12th IFAC International Symposium on
Advanced Control of Chemical Processes
July 14-17, 2024. Toronto, Canada

Interpretable Propagation Path Neural Network for Fault Detection and Diagnosis

Benjamin Nguyen * **Moncef Chioua** *

* *Department of Chemical Engineering, Polytechnique Montréal, Montréal, Canada, (e-mail: benjamin.nguyen@polymtl.ca)*

Abstract: Automated fault detection and diagnosis (FDD) in modern chemical process systems are essential for ensuring safe and reliable operation. Deep learning methods for FDD are gaining traction due to their high fault classification performances, but these methods lack interpretability which hinders their practical use. In this work, an interpretable neural network model for FDD is proposed which classifies the fault based on the fault propagation path. In our approach, fault propagation paths are embedded into the model as directed graphs in a graph neural network. This framework enforces connections between hidden layer nodes, giving them and their activations a physical and interpretable meaning. We evaluate the performance and interpretability of the proposed model on the benchmark Tennessee Eastman Process where it achieves a 92.9% classification accuracy on select fault datasets.

Keywords: interpretability; propagation path; graph neural network; fault diagnosis; chemical process;

1. INTRODUCTION

Abnormal events or faults in industrial process systems can result in risks to safety and the environment, as well as high maintenance costs and downtime. Therefore, automated fault detection and diagnosis (FDD) systems are essential for the quick recovery of the process system. Established methods for FDD include multivariate statistical methods such as Principle Component Analysis (PCA) (Kresta et al., 1991) and Partial Least Squares (PLS) (MacGregor and Kourti, 1995) but with the advent of Industry 4.0, deep neural networks are emerging as an increasingly viable option. Methods such as Long Short Term Memory (LSTM) networks (Zhao et al., 2018), and Convolutional Neural Networks (CNN) (Wu and Zhao, 2018) have been shown to produce higher accuracies in FDD tasks than the established methods due to their ability to model non-linearities and automatically extract features (Zhao et al., 2018).

However, deep neural networks suffer from a lack of interpretability due to their complex "black box" internal structure. For FDD models, the ability to explain a decision is crucial for industrial acceptance, given the high consequence of an incorrect action (Hoffmann et al., 2021). Operators and engineers must trust that the model follows a logical reasoning that is aligned with process knowledge. With established methods such as PCA, the principle components within the model are composed of linear combinations and can be interpreted by producing contribution plots identifying the variables driving the detection (Westerhuis et al., 2000). In deep neural networks, the internal behaviour is much more difficult to decipher due to the non-linear activation functions and the multitude of unintelligible nodes and weights (Molnar, 2022).

To fulfill the interpretability requirement for FDD, modern methods are applying *explainability* techniques to interpret the decisions of the black-box model. These techniques either approximate the black-box model using an interpretable model such as a decision tree, or compute a gradient of the prediction with respect to the input variables (Molnar, 2022). Similar to PCA, the explanations are in the form of a contribution plot showing which variables drive the classification for a given sample. Examples of applications in process systems FDD literature include Local Interpretable Model-agnostic Explanations (LIME)(Bhakte et al., 2023), Gradient-weighted Class Activation Mapping (Grad-CAM) (Wu and Zhao, 2022), Layerwise Relevance Propagation (LRP) (Agarwal et al., 2021), and Shapley values (Bhakte et al., 2022). However, these explainability techniques only provide an approximate explanation for a given sample. The black-box model internals are still unknown which leads to uncertainty in both the prediction and the explanation (Rudin, 2019). Having a model that is inherently interpretable would eliminate the need for an external technique and provide a more robust and faithful explanation (Rudin, 2019).

To create an inherently interpretable model, constraints are usually put on the structure, making the inner components and decision-making process easier to understand (Rudin, 2019). Examples include decision trees and linear regressions which are limited to splits in single variables and linear relationships (Molnar, 2022). Constraints, in the form of sparse connections, can also be applied to neural networks to make them more interpretable such as in weight-pruned neural networks (Filan et al., 2021) or Graph Neural Networks (GNNs) (Battaglia et al., 2018). Structured sparse connections can provide sensible relations between the input variables and hidden layer nodes, allowing them to retain a physical meaning (Battaglia

et al., 2018). When applied to chemical process systems, the nodes can represent process variables or units if they are connected following the physical process. Examples in the process systems literature include Wu et al. (2020) who incorporated the process structural knowledge into the architecture of a recurrent neural network for model predictive control. In their weight-constrained model, groups of hidden layer nodes represented process units by constraining the connections of downstream input variables, not belonging to the process unit. Other examples include Wu and Zhao (2021) and more recently, Jia et al. (2023) who used GNNs for FDD. The nodes in the GNN represent process variables through constraints outlined by the process topology. Performance improvements were achieved by these models but the mechanisms to explore their inherent interpretability were not investigated.

Therefore, we propose an interpretable neural network model for FDD in chemical process systems. Similar to Jia et al. (2023) and Wu and Zhao (2021), the proposed model uses GNNs where nodes represent process variables but instead of using a single graph of the full process topology, a set of graphs outlining the propagation paths of separate faults are utilized. In the context of FDD, the sequence of affected variables in a propagation path can discriminate between faults, and the subset of variables in each path are more interpretable than the full process topology. In the proposed model, the set of propagation paths are integrated through an ensemble of graph convolutional layers where each layer guides the model to produce a representation following the corresponding propagation path. The representations can then be interpreted through the node activations to understand the model reasoning. The contributions of this paper are the following:

- An interpretable FDD model which learns and classifies based on the propagation path of a fault in addition to time series process data
- A framework to interpret node activations which represent process variables to understand and verify the proposed model decision

The remainder of the paper is organized as follows: Section 2 presents the proposed fault propagation-based model and interpretation method. Section 3 evaluates the effectiveness of the proposed model on the benchmark Tennessee Eastman Process. Performance is compared to standard deep learning methods for FDD, and a sample interpretation is demonstrated. Section 4 concludes this study and discusses future work.

2. METHODOLOGY

This section outlines the framework for the proposed interpretable neural network model. The model first captures abnormal temporal features in the Temporal Convolutional (TC) layer and then propagates these features throughout the fault propagation paths in the Spatial Propagation (SP) module. The representations of each layer are then grouped by fault in the classification layer which outputs a fault class probability. The model is applicable to continuous processes with a known steady state where significant deviations are related to abnormal situations. The full model is shown in Figure 1, and the following sub-sections describe its components in detail.

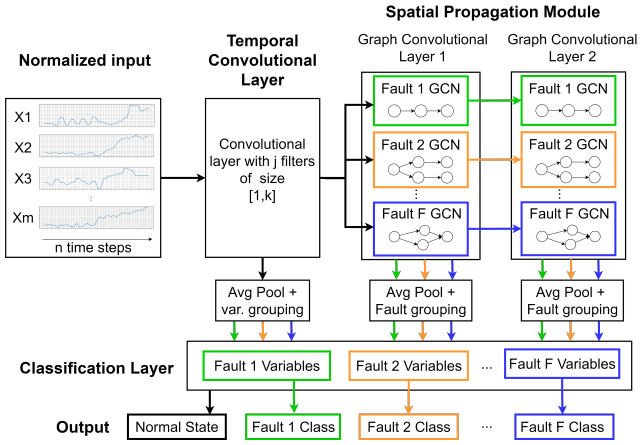

Fig. 1. Model schematic from the normalized input to the output classes. Colored arrows represent the outputs from the corresponding fault propagation paths.

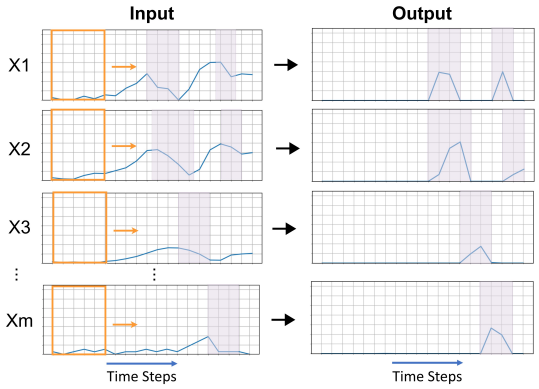

Fig. 2. Input and output of a filter in the TC layer from the case study in Section 3.

2.1 Temporal Convolutional (TC) Layer

First, abnormal temporal features in each process variable are captured using a convolutional layer. These features identify: 1) the process variables affected by the fault in the input sample, 2) specific temporal features that can assist the classification layer in discriminating between faults.

The input to the proposed model is a matrix X with m process variables and n time steps. It enters the Temporal Convolutional (TC) layer which has j filters with a size of $[1, k]$ (Krizhevsky et al., 2012). These 1D filters slide along the time axis of the m variables detecting temporal features while also de-noising the input data with the ReLU activation function. Figure 2 shows the input and output of one of the filters for the case-study model presented in Section 3. The filter, indicated by the orange square, detects the decreasing slopes shown in the shaded areas. This temporal feature aids in diagnosing faults with random variations where downward slopes are often present. Since the 1D filters only process one variable at a time, their meaning is preserved in the output. This method is adapted from Song et al. (2023) who also required the process variables to remain independent for causality extraction. Once exiting the TC layer, the extracted temporal features are a 3D tensor of size $[m, n, j]$ which is fed to the spatial propagation module.

2.2 Spatial Propagation (SP) Module

In the Spatial Propagation (SP) module, the temporal features captured in the TC layer are propagated to the connected process variables outlined in the embedded propagation paths. First, the temporal features in the $[m, n, j]$ tensor are concatenated for each process variable to form a matrix of size $[m, n \times j]$. Next, the matrix is fed to F parallel GC layers where F is the number of faults the model is trained on. In the GC layers, each of the m process variables correspond to a node, and their $n \times j$ vector of temporal features are embedded as node features. Then, graph convolutions are performed where the node features are propagated to neighboring nodes based on their respective propagation paths. The paths are represented by an adjacency matrix, A_f, where the subscript f is the fault in question. These are $[m, m]$ binary matrices with "1"s representing causal connections between process variables in fault 's propagation path and "0"s representing no connection. Finally, the aggregated node features are updated using a set of learned weights and sent through a ReLU activation function. The equation for the updated state, $H_f^{(l)}$, after being processed by GC layer l is the following:

$$H_f^{(l)} = \sigma\left(D_f^{-1/2} A_f D_f^{-1/2} H_f^{(l-1)} W_f\right) \tag{1}$$

Where the superscript l is the GC layer in question, σ is the ReLU activation function, $\tilde{A}_f$ is the adjacency matrix with added self-connections, $\tilde{D}_f$ is the diagonal matrix used to normalize $\tilde{A}_f$, $H_f^{(l-1)}$ is the previous state and W_f are the learned weights (Kipf and Welling, 2016). The output of each GC layer is a 3D tensor of size $[F, m, n \times j]$ representing the states of the F propagation paths.

Within the SP module, we also propose a method to preserve the time ordered attributes of the temporal features in each of the nodes by pruning connections in the weight matrix, W_f. In the vanilla GC framework by Kipf and Welling (2016), W_f is dense making the update function from the aggregated previous state fully connected as shown on the left in Figure 3. The updated node features lose their time ordered attributes by being functions of all the node features in the previous state. Therefore, we propose to prune W_f such that the updated node features are only functions of s previous node features for a given filter output vector as shown on the right in Figure 3. s is the amount of past information needed to process a temporal feature. The visualization of these attributes in the node activations allow for the interpretation of sequence, especially when time delays between affected variables in the propagation path exist. The sequence can then be used to differentiate between faults.

2.3 Classification Layer

The propagation sequence represented by the activations in the Temporal Convolutional (TC) and graph convolutional (GC) layers are combined and utilized by the classification layer to classify the input sample. This framework of outputting the intermediate layers ensures that the sequence of activations between layers are used for classification. Simultaneously, it prevents an "oversmoothing"

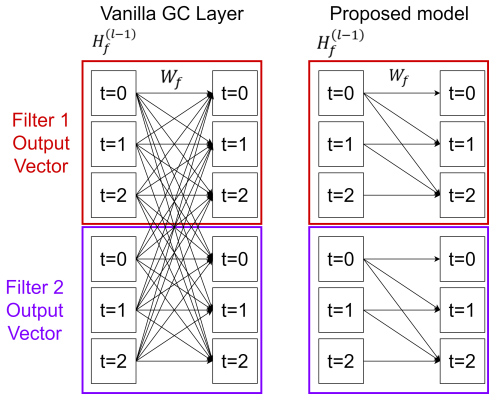

Fig. 3. Schematic comparing the node update function of the classical GC layer by Kipf and Welling (2016) and the proposed model's modification with $s = 3$.

issue in graph neural networks where the node features become too similar after multiple graph convolutional layers (Hamilton, 2020). Therefore, more information is provided to the classifier to learn and distinguish between faults.

First the node features corresponding to each of the j filters in every node are average pooled. The size of the TC layer output is reduced from an $[m, n, j]$ tensor to an $[m, j]$ TC matrix and the GC layer output tensors are reduced from $[F, m, n \times j]$ to $[F, m, j]$. Then, each of the pooled TC and GC layer tensors are split into F fault matrices. Note that in each of these matrices, only the subset of process variables present in their respective propagation paths are included. Figure 1 represents these matrices with coloured arrows corresponding to their propagation paths. Then the fault matrices from each layer are grouped by fault, concatenated, and flattened to form a final vector of length $(\sum_{f=1}^{F} m_f) \times l \times j$, where l is the number of layers and m_f is the number of process variables in fault f's propagation path. This final classification vector is fed into a linear classifier with $F + 1$ output classes which includes the normal state class. This layer is pruned to connect only the corresponding flattened fault matrix to their respective fault class. This framework avoids complex classification criteria of a given fault based on the activations of process variables not within the fault class.

2.4 Model Decision Interpretation

The proposed model framework is designed to allow the user to understand how the model comes to a decision. The basis of decision-making process are the activations of nodes following the fault propagation path. The node activations can be analyzed to verify that the proposed model is working in this manner. The procedure for interpretation is the following.

When the proposed model detects a fault, j (number of filters in TC layer) sets of activations within the TC and graph convolutional layers are produced. Each set corresponds to an abnormal temporal feature used by the model to classify the fault sample. The contributions of each temporal feature to the chosen output class are calculated from the pooled outputs and the classification layer weights. The activations set with the highest contribution are therefore analyzed. Insight into the types of tempo-

ral features the model values for the classified fault can also be obtained through the level of contribution. Then, the activations are filtered for each node feature using a threshold calculated with the normal state validation set:

$$g^{m,t} = a^{m,t}_{normal,mean} + z\sigma^{m,t}_{normal} \tag{2}$$

Where for node m and node feature t, $g^{m,t}$ is the threshold value, $a^{m,t}_{normal,mean}$ is the mean activation value, $\sigma^{m,t}_{normal}$ is the standard deviation, and z is the threshold factor. z is a hyperparameter determining the number of standard deviations to set the threshold. After filtering, the node activations of the chosen propagation path are analyzed to verify the model decision. The user can reason that if the nodes are activated from the root cause variable along the propagation path, then the representation produces a high output probability for the chosen class. The user can also compare the node activations to other paths identifying less activations not following the propagation path resulting in lower output probabilities. The user is therefore able to identify the decision-making mechanism for the proposed model.

3. CASE STUDY: TENNESSEE EASTMAN PROCESS

The proposed approach was tested on the Tennessee Eastman Process (TEP) (Downs and Vogel, 1993) which is a benchmark simulation used for fault detection and diagnosis (Chiang et al., 2000). First a description of TEP and dataset are given, followed by the performance metrics, pre-processing, and training method. Next, the classification performance results are presented and compared to standard black-box models. Finally, the interpretability is demonstrated with a sample fault.

3.1 Process Description and Dataset

The TEP produces two products, G and H, from four reactants, A, C, D, and E, along with an inert, B, and byproduct, F. It involves five major units: the reactor, condenser, separator, stripper, and compressor. 52 variables can be obtained from this process which include 22 process sensors, 18 composition measurements, and 11 manipulated variable measurements. The control structure is that of Lyman and Georgakis (1995). The schematic of the process is shown in Figure 4 along with the fault 6 propagation path from Sun et al. (2020) discussed in Section 3.5.

The original dataset consists of 20 different fault states plus a fault-free normal state. Each state has a set of 25-hour operation intervals which are called "runs". The runs differ through random seed and the process variables have a sampling time of 3 minutes equating to 500 samples per run. In this study, only the first 170 samples of each run were used for training and evaluation to allow the model to focus on learning the initial propagation of the faults. Only faults with identifiable propagation paths in the literature were evaluated since there are no established ground-truth propagation paths. They include faults 1 and 7 from Chen et al. (2018), 4, 5, 12 and 13 from Mori et al. (2014), 6 from Sun et al. (2020), and 8 from Li et al. (2016). These references use causal discovery methods like transfer

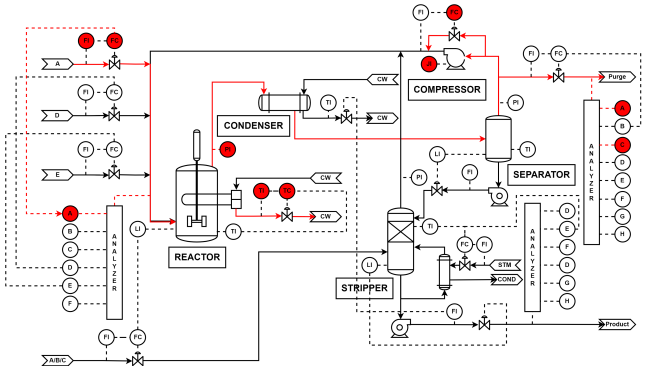

Fig. 4. TEP schematic adapted from Chiang et al. (2000) with the fault 6 propagation path from Sun et al. (2020) shown in red

entropy, Granger causality, or Bayesian networks to find the propagation paths for the faults.

3.2 Performance Metrics

The metrics used to evaluate model performance were the accuracy and false alarm rate (FAR):

$$\text{Accuracy} = \frac{\#\text{ of correctly classified samples}}{\text{total }\#\text{ of samples}} \tag{3}$$

$$\text{FAR} = \frac{\#\text{ of misclassified samples at normal state}}{\text{total }\#\text{ of samples at normal state}} \tag{4}$$

Having a low FAR is essential for preventing operator alarm fatigue and increasing trust in the model. An FAR limit of 0.02 is set for the final models.

3.3 Data Pre-processing and Training

For each fault, 22 runs were used in total with 18 runs for training, 2 for validation, and 2 for testing. The performance improvement was marginal with a higher number of runs. For scaling, the entire dataset is z-score normalized using the sample mean and standard deviation of the normal state training data. Then the absolute value is taken to avoid a "dead node" issue (Lu et al., 2019) where some nodes do not activate throughout the entire training set, complicating the interpretation. The scaling equation is therefore the following:

$$x_i^{'} = |\frac{x_i - \bar{x_i}}{s_i}| \tag{5}$$

Where for variable i, x_i' is the normalized value, $\bar{x_i}$ is the normal state mean, s_i is the normal state standard deviation, and $||$ is the absolute value. The data is then converted into 2D sliding windows of 52 variables by 20 time-steps with a stride of one to minimize detection delay. In terms of model parameters, three convolutional filters of size $[1, 5]$ in the Temporal Convolutional layer were used, along with two graph convolutional layers with an s of three. Next, the model was trained until convergence at 50 epochs using the Adam optimizer (Kingma and Ba, 2014) with a batch size of 36 and a learning rate of 0.01. Within the loss function, the weight of the normal state class was five times the other classes to emphasize a low false alarm rate and the selective pruning occurred at the end of every batch.

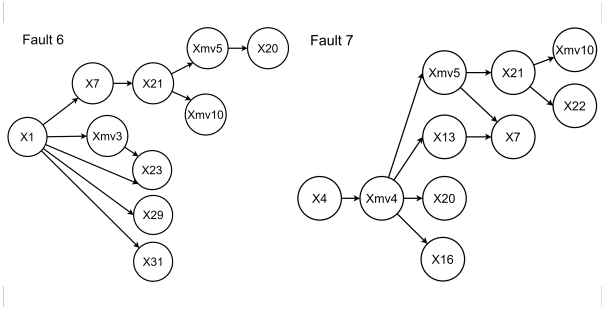

Fig. 5. Graphs of the fault 6 (left) and fault 7 (right) propagation path adapted from Sun et al. (2020) and Chen et al. (2018)

3.4 Classification Results

The final model chosen is the model with the highest validation accuracy and an FAR less than 0.02. The proposed model achieves a 91.9% accuracy on the test set. As a performance comparison, several standard black-box neural network models are also trained and tested on the same dataset. Specifically, an LSTM (Zhao et al., 2018), a CNN (Wu and Zhao, 2018), and a Graph Convolutional Network (GCN) (Wu and Zhao, 2021). The graph used for the GCN is adapted from Jia et al. (2023) to reflect using the process topology of the available process variables. Table 3.4 gives the results comparing the models. Overall,

Table 1. Overall model accuracies

Dataset	LSTM	CNN	GCN	Proposed model
Training	92.7%	95.0%	96.0%	94.1%
Validation	93.9%	88.3%	93.9%	91.9%
Test	91.4%	87.7%	95.5%	92.9%

the proposed model performs worse than the GCN on the test set at 92.9% accuracy while performing better than the CNN and the LSTM. By using the full topology of the process, the GCN may extract information outside of the propagation path, thus leading to a higher performance. However, the performance of the proposed model is still deemed acceptable while being interpretable.

3.5 Proposed Model Interpretation - fault 6

In this section, a model decision from fault 6 is interpreted. According to Sun et al. (2020), when fault 6 occurs, there is a sudden loss of A feed (X1) which is attempted to be corrected by the A feed valve (Xmv3). The inlet A composition is also affected (X23) along with the A purge (X29) and C purge compositions (X31). Meanwhile, the reactor is the first unit to be affected with the reactor pressure (X7), cooling water temperature (X21), and cooling water valve (Xmv10) deviating from normal. It then reaches the compressor affecting recycle valve (Xmv5) and compressor work (X20) (Figure 4 & Figure 5).

The fault is immediately detected and diagnosed correctly on the first sample, 3 minutes into the fault. Within the fault 6 classification layer, filter 1 has the highest contribution for this sample and therefore its activations are analyzed. The column on the left in Figure 6 shows these activations which have a threshold factor, z, set to 5 standard deviations. The node features of the corresponding nodes representing process variables are on the x-axis

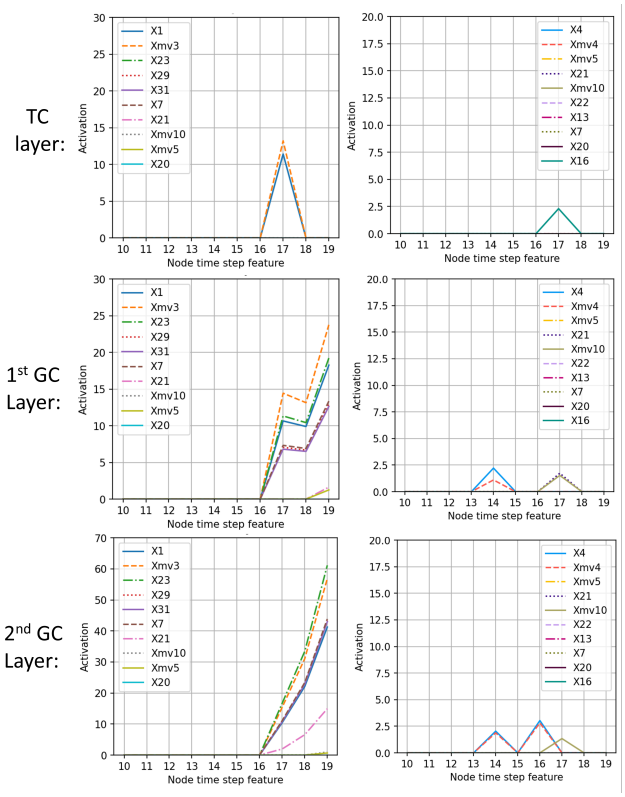

Fig. 6. TC and GC node activations for fault 6 (left) and fault 7 (right) propagation paths at t=1 for fault 6

while the activation values higher than the threshold are on the y-axis. In the Temporal Convolutional layer, the root cause node X1 and corresponding valve node Xmv3 are activated. The activations then propagate into the next steps of the path in the first GC layer with variables X23, X29, X31, X7 and small activations in X21 and Xmv5. Then in the second GC layer, Xmv10 is activated while the activations of the other nodes are amplified. These node activations follow the fault 6 propagation graph from the root cause, and therefore the reasoning behind the fault 6 classification can be observed.

The activations of other propagation paths can also be analyzed to verify why another class was not chosen. For the same fault 6 sample at t=1, the right column on Figure 6 shows the activations of the fault 7 path (Figure 6, right). In the Temporal Convolutional layer, the reactor pressure (X16) is activated followed by the A/B/C feed rate (X4), A/B/C feed valve (Xmv4), reactor cooling water valve (Xmv10) and temperature (X21) in the first GC layer and X21 and X16 disappearing in the second GC layer. These activations do not follow the propagation path of fault 7 and therefore, it is reasonable that the model chose fault 6 over fault 7. This verification procedure can also be repeated with the other propagation paths as well.

4. CONCLUSION

In this work, an interpretable neural network model was proposed for FDD on chemical process systems. It successfully provides prediction interpretations through the internal node activations which are enforced through graph convolutional layers and network pruning to represent the

propagation paths of given faults. Tests on the benchmark Tennessee Eastman Process show that it can achieve acceptable performance while being interpretable through the activation sequence. Future work would address limitations with faults of similar propagation paths but differing fault pattern (eg. step vs. random variation) and optimizing and interpreting hyperparameters.

REFERENCES

Agarwal, P., Tamer, M., and Budman, H. (2021). Explainability: Relevance based dynamic deep learning algorithm for fault detection and diagnosis in chemical processes. *Computers & Chemical Engineering*, 154, 107467.

Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., and Faulkner, R. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Bhakte, A., Chakane, M., and Srinivasan, R. (2023). Alarm-based explanations of process monitoring results from deep neural networks. *Computers & Chemical Engineering*, 179, 108442.

Bhakte, A., Pakkiriswamy, V., and Srinivasan, R. (2022). An explainable artificial intelligence based approach for interpretation of fault classification results from deep neural networks. *Chemical Engineering Science*, 250, 117373.

Chen, H.S., Yan, Z., Yao, Y., Huang, T.B., and Wong, Y.S. (2018). Systematic procedure for granger-causality-based root cause diagnosis of chemical process faults. *Industrial & Engineering Chemistry Research*, 57(29), 9500–9512.

Chiang, L.H., Russell, E.L., and Braatz, R.D. (2000). *Fault detection and diagnosis in industrial systems*. Springer Science & Business Media.

Downs, J.J. and Vogel, E.F. (1993). A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17(3), 245–255. doi: https://doi.org/10.1016/0098-1354(93)80018-I.

Filan, D., Casper, S., Hod, S., Wild, C., Critch, A., and Russell, S. (2021). Clusterability in neural networks. *arXiv preprint arXiv:2103.03386*.

Hamilton, W.L. (2020). *Graph representation learning*. Morgan & Claypool Publishers.

Hoffmann, M.W., Drath, R., and Ganz, C. (2021). Proposal for requirements on industrial ai solutions. In *Machine Learning for Cyber Physical Systems: Selected papers from the International Conference ML4CPS 2020*, 63–72. Springer Berlin Heidelberg.

Jia, M., Hu, J., Liu, Y., Gao, Z., and Yao, Y. (2023). Topology-guided graph learning for process fault diagnosis. *Industrial & Engineering Chemistry Research*, 62(7), 3238–3248.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T.N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kresta, J.V., Macgregor, J.F., and Marlin, T.E. (1991). Multivariate statistical monitoring of process operating performance. *The Canadian journal of chemical engineering*, 69(1), 35–47.

Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Li, G., Qin, S.J., and Yuan, T. (2016). Data-driven root cause diagnosis of faults in process industries. *Chemometrics and Intelligent Laboratory Systems*, 159, 1–11.

Lu, L., Shin, Y., Su, Y., and Karniadakis, G.E. (2019). Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*.

Lyman, P.R. and Georgakis, C. (1995). Plant-wide control of the tennessee eastman problem. *Computers & Chemical Engineering*, 19(3), 321–331. doi: https://doi.org/10.1016/0098-1354(94)00057-U.

MacGregor, J.F. and Kourti, T. (1995). Statistical process control of multivariate processes. *Control Engineering Practice*, 3(3), 403–414. doi: https://doi.org/10.1016/0967-0661(95)00014-L.

Molnar, C. (2022). *Interpretable machine learning: A Guide for Making Black Box Models Explainable*. 2 edition.

Mori, J., Mahalec, V., and Yu, J. (2014). Identification of probabilistic graphical network model for root-cause diagnosis in industrial processes. *Computers & chemical engineering*, 71, 171–209.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5), 206–215.

Song, P., Zhao, C., and Huang, B. (2023). Mpge and rootrank: A sufficient root cause characterization and quantification framework for industrial process faults. *Neural Networks*, 161, 397–417.

Sun, W., Paiva, A.R., Xu, P., Sundaram, A., and Braatz, R.D. (2020). Fault detection and identification using bayesian recurrent neural networks. *Computers & Chemical Engineering*, 141, 106991.

Westerhuis, J.A., Gurden, S.P., and Smilde, A.K. (2000). Generalized contribution plots in multivariate statistical process monitoring. *Chemometrics and intelligent laboratory systems*, 51(1), 95–114.

Wu, D. and Zhao, J. (2021). Process topology convolutional network model for chemical process fault diagnosis. *Process Safety and Environmental Protection*, 150, 93–109. doi:10.1016/j.psep.2021.03.052.

Wu, D. and Zhao, J. (2022). *Understand how CNN diagnoses faults with Grad-CAM*, volume 49, 1537–1542. Elsevier.

Wu, H. and Zhao, J. (2018). Deep convolutional neural network model based chemical process fault diagnosis. *Computers and Chemical Engineering*, 115, 185–197. doi:10.1016/j.compchemeng.2018.04.009.

Wu, Z., Rincon, D., and Christofides, P.D. (2020). Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. *Journal of Process Control*, 89, 74–84.

Zhao, H., Sun, S., and Jin, B. (2018). Sequential fault diagnosis based on lstm neural network. *IEEE Access*, 6, 12929–12939. doi:10.1109/ACCESS.2018.2794765.