Shared Parameter Network: An efficient process monitoring model

Lucky E. Yerimah [a,], Sambit Ghosh [a] and B. Wayne Bequette [a1]

[a] Howard P. Isermann Department of Chemical and Biological Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180

Abstract

The use of deep learning methods for fault detection and diagnosis (FDD) has continued to gain research attention due to the continuous availability of data and the need for reliable FDD methods. Recurrent neural network (RNN) units continue to dominate the neural network of choice in FDD models because of their ability to capture temporal dependence in time series data. In our previous work, we proposed a probabilistic bidirectional recurrent network (PBRN) for process prediction and detection of faults. A key limitation of PBRN is the number of model parameters in the RNN units. In this paper, we reduce the computational requirements and training time of the original PBRN by introducing a shared parameter network (SPN). SPN uses a shared parameter space to learn relevant features for both process prediction and fault detection. This shared parameter setting significantly reduces the model size and training time. Using SPN, we achieved a 76 percent reduction in the number of parameters and a 48 percent reduction in the time required for training. The validation loss profile during training and the fault detection performances of both models also demonstrate the superiority of SPN for process prediction and fault detection.

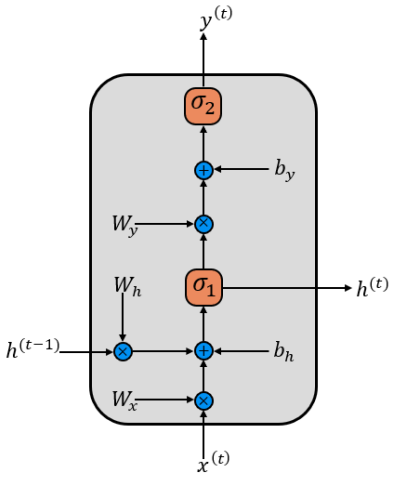
Keywords

Fault detection and diagnosis (FDD), recurrent neural network (RNN), probabilistic bidirectional recurrent network (PBRN), shared parameter network (SPN).

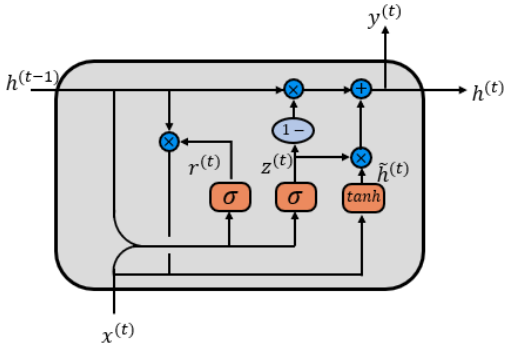
Introduction

Fault detection and diagnosis have continued to be an active research area in the process systems engineering research community. This is due to the continuous demand for preventive abnormal event handling systems and methods that promote efficient manufacturing operations (Bi et al. (2022)). Fault detection and diagnosis methods are mostly data-driven. These data-driven methods can be grouped into statistical process control (SPC) methods and Artificial Intelligence (AI) methods. SPC methods such as Principal Component Analysis (PCA), Partial Least Squares (PLS), and Canonical variate analysis (CVA) have been extensively studied for fault detection and diagnosis using the simulated Tennessee Eastman process (TEP) as a case study (Qin (2003); Li et al. (2010); Ruiz-Cárcel et al. (2015)). These methods work by projecting the historical process data into lower dimensional spaces and using the subspaces to compute metrics that indicate the condition of the process (Chiang et al. (2001); Ghosh et al. (2022)). Limitations of SPC methods include the assumptions of linear process behaviors and the inability to handle temporal dependence in the process data. Nonlinear and dynamic variants of SPC methods such as Dynamic PCA, Dynamic PLS, and Kernel PCA have been proposed by researchers, but it is difficult to find the inverse mappings from the feature space to the input space using these methods. Also, computing the matrix inverse for high-dimensional process data is computationally intensive (Cheng et al. (2019)). Recent advances in the use of deep learning models to approximate nonlinear relationships in data have provided opportunities for improving the accuracies of fault detection and diagnosis methods. Recurrent neural networks (RNN) are a class of deep learning models developed for primarily handling time series data like historical process data. They can account for temporal dependence in time series data. Deep learning models can be applied to supervised learning problems when labels are provided along with the data and unsupervised learning when labels are not available. Process data are often unlabeled hence unsupervised learning methods such as autoencoders (AE) are used with RNN units to learn from unlabeled data (Cheng et al. (2019); Yerimah et al. (2022)). RNN units such as gated recurrent units (GRU) have been proposed for fault detection and diagnosis using the TEP simulated data (Cheng et al. (2019)). In our previous study, we proposed a probabilistic bidirectional recurrent network (PBRN) for process prediction and detection of faults (Yerimah et al. (2022)). PBRN uses a bidirectional RNN in a probabilistic and deterministic framework for process prediction and detection of faults. In this work, we improve the performance of PBRN by introducing parameter sharing between the two RNN networks.

[1] Corresponding author: B. Wayne Bequette (E-mail: bequette@rpi.edu).



(a) Basic Structure of RNN



(b) Gated recurrent unit

Figure 1: Recurrent neural networks. Reprinted with permission (Yerimah et al. (2022))

This significantly reduces the model size, reduces the training time, and also allows for learning shared features that are most important to both process prediction and fault detection.

Background

Recurrent Neural Networks

A recurrent neural network (RNN) is a modified neural network to handle data with temporal dependence. RNN contains a hidden state which allows it to capture the time dependence in data. Figure 1a shows the basic structure of an RNN.

weights $\mathbf{W_h}$,$\mathbf{W_x}$,$\mathbf{W_y}$ and bias vectors $\mathbf{b_h}$,$\mathbf{b_y}$ represent the parameters of the unit as shown in Figure 1a. Equations (1) and (2) show the relationship between the parameters and hidden units and outputs.

$$h^{(t)} = \sigma_1\left(W_h h^{(t-1)} + W_x x^{(t)} + b_h\right) \tag{1}$$

$$y^{(t)} = \sigma_2\left(W_y h^{(t)} + b_y\right) \tag{2}$$

where $y^{(t)}$ represents the RNN output at time t. σ_1 and σ_2 are the activation functions for the hidden state and output, respectively. Activation functions introduce nonlinearity to the hidden units and outputs.

Given a training dataset $= \{\mathbf{x_i}, \mathbf{y_i}\}_{i=1}^{N}$, model parameters of an RNN are learned using the backpropagation through time (BPTT) through the minimization of the loss function between the predicted and actual output as shown in Equation 3 (Werbos (1990)).

$$\mathcal{L} = \frac{1}{2N}\sum_{i=1}^{N}\left\|\hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)}\right\|_2^2 \tag{3}$$

where $\mathcal{L}$ is equivalent to $1/2$ of the average squared prediction error between the predicted output $\hat{y}$ and the actual output y.
Gated recurrent units are a type of RNN for efficiently capturing the time dependence in a time series data [ME]. A GRU uses a gating method for adaptively deciding what information to retain in the unit's memory. The gates include an update gate $z^{(t)}$ for computing information from the past that will be passed along to the future. The reset gate $r^{(t)}$ determines information from the past that will be forgotten. Figure 1b shows the structure of a GRU and including its gates.

Autoencoders

Autoencoders (AE) are deep learning model architectures that contain encoder and decoder neural networks. When data is passed through the input of the AE, a reconstructed version of the data is produced in the output and this makes AE to be classified as a generative model. Given a data set $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathcal{R}^d$ is a d dimensional vector of sensor measurements, the encoder f_θ produces a latent vector $\mathbf{z}_i$ that represents the most salient features of the data. Where $\mathbf{z}_i \in \mathcal{R}^m$, where $m < d$. The decoder g_ϕ uses the latent space to reconstruct the original data as $\hat{\mathbf{x}}$ as shown in Figure 2a.

2.2.1 Variational autoencoders

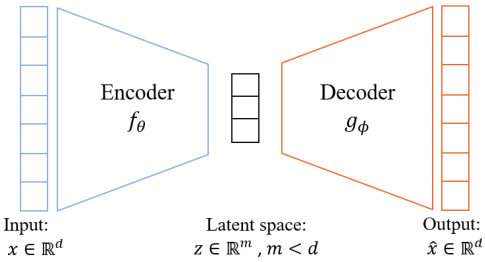
Variational autoencoders (VAE) are probabilistic AE primarily built to handle latent space sensitivity. VAE consists of a probabilistic encoder for outputting the distribution (μ, σ) of the latent space $\mathbf{z}$, and a decoder for sampling from the distribution to reconstruct the original data $\mathbf{x}$ as shown in Figure 2b. Training a VAE involves maximizing the evidence lower bound (ELBO) of the log-likelihood of each data sample $p_\theta(\mathbf{x}^{(i)})$. The ELBO $\mathcal{L}\left(\theta, \phi, \mathbf{x}^{(i)}\right)$ is expressed as:

$$\begin{aligned}\mathcal{L}\left(\theta, \phi, \mathbf{x}^{(i)}\right) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right] - \mathcal{D}_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})\|p_\theta(\mathbf{z})\right) \\ &= \mathcal{L}_{REC}^{(i)} + \mathcal{L}_{KL}^{(i)}\end{aligned} \tag{4}$$

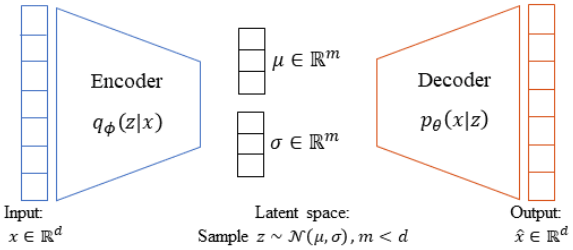
where $\mathcal{L}_{REC} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right]$ and $\mathcal{L}_{KL} = -\mathcal{D}_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})\|p_\theta(\mathbf{z})\right)$

Using several assumptions and simplifications for practical implementation of VAE, Equation 4 is reduced to a closed form solution as:

$$\mathcal{L}_{REC}^{(i)} = \|\mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)}\|_2^2 \tag{5}$$



(a) Deterministic Autoencoder



(b) Variational Autoencoder

Figure 2: Autoencoder neural networks. Reprinted with permission (Yerimah et al. (2022)).

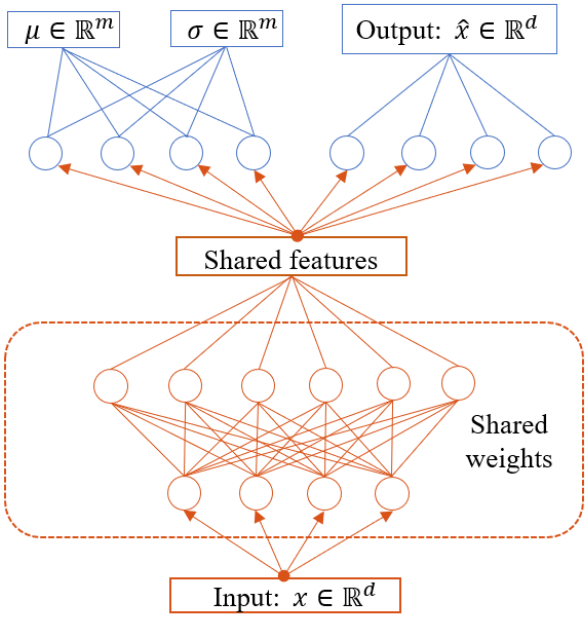


Figure 3: Proposed shared parameter network. The model uses a shared input space to provide features for both reconstruction and latent space distribution.

and,

$$2\mathcal{L}_{KL}^{(i)} = \|\mu^{(i)}\|_2^2 + d + \sum_j^d \sigma_j^{(i)} - \log \sigma_j^{(i)} \tag{6}$$

For the complete derivation of Equations 5 and 6, the reader is referred to our previous work (Yerimah et al. (2022)). A consequence of the simplifying assumptions of VAE means that the learned latent space does not exactly match the assumed prior.

Probabilistic Bidirectional Recurrent Network (PBRN)

PBRN is a novel AI-based sensor predictor and fault detector model (Yerimah et al. (2022)). PBRN uses two bidirectional RNN models that are arranged to solve the problem of assumed prior and learned latent space mismatch of VAE. The first bidirectional RNN works as a deterministic model and reconstructs the input data directly from the input space. The second bidirectional RNN takes the reconstructed data as an input and produces an output distribution that can be sampled for fault detection. To train the PBRN, the simplified ELBO expressions in Equations 5 and 6 are used to learn the parameters of the model. A limitation of the PBRN is the ability to handle a longer prediction horizon (Yerimah et al. (2022)). Increasing the model size can help the model to better handle a longer prediction horizon but this comes at the expense of requiring more computation power and longer training times.

Parameter Sharing Neural Networks

Training neural networks require significant computational resources such as GPU memory. Increasing the number of weights of a neural network also increases the GPU memory required to train and save the network weights. This limits the model sizes that can be trained and also increases the training time of the networks. One solution to increase model sizes without increasing the number of parameters is to introduce parameter sharing across the layers (Savarese and Maire (2019)). Aside from the reduction in model size, parameter sharing also provides the advantage of using the same feature detector in the input data across different layers of the neural network. It also allows for using the same features for multi-task learning (Zhuang et al. (2015)).

Method

We propose a shared parameter network (SPN) to improve the performance of the PBRN model without significantly increasing the model size and the training time.

Proposed Method

The shared parameter network consists of RNN layers for representing the shared feature space for both input reconstruction and latent distribution as shown in Figure 3. The reconstructed output represents the deterministic side of the model while the latent distribution represents the probabilistic side of the model. To train the model parameters, a loss function containing the reconstruction error and the simplified ELBO error is used. The SPN training algorithm is similar to that of PBRN except for that shared parameters and the parameters of each output are learned. The complete training algorithm is given in Algorithm 1.

Algorithm 1: : SPN Training Algorithm

input : Dataset $\{x_i\}_{i=1}^{N}$, Learning rate η, Number of iterations Epoch, Sequence length τ
output: Model parameters Θ
$\mathbf{X} = \{x_i\}_{i=1}^{N-\tau}$, $\mathbf{Y} = \{x_i\}_{i=1+\tau}^{N}$, Split data into train and validation sets
generate J batches $\{x_j, y_j\}_{i=1}^{J}$ from the train set
$\Theta \leftarrow$ Initialize parameters
for *training time* $= 1$ ***to*** *Epoch* **do**
  for *batch* $\{x_j^{(i)}, y_j^{(i+\tau)}\}$ *in train set* **do**
    compute $\hat{x}_j^{(i+\tau)}$
    compute (μ, σ) and sample $z_j^{(i+\tau)} \sim \mathcal{N}(\mu, \sigma)$
    compute $\mathcal{L}(\hat{x}_j^{(i+\tau)}, y_j^{(i+\tau)}, \Theta)$

$$\nabla_\Theta \mathcal{L}(\hat{x}_j^{(i+\tau)}, y_j^{(i+\tau)}, \Theta) = \sum_{t=1}^{T} \frac{d\mathcal{L}(\hat{x}_j^{(i+\tau)}, y_j^{(i)}, \Theta)}{d(\Theta)}$$

    Update parameters using

$$\Theta \leftarrow \nabla_{\theta,\phi} \mathcal{L}(\hat{x}_j^{(i+\tau)}, y_j^{(i+\tau)}, \Theta)$$

    compute validation loss $\mathcal{L}_{\text{val}}^{\text{time}}$
  end
  if $\mathcal{L}_{val}^{time} < \mathcal{L}_{val}^{time\text{-}1}$ **then**
    save (Θ)
  end
end

SPN Fault Detection

For online fault detection, the reconstruction error and ELBO error are used to monitor test data for fault or no-fault. These errors can be computed using Equations 5 and 6 respectively. Using normal operating condition test data, thresholds for Equations 5 and 6 can be computed as follows:

$$\mathcal{L}_{\text{KL,lim}} = \frac{\beta}{2N} \sum_{i=1}^{N} \left(\|\mu^{(i)}\|_2^2 + d + \sum_{j}^{d} \sigma_j^{(i)} - \log \sigma_j^{(i)} \right) \tag{7}$$

$$\mathcal{L}_{\text{REC,lim}} = \frac{\lambda}{N} \sum_{i=1}^{N} \left\| x^{(i)} - \hat{x}^{(i)} \right\|_2^2 \tag{8}$$

β and λ are tunable parameters between $[0, 1]$ which are adjusted to control false positive and missed detection rates.

Case Study: Air Separation Unit

Process Description

The air separation unit (ASU) is a manufacturing process that yields industrial quantities of nitrogen, oxygen, and argon in high-purity liquids and gases. The ASU uses a triple-column system and a primary heat exchanger for high degrees of thermal and material integration. For a more detailed description of the ASU, the reader is referred to Cao et al. (2016) (Cao et al. (2016)). Real manufacturing data from the ASU is used as the case study for this work. The data includes 70 selected sensor tags that capture the most important information for process monitoring.

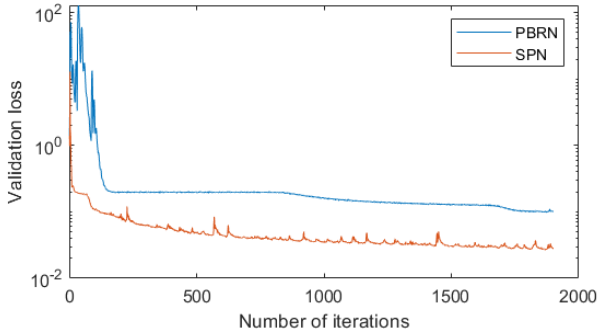


Figure 4: Validation loss performance of PBRN and SPN. The SPN achieved a lower validation loss and is therefore superior for process prediction

Model Setup

The model shared parameter space uses two layers of RNN units. The first layer contains 300 units of GRU while the second layer contains 60 units of GRU. The input to the reconstruction side of the model contains 70 dense units of neural networks which is equal to the dimension of the reconstructed data while the latent distribution uses 60 dense units of neural networks. The implementation of the model is carried out in TensorFlow v2.10.0 open source software library for machine learning development. An adaptive learning rate optimizer (Adam) is used for optimizing the model parameters and the hyper-parameters are tuned using a trial and error method. For training and validation, 2 months of real manufacturing and normal operating condition data are used. The test data contains both normal operating conditions and faulty data.

Training Performance Comparison

The SPN contains a total of 1,272,934 parameters while the PBRN contains a total of 5,248,390 parameters. Thus the PBRN is over four times greater than SPN. Also, the training time for SPN is 139 seconds while PBRN requires 265 seconds to train for the same number of iterations. These values show a significant reduction in GPU requirements and training time for SPN when compared to PBRN. Figure 4 shows a plot of the validation loss during training. Although SPN contains fewer parameters, it achieves a lower validation loss than PBRN. This improved performance can be attributed to the tied relationship between the reconstruction loss and KL loss. The KL loss acts as a regularizer on the reconstruction loss and the improved regularization of the SPN model is attributed to the shared feature space which provides a dependence that was absent in the original PBRN model (Yerimah et al. (2022)).

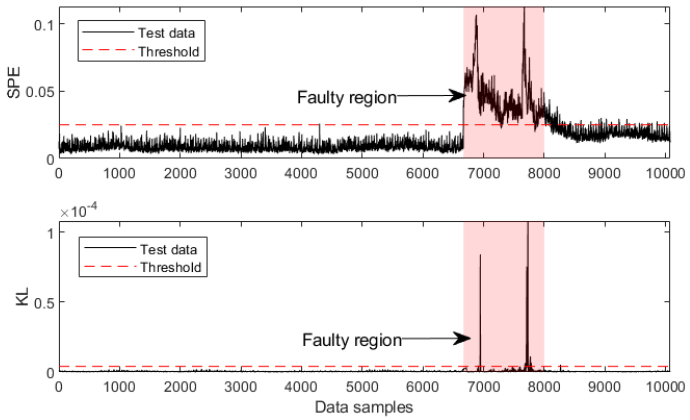
Table 1: Comparison of PBRN and SPN. Both models methods are each better than the other for both faults.

		PBRN		SPN	
	Faults	SPE	KL	SPE	KL
True positives	A	99.4	96.9	97.4	87.4
	B	95.1	95.7	97.7	97.4
False positives	A	0.4	2.9	2.3	0.02
	B	4.9	4.0	2.0	0.3

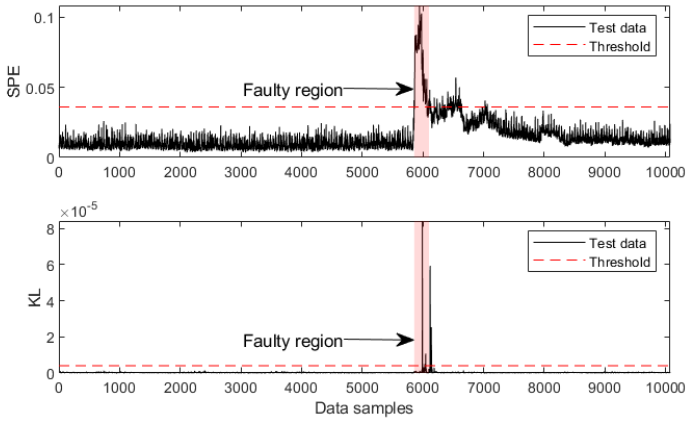
Fault Detection Performance Comparison

Using PBRN as the state-the-art, we compare the fault detection performance of SPN using test data containing two real faults; Faults A and B. We make use of anonymous tags A and B to refer to these faults for proprietary reasons. The architectures of both models were determined using a grid search as described in Section 3.1. The fault detection metrics SPE and KL are calculated using Equations 5 and 6 respectively and the thresholds are determined using Equations 7 and 8. The tuning parameters β and γ are tuned manually to give the best trade-off between false positives and missed detection. Table 1 gives the true positive rates and false alarms for PBRN and SPN. A data point is flagged as faulty if it exceeds the threshold within the faulty region while data points that exceed the threshold outside of the faulty region are flagged as false alarms.

Both methods provide very high true positive rates due to their ability to capture nonlinear relationships and temporal dependence in the process data. In terms of true positives, PBRN outperforms SPN for fault detection of Fault A, but both methods show similar performance in false alarms. On the other hand, SPN outperforms PBRN in true positives and false alarms of Fault B. Figure 5 shows the fault detection plots of SPN. From Figure 5a, we see that faulty SPE values are closer to normal operating condition SPE values and this makes Fault A slightly more difficult to detect than Fault B. The slightly higher performance of PBRN over SPN in Fault A can therefore be attributed to the larger model size capacity for discriminating between faulty and normal SPE values. Since SPN gives superior performance for Fault B and a close to PBRN performance for Fault A, and SPN also provides a significant reduction in GPU requirements and training time, we can hence conclude that SPN overall, outperforms PBRN.



(a) Fault A



(b) Fault B

Figure 5: The fault detection performance of SPN. The shaded region represents the faulty regions. A data point is counted as true positive if it exceeds the threshold within the faulty region and a data point is counted as false alarm if it exceeds the threshold outside of the faulty region.

Conclusion

Higher detection accuracy can often be achieved with larger model sizes at the cost of requiring more computational resources and training time. We have demonstrated in this work that higher performances can also be achieved by introducing novel architectures such as shared parameter layers. The total number of parameters in the SPN and the original PBRN reveal a 76 percent reduction in model size. We also achieved a 48 percent reduction in the training time of the model. The validation profiles of SPN and PBRN show that SPN still outperforms PBRN in process prediction and the fault detection results also reveal that SPN is similar to PBRN in terms of performance. These results show that the shared parameter layers provided more relevant features for process prediction and fault detection than PBRN layers. Using RNN models for multi-time-step prediction decreases the prediction accuracy of the models. Future work in this research involves investigating larger model sizes using the SPN architecture to improve on the detection performance of the current SPN model and to perform long horizon predictions.

Acknowledgements

Part of this work was done with the support of the US Department of Energy through the Clean Energy Smart Manufacturing Innovation Institute (CESMII) Chemical Processing – Smart Air Separation Unit project. Award Number: 4550 G WA324 is greatly appreciated. The authors also acknowledge the Smart Operations, Center of Excellence (COE), Linde, Tonawanda, New York, USA for access to real

plant data.

Disclaimer

References

Bi, X., R. Qin, D. Wu, S. Zheng, and J. Zhao (2022). One step forward for smart chemical process fault detection and diagnosis. *Computers & Chemical Engineering 164*, 107884.

Cao, Y., C. L. E. Swartz, J. Flores-Cerrillo, and J. Ma (2016). Dynamic modeling and collocation-based model reduction of cryogenic air separation units. *AIChE Journal 62*(5), 1602–1615.

Cheng, F., Q. P. He, and J. Zhao (2019). A novel process monitoring approach based on variational recurrent autoencoder. *Computers & Chemical Engineering 129*, 106515.

Chiang, L., E. Russell, and R. Braatz (2001). *Fault Detection and Diagnosis in Industrial Systems*. Springer Verlag, U.K.

Ghosh, S., L. E. Yerimah, Y. Wang, Y. Cao, J. Flores-Cerrillo, and B. W. Bequette (2022). A graph signal processing-based multiple model kalman filter (gsp-mmkf) tool for predictive analytics–an air separation unit process application. *Journal of Advanced Manufacturing and Processing*, e10121.

Li, G., S. J. Qin, and D. Zhou (2010). Geometric properties of partial least squares for process monitoring. *Automatica 46*(1), 204–210.

Qin, S. J. (2003). Statistical process monitoring: basics and beyond. *J. Chemometrics 17*, 480–502.

Ruiz-Cárcel, C., Y. Cao, D. Mba, L. Lao, and R. Samuel (2015). Statistical process monitoring of a multiphase flow facility. *Control Engineering Practice 42*, 74–88.

Savarese, P. and M. Maire (2019). Learning implicitly recurrent cnns through parameter sharing. *arXiv preprint arXiv:1902.09701*.

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE 78*(10), 1550–1560.

Yerimah, L. E., S. Ghosh, Y. Wang, Y. Cao, J. Flores-Cerrillo, and B. W. Bequette (2022). Process prediction and detection of faults using probabilistic bidirectional recurrent neural networks on real plant data. *Journal of Advanced Manufacturing and Processing*, e10124.

Zhuang, F., D. Luo, X. Jin, H. Xiong, P. Luo, and Q. He (2015). Representation learning via semi-supervised autoencoder for multi-task learning. In *2015 IEEE International Conference on Data Mining*, pp. 1141–1146. IEEE.