighbors. Kipf and Welling [47] proposed the following layer-wise propagation rule:

\[
H^{(l+1)} = \alpha (\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} H^{(l)}) W^{(l)}
\]

where \( \tilde{A} = A + I_N \) is the adjacency matrix with added self-connections, \( \tilde{D} \) is the diagonal matrix with elements \( \tilde{D}_{ii} = \sum_j \tilde{A}_{ij} \) and \( W^{(l)} \) is the trainable parameter matrix. \( H^{(l)} \) is the output matrix of the previous layer; \( H^{(0)} = X \).

GCNs have emerged as a powerful approach for learning representations on graph-structured data. NAGCN introduced a novel neighborhood adaptive kernel for graph convolutional networks, enhancing node classification by efficiently learning and integrating relevant neighborhood information [48]. Additionally, SGLNet presented an innovative spiking neural network model combining adaptive graph convolution and LSTM, demonstrating superior performance in EEG signal classification for brain-computer interfaces [49].

Several papers on multivariate time series forecasting [50], [51], [52] using spectral temporal graph neural networks and spatial temporal graph neural networks have been published in recent years. These works solve graph regression problems with benchmark datasets such as traffic, electricity consumption, solar energy generation, and exchange rates. Adjacency matrices were obtained by trainable parameters during the models training process.

**B. GRAPH NEURAL NETWORKS**

Multivariate sensor data can be represented as graphs, where nodes correspond to sensors, and edges represent dependencies between them. The most commonly used mathematical notation for a graph is \( G = (V, E) \), where \( V \) is a set of nodes and \( E \) is a set of edges. The number of nodes is denoted by \( N = |V| \). The neighborhood of a node \( u \in V \) is defined as \( \mathcal{N}(u) = \{ v \in V | (u, v) \in E \} \), where \( (u, v) \) is denoted as an edge between \( u \) and \( v \). The structure of a graph can be represented by adjacency matrix \( A = (A_{ij}) \) with \( A_{ij} = c > 0 \) if \( (u_i, u_j) \in E \) and \( A_{ij} = 0 \) if \( (u_i, u_j) \notin E \) (Fig. 2).

For a long time, signed directed graphs (SDG) have been widely used for failure path analysis, however, they require expert knowledge of equipment and processes [41]. In contrast, data-driven methods rely solely on historical data. An example is representing relationships between sensors by clustering sensors into the groups and finding the correlation between them [42]. Another way to represent relationships between sensors is an adjacency matrix. Graph neural networks model the dependencies between nodes on historical data using predefined adjacency matrix. The first concept of GNN was proposed in 2009 [43]. It can be considered as a generalization of convolutional neural networks. GNNs began to develop rapidly [44], [45], [46] following the publication of Kipf and Welling [47], who described graph convolutional networks (GCNs) from a mathematical perspective.

The main idea behind GCNs is the aggregation of information in a node from its neighbors. Kipf and Welling [47] proposed the following layer-wise propagation rule:

\[
H^{(l+1)} = \alpha (\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} H^{(l)}) W^{(l)}
\]

where \( \tilde{A} = A + I_N \) is the adjacency matrix with added self-connections, \( \tilde{D} \) is the diagonal matrix with elements \( \tilde{D}_{ii} = \sum_j \tilde{A}_{ij} \) and \( W^{(l)} \) is the trainable parameter matrix. \( H^{(l)} \) is the output matrix of the previous layer; \( H^{(0)} = X \).

GCNs have emerged as a powerful approach for learning representations on graph-structured data. NAGCN introduced a novel neighborhood adaptive kernel for graph convolutional networks, enhancing node classification by efficiently learning and integrating relevant neighborhood information [48]. Additionally, SGLNet presented an innovative spiking neural network model combining adaptive graph convolution and LSTM, demonstrating superior performance in EEG signal classification for brain-computer interfaces [49].

Several papers on multivariate time series forecasting [50], [51], [52] using spectral temporal graph neural networks and spatial temporal graph neural networks have been published in recent years. These works solve graph regression problems with benchmark datasets such as traffic, electricity consumption, solar energy generation, and exchange rates. Adjacency matrices were obtained by trainable parameters during the models training process.

**C. GRAPH NEURAL NETWORKS FOR FAULT DIAGNOSIS**

The diagram of an example of applying GNN to multivariate sensor data is shown in Fig. 3. Production equipment is represented as a graph described by an adjacency matrix. An adjacency matrix is fed into the GCN input together with the feature matrix \( X^{m \times n} \), where \( m \) is the window size of the time series and \( n \) is the number of sensors. The output of GCN is the type of fault.

Zhang et al. [53] proposed graph attention networks (GATs) for imbalanced industrial fault diagnosis. The authors
obtained the adjacency matrix by computing the cosine similarity between the embeddings of different sensor nodes. Then a top-K selection strategy was applied to retain the most significant interactions. Chen et al. [8] made a review on graph neural network-based fault diagnosis. Four GNN architectures have been designed for fault multi-class classification task: graph convolutional network, GAT, graph sample and aggregate (GraphSage) and spatial temporal graph convolutional network (STGCN). The adjacency matrices were obtained by KNN and KNN + Graph autoencoder (GAE) methods. KNN approach was applied to the similarity of time-frequency features. GNN-based methods showed better performance than the six baseline machine learning approaches. Li et al. [9] proposed a practical guideline and a GNN-based FDD framework. In this framework, two types of graph construction methods for multivariate time series where discovered: KNN-based and Radius-based. Liu and Jafarpour [54] constructed a causal map in the form of an adjacency matrix using conditional Granger causality analysis. Gao et al. [55] constructed the adjacency matrix by embedding physical knowledge of sensor relationships into the graph. This approach incorporated the intrinsic physical dependencies among sensors, enhancing the model’s ability to capture relevant features for fault diagnosis.

All these methods were applied to the training dataset, and the resulting adjacency matrix was then used as input for the GNN. Qing et al. [56] proposed to construct initial graph by KNN method and then modify it by trainable weights. In our study, the adjacency matrix initially consists of random trainable parameters that are optimized along with the entire model.

III. MODEL ARCHITECTURES
This section describes the GNN architectures employed in our experiments. In Subsection A, we present our general architecture designed to solve the FDD task. It consists of two main components: the graph structure learning (GSL) block and the GNN block. One of the outcomes of training such a model is the adjacency matrix obtained from the output of the GSL block. To evaluate how well this adjacency matrix describes the data, we applied it in a transfer learning context with a simpler graph neural network described in Subsection B. Subsection C describes an architecture that uses multiple adjacency matrices and the motivation for its design. The code of our models is available at https://github.com/AIRI-Institute/gnn-tam.

A. GENERAL ARCHITECTURE
Our general architecture consists of GSL and GNN blocks (Fig. 4). We use the GSL block to represent our data as a graph structure. This graph structure, in the form of an adjacency matrix, is then used for graph convolutions in the GNN block. The structure of both parts and the model training process are described in more detail below.

1) GSL BLOCK
GNNs operate on data that can be represented as a graph structure. However, this structure is not always known in advance. In our study, we employed a GSL block to derive it from multivariate sensor data. Moreover, there are different types of graphs describing different types of relationships between nodes. In the context of industrial equipment, two nodes in the form of signals can have directed, uni-directed and undirected connections. Which type of graph is best suited to describe a specific dataset is not always obvious. Therefore, we explored five different options for GSL block that allow obtaining specified types of connections between graph nodes.

The operation of the GSL block is based on randomly initialized parameters that are trained simultaneously with the other parameters of the architecture. The output of the block is a weighted adjacency matrix. These weights not only signify connections between nodes but also capture the strength of these relationships. This approach enables a more accurate approximation of technological processes and equipment operation.

Motivated by [52], [57], [58], and [59], we describe four options for the GSL block:
• ReLU(W). The idea is that the adjacency matrix is a parameter matrix that contains $N^2$ parameters. This method does not impose any restrictions on the type of connections between the nodes of the graph.

$$A = \text{ReLU}(W)$$  \hspace{1cm} (2)

• Uni-directed $A$. This approach assumes that the relations between the graph nodes are unidirectional. There cannot be more than one directed edge between two nodes.

$$M_1 = \tanh(\alpha E_1 \Theta_1)$$  \hspace{1cm} (3)

$$M_2 = \tanh(\alpha E_2 \Theta_2)$$  \hspace{1cm} (4)

$$A = \text{ReLU}(\tanh(\alpha(M_1 M_2^T - M_2 M_1^T)))$$  \hspace{1cm} (5)

$E_1$ in (3) and $E_2$ in (4) are randomly initialized trainable node embeddings. $\Theta_1$ in (3) and $\Theta_2$ in (4) are model parameters and $\alpha$ is a hyperparameter to control the saturation rate of the activation function. Small values of $\alpha$ allow us to adjust the order of the weights, while large values bring the weights closer to 1. Expression $(M_1 M_2^T - M_2 M_1^T)$ in (5) gives asymmetric properties to the resulting adjacency matrix.

• Undirected $A$. The undirected graph assumes that the edges between its nodes do not have direction.

$$A = \text{ReLU}(\tanh(\alpha(M_1 M_1^T)))$$  \hspace{1cm} (6)

• Directed $A$. The directed graph assumes that all its edges have their own directions.

$$A = \text{ReLU}(\tanh(\alpha(M_1 M_2^T)))$$  \hspace{1cm} (7)

Furthermore we propose to use hyperbolic tangent as activation function similar to ReLU(W). The performance of this option for GSL was also assessed by Kim et al. [60].

• Tanh(W). The adjacency matrix with both positive and negative weights showed promising results during the experiments.

$$A = \tanh(\alpha W)$$  \hspace{1cm} (8)

One approach to managing the complexity of the adjacency matrix is to restrict the maximum number of edges per node, thereby increasing sparsity. This is beneficial when the matrix contains excessive connections or when only strong dependencies between nodes are of interest. To achieve this, a strategy is proposed: for each node, retain only the top $k$ edges with the highest weights, setting the weights of remaining edges to zero.

2) GNN BLOCK

To compare different methods of obtaining an adjacency matrix, we utilized a graph neural network featuring two GCN layers. The network includes two read-out layers that generate a graph representation by applying a $\text{min}$ function to node representations. Batch normalization was incorporated to enhance the efficiency of the $\text{min}$ function. Both read-out layers are connected by skip connection. The result of the addition is passed through a fully connected layer to determine whether the graph represents a normal state or one of several fault classes.

3) TRAINING AND OPTIMIZING THE MODEL PARAMETERS

The GSL block is responsible for learning the graph structure from the data, while the GNN block performs graph convolutions using the learned adjacency matrix. The training process for our model involves optimizing the parameters of both the GSL and GNN blocks simultaneously by minimizing the common loss function (Algorithm 1). This joint optimization ensures that the learned graph structure is well-suited for the downstream task performed by the GNN block. Optimal hyperparameters such as $\alpha$ for the GSL block and the number of hidden parameters in the GCN layers of the GNN block can be found by grid search.

**Algorithm 1 Training of the General Model**

**Require:** Training dataset $D$, model architecture with GSL and GNN blocks, learning rate $\eta$, batch size $b$, number of epochs $E$

**Ensure:** Trained model parameters $\Theta = \{\Theta_{GSL}, \Theta_{GNN}\}$

1: Initialize model parameters $\Theta$ randomly
2: for epoch $e = 1$ to $E$ do
3: Shuffle the training dataset $D$
4: for each batch $B \in D$ of size $b$ do
5: Forward pass through GSL block: Compute adjacency matrix $A = GSL(\Theta_{GSL})$
6: Forward pass through GNN block: Compute predictions $\hat{y} = \text{GNN}(x, A; \Theta_{GNN})$ for $x \in B$
7: Compute loss $L(\hat{y}, y)$ for true labels $y \in B$
8: Backward pass: Compute gradients $\nabla_{\Theta} L$
9: Update parameters: $\Theta \leftarrow \Theta - \eta \nabla_{\Theta} L$
10: end for
11: Optionally, evaluate model on validation set and adjust $\eta$ or other hyperparameters
12: end for
13: return Trained model parameters $\Theta$

After training, the resulting adjacency matrix can be saved and evaluated for its fidelity in capturing relationships within the source data. For instance, it can be compared against a process diagram or utilized in transfer learning scenarios involving different architectural models.

**B. SIMPLIFIED ARCHITECTURE FOR TRANSFER LEARNING EVALUATION**

To assess the potential of using trainable adjacency matrices in transfer learning, we designed a simplified GNN architecture (Fig. 5). This architecture features reduced parameter complexity in the GCN layers and accepts shorter signal sequences as input. The adjacency matrices derived from training our general GNN architecture are utilized as predefined matrices in this model. We then compare the
results with those obtained using a baseline matrix generated by correlation.

The training process of the simplified model is similar to that of our general architecture, except that there is no GSL block, and the adjacency matrix is already calculated.

**FIGURE 5.** Simplified model architecture with two GCN layers and a predefined adjacency matrix.

**C. ARCHITECTURE WITH MULTIPLE ADJACENCY MATRICES**

The technological process can be complex and dynamically changing. The relationships between signals in such equipment cannot be described by a single static graph. We proposed an approach in which the model learns several graph structures, which yielded some positive results in our experiments.

To create a model with multiple adjacency matrices, we propose an architecture that includes several graph neural network modules trained in parallel (Fig. 6). Each GNN module, based on our general GNN architecture (Subsection IIIA), operates independently. The outputs from these modules are concatenated and passed to a fully connected output layer. We used Tanh(W) option for the GSL block. The training process for this architecture is described in Algorithm 2.

**Algorithm 2 Training of the Model With Multiple Adjacency Matrices**

**Require:** Training dataset $D$, model architectures with GSL and GNN blocks, learning rate $\eta$, batch size $b$, number of epochs $E$

**Ensure:** Trained model parameters $\Theta = \{\theta^{(l)}_{GSL}, \theta^{(l)}_{GNN}, \theta^{(k)}_{GSL}, \theta^{(k)}_{GNN}, \theta_{FC}\}$

1: Initialize model parameters $\Theta$ randomly
2: for epoch $e = 1$ to $E$ do
3:   Shuffle the training dataset $D$
4:   for each batch $B \in D$ of size $b$ do
5:     Initialize list of hidden states $h_{list} = []$
6:     for each model $m = 1$ to $k$ do
7:       Forward pass through GSL block: Compute adjacency matrix $A^{(m)} = GSL(\theta^{(m)}_{GSL})$
8:       Forward pass through GNN block: Compute hidden states $h^{(m)}(x) = GNN(x, A^{(m)}; \theta^{(m)}_{GNN})$ for $x \in B$
9:     Append hidden states to list: $h_{list} \leftarrow h_{list} \cup \{h^{(m)}\}$
10:    end for
11:   Concatenate hidden states: $h_{concat} = concat(h_{list})$
12:   Forward pass through fully connected layer: Compute final predictions $\hat{y} = FC(h_{concat}; \theta_{FC})$
13:   Compute loss $L(\hat{y}, y)$ for true labels $y \in B$
14:   Backward pass: Compute gradients $\nabla_{\Theta} L$
15:   Update parameters: $\Theta \leftarrow \Theta - \eta \nabla_{\Theta} L$
16: end for
17: Optionally, evaluate model on validation set and adjust $\eta$ or other hyperparameters
18: end for
19: return Trained model parameters $\Theta$

**IV. EXPERIMENTAL SETUP**

This section outlines the setup of our experiments. It includes details about the dataset used and the data preprocessing steps. Additionally, the training procedures for our models and the evaluation metrics employed are discussed. Finally, we list the baselines utilized.

**A. DATASET DESCRIPTION**

We validated our models using the Tennessee Eastman Process dataset, which serves as a benchmark for FDD
A. Kovalenko et al.: Graph Neural Networks With Trainable Adjacency Matrices

FIGURE 7. Additionally marked Tennessee Eastman Process diagram (L. Ma, J. Dong and K. Peng [61], 2019).

approaches in this study. TEP was created by the Eastman Chemical Company of Tennessee in 1993 [62] and became a widely used benchmark in the academic community. The industrial plant depicted in the TEP dataset consists of five main units: reactor, condenser, stripper, compressor, and separator. The process schema is illustrated in Fig. 7.

Eight components \( A, B, C, D, E, F, G, \) and \( H \) are involved in the chemical process with the following relationships (g – gaseous, liq – liquid):

\[
\begin{align*}
A(g) + C(g) + D(g) & \rightarrow G(\text{liq}) & (9) \\
A(g) + C(g) + E(g) & \rightarrow H(\text{liq}) & (10) \\
A(g) + E(g) & \rightarrow F(\text{liq}) & (11) \\
3D(g) & \rightarrow 2F(\text{liq}) & (12)
\end{align*}
\]

Gaseous reactants \( A, C, D \) and \( E \) are involved in the formation of liquid products \( G \) in (9) and \( H \) in (10). \( B \) is an inactive component that does not participate in the chemical reaction and \( F \) in (11) is a byproduct of the process. Gaseous components enter the reactor where the main chemical reaction takes place. Heat released during the reaction is removed by means of a built-in cooling unit. The resulting product and reagent residues enter the condenser for cooling and condensation. Noncondensed components are returned to the reactor through the separator. Inert gases and byproducts are removed from the system as vapor. After the separator, the condensed components enter the stripper, where they are removed from excess reagents and left in the form of products \( G \) and \( H \).

Each multivariate time series simulation includes 41 measured variables and 11 control variables, recorded from multiple sensors at 3-minute intervals. The original TEP dataset did not provide a sufficient number of time series examples for training deep neural networks. Therefore, we utilized the extended TEP dataset [63], which includes 100 simulations for each of the 28 fault types (Table 1). Each simulation comprises 2000 data points, with fault states appearing after the 600th point.

**B. DATA PREPARATION**

The initial TEP dataset was normalized using the standard score formula. We employed a sliding window technique to extract samples from the time series data. Each sample consists of a fixed number of consecutive time steps, capturing the temporal dependencies among sensor readings. Each window is labeled based on the fault type it corresponds to: if any part of the window falls within a fault condition, the entire window is labeled accordingly. The dataset is split into 80% for training and 20% for testing.

**C. MODELS TRAINING**

All models were trained on an Nvidia RTX A4000 GPU to accelerate the computational process. We used the Adam optimizer with a learning rate of 0.001 and a batch size of 512 to balance memory usage and training efficiency. Cross-entropy was employed as a loss function for all architectures.
D. EVALUATION METRICS

During evaluation, the model receives a sample as input and predicts one of the process states. Comparison with labeled test data can result in four main outcomes:

- True positive (TP): The faulty process state is correctly diagnosed.
- False Positive (FP): A normal process state is diagnosed as faulty.
- True Negative (TN): A normal process state is correctly diagnosed.
- False Negative (FN): A faulty process state is diagnosed as normal.

The following metrics, crucial in FDD tasks, were used to assess model performance:

- **True Positive Rate (TPR)**: Probability that the faulty state is diagnosed correctly:
  \[
  TPR = \frac{TP}{TP + FN} \tag{13}
  \]

- **False Positive Rate (FPR)**: Probability that the normal state is detected as a fault:
  \[
  FPR = \frac{FP}{FP + TN} \tag{14}
  \]

- **Average Detection Delay (ADD)**: Average time delay between the moment when the process state changes (from normal to faulty condition) and when the model detects this change.

We used two types of tables for model comparison. The first table includes TPRs and FPRs calculated separately for each fault type. The second table includes detection TPR, detection FPR, and ADD. Detection TPR/FPR represents the probability that the faulty state is detected correctly/incorrectly, irrespective of correct fault classification.

E. BASELINES

To evaluate our approach, we compared it against three baseline methods for obtaining the adjacency matrix: correlation, KNN, and attention. Specifically, a Pearson correlation matrix was constructed based on the dataset, and values below 0.3 were filtered out. In the KNN method, euclidean distances were used, and 18% of the nearest neighbors (edges) were included in the adjacency matrix. The attention mechanism dynamically computes an adjacency matrix based on pairwise attention scores derived from node features, allowing the graph structure to adapt to the input data.

To evaluate the competitiveness of GNN-based models, we compared them against several baseline models: MLP, 1DCNN [64], and GRU [40] architectures. Previous research [40] has demonstrated that a network utilizing Gated Recurrent Units (GRU) can achieve the best results in supervised FDD tasks using the TEP dataset. We selected the GRU architecture that referred to “GRU: type 2” in that study.

V. RESULTS AND DISCUSSION

Our experiments proceeded as follows: first, we trained the general GNN architecture and baselines, comparing their metrics. We saved and analyzed the adjacency matrices obtained after training, then utilized them in transfer learning with a simplified GNN model. Finally, we proposed an architecture using multiple adjacency matrices and evaluated its performance.

A. FAULT DIAGNOSIS WITH OUR GENERAL GNN ARCHITECTURE

The proposed GNN-based model was trained using five variants of the GSL block. We determined that the hyper-parameter $\alpha$ provided optimal performance when set to 0.1. The number of hidden parameters in GCN layers was equal to 1024. Sliding window size was fixed at 100. All models were trained for 40 epochs with 3 training iterations per model, averaging the metric values.

Table 2 presents the results including Detection TPR, Detection FPR, Average Detection Delay (ADD), and the number of trainable parameters for each model. The GNN + Tanh(W) model achieved the highest Detection TPR scores. Interestingly, all models employing GSL blocks demonstrated TPR/FPR scores comparable to or better than those of the GRU model. Moreover, significant improvements were observed in the crucial ADD metric. This experiment underscores the competitiveness of graph neural networks in FDD tasks, showing superior performance on the TEP dataset. Detailed results for each fault type are provided in Table 3.

The models employing GSL block outperformed the baseline using the correlation matrix. This suggests...
that our trained adjacency matrices can contain more information about the relationships between signals than just linear correlation. To test this hypothesis, we analyzed the resulting graph structures in the next section by comparing them with the process diagram. Additionally, we explored the application of trained adjacency matrices in transfer learning with different model architectures.

Various approaches to training adjacency matrices enable us to impose structural constraints such as undirected, directed, and unidirectional edges. Future research could explore integrating domain expert knowledge into these models. Masks could designate which types of edges can connect specific nodes, tailored to the characteristics of the technological equipment. Another potential avenue for research involves feature selection using GNN architectures.

We did not limit the number of edges coming from nodes in the models listed in the results table. However, during parameter selection, we observed that restricting nodes to 5 or 3 outgoing edges didn’t significantly reduce model accuracy. It would be interesting to examine the information content of the data if we removed signals corresponding to nodes that have no incoming or outgoing edges, or if these edges have very small weights.

### TABLE 2. Values of detection TPR/FPR, and ADD for each model. Best results are in bold.

| Model                  | Detection TPR | Detection FPR | ADD  | Num. of trainable par. |
|------------------------|---------------|---------------|------|------------------------|
| Ours                   | 0.9265        | 0.0317        | 20.67| 11135661               |
| GNN + ReLU(W)          |               |               |      |                        |
| GNN + Uni-directed A   | 0.9189        | 0.0113        | 31.27| 1213557                |
| GNN + Undirected A     | 0.9262        | 0.0130        | 29.53| 1198257                |
| GNN + Directed A       | 0.9301        | 0.0249        | 21.08| 1213557                |
| GNN + Tanh(W)          | 0.9346        | 0.0406        | 18.95| 11135661               |
| Baselines              |               |               |      |                        |
| GNN + Correlation      | 0.8882        | 0.0064        | 46.86| 11135661               |
| GNN + KNN              | 0.8901        | 0.0206        | 33.04| 11135661               |
| GNN + Attention        | 0.9226        | 0.0234        | 25.30| 11135661               |
| MLP                    | 0.8895        | 0.0116        | 28.06| 2688189                |
| 1DCNN                  | 0.8967        | 0.0151        | 25.07| 226941                 |
| GRU                    | 0.9247        | 0.0387        | 35.46| 384669                 |

### FIGURE 8. Relationships between sensors received by the GSL block.

by comparing them with the process diagram. Additionally, we explored the application of trained adjacency matrices in transfer learning with different model architectures.

Various approaches to training adjacency matrices enable us to impose structural constraints such as undirected, directed, and unidirectional edges. Future research could explore integrating domain expert knowledge into these models. Masks could designate which types of edges can connect specific nodes, tailored to the characteristics of the technological equipment. Another potential avenue for research involves feature selection using GNN architectures.

We did not limit the number of edges coming from nodes in the models listed in the results table. However, during parameter selection, we observed that restricting nodes to 5 or 3 outgoing edges didn’t significantly reduce model accuracy. It would be interesting to examine the information content of the data if we removed signals corresponding to nodes that have no incoming or outgoing edges, or if these edges have very small weights.
B. ANALYSIS OF TRAINED ADJACENCY MATRICES

When comparing the obtained adjacency matrices with the technological process diagram, we observed both correspondences and discrepancies. Differences may arise because hidden dependencies are not always visible in the diagram. However, logical connections can still be identified. For instance, during our experiments, we trained a model allowing a maximum of 3 outgoing edges per node, with a directed approach chosen for the GSL block. Fig. 8 illustrates outgoing edges from nodes 51 and 52 as examples. Sensor 51 monitors coolant flow rate in the reactor, impacting the temperature measured by sensor 9 during chemical reactions. Temperature changes influence pressure (sensor 7) within closed volumes. Moreover, the reactor and stripper tanks are interconnected, affecting pressure readings (sensor 16). Similarly, sensor 52, which measures coolant flow in the condenser, impacts sensors 7 and 9 in the reactor. Post-condenser, the substance enters the separator, where coolant temperature (sensor 22) is measured. These edges identified by the model are logical, demonstrating the utility of the GSL block in uncovering dependencies between equipment components.

However, the adjacency matrices obtained at different iterations may differ. The previous model with limited edges was trained iteratively 5 times. All obtained adjacency matrices have common features (some common nodes have large weights, some less). Similarities are shown in Fig. 9. The rows contain the sums of edge weights belonging to the numbered sensors/nodes. All values were scaled by min-max normalization. It can be seen from the picture that all adjacency matrices have similar node importance, indicating that some of the sensors have a greater influence on the entire system, and some less. This also confirms that the GSL block does indeed learn the interaction structure in the workflow. The resulting adjacency matrices are not similar to the adjacency matrix obtained by the simple correlation method. Therefore, they can display hidden dependencies between sensors. Such data can be very useful for engineers working with this equipment. In further research, methods for stabilizing the results at different iterations of model training can be studied.

C. TRANSFER LEARNING WITH TRAINED ADJACENCY MATRICES

To generate an adjacency matrix, the parameters of the GSL block are trained alongside those of the other layers, adapting to their values. The adjacency matrix must accurately reflect the dependencies between nodes to ensure that subsequent GCN layers can correctly process the incoming data. However, during backpropagation, the trainable weights of the adjacency matrix are influenced by the weights of the GCN layers. To assess this interaction, we propose a simplified model with a different structure and parameters, employing a sliding window size of 10 (see Fig. 5). The results presented in Table 4 demonstrate that the trained adjacency matrices outperform the predefined matrix in terms of true positive rate (TPR) and false positive rate (FPR) scores. This suggests that, despite the influence of GCN layer weights, the trained adjacency matrices effectively capture data dependencies.

These findings indicate that trained adjacency matrices could be viable for transfer learning. The simplified GNN model, for example, includes 4,557 trainable parameters. For the ReLU(W) option of the GSL block, the number of trainable parameters scales with the square of $N$, where $N$ represents the number of sensors (52 in our case). Adding a ReLU(W) GSL block would increase the trainable parameters to 7,261 for 52 sensors. For equipment with 100 or 1,000 sensors, this would result in training an additional 10,000 or 1,000,000 parameters, respectively. While this is feasible for compute servers, it could present limitations when training on embedded devices, an area warranting further research.

D. GNN ARCHITECTURE WITH MULTIPLE ADJACENCY MATRICES FOR FAULT DIAGNOSIS

The variations observed in the obtained adjacency matrices led to the idea that there cannot be a single ideal adjacency matrix. The processes within such complex technological
systems are dynamic and can evolve over time. Depending on the stage of the process, the relationships between sensors may fluctuate. This led us to explore the concept of learning multiple adjacency matrices in parallel.

We tested a variant with 10 GNN modules, where the number of parameters in GCN layers was reduced from 1024 to 64. As a result, the total number of trainable parameters in this modified architecture significantly decreased compared to the original model (153,949 versus 1,185,661 parameters). The new model was trained for 35 epochs, and the results are summarized in Tables 5 and 7. Despite having fewer parameters to learn, the proposed architecture achieved similar performance on the dataset used.

In the subsequent stage of the experiment, we further reduced the training dataset to only 10% of simulations for each fault type. Each model was iteratively trained 10 times and the metrics were averaged. The results are shown in Tables 6 and 8. On the reduced dataset, the proposed architecture is superior in more fault types than on the full dataset (Tables 7 and 8).

This part of the experiment demonstrates that the architecture with multiple parallel GNN modules, despite having significantly fewer parameters, excels in identifying data dependencies, particularly on the reduced TEP dataset. Future research could explore whether such an architecture offers advantages when data availability is limited. Additionally, investigating the feasibility of transferring this architecture to embedded devices with constrained computing resources would be beneficial. These directions could further enhance the applicability and efficiency of GNN-based models in practical settings.

VI. CONCLUSION

In this paper, we proposed a novel approach to fault detection and diagnosis (FDD) in industrial processes using Graph Neural Networks (GNNs) with trainable adjacency matrices. The proposed method was tested on a fault diagnosis task, demonstrating its capability to outperform traditional approaches in identifying data dependencies, particularly on a reduced dataset.

The architecture was designed to be scalable by training multiple GNN modules in parallel. This approach allowed for a reduction in the number of parameters, which in turn resulted in a decrease in computational cost without sacrificing performance. The experimental results showed that the proposed method achieved comparable or better performance on both full and reduced datasets, highlighting its potential for practical applications in industrial settings.

Future research could consider further optimization of the architecture, such as exploring different types of data dependencies or applying the method to a wider range of industrial processes. Additionally, investigating the transferability of the model to embedded devices with limited computational resources could be beneficial. Overall, the proposed approach opens up new possibilities for fault diagnosis in dynamic systems, offering a robust and efficient solution for industrial applications.
Neural Networks (GNNs) with trainable adjacency matrices. This approach addresses the limitations of traditional FDD methods and existing deep learning techniques by effectively modeling the relationships between sensors, capturing both local and global dependencies. We explored different methods to train adjacency matrices and proposed an architecture utilizing multiple adjacency matrices, which reduces the number of trainable parameters while maintaining high prediction accuracy.

We evaluated our method on the Tennessee Eastman Process (TEP) dataset, demonstrating state-of-the-art performance. Our results show that GNNs with trainable adjacency matrices improve fault diagnosis accuracy compared to existing methods. This approach provides a robust framework for leveraging the rich, multivariate sensor data generated by modern industrial equipment.

Future research will focus on validating the proposed approach with real-world industrial datasets to assess its practical applicability and robustness across diverse scenarios. Additionally, we aim to optimize the computational efficiency of our models to facilitate real-time fault detection and diagnosis on embedded devices.

REFERENCES

[1] D. Divya, B. Marath, and M. B. S. Kumar, “Review of fault detection techniques for predictive maintenance,” J. Quality Maintenance Eng., vol. 29, no. 2, pp. 420–441, Apr. 2023.
[2] N. Trapani and L. Longo, “Fault detection and diagnosis methods for sensors systems: A scientific literature review,” IFAC-PapersOnLine, vol. 56, no. 2, pp. 1253–1263, 2023.
[3] V. Pozdnyakov, A. Kovalenko, I. Makarov, M. Drobyshevskiy, and K. Lukyanov, “Adversarial attacks and defenses in fault detection and diagnosis: A comprehensive benchmark on the Tennessee Eastman Process,” IEEE Open J. Ind. Electron., vol. 5, pp. 428–440, 2024.
[4] L. Maretto, M. Faccio, and D. Battini, “The adoption of digital technologies in the manufacturing world and their evaluation: A systematic review of real-life case studies and future research agenda,” J. Manuf. Syst., vol. 68, pp. 576–600, Jun. 2023.
[5] D. Zhevenko, M. Kazantsev, and I. Makarov, “Interaction models for remaining useful lifetime estimation,” J. Ind. Inf. Integ., vol. 33, Jun. 2023, Art. no. 100444.
[6] K. Schwab, The Fourth Industrial Revolution. New York, NY: Crown, 2017.
[7] Y. Zhu, S. Zhao, Y. Zhang, C. Zhang, and J. Wu, “A review of statistical-based fault detection and diagnosis with probabilistic models,” Symmetry, vol. 16, no. 4, p. 455, Apr. 2024.
[8] Z. Chen, J. Xu, C. Alippi, S. X. Ding, Y. Shardt, T. Peng, and C. Yang, “Graph neural network-based fault diagnosis: A review,” 2021, arXiv:2111.08185.
[9] T. Li, Z. Zhou, S. Li, C. Sun, R. Yan, and X. Chen, “The emerging graph neural networks for intelligent fault diagnostics and prognostics: A guideline and a benchmark study,” Mech. Syst. Signal Process., vol. 168, Apr. 2022, Art. no. 108653.
[10] M. Jia, J. Hu, Y. Liu, Z. Gao, and Y. Yao, “Topology-guided graph learning for process fault diagnosis,” Ind. Eng. Chem. Res., vol. 62, no. 7, pp. 3238–3248, Feb. 2023.
[11] Y.-J. Park, S.-K.-S. Fan, and C.-Y. Hsu, “A review on fault detection and process diagnostics in industrial processes,” Processes, vol. 8, no. 9, p. 1123, Sep. 2020.
[12] S. J. Qin, “Data-driven fault detection and diagnosis for complex industrial processes,” IFAC Proc. Volumes, vol. 42, no. 8, pp. 1115–1125, 2009.
[13] D. Gonzalez-Jimenez, J. del-Olmo, J. Poza, F. Garamiola, and P. Madina, “Data-driven fault diagnosis for electric drives: A review,” Sensors, vol. 21, no. 12, p. 4024, Jun. 2021.
[14] A. Lundgren and D. Jung, “Data-driven fault diagnosis analysis and open-set classification of time-series data,” Control Eng. Pract., vol. 121, Apr. 2022, Art. no. 105006.
[15] S. X. Ding, Modelling of Technical Systems. London, U.K.: Springer, 2013.
[16] P. M. Frank, “Principles of model-based fault detection,” IFAC Proc. Volumes, vol. 25, no. 10, pp. 213–220, Jun. 1992.
[17] A. Kuestenmacher and P. G. Pflüger, “Model-based fault diagnosis techniques for mobile robots,” IFAC-PapersOnLine, vol. 49, no. 15, pp. 50–56, 2016.
[18] D. Asadi, “Model-based fault detection and identification of a quadrotor with rotor fault,” Int. J. Aeronaut. Space Sci., vol. 23, no. 5, pp. 916–928, Nov. 2022.
[19] C. Nan, F. Khan, and M. T. Iqbal, “Real-time fault diagnosis using knowledge-based expert system,” Process Saf. Environ. Protection, vol. 86, no. 1, pp. 55–71, Jan. 2008.
[20] H. Verbruggen, S. Tzafestas, and E. Zanni, “Knowledge-based fault diagnosis of technological systems,” in Artificial Intelligence in Industrial Decision Making. Dordrecht, The Netherlands: Springer, 1995.
[21] W. Li, H. Li, S. Gu, and T. Chen, “Process fault diagnosis model with knowledge-based approaches: Advances and opportunities,” Control Eng. Pract., vol. 105, Dec. 2020, Art. no. 104637.
[22] E. L. Russell, L. H. Chiang, and R. D. Braatz, “Principal component analysis,” in Data-Driven Methods for Fault Detection and Diagnosis in Chemical Processes. London, U.K.: Springer, 2000, pp. 33–52.
[23] E. L. Russell, L. H. Chiang, and R. D. Braatz, “Partial least squares,” in Data-Driven Methods for Fault Detection and Diagnosis in Chemical Processes. London, U.K.: Springer, 2000, pp. 67–80.
[24] E. L. Russell, L. H. Chiang, and R. D. Braatz, “Canonical variate analysis,” in Data-Driven Methods for Fault Detection and Diagnosis in Chemical Processes. London, U.K.: Springer, 2000, pp. 81–95.
[25] C. Ji and W. Sun, “A review on data-driven process monitoring methods: Characterization and mining of industrial data,” Processes, vol. 10, no. 2, p. 335, Feb. 2022.
[26] S. J. Qin, “Survey on data-driven industrial process monitoring and diagnosis,” Annu. Rev. Control, vol. 36, no. 2, pp. 220–234, Dec. 2012.
[27] B. Cai, L. Huang, and M. Xie, “Bayesian networks in fault diagnosis,” IEEE Trans. Ind. Inform., vol. 13, no. 5, pp. 2227–2240, Oct. 2017.
[28] H.-S. Chen, C. Zhao, Z. Yan, and Y. Yao, “Root cause diagnosis of oscillation-type plant faults using nonlinear causality analysis,” IFAC-PapersOnLine, vol. 50, no. 1, pp. 13898–13903, Jul. 2017.
[29] M. Zhao and J. Liu, “Adaptive graph regularized low–rank matrix factorization with noise and outliers for clustering,” IEEE Access, vol. 8, pp. 171851–171863, 2020.
[30] S. Zheng and J. Zhao, “A new unsupervised data mining method based on the stacked autoencoder for chemical process fault diagnosis,” Comput. Chem. Eng., vol. 135, Apr. 2020, Art. no. 106755.
[31] Y. Liu and Z. Ge, “Weighted random forests for fault classification in industrial processes with hierarchical clustering model selection,” J. Process Control, vol. 64, pp. 62–70, Apr. 2018.
[32] T. Sukchotrat, S. B. Kim, and F. Tsung, “One-class classification-based control charts for multivariate process monitoring,” IEE Trans., vol. 42, no. 2, pp. 107–120, Nov. 2009.
[33] D. Ha, U. Ahmed, H. Pyun, C.-J. Lee, K. H. Baek, and C. Han, “Multi-mode operation of principal component analysis with K-nearest neighbor algorithm to monitor compressors for liquefied natural gas mixed refrigerant processes,” Comput. Chem. Eng., vol. 106, pp. 96–105, Nov. 2017.
[34] H. Tao, P. Wang, Y. Chen, V. Stojanovic, and H. Yang, “An unsupervised fault diagnosis method for rolling bearing using STFT and generative neural networks,” J. Franklin Inst., vol. 357, no. 11, pp. 7286–7307, Jul. 2020.
[35] M. Golyadkin, V. Pozdnyakov, L. Zhukov, and I. Makarov, “SensorScan: Self-supervised learning and deep clustering for fault diagnosis in chemical processes,” Artif. Intell., vol. 324, Nov. 2023, Art. no. 104012.
[36] H. Wu and J. Zhao, “Deep convolutional neural network model based chemical process fault diagnosis,” Comput. Chem. Eng., vol. 115, pp. 185–197, Jul. 2018.
[37] X. Gao, F. Yang, and E. Feng, “A process fault diagnosis method using multi-time scale dynamic feature extraction based on convolutional neural network,” Can. J. Chem. Eng., vol. 98, no. 6, pp. 1280–1292, 2020.
[38] M. M. Azad, P. Kumar, and H. S. Kim, “Delamination detection in CFRP laminates using deep transfer learning with limited experimental data,” J. Mater. Res. Technol., vol. 29, pp. 3024–3035, Mar. 2024.

[39] M. M. Azad and H. S. Kim, “Hybrid deep convolutional networks for the autonomous damage diagnosis of laminated composite structures,” Comput. Struct., vol. 329, Feb. 2024, Art. no. 117792.

[40] I. Lomov, M. Lyubimov, I. Makarov, and L. E. Zhukov, “Fault detection in Tennessee Eastman Process with temporal deep learning models,” J. Ind. Inf. Integr., vol. 23, Sep. 2021, Art. no. 100216.

[41] C. C. Chang and C. C. Yu, “On-line fault diagnosis using the signed directed graph,” Ind. Eng. Chem. Res., vol. 29, no. 7, pp. 1290–1299, Jul. 1990.

[42] Q.-X. Zhu, X.-W. Wang, K. Li, Y. Xu, and Y.-L. He, “Enhanced multicorrelation block process monitoring and abnormally root cause analysis for distributed industrial process: A visual data-driven approach,” J. Process Control, vol. 118, pp. 1–15, Oct. 2022.

[43] F. Scarcelli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” IEEE Trans. Neural Netw., vol. 20, no. 1, pp. 61–80, Jan. 2009.

[44] I. Makarov, O. Gerasimova, P. Sulimov, and L. E. Zhukov, “Co-authorship network embedding and recommending collaborators via network embedding.,” in Proc. 7th Int. Conf. Anal. Images, Social Netw. Tests (AIST), Cham, Switzerland: Springer, Jul. 2018, pp. 32–38.

[45] I. Makarov and O. Gerasimova, “Link prediction regression for weighted co-authorship networks,” in Proc. 15th Int. Work-Conf. Artif. Neural Netw. (IWANN), Cham, Switzerland: Springer, Jul. 2019, pp. 667–677.

[46] K. Tikhomirova and I. Makarov, “Community detection based on the nodes role in a network: The telegram platform case,” in Proc. 9th Int. Conf. Anal. Images, Social Netw. Tests (AIST), Moscow, Russia, Cham, Switzerland: Springer, Oct. 2020, pp. 294–302.

[47] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, arXiv:1609.02907.

[48] P. Gong and L. Ai, “Neighborhood adaptive graph convolutional network for node classification,” IEEE Access, vol. 7, pp. 170578–170588, 2019.

[49] P. Gong, P. Wang, Y. Zhou, and D. Zhang, “A spiking neural network with adaptive graph convolution and LSTM for EEG-based brain-computer interfaces,” IEEE Trans. Neural Syst. Rehabil. Eng., vol. 31, pp. 1440–1450, 2023.

[50] D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu, C. Huang, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, “Spectral temporal graph neural network for multivariate time-series forecasting,” in Proc. Adv. Neural Inf. Process. Syst., vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds. Glasgow, U.K.: Curran Associates, 2020, pp. 17766–17778.

[51] Y. Cui, K. Zheng, D. Cui, J. Xie, L. Deng, F. Huang, and X. Zhou, “METRO: A generic neural network framework for multivariate time series forecasting,” Proc. VLDB Endowment, vol. 15, no. 2, pp. 224–236, Oct. 2021.

[52] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, “Connecting the dots: Multivariate time series forecasting with graph neural networks,” in Proc. 26th ACM SIGKDD Conf. Knowl. Discovery Data Mining, 2020, pp. 753–763.

[53] C. Zhang, Y. Li, L. Fan, L. Shen, and T. Qin, “Embedding-enhanced graph attention networks for imbalanced industrial fault diagnosis,” IEEE Access, early access, Apr. 22, 2024, doi: 10.1109/ACCESS.2024.3392422.

[54] Y. Liu and B. Jafarpour, “Graph attention network with Granger causality map for fault detection and root cause diagnosis,” Comput. Chem. Eng., vol. 180, Jan. 2024, Art. no. 108453.

[55] Y. Guo, Z. Zhong, M. Ma, Z. Zhang, Y. Zhang, C. Wang, and Z. Wang, “Physics-embedded recurrent graph neural network for fault diagnosis of complex systems,” IEEE Access, vol. 12, pp. 122426–122436, 2024.

[56] H. Qing, N. Zhang, Y. He, Y. Xu, and Q. Zhu, “IC points weight learning-based GCN and improving feature distribution for industrial fault diagnosis,” Expert Syst. Appl., vol. 255, Dec. 2024, Art. no. 124681.

[57] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Graph WaveNet for deep spatial–temporal graph modeling.,” in Proc. 26th Int. Joint Conf. Artif. Intell., Aug. 2019, pp. 1907–1913.

[58] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, “Attention based spatial–temporal graph convolutional networks for traffic flow forecasting,” in Proc. AAAI Conf. Artif. Intell., vol. 33, 2019, p. 922.

[59] L. Shi, Y. Zhang, J. Cheng, and H. Lu, “Two-stream adaptive graph convolutional networks for skeleton-based action recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 12018–12027.

[60] G. Kim, S. Kang, G. Park, and B.-C. Min, “Electric vehicle battery state of charge prediction based on graph convolutional network,” Int. J. Automot. Technol., vol. 24, no. 6, pp. 1519–1530, Dec. 2023.

[61] L. Ma, J. Dong, and K. Peng, “A novel key performance indicator oriented hierarchical monitoring and propagation path identification framework for complex industrial processes,” ISA Trans., vol. 96, pp. 1–13, Jan. 2020.

[62] J. J. Downs and E. F. Vogel, “A plant-wide industrial process control problem,” Comput. Chem. Eng., vol. 17, no. 3, pp. 245–255, Mar. 1993.

[63] C. Reinartz, M. Kulahci, and O. Ravin, “An extended Tennessee Eastman simulation dataset for fault-detection and decision support systems,” Comput. Chem. Eng., vol. 149, Jun. 2021, Art. no. 107281.

[64] S. Huang, J. Tang, J. Dai, Y. Wang, and J. Dong, “1DCNN fault diagnosis based on cubic spline interpolation pooling,” Shock Vibrat., vol. 2020, pp. 1–13, Feb. 2020.

A. Kovalenko et al.: Graph Neural Networks With Trainable Adjacency Matrices

ALEKSAKD KOVALENO received the specialist degree in microelectronics from Saint Petersburg Electrotechnical University “LETI,” Saint Petersburg, Russia, in 2010, and the master’s degree in data science from HSE University, Moscow, Russia, in 2022.

Since 2012, he has been with industrial equipment, as an Electronics Service Engineer. He is currently a Junior Research Scientist with the Artificial Intelligence Research Institute, Moscow, Russia. His research interests include digital twins and deep learning for sensor data analysis.

VITALY POZDNYAKOV received the master’s degree in data science from HSE University, Moscow, Russia, in 2021.

Since 2021, he has been a Junior Research Scientist with the Laboratory of Methods for Big Data Analysis, HSE University, and the ISP RAS Research Center for Trusted Artificial Intelligence. He is currently a Junior Research Scientist with AIRI, Moscow, and a Visiting Lecturer with HSE University. His research interests include probabilistic forecasting, fault diagnosis in large-scale industrial processes, and defense methods of fault diagnosis systems.

ILYA MAKAROV (Member, IEEE) received the specialist degree in mathematics from Lomonosov Moscow State University, Moscow, Russia, in 2011, and the Ph.D. degree in computer science from the University of Ljubljana, Ljubljana, Slovenia, in 2021.

Since 2011, he has been a Lecturer with the School of Data Analysis and Artificial Intelligence, HSE University, Moscow, where he was the School Deputy Head, from 2012 to 2016, and is an Associate Professor and a Senior Research Fellow. He was also the Program Director of the BigData Academy MADE from VK and a Researcher with the Samsung-PDMI Joint AI Center, St. Petersburg Department of V.A., Steklov Mathematical Institute, Russian Academy of Sciences, Saint Petersburg, Russia. He is currently a Senior Research Fellow with the Artificial Intelligence Research Institute (AIRI), Moscow, where he leads the research in industrial AI. He became the Head of the AI Center and Data Science Tech Master Program in NLP, National University of Science and Technology MISIS, Moscow.