National Research University Higher School of Economics
Faculty of Computer Science
Program 'Master of Data Science'

MASTER'S THESIS

Industrial ML: graph neural networks for fault diagnosis in multivariate sensor data

Student: Alexander Kovalenko

Supervisor: PhD, Associate Professor, Ilya Makarov

Moscow, 2022

Abstract

Timely detected deviations in the technological process, as well as the earliest detection of the cause of the fault, significantly reduce the costs of modern production in industry. Production equipment consists of parts interacting with each other and can be represented as a graph. Graph nodes can be represented as data from the different sensors, and edges can display the influence of these data on each other. In this work, the possibility of applying graph neural networks to the problem of fault classification in a chemical process is investigated. For a neural network based on the interaction between the graph nodes, an adjacency matrix is required. Such data are usually not predefined, therefore, several methods have been proposed for obtaining adjacency matrices during the training process. It was also found that one adjacency matrix cannot describe the dynamic production process. Equipment can operate in different modes and can have different interactions between its parts. A new method was proposed based on parallel learning of several adjacency matrices.

Contents

1 Introduction

One of the main criteria for success in the industry is the smooth operation of equipment and the absence of defective products. However, equipment often stops due to a malfunction and finding its cause and eliminating it takes a significant amount of time. In the technological process, parameters can deviate, affecting the result and leading to a defective product. All this leads to financial losses due to equipment downtime and damaged raw materials. Sometimes the causes of faults are completely non-obvious and have hidden dependencies that create 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, such problems are usually called fault detection and diagnosis (FDD).

The latest scientific discoveries and developments in the fields of electronics and computer science provide us with new opportunities in various areas of our life, including industry. New types of automation sensors have been available and computing capacity has increased, enabling the application of machine learning models in practice. The concept of Industry 4.0 focuses on interconnectivity, automation, machine learning, and real-time data in manufacturing. Modern and modernized production equipment consists of hundreds of sensors, data from which can be used to increase the quality and productivity of production lines.

The multivariate sensor data gives information about the status of various equipment components affecting the production process. Using this data together with machine learning approaches outperforms classical statistical FDD methods that have been used in the past decades. Production equipment consists of many devices that work together and deviations in the operation of one of them can affect the operation of the others. All these changes are also displayed in the signals from the sensors of these devices. The received signals have hidden dependencies and can correlate with each other. Such data can be represented as a graph where nodes are information from sensors and edges are connections describing their influence on each other.

Approaches based on graph neural networks (GNN) for fault diagnosis task are proposed in this work. This type of neural network shows good results when working with graph-structured data. To train a GNN, an adjacency matrix that describes the relationship between the nodes of the graph is needed. These data are very important, but not always known and can be of different quality. Over the past year, very few papers have been published on the use of GNN in fault diagnosis problems. In these papers, the adjacency matrices are known in advance or obtained by various methods before the start of GNN training process. In this work, options to obtain an adjacency matrix during the training process were proposed. Adjacency matrix can be created in various ways by using trainable parameters.

Not always the adjacency matrix can fully describe all possible relationships between graph nodes. Production equipment can have complex dynamic dependencies between its devices. Different modes of operation can be described by

different adjacency matrices. A novel idea was proposed to learn in parallel and use several adjacency matrices during the training and evaluation processes.

All GNN models were trained on the popular FDD benchmark to determine the type of fault. The results were compared with the baselines such as principal component analysis, multilayer perceptron (MLP), and 1d convolutional neural network (1DCNN).

2 Related works

Fault detection and diagnosis (FDD) methods provide great benefits in reducing production costs and improving quality and productivity. The data structures used by these methods can be represented as graph structures. In the same time, great development of graph neural networks in recent years may allow them to be used as approaches for fault detection and diagnosis problems. Recent papers in the fields of FDD and GNN are reviewed in the next two sections.

2.1 Fault detection and diagnosis

Fault detection is the process of identifying that an equipment or process is not in a normal state, and fault diagnosis is the process of determining the root case of the fault (Fig.1). Many different FDD techniques and approaches have been developed over the past decades [15]. These methods can be classified into data-driven, model based and knowledge-based groups. 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/processing equipment.

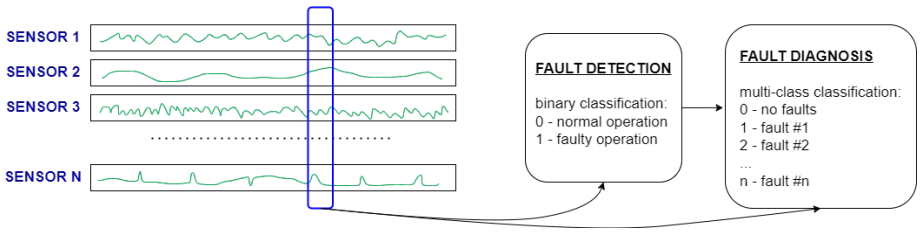

Figure 1: Fault diagnosis task can include the detection step or can be divided in two subsequent tasks.

To obtain more information about the fault, the faulty variable should be isolated. There are two main data-driven methods used to isolate the faulty variable, supervised and unsupervised. Supervised methods require labeled data

with fault information, whereas unsupervised methods are used when there is no such data. Classical statistical methods such as principal component analysis (PCA), partial least squares (PLS), independent component analysis (ICA), fisher discriminant analysis (FDA), subspace-aided approach (SAP) [22], and modern neural networks [21] are used for fault detection tasks.

Cheng Ji and Wei Sun reviewed classical and recent research on data-driven process monitoring methods from the perspective of "characterizing and mining industrial data" [10]. In this paper, three types of root cause diagnosis based on statistical analysis methods are described: contribution plot-based methods, probability reasoning-based methods, and causal reasoning-based methods. These approaches have both advantages and disadvantages, but the overall limitation of computational complexity increases with the number of variables in multivariate data.

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 [18, 7]. Ildar Lomov et al. [13] 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.

2.2 Graph neural networks

Multivariate sensor data can be represented in the form of graph, where the nodes are sensors and the edges are the relationships between them. The most commonly used mathematical notation of the 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 $v \in V$ is defined as $\mathcal{N} = \{u \in V | (v, u) \in E\}$, where (v, u) is denoted as an edge between v and u. The structure of a graph can be represented by adjacency matrix $A \in R^{N \times N}$ with $A_{ij} = c > 0$ if $(v_i, u_j) \in E$ and $A_{ij} = 0$ if $(v_i, u_j) \notin E$ (Fig. 2).

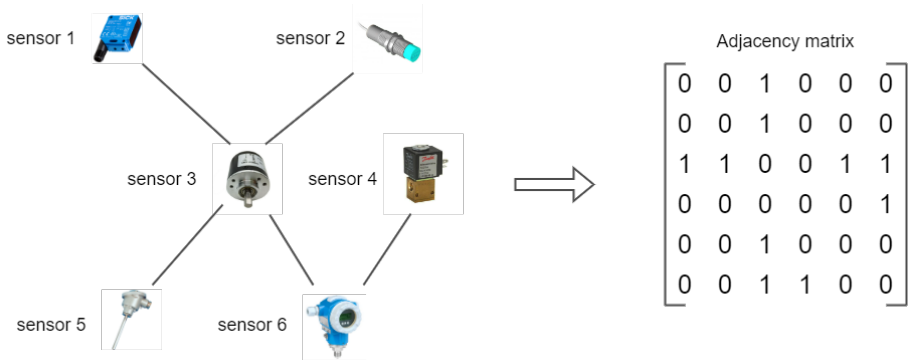

Figure 2: Relationships between devices in the form of an adjacency matrix.

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 [2]. Modern graph neural networks can be used to build models based only on historical data. The first concept of GNN was proposed in 2009 [6]. It can be considered as a generalization of convolutional neural networks. GNNs began to develop rapidly after the publication of Kipf [11], in which he described graph convolutional networks (GCN) from a mathematical point of view.

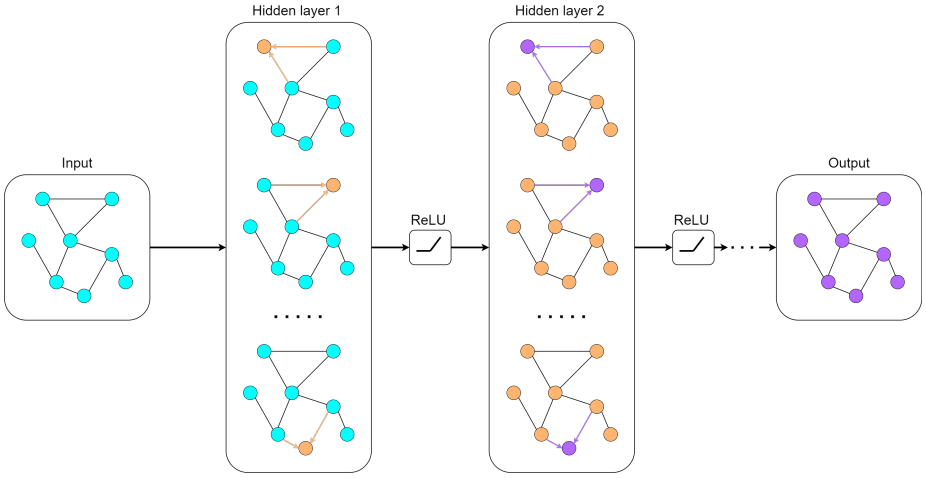

Figure 3: Standard GCN achitectue.

The main idea behind the GCN is the aggregation of information in a node from its neighbors, Fig.3. Kipf et al. [11] proposed the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

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

Graph neural networks can be used in three main task levels:

- Node level: node classification.
- Edge level: link prediction – finding of missing edges.
- Graph level: graph classification or regression.

Several papers on multivariate time series forecasting [1, 4, 19] using spectral temporal graph neural networks and spatial temporal graph neural networks have been published in the past few 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.

Zhiwen Chen et al. [3] 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, graph attention network (GAT), graph sample and aggregate (GraphSage) and spatial temporal graph convolutional network (STGCN). The adjacency matrices were obtained by k nearest neighbors (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. Tianfu Li et al. [12] 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.

3 Model description

The adjacency matrix must be known in order to apply graph neural networks. In multivariate time series fault diagnosis problems, such data are very rare. In some papers [3][12], authors use classical statistical and other modern methods (KNN-based, Radius-based, KNN + graph auto-encoder etc.) to build an adjacency matrix before applying it to GNN. All these methods are used on the training data and feed the result to the neural network before training starts. In this work, the so-called graph structure learning layer is discovered for FDD task. This implies that the adjacency matrix is obtained during the training of the GNN and can be adjusted when training on new data.

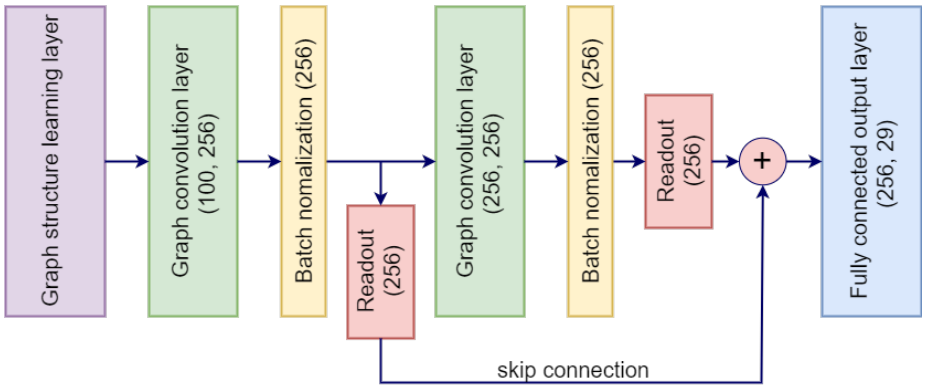

Figure 4: The model architecture with two GCN layers and graph structure learning module.

To compare different ways to obtain an adjacency matrix, the general architecture of a graph neural network with two GCN layers is used (Fig.4). There are two read-out layers that create a graph representation by applying a *min* function to node representations. Batch normalization was added to make the

min function more efficiently. Both read-out layers are connected by skip connection. The result of the addition is fed to a fully connected layer that indicates whether the graph belongs to the normal state or to one of the fault classes. Next, motivated by [19, 20, 8, 17] various options for the graph structure learning layer described.

- $ReLU(W)$. The idea is that the adjacency matrix is a parameter matrix that contains N^2 parameters.

$$A = ReLU(W)$$

  The result is a weighted adjacency matrix.

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

$$M_1 = \tanh(\alpha E_1 \Theta_1)$$

$$M_2 = \tanh(\alpha E_2 \Theta_2)$$

$$A = ReLU(\tanh(\alpha(M_1 M_2^T - M_2 M_1^T)))$$

  E_1 and E_2 are randomly initialized node embeddings that are learnable during training. Θ_1 and Θ_2 are model parameters, and α is a hyperparameter to control the saturation rate of the activation function. Small α values allow us to adjust the order of the weights, while large values bring the weights closer to 1. Expression $(M_1 M_2^T - M_2 M_1^T)$ gives asymmetric properties to the resulting adjacency matrix.

- $UndirectedA$. The undirected graph assumes that the edges between its nodes do not have direction.

$$A = ReLU(\tanh(\alpha(M_1 M_1^T)))$$

- $DirectedA$. The directed graph assumes that all its edges have their own directions.

$$A = ReLU(\tanh(\alpha(M_1 M_2^T)))$$

It is possible to limit the maximum number of edges in each node and make the adjacency matrix more sparse. This can be useful when the resulting adjacency matrix has too many connections, or when only edges with strong dependency between nodes need to be found. To remove edges with small weights from the graph, the following strategy is proposed.

$$\text{for } i = 1, 2, ..., N$$

$$\text{idx} = \text{argtopk}(A[i, :])$$

$$A[i, -idx] = 0$$

The function argtopk() returns the indexes of k highest edges weights for each node. These edges are stored in the adjacency matrix while the rest of the weights are set to zero.

4 Dataset description

The Tennessee Eastman Process (TEP) dataset is used to evaluate FDD approaches in this work. TEP was created by the Eastman Chemical Company of Tennessee in 1993 [5] and became a widely used benchmark in the academic community. This is a flowchart for an industrial plant consisting of five main units: reactor, condenser, stripper, compressor and separator. The process schema is shown in Fig.5.

Eight components A, B, C, D, E, F, G and H are involved in the chemical process with the following relationships:

$$A(g) + C(g) + D(g) \rightarrow G(liq)$$

$$A(g) + C(g) + E(g) \rightarrow H(liq)$$

$$A(g) + E(g) \rightarrow F(liq)$$

$$3D(g) \rightarrow 2F(liq)$$

Gaseous reactants A, C, D and E are involved in the formation of liquid products G and H. B is an inactive component that does not participate in the chemical reaction and F is a by-product 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. Non-condensed components are returned back to the reactor through the separator. Inert gases and byproducts are removed from the system as vapor. After

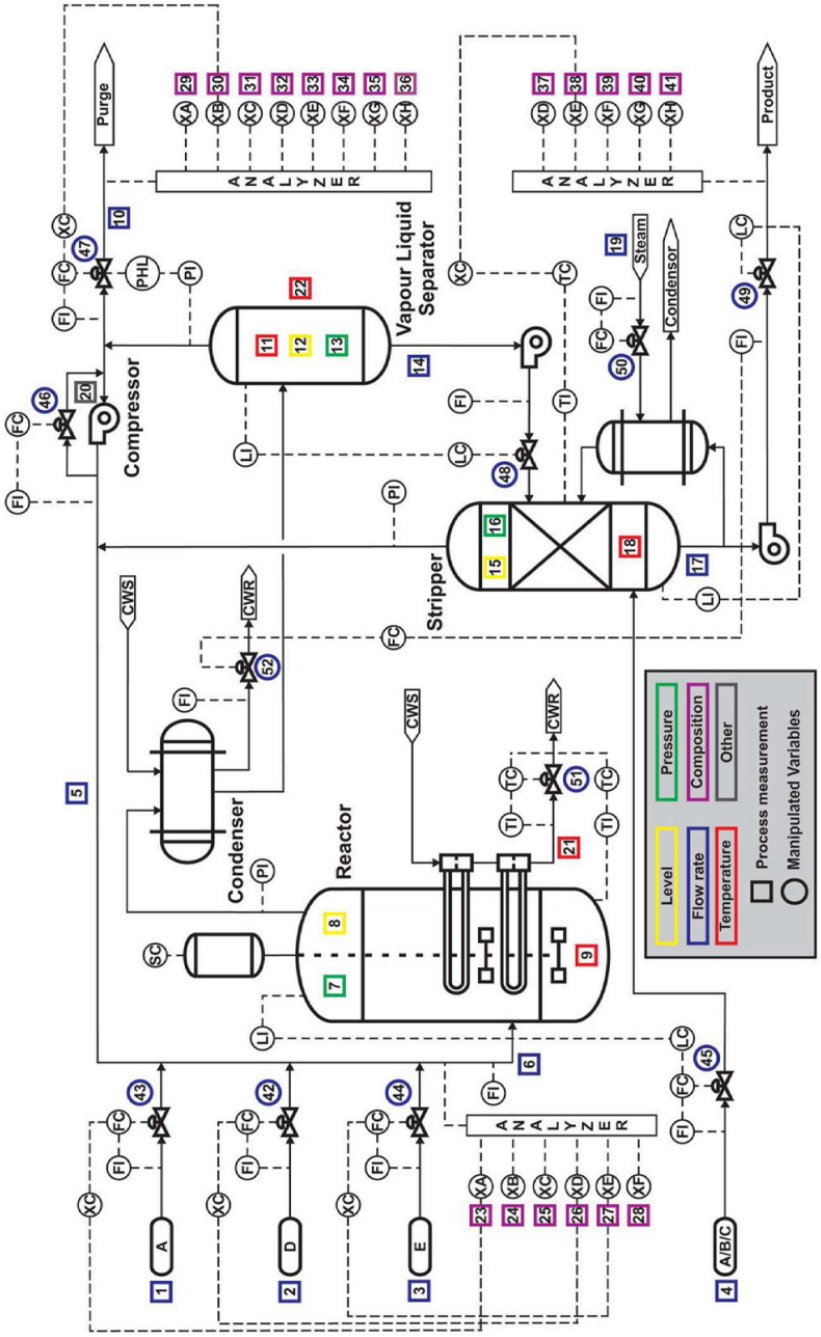

Figure 5: Additionally marked Tennessee Eastman Process diagram (L. Ma, J. Dong and K. Peng [14], 2019).

the separator, the condensed components enter the stripper, where they are cleaned of excess reagents and left in the form of products G and H.

The TEP dataset consists of multivariate time series simulations referring to either the normal state of the process or one of 28 fault types (Table 1). Each multivariate time series simulation has 41 measured variables and 11 control variables from different sensors recorded with 3-minute intervals.

Initial TEP dataset contains insufficient number of time series examples for training deep neural networks. The authors of [16] have extended it by generating more examples for each failure/normal state. The additional Tennessee Eastman Process dataset contains 500 data points for each training simulation and 960 data points for test cases. The fault states appear after 20 and 160 data points in the training and testing simulations, respectively.

Table 1: Description of the Normal/Fault states in TEP dataset.

State ID	Description	Type
0	Normal state	None
1	A/C feed ratio, B composition constant	Step
2	B composition, A/C ratio constant	Step
3	D feed temperature	Step
4	Reactor cooling water inlet temperature	Step
5	Condenser cooling water inlet temperature	Step
6	A feed loss	Step
7	C header pressure loss-reduced availability	Step
8	A, B, C feed composition	Random Variation
9	D feed temperature	Random Variation
10	C feed temperature	Random Variation
11	Reactor cooling water inlet temperature	Random Variation
12	Condenser cooling water inlet temperature	Random Variation
13	Reaction kinetics	Slow Drift
14	Reactor cooling water valve	Sticking
15	Condenser cooling water valve	Sticking
16	Unknown	Unknown
17	Unknown	Unknown
18	Unknown	Unknown
19	Unknown	Unknown
20	Unknown	Unknown
21	Unknown	Unknown
22	Unknown	Unknown
23	Unknown	Unknown
24	Unknown	Unknown
25	Unknown	Unknown
26	Unknown	Unknown
27	Unknown	Unknown
28	Unknown	Unknown

5 Experiment

During the research, the experiment was divided into three parts. At the first step, the efficiency of the GNN-based model for the fault diagnosis task was investigated. Next, the quality of obtained adjacency matrices was studied by using them in the GNN model with another structure. Finally, another architecture with several graph structure learning layers was proposed.

5.1 Fault diagnosis

The proposed GNN-based model was trained with four different variants of graph structure learning layer. Hyper-parameter α was set to 0.1 after comparing different values. The TEP dataset was normalized by the standard score formula:

$$z = \frac{x - \mu}{\sigma},$$

where μ is the mean value and σ is the standard deviation of the feature. Node features are presented as time series values with a window size equal to 100. Adam with an initial learning rate of 0.001 was chosen as the optimization algorithm. All models were trained during 50 epochs.

5.1.1 Metrics

During the evaluation process, the model receives a sample as input and labels it as one of the process states. When comparing the result with labeled test data, four main outcomes are possible:

- True positive (TP) is the case when faulty process state was diagnosed correctly.
- False positive (FP) is the case when normal process state was diagnosed as a fault.
- True negative (TN) is the case when normal process state was diagnosed correctly.
- False negative (FN) is the case when faulty process state was diagnosed as normal one.

The following metrics are very important in FDD tasks and were used to evaluate the quality of the models:

- True Positive Rate (TPR) is the probability that the fault state is diagnosed correctly:

$$TPR = \frac{TP}{TP + FN}$$

- False Positive Rate (FPR) is the probability that the normal state is detected as a fault:

$$FPR = \frac{FP}{FP + TN}$$

- Average Detection Delay (ADD) is an average time delay between the moment when the process state changed (from normal to faulty condition) and the moment when this change was detected by the model.

To compare the models, TPR and FPR were calculated separately for each type of fault as well as the average value among all faults.

5.1.2 Baseline models

The model with a predefined adjacency matrix was taken as a baseline. To obtain the adjacency matrix, the correlation method was used. Based on the dataframe, the Pearson correlation matrix was built. Values less than 0.3 were removed. This adjacency matrix generation strategy performed better in preliminary tests compared to other methods that have been tried.

To test that GNN-based models can compete with other types of neural networks, MLP-based and 1DCNN [9] based models where also added to the baseline. An architecture with two hidden layers was chosen for MLP-based model (Fig.6).

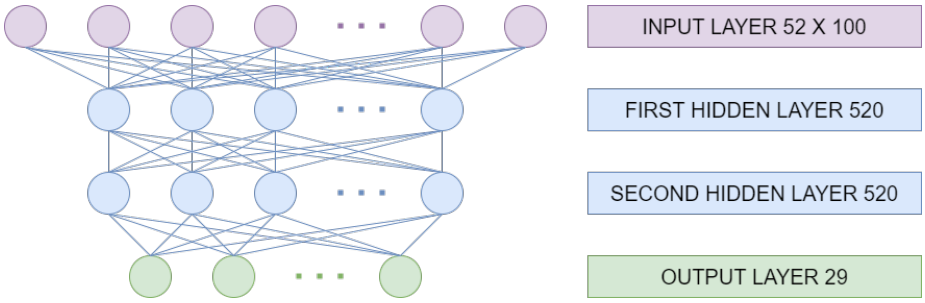

Figure 6: MLP with two hidden layers.

In recent years, convolutional neural networks have shown very good results in fault diagnosis tasks. However, the purpose of this work is not to create a supercomplex model that outperforms analogues, but rather a general assessment of the applicability of GNNs in FDD problems. As a CNN baseline, several standard architectures with one and two 1d convolutional layers were created. The best TPR/FPR scores were obtained by the model with two 1d convolutional and maxpooling layers. Architecture depicted in Fig.7.

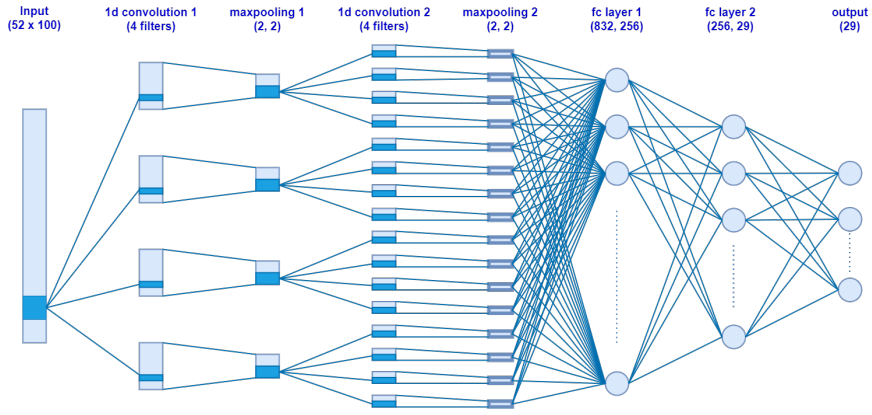

Figure 7: 1DCNN baseline architecture.

The principal component analysis method was chosen as the classical statistical baseline. The PCA method cannot be trained by stochastic gradient descent and has high memory complexity, so it was tested on a reduced dataset with only 20 possible types of fault.

5.1.3 Results

The average values of TPR/FPR, ADD and the number of trainable parameters for each model are shown in Table 2. The best average TPR score was obtained by GNN + Direct A model. All models with graph structure learning layers showed better results than the models from the baseline. The experiment confirmed that neural networks outperform classical statistical methods in FDD tasks and graph neural networks can compete with other types of NN. The detailed results for each fault type are shown in Table 3.

Table 2: Average values of TPR/FPR, and ADD for each model.

Model	TPR	FPR	ADD	Num. of trainable par.
GNN + Relu(W)	0.9225	0.0046	36.77	102013
GNN + Uni-directed A	0.9210	0.0037	31.66	129909
GNN + Undirected A	0.9259	0.0072	32.06	114609
GNN + Directed A	0.9272	0.0071	35.55	129909
GNN + Correlation	0.8882	0.0064	46.86	99309
MLP	0.8895	0.0136	28.06	2688189
CNN1D	0.8967	0.0151	25.07	226941
PCA	0.6363	0.0382	24.54	-

Table 3: Results of fault diagnosis for each type of fault.

State ID	GNN + ReLU	GNN + Uni-dir.	GNN + Undir.	GNN + Dir.	GNN + Corr.	MLP	1DCNN
1	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00
2	1.00/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00
3	0.97/0.00	0.97/0.00	0.98/0.00	0.98/0.00	0.95/0.00	0.97/0.00	0.96/0.00
4	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	0.99/0.00
5	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	1.00/0.00	0.98/0.00	0.97/0.00
6	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00
7	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00
8	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00
9	0.60/0.00	0.68/0.00	0.57/0.00	0.55/0.00	0.54/0.00	0.70/0.00	0.49/0.00
10	0.97/0.00	0.98/0.00	0.98/0.00	0.97/0.00	0.96/0.00	0.97/0.00	0.98/0.00
11	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.98/0.00	0.99/0.00
12	0.99/0.00	0.98/0.00	0.98/0.00	0.99/0.00	0.98/0.00	0.98/0.00	0.98/0.00
13	0.97/0.00	0.97/0.00	0.97/0.00	0.97/0.00	0.97/0.00	0.97/0.00	0.97/0.00
14	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00	0.99/0.00	0.99/0.00
15	0.31/0.00	0.28/0.00	0.38/0.01	0.37/0.01	0.00/0.00	0.01/0.01	0.08/0.00
16	0.92/0.00	0.86/0.00	0.90/0.00	0.91/0.00	0.65/0.00	0.67/0.00	0.67/0.00
17	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00
18	0.96/0.00	0.96/0.00	0.96/0.00	0.97/0.00	0.96/0.00	0.96/0.00	0.96/0.00
19	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00
20	0.97/0.00	0.97/0.00	0.97/0.00	0.97/0.00	0.97/0.00	0.97/0.00	0.97/0.00
21	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.01/0.00	0.00/0.00
22	0.64/0.00	0.61/0.00	0.73/0.00	0.72/0.00	0.69/0.00	0.47/0.00	0.61/0.00
23	0.94/0.00	0.96/0.00	0.94/0.00	0.99/0.00	0.71/0.00	0.63/0.00	0.78/0.00
24	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00
25	1.00/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00	0.98/0.00	0.99/0.00
26	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00	0.97/0.00	0.98/0.00
27	0.99/0.00	0.99/0.00	1.00/0.00	0.99/0.00	0.99/0.00	0.98/0.00	0.99/0.00
28	0.95/0.00	0.96/0.00	0.96/0.00	0.96/0.00	0.85/0.00	0.84/0.01	0.90/0.00

All models with graph structure learning layers shown better scores than the baseline with predefined adjacency matrix. The adjacency matrices obtained during training can be very useful in faster troubleshooting of equipment malfunctions. Their quality are assessed in the next section.

5.2 Quality of obtained adjacency matrices

To obtain an adjacency matrix, the graph structure learning layer parameters are trained together with the parameters of the other layers and can be adapted to their values. On the one hand, the adjacency matrix must correctly indicate the dependencies between the nodes so that subsequent GCN layers correctly process the incoming data. On the other hand, during backpropagation, the trainable weights of the adjacency matrix depend on the weights of the GCN layers. To evaluate this negative influence, a simplified model with a different structure, another set of parameters and window size equals to 10 is proposed (Fig.8). The adjacency matrices obtained during fault diagnosis step in the previous subsection are fed into the model as predefined ones, and the results is compared with the baseline (matrix obtained by correlation). The results in Table 4 show that the trained adjacency matrices outperform the predefined

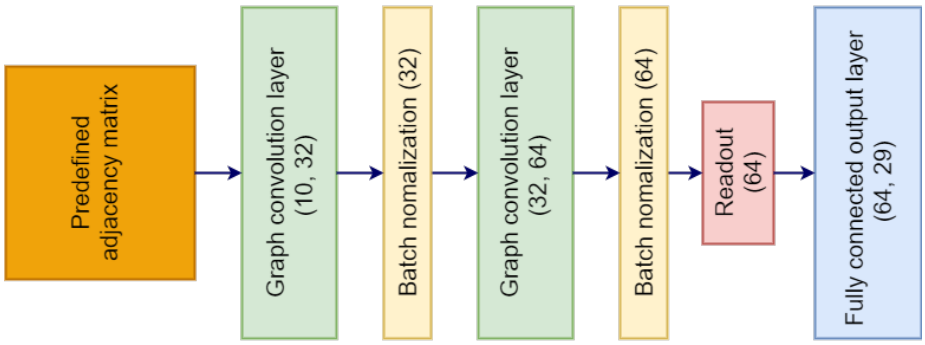

Figure 8: Simplified model architecture with two GCN layers and predefined adjacency matrix.

one in TPR and FPR scores. This may indicate that, despite the influence of the weights of the GCN layers, the trained adjacency matrices can well display the dependencies in the data.

Table 4: Experiment with trained and predefined adjacency matrices.

Model	TPR	FPR	ADD	Num. of trainable par.
GNN + Relu(W)	0.7667	0.0008	46.76	4557
GNN + Uni-directed A	0.7827	0.0049	36.21	4557
GNN + Undirected A	0.7735	0.0020	41.17	4557
GNN + Directed A	0.7884	0.0012	42.95	4557
GNN + Correlation	0.7491	0.0076	39.82	4557

When comparing the obtained adjacency matrices and the scheme of technological process, both correspondences and differences can be seen. Differences can be explained by the fact that it is not always possible to see hidden dependencies in the diagram. However, some logical connections can still be found. During the experiment, a model was trained with the Directed A learning layer and the maximum number of edges for one node equal to 3 by using the power of the argtopk function. Fig.9 shows the outgoing edges for nodes 51 and 52. Sensor 51 displays the flow rate of coolant in the reactor, which in turn affects the temperature released during the chemical reaction measured by sensor 9. When the temperature of a substance in a closed volume changes, the pressure changes. Accordingly, when the coolant flow changes, not only the temperature changes, but also the pressure recorded by the sensor 7. The reactor and stripper tanks are connected, so the pressure also changes on the readings of sensor 16. Sensor 52 measures the coolant flow in the condenser, which also affects sensors 7 and 9 in the reactor. After the condenser, the substance enters the separator,

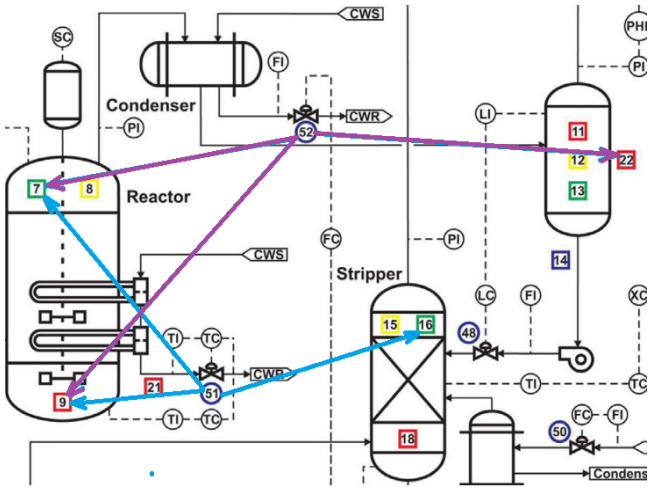

Figure 9: Dependencies received by the graph structure learning layer.

where the temperature of coolant is measured by sensor 22. The edges obtained using the model are logical and prove that the graph sctructure learning layer can be used to find hidden dependencies between parts of equipment.

All adjacency matrices obtained by the different graph structure learning layers in the Fault diagnosis part of the experiment can have common features (some common nodes have large weights, some less). Similarities are shown in Fig.10. The rows contain the sums of the weights of the edges belonging to the numbered sensors/nodes. 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 graph structure learning layers does indeed learn the interaction structure in the workflow. However, the total number of edges between each node may be different. These results led to the idea that there cannot be an ideal adjacency matrix. The processes of such a complex system are not static and can change over time. Depending on the stage of the process, the dependencies between sensors may change. In the next step, the possibility of learning several adjacency matrices in parallel was studied.

sensor/node:	1	2	3	. . .	20	21	22	23	. . .	50	51	52
Uni-direced A	1.2	5.7	5.8	. . .	3.8	0.1	0.9	3.4	. . .	2.9	0.9	2.5
Undireced A	0.8	13.4	4.7	. . .	6.5	0.4	0.8	5.2	. . .	2.9	1.7	4.6
Direced A	1.1	7.0	5.3	. . .	4.0	0.6	1.1	7.5	. . .	4.1	1.0	3.2

Figure 10: Similarities of obtained adjacency matrices.

5.3 Model with several graph structure learning layers

To train several adjacency matrices, an architecture with several trained in parallel graph neural networks (GNN modules) is proposed (Fig.11). The model described in the Model section was taken as an instance of the GNN module. Module outputs are combined by concatenation and fed to the fully connected output layer. Variants with 4, 10 GNN modules and different number of hidden parameters in the GCN layers were tested. Relu W was chosen as the graph structure learning layer. All models were trained during the 50th epochs and the best scores where chosen. The results with average values are shown in Table 5. More detailed results can be found in Table 6.

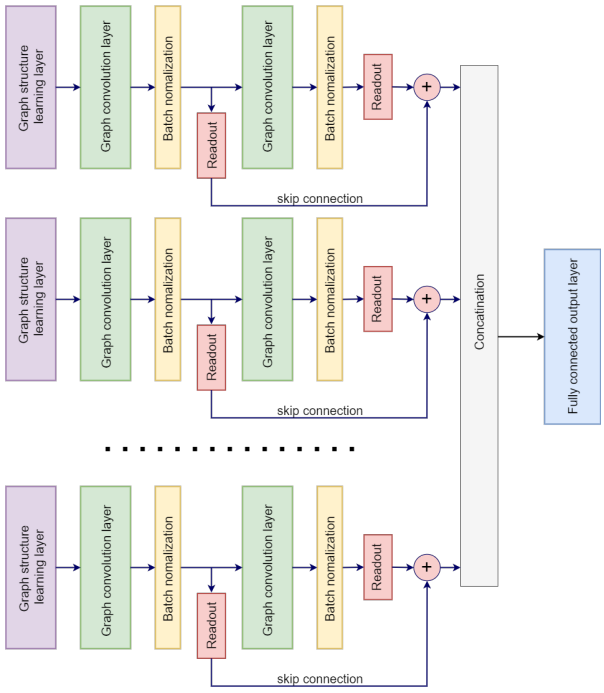

Figure 11: Model architecture with several graph structure learning layers.

Table 5: Models with several GNN modules.

Model	TPR	FPR	ADD	Num. of trainable par.
4 x (GNN(64) + Relu(W))	0.9288	0.0110	29.86	61597
4 x (GNN(256) + Relu(W))	0.9337	0.0490	15.67	407965
10 x (GNN(64) + Relu(W))	0.9362	0.0413	19.50	153949
GNN + Relu(W)	0.9225	0.0046	36.77	102013

Table 6: Results of fault diagnosis for each type of fault.

State ID	4 x GNN(64) + ReLU	4 x GNN(256) + ReLU	10 x GNN(64) + ReLU	GNN + ReLU
1	1.00/0.00	1.00/0.00	0.99/0.00	1.00/0.00
2	1.00/0.00	0.99/0.00	0.99/0.00	1.00/0.00
3	0.98/0.00	0.97/0.00	0.99/0.00	0.97/0.00
4	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00
5	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00
6	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00
7	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00
8	1.00/0.00	0.98/0.00	1.00/0.00	0.98/0.00
9	0.60/0.00	0.70/0.00	0.43/0.00	0.60/0.00
10	0.98/0.00	0.98/0.00	0.98/0.00	0.97/0.00
11	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00
12	0.98/0.00	0.99/0.00	0.99/0.00	0.99/0.00
13	0.97/0.00	0.97/0.00	0.95/0.00	0.97/0.00
14	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00
15	0.34/0.01	0.42/0.01	0.50/0.02	0.31/0.00
16	0.94/0.00	0.94/0.01	0.92/0.00	0.92/0.00
17	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00
18	0.96/0.00	0.96/0.00	0.96/0.00	0.96/0.00
19	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00
20	0.97/0.00	0.97/0.00	0.97/0.00	0.97/0.00
21	0.00/0.00	0.01/0.01	0.02/0.01	0.00/0.00
22	0.71/0.00	0.46/0.00	0.76/0.00	0.64/0.00
23	0.98/0.00	0.97/0.01	0.97/0.00	0.94/0.00
24	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00
25	0.99/0.00	0.99/0.00	0.98/0.00	1.00/0.00
26	0.98/0.00	0.98/0.00	0.98/0.00	0.98/0.00
27	0.99/0.00	0.99/0.00	0.99/0.00	0.99/0.00
28	0.97/0.00	0.97/0.00	0.95/0.00	0.95/0.00

This part of the experiment show that the architecture with several parallel GNN modules, having a comparable number of parameters, is better at finding dependencies in the data. For example, the proposed models began to learn the fault 21, which did not respond to previous architectures. The optimal number of GNN modules and hidden parameters of GCN layers can be discovered in further studies.

6 Conclusion

GNN-based models outperform classical statistical models and show good performance compared to other types of neural networks in FDD tasks with structured data. For use in GNN, this type of data requires information about the relationship of nodes in the form of an adjacency matrix, which is not always known. The proposed graph structure learning layer allows to solve this problem. The adjacency matrices obtained during the training showed a better result than that one obtained by the correlation method. The possibility of obtaining hidden relationships between equipment sensors can open up wide opportunities in process control, as well as speed up the repair and adjustment of equipment. In further works, models with graph structure learning layers can be tested on datasets where adjacency matrices are known in advance. Other types of GNN layers can also be considered.

The adjacency matrices obtained during training have similar features (the same nodes have a greater influence on other parts of the equipment, others less), but there are also noticeable differences. This can be explained both by the fact that the influences between nodes can be expressed differently through intermediate nodes and by the fact that the equipment can operate in different modes and the dependencies are not static. During the research, the idea of learning multiple adjacency matrices came up. A model consisting of several parallel GNN modules with graph structure learning layers was proposed. This architecture has shown that with a similar number of trainable parameters, it can better find dependencies in the data.

Graph neural networks have shown good results and prospects in processing data from various types of sensors in technological equipment. Further research may be aimed not only at improving the accuracy of the classification of fault types, but also at ways of interpreting the results of the models. The ability to obtain adjacency matrices can open up great possibilities in determining the root cause of a malfunction, as well as the fault propagation path.

References

[1] Defu Cao et al. "Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting." In: *NeurIPS.* (2020).

[2] C.C. Chang and C.C Yu. "On-line fault diagnosis using the signed directed graph." In: *Ind. Eng. Chem. Res.* (1990), pp. 1290–1299.

[3] Zhiwen Chen et al. "Graph neural network-based fault diagnosis: a review." In: (2021), pp. 1–17.

[4] Y Cui et al. "METRO: a generic graph neural network framework for multivariate time series forecasting." In: *Proceedings of the VLDB Endowment 15 (2)* (2021), pp. 224–236.

[5] J.J. Downs and E.F. Vogel. "A plant-wide industrial process control problem." In: *Computers and Chemical Engineering 17 (3)* (1993), pp. 245–255.

[6] Scarselli Franco et al. "The graph neural network model." In: *IEEE Transactions on Neural Networks, vol. 20* (2009), pp. 1–22.

[7] Xin Gao, Fan Yang, and E. Feng. "A process fault diagnosis method using multi-time scale dynamic feature extraction based on convolutional neural network." In: *Canadian Journal of Chemical Engineering* (2020), pp. 1280–1292.

[8] Shengnan Guo et al. "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting." In: *Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33* (2019), pp. 922–929.

[9] Shuzhan Huang et al. "1DCNN Fault Diagnosis Based on Cubic Spline Interpolation Pooling." In: *Hindawi, Shock and Vibration* (2020), pp. 1–13.

[10] Cheng Ji and Wei Sun. "A Review on Data-Driven Process Monitoring Methods: Characterization and Mining of Industrial Data". In: *Processes 2022 10(335)* (2022), pp. 1–36.

[11] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." In: *Proceedings of the 5th International Conference on Learning Representations, ICLR '17.* (2017).

[12] Tianfu Li et al. "The emerging graph neural networks for intelligent fault diagnostics and prognostics: A guideline and a benchmark study." In: *Mechanical Systems and Signal Processing Volume 168* (2022), p. 108653.

[13] Ildar Lomov et al. "Fault detection in Tennessee Eastman process with temporal deep learning models." In: *Journal of Industrial Information Integration (23)* (2021), Article 100216.

[14] L. Ma, J. Dong, and K. Peng. "A novel key performance indicator oriented hierarchical monitoring and propagation path identification framework for complex industrial processes." In: *ISA Transactions* (2019). DOI: `https://doi.org/10.1016/j.isatra.2019.06.004`.

[15] You-Jin Park, Shu-Kai S. Fan, and Chia-Yu Hsu. "A review on fault detection and process diagnostics in industrial processes." In: *Processes 2020 8(1123)* (2020), pp. 1–26.

[16] C.A. Rieth et al. "Issues and advances in anomaly detection evaluation for joint human-automated systems." In: *International Conference on Applied Human Factors and Ergonomics, Springer* (2017), pp. 52–63.

[17] Lei Shi et al. "Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition." In: *CVPR* (2019), pp. 12026–12035.

[18] Hao Wu and Jinsong Zhao. "Deep convolutional neural network model based chemical process fault diagnosis." In: *Computers and Chemical Engineering* (2017), pp. 185–197.

[19] Zonghan Wu et al. "Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks." In: *KDD* (2020), pp. 753–763.

[20] Zonghan Wu et al. "Graph WaveNet for Deep Spatial-Temporal Graph Modeling." In: *IJCAI* (2019).

[21] Xingwei Xu et al. "Intelligent monitoring and diagnostics using a novel integrated model based on deep learning and multi-sensor feature fusion." In: *Measurement Volume 165* (2020), p. 108086.

[22] Shen Yina et al. "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process." In: *Journal of Process Control 22* (2012).