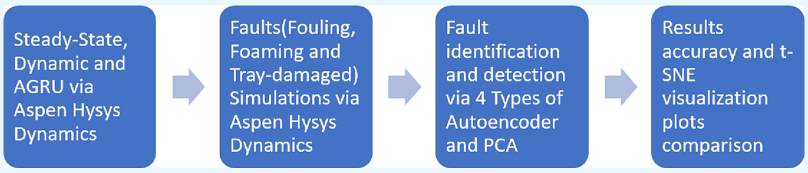
Fault Detection and Identification in an Acid Gas Removal Unit Using Deep Autoencoders

Tan Kaiyun Kathlyn, Haslinda Zabiri,* Chris Aldrich, Xiu Liu, and Ahmad Azharuddin Azhari Mohd Amiruddin

Cite This: *ACS Omega* 2023, 8, 19273−19286 Read Online

ACCESS | Metrics & More | Article Recommendations


Steady-State, Dynamic and AGRU via Aspen Hysys Dynamics
Faults(Fouling, Foaming and Tray-damaged) Simulations via Aspen Hysys Dynamics
Fault identification and detection via 4 Types of Autoencoder and PCA
Results accuracy and t-SNE visualization plots comparison

ABSTRACT: An acid gas removal unit (AGRU) in a natural gas processing plant is designed specifically to remove acidic components, such as carbon dioxide (CO_2) and hydrogen sulfide (H_2S), from the natural gas. The occurrence of faults, such as foaming, and to a lesser extent, damaged trays and fouling, in AGRUs is a commonly encountered problem; however, they are the least studied in the open literature. Hence, in this paper, shallow and deep sparse autoencoders with SoftMax layers are investigated to facilitate early detection of these three faults before any significant financial loss. The dynamic behavior of process variables in AGRUs in the presence of fault conditions was simulated using Aspen HYSYS Dynamics. The simulated data were used to compare five closely related fault diagnostic models, i.e., a model based on principal component analysis, a shallow sparse autoencoder without fine-tuning, a shallow sparse autoencoder with fine-tuning, a deep sparse autoencoder without fine-tuning, and a deep sparse autoencoder with fine-tuning. All models could distinguish reasonably well between the different fault conditions. The deep sparse autoencoder with fine-tuning was best able to do so with very high accuracy. Visualization of the autoencoder features yielded further insight into the performance of the models, as well as the dynamic behavior of the AGRU. Foaming was relatively difficult to distinguish from normal operating conditions. The features obtained from the fine-tuned deep autoencoder in particular can be used to construct bivariate scatter plots as a basis for automatic monitoring of the process.

1. INTRODUCTION

An acid gas removal unit (AGRU) in a natural gas processing plant is designed to remove acidic components, such as carbon dioxide (CO_2) and hydrogen sulfide (H_2S), from natural gas. Natural gas processing plants are operated with the fundamental goal of producing gas that meets specific quality criteria satisfying the needs of their customers. The quality of the processed gas is determined by the type and concentration of the contaminants present. For instance, a high concentration of carbon dioxide in natural gas results in a lower gas heating value, thus lowering its quality value.

Several technologies are currently applied in the monitoring of acid gas removal units, including gas chromatography and spectroscopy,[1] to measure the composition of gas streams; various types of sensors are used to monitor temperature, pressure, flow rate, and other variables in acid gas removal units. In addition to these, process modeling[2] is used to simulate the behavior of acid gas removal units and predict their performance under different operating conditions, as well as data-driven fault diagnostic methods, such as those considered in this paper. Generally, the limitations of these monitoring technologies can include factors such as cost, accuracy, complexity, and maintenance requirements.

Process monitoring and control of AGRUs play a critical role in their safe and efficient operation.[2] Common challenges when monitoring chemical process plants, including AGRUs, are their nonlinearity, the high dimensionality of real industrial data, and possible multimodal behavior that needs to be accounted for. While many different models have been proposed to address these problems in general, the reporting of research specifically related to fault detection and identification in AGRUs in the open literature is limited. Pradittiamphon and Wongsa[3] have found that the use of partial least squares (PLS) to detect variations of natural gas composition and contamination of amine solutions by hot oil could result in the early detection and isolation of faults in time to prevent otherwise costly

Received: December 21, 2022
Accepted: April 6, 2023
Published: May 22, 2023

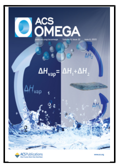


Table 1. Common Faults in AGRUs

type of fault	source of fault	fault simulation
foaming	presence of contaminants in solvent, such as condensed hydrocarbon	decreasing the foaming factor or increasing the differential pressure across the absorber or stripping column abruptly
damagedtray	corrosion of tray due to deposit of solids, which are produced from solvent degradation	decreasing the tray efficiency of the absorber or stripping column
fouling	accumulation of sludge or scale on trays which contains contaminants, such as heavy hydrocarbon and polymeric solvent degradation product	increasing the differential pressure across the absorber or stripping column gradually
solvent loss	leakage or spills	splitting the affected flow line into two streams; one stream is directed to the original destination, while another stream is routed to a sink (representing leakage)
variation of feedgas composition	upstream source gas process upset	changing the hydrocarbon composition in the feed gas

intervention. Another study conducted by Hakimi et al.[4] have made use of artificial neural networks to successfully detect faults in a CO_2 absorption/stripping column based on data simulated on an Aspen Plus environment. A different approach, based on artificial immune systems was used by Al-Sinbol and Perhinschi.[5−7] The approach was validated with a rigorous Dynsim model of an acid gas removal unit and was able to detect 14 different abnormal conditions, including solid deposits and leakages occurring at typical locations throughout the system.

Natural gas processing systems have been monitored by a large variety of approaches that can broadly be grouped into two main categories, viz. model-based and data-based methods. The latter comprise variants of principal component analysis,[8−10] partial least squares,[11] and neural networks including autoencoders.[12,13] More recently, the use of deep neural networks in fault diagnosis has attracted attention,[14] but while these networks can provide performance superior to traditional machine learning methods, they need generally very large amounts of data to perform optimally and are generally costly to develop and maintain, owing to computationally intensive requirements. Similarly, other models are virtually all subject to trade-offs between the degree to which they are capable of capturing nonlinear behavior, robustness to shifts in the distributions of the data resulting from changes in operating conditions, their requirements with regard to volumes of data, cost of development and maintenance, interpretability, etc. Therefore, despite the rapid increase in the use of traditional and deep machine learning methods in these industries, there is still significant scope for further development in data-based methods, given the limited studies currently documented in the literature. This includes not only the application of different models but also their application to different possible fault scenarios.

Therefore, in this study, the dynamic behavior of an AGRU subject to faults related to foaming, damaged trays, and fouling was investigated. In addition, an important class of models used for fault detection are considered, namely, autoencoders augmented with SoftMax output layers. In particular, it is shown that these deep augmented autoencoders offer a decided advantage over autoencoders with simpler structures and that visualization of the process dynamics based on features generated by such autoencoders can form a powerful basis for process monitoring.

The rest of the paper is organized as follows; Section 2 discusses a general description of common faults occurring in AGRUs. Section 3 describes the analytical methodology of fault detection and identification studied in this paper, and Section 4 presents the results and discussion of this study, followed by conclusions.

2. FAULTS IN ACID GAS REMOVAL UNITS (AGRUS)

Operating anomalies (or faults) such as foaming, damaged trays, fouling, solvent loss, and feed gas composition variation are possible faults that may occur in the AGRU system. Table 1 summarizes the faults and the ways in which they may be simulated.

Three of the five fault conditions presented in Table 1, namely, foaming, damaged trays, and fouling, are investigated, as described in the following sections.[15] The occurrence of faults, such as foaming, and to a lesser extent, damaged trays and fouling, in AGRUs is a commonly encountered problem; however, they are the least studied in the literature.

2.1. Foaming.

Foaming is a common issue in many AGRU units and may reduce the CO_2 removal efficiency in the absorber column. Amine solvent solution tends to become frothy on the absorber or stripping column tray when it is vigorously agitated especially with the presence of contaminants in the solvent, such as condensed hydrocarbon. The froth produced will then be stabilized, altering the amine solution surface properties instead of breaking in the tray downcomer, resulting in an analogous situation of jet flooding. The possible way to simulate foaming anomaly is by reducing the foaming factor of the absorber or stripping column or by increasing the differential pressure across the absorber or stripping column abruptly.

2.2. Damaged Tray.

Corrosion of the tray due to deposit of solids, which are produced from the solvent degradation, is the most common source, which leads to the damaged tray of the absorber or stripping column. Solids present in the solvent solution tend to be abrasive to the protective layer of the tray, thus exposing the tray to potential corrosion. The possible way to simulate damaged tray anomaly is by decreasing the tray efficiency of the absorber or stripping column. The frequency at which damaged tray happens may be lower than those of foaming.

2.3. Fouling.

The accumulation of sludge or scale on the absorber or stripping column tray in the area of the unit with relatively still or low velocity is one of the common problems happening in an AGRU. The sludge or scale consists of a mixture of contaminants, such as heavy hydrocarbon and polymeric solvent degradation product. The formation of scale on the tray results in gradual choking of the hole area where gas is required to flow up the column, resulting in jet flooding and gas capacity loss in the column. The most direct way to simulate fouling is by increasing the differential pressure across the absorber or stripping column gradually in a slow manner. Similarly, fouling issues may occur at a much lesser rate than foaming.

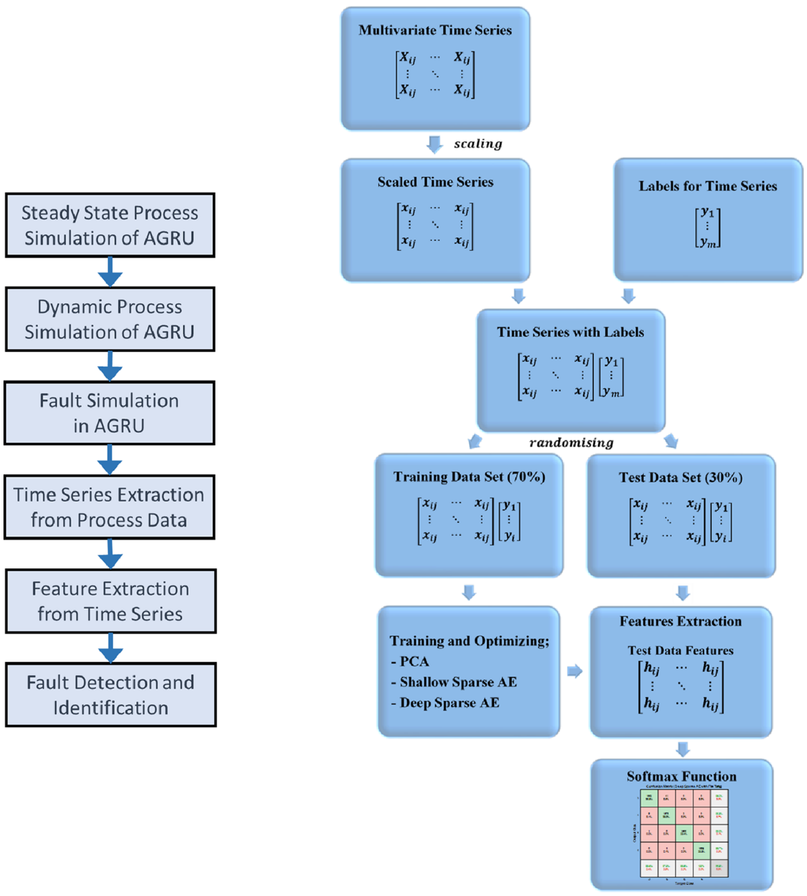


Figure 1. Overall methodology framework, i.e., schematic workflow (left) and detailed flowchart from preprocessing till classification stage where the machine learning model is involved, where AE represents "autoencoder" (right).

3. ANALYTICAL METHODOLOGY FOR SUPERVISED FAULT DETECTION

In this study, there were four main stages involved: (i) dynamic process simulation of the AGRU system, (ii) preprocessing of multivariate time series, (iii) feature extraction from the preprocessed time series data, and (iv) construction of classification models for fault detection. The general analytical framework for model development is illustrated as shown in Figure 1 on the left, while a more detailed workflow related to the processing of the simulated data is as shown in Figure 1 on the right.

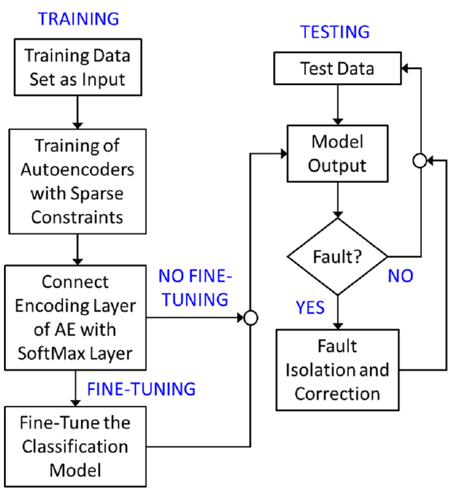

TRAINING
Training Data Set as Input
Training of Autoencoders with Sparse Constraints
Connect Encoding Layer of AE with SoftMax Layer
NO FINE-TUNING
FINE-TUNING
Fine-Tune the Classification Model
TESTING
Test Data
Model Output
Fault?
NO
YES
Fault Isolation and Correction

Figure 2. Implementation of the fault identification models.

In essence, the features extracted from the multivariate time series data signal by the machine learning model are treated as the diagnostics that can be used to detect the presence of fault and identify the type of fault occurred. The workflow associated with development and implementation of the models is shown in Figure 2. The methodology is further described in more detail in the following section.

3.1. Process Simulation of Acid Gas Removal Unit (AGRU).

In this study, diethanolamine (DEA) solution was used as the solvent for the process simulation of the AGRU system.[16] First, the AGRU system simulation model developed by Abbas et al.[16] was reproduced and simulated using all the operating conditions stated. As not all of the operating parameters were presented, the system model was treated as a foundation model to be further developed. Modification on the system was performed in accordance with heuristic rules applied in typical AGRU system design and operation.[17] The modification was necessary to produce a process model that could represent the real industrial AGRU as closely as possible.

The AGRU system was initially simulated using Aspen HYSYS at a steady state as shown in Figure 3a. Adjustments were subsequently made to the steady state model in order for it to run dynamically, which included specification of pressure and flow parameters, proper sizing of the equipment, and design of the control schemes. Conventional proportional-integral-derivative (PID) controllers were introduced into the system

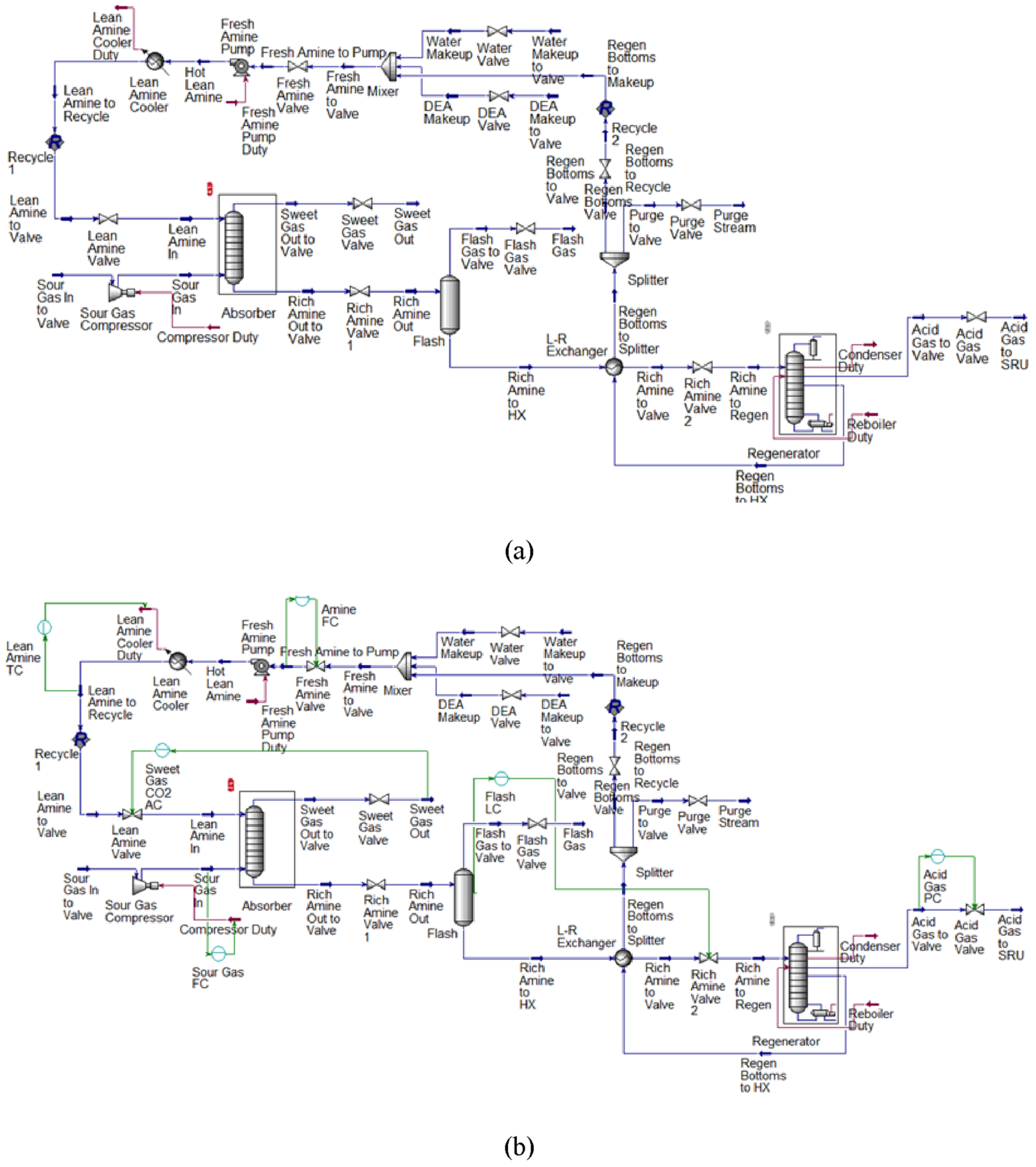


Figure 3. (a) Steady state and (b) dynamic process simulation model of the AGRU system.

with appropriate tuning. The AGRU system was then simulated using Aspen HYSYS Dynamics as depicted in Figure 3b with noise being introduced into the system to make the model more realistic.

The process model is then allowed to be run dynamically until it achieved its desired steady state condition. At this stage, simulated faults were introduced into the system individually and the resulting dynamic behavior of each relevant stream variable was observed and recorded. The simulation was allowed to run continuously, giving the controllers sufficient time to bring the deviated process system back to its desired steady state condition. Once steady state was achieved, the same type of fault was introduced into the system once again and the procedure repeated for a total of four times in order to generate a sufficient amount of data representative of the fault condition.

The above steps were repeated with damaged trays and fouling. It should be noted that all of the aforementioned faults were introduced in the absorber unit. The AGRU simulated data used in this study comprised 13 input variables and 4 output/target classes. For fault detection and identification in the AGRU, the input variables selected were the CO_2 composition of the sweet gas, as well as the total flowrate, temperature, and pressure of sweet gas stream, lean solvent stream, rich solvent stream and acid gas stream (3 process variables × 4 process streams + CO_2 composition from sweet gas stream), making up a total of 13 input variables, as shown in Table 2. The four

Table 2. Input Variable and Output/Target Class

input variable (x_i)	output/target class
x_1: sweet gas stream CO_2 composition (kmol/kmol)	normal
x_2: sweet gas stream total flowrate (kmol/hr)	foaming
x_3: sweet gas stream temperature (°C)	damaged tray
x_4: sweet gas stream pressure (bar)	fouling
x_5: lean solvent stream total flowrate (kmol/hr)	
x_6: lean solvent stream temperature (°C)	
x_7: lean solvent stream pressure (bar)	
x_8: rich solvent stream total flowrate (kmol/hr)	
x_9: rich solvent stream temperature (°C)	
x_{10}: rich solvent stream pressure (bar)	
x_{11}: acid gas stream total flowrate (kmol/hr)	
x_{12}: acid gas stream temperature (°C)	
x_{13}: acid gas stream pressure (bar)	

Table 3. Labeling of Fault Classes

AGRU system condition	denoted label	indexed label
normal	N	[1 0 0 0]
foaming	A	[0 1 0 0]
damaged tray	B	[0 0 1 0]
fouling	C	[0 0 0 1]

output/target classes were normal, foaming, damaged tray, and fouling conditions.

In this study, a total of $n = 22{,}666$ samples were generated and $m = 21{,}332$ samples (excluding the initial 1334 samples, which represent the nonsteady state) were treated as the base input data for subsequent development of machine learning models for fault diagnosis.

3.2. Preparation of the Data.

MATLAB's Deep Learning Toolbox (2021a) was used to build the models. First of all, preprocessing of data was performed. Normalization of the raw multivariate time series data was done to yield variables with zero mean and unit variance, as shown in eq 1, where x_{ij} and X_{ij} are the scaled and raw data point respectively at the i^{th} sample and j^{th} variable, and μ_j and σ_j are the mean and standard deviation, respectively, of the j^{th} variable across m samples.

$$x_{ij} = \frac{X_{ij} - \mu_j}{\sigma_j};\ i = 1, 2, \cdots, m;\ j = 1, 2, \cdots, n \qquad (1)$$

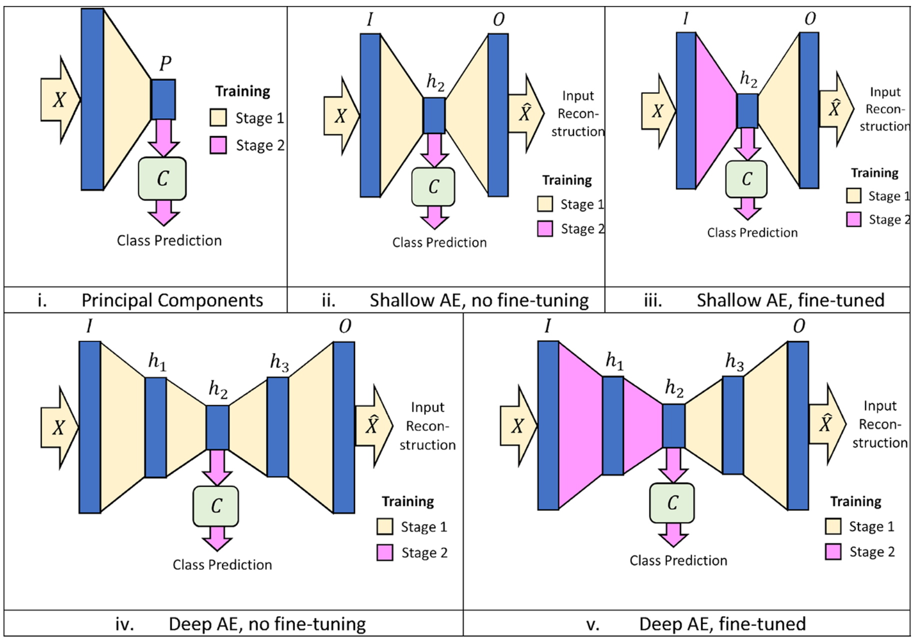
The fault conditions are labeled as shown in Table 3.

The scaled and normalized data together with their labels were randomly divided into a training set (70% of the data or 14,936 samples) and a test set (30% of the data or 6396 samples).

3.3. Analytical Methodology.

The overall analytical methodology consisted of the use of autoencoders to extract features from the operational variables in the AGRU and then using these features as predictors in a supervised fault classification model. Five related approaches were used to achieve this. The first was based on the use of principal component analysis (PCA), and the other two were based on a shallow and a deep sparse autoencoder, each of which was trained in two different ways.

A two-stage training methodology was followed. In the first stage, an autoencoder was trained in unsupervised mode with target function the reconstruction error of the predictors or input data. The autoencoder in this first stage is defined by an input layer (I) with the same number of nodes as the number of input variables, either one (h_2) or three hidden layers (h_1, h_2, and h_3) and an output layer (O) with the same number of nodes as the input layer. All the models were also provided with a classification (SoftMax) layer. The outputs of the bottleneck



C: classifier (SoftMax layer), I: input layer, h_1: first (encoding) hidden layer, h_2: second (bottleneck) hidden layer, h_3: third (decoding) hidden layer, O: output layer of autoencoder, X: matrix of process data, $\hat{X}$: reconstructed matrix of process data,

Figure 4. Models used in fault diagnosis of the AGRU. (i) Principal component model with scores serving as predictors to the classifier, (ii) shallow autoencoder with no fine-tuning, (iii) shallow autoencoder with fine-tuning, (iv) deep autoencoder with no fine-tuning, (v) deep autoencoder with fine-tuning.

Table 4. Training Parameters of Shallow Sparse Autoencoder

training parameter	value/description			
number of hidden nodes in the bottleneck layer	1	2	3	4
activation/transfer function in encoder	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)
activation/transfer function in decoder	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)
training algorithm	scaled conjugate gradient descent (trainscg)	scaled conjugate gradient descent (trainscg)	scaled conjugate gradient descent (trainscg)	scaled conjugate gradient descent (trainscg)
L2 regularizer coefficient	0.001	0.001	0.001	0.001
sparsity regularizer coefficient	16	16	16	16
desired sparsity parameter	0.075	0.05	0.1	0.05
max epochs	1000	1000	1000	1000

Table 5. Process Variables Used as Predictors

stream	variable
sweet gas	CO_2 composition (kmol/kmol)
	total flowrate (kmol/hr)
	temperature (°C)
	pressure (bar)
lean solvent	total flowrate (kmol/hr)
	temperature (°C)
	pressure (bar)
rich solvent	total flowrate (kmol/hr)
	temperature (°C)
	pressure (bar)
acid gas	total flowrate (kmol/hr)
	temperature (°C)
	pressure (bar)

layer (h_2) served as input to this classification layer (c), which had as the output four predicted process conditions or classes (normal operation and the three fault conditions).

The second hidden layer is also referred to as the bottleneck or feature extraction layer of the autoencoder, while layers h_1 and h_3 can be referred to as the encoding and decoding hidden layer of the network. These nodes in the hidden layers typically have sigmoidal activation functions, which allow nonlinear encoding and decoding of the data. The three-hidden layer autoencoder will be referred to as a deep autoencoder, although deep autoencoders would typically have more than three hidden layers.

It should also be noted that when layers h_1 and h_3 are absent and the network and the bottleneck layers have linear nodes, the model is equivalent to principal component analysis. Feature extraction is carried out to extract the most significant and essential information from the input data set with the goal of obtaining a more effective representation of the features via linear or nonlinear features learning.

These models are illustrated in Figure 4. Figure 4i is a principal component model, which is fitted directly to the data, as explained in Section 3.3.1, as this is a more efficient approach than fitting the equivalent autoencoder mentioned above. The principal components (P) served as input to the SoftMax classifier (c), and during the second stage of training, only the weights of this layer are trained. The second variant (ii) is a shallow autoencoder with a bottleneck layer (h_2) only. After the first stage training of the autoencoder to reconstruct the input data, the features generated from its bottleneck layer (h_2) served as input to the SoftMax layer (c). The third variant (iii) is identical to (ii), except that during second stage training, both the SoftMax weight layer and the weights of the encoding section (h_2) of the autoencoder are trained. The weights of h_2 are not trained ab initio but are further trained or fine-tuned, following stage 1 training. Model (iv) is the same as model (ii), except that the autoencoder has three hidden layers. In this case, again, the output of the bottleneck layer (h_2) serve as input to the SoftMax classification layer and only the weights of this layer are trained in the second stage.

In the same way, model (v) is identical to model (iii), except that it also has three hidden layers. Unlike model (iv), during the second stage of training, the layers of the SoftMax layer, as well as those of the h_2 and h_2 hidden layers, are trained. Model (v) is the most powerful version of the five models, as it has a deep structure that is fine-tuned to identify the fault conditions. More detail on each of the models is given in subsections 3.3.1−3.3.3.

3.3.1. Principal Component Analysis (PCA). With principal component analysis (PCA), a new set of variables or principal components is obtained from linear combinations of the original variables and they are ordered by the amount of variance they explain in the data. The first principal component explains the most variance, and each subsequent component explains as much of the remaining variance as possible. A reduced set of features can be obtained by selecting only the top principal components.

More formally, the decomposition of the data matrix **X** via PCA is given by eq 2, where $\mathbf{T}_k$ is the score matrix with a dimension of $m \times k$, $\mathbf{P}_k$ is the loading matrix with a dimension of $n \times k$, $\mathbf{E}_k$ is the residual matrix with a dimension of $m \times n$, and $k \ll n$ is the total number of principal components retained.

$$\mathbf{X} = \mathbf{T}_k\mathbf{P}_k^{\mathrm{T}} + \mathbf{E}_k \tag{2}$$

Retention of k principal components was based on the cumulative variance (CV) of the components, computed according to eq 3. K is the minimum number of components required to ensure CV ≥ 0.8.[18] In eq 3, CV_k is the cumulative variance across the first k number of retaining principal components, and λ_l the lth eigenvalue of the model, with l = 1, 2, ... k.

$$\mathrm{CV}_k = \frac{\sum_{l=1}^{k}\lambda_l}{\sum_{l=1}^{n}\lambda_l} \tag{3}$$

The training data set was input into the PCA model. The principal component variances, which are the eigenvalues λ_l, were computed, and a CV value of approximately 0.8 was used a criterion for the number of principal components to retain. The CV was computed from the eigenvalues starting from CV_1 until a CV of >0.8 was reached, i.e., CV_4= 0.853.

3.3.2. Shallow Sparse Autoencoder. An autoencoder maps an input vector x into a latent representation (features) H as indicated by eq 4. This is followed by decoding the features H to reconstruct the original sample vector x as outputs $\hat{x}$, as indicated in eq 5.

$$H = \Phi(W_1x + b_1) \tag{4}$$

$$\hat{x} = \Phi(W_2h + b_2) \tag{5}$$

$$\Phi(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

In eqs 4 and 5, Φ is an activation function, which is typically a sigmoidal function for both encoding and decoding processes as expressed in eq 6. W_1 and W_2 are the weight matrices associated with encoding and decoding, respectively, and likewise, b_1 and b_2 are bias vectors associated with encoding and decoding, respectively.

Optimization of these parameters (W_1, b_1, W_2, and b_2) is carried out with the fundamental objective of the average reconstruction error, as shown by eq 7, where $L(x_{ij}, x'_{ij})$ is the loss function, which is set to be a squared error function in this study as expressed in eq 8. A scaled conjugate gradient descent algorithm was applied in the autoencoder training and optimization.

$$[W_1^*, b_1^*, W_2^*, b_2^*] = arg\ min_{W_1,b_1,W_2,b_2}\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}L(x_{ij}, x'_{ij})}{m} \tag{7}$$

$$L(x_{ij}, x') = (x_{ij} - x'_{ij})^2 \tag{8}$$

To prevent overfitting of the data, optimization was made subject to a sparsity constraint[19,20] as indicated by eq 9.

$$[W_1^*, b_1^*, W_2^*, b_2^*] = arg\ min_{W_1,b_1,W_2,b_2}\left(\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(x_{ij} - x'_{ij})^2}{m} + \alpha\Omega_{\text{weights}} + \beta\Omega_{\text{sparsity}}\right) \tag{9}$$

$$\Omega_{\text{weights}} = \frac{\sum_{a=1}^{p}\sum_{i=1}^{m}\sum_{j=1}^{n}(W_{ij}^{(a)})^2}{2} \tag{10}$$

$$\Omega_{\text{sparsity}} = \sum_{c=1}^{q}\text{KL}(\rho \parallel \hat{\rho}_c) \tag{11}$$

As shown in eqs 9−11, α is the L$_2$ regularization coefficient, Ω_{weights} is the L$_2$ regularizer, β is the sparsity coefficient, Ω_{sparsity} is the sparsity regularizer, p is the total number of hidden layer(s), W is the weight matrix, and q is the total number of hidden nodes in the hidden layer. KL($\rho \parallel \hat{\rho}_c$) and $\hat{\rho}_c$ are further expressed mathematically in eqs 12 and 13, respectively.

$$\text{KL}(\rho \parallel \hat{\rho}_c) = \rho\log\left(\frac{\rho}{\hat{\rho}_c}\right) + (1 - \rho)\log\left(\frac{1 - \rho}{1 - \hat{\rho}_c}\right) \tag{12}$$

$$\hat{\rho}_c = \frac{\sum_{i=1}^{m}h_cx_i}{m} \tag{13}$$

As shown in eqs 12 and 13, KL is the Kullback−Leibler divergence, ρ is the desired sparsity parameter, which is typically a relatively small value ($\rho \approx 0$), and $\hat{\rho}_c$ is the average activation of the hidden node c. It is noted that the ρ value implies the desired model input sample proportion that a hidden unit acts upon. In this study, KL was used to enforce the sparsity enforce constraint, $\hat{\rho}_c \approx \rho$.

Upon completion of the first stage training as discussed above, the features extracted from the bottleneck layer (h_2) of the autoencoder served as the inputs to the classification model. The training parameters of the shallow sparse autoencoder are as shown in Table 4.

3.3.3. Deep Sparse Autoencoders. To develop a deep sparse autoencoder, two individual shallow sparse autoencoders with the implementation of the similar development approach as discussed in Section 3.3.2 were first constructed. Layerwise training/pretraining was conducted on one hidden layer at a time in a sequential manner. The model input data x was presented to the first shallow sparse autoencoder, whereupon the features captured by the first shallow sparse autoencoder were then treated as the inputs of the second shallow sparse autoencoder for further feature extraction. The two shallow sparse autoencoders were subsequently stacked together to create a deep sparse autoencoder.

3.4. Classification (Fault Detection and Identification).

The classifier assigned the unknown AGRU conditions to their respective classes; A (normal), B (foaming), C (damaged tray), and D (fouling). The SoftMax function was used for this purpose by introducing a SoftMax layer at the end of each of the PCA and autoencoder models.

A SoftMax function, which is also known as a normalized exponential, is recognized in multiclass generalization of the logistic sigmoid function.[21] The Softmax function is used to represent a discrete variable with u possible values as a probability distribution. The inputs of the SoftMax function consisted of components or features, which were recomputed to be interpretable as probabilities, i.e., each component having a value between 0 and 1 and all component values summing to unity.

This SoftMax function was used to map the features extracted by the PCA and autoencoder models onto a probability distribution over the four classes (normal, foaming, damaged tray, fouling). The fault condition with the highest probability was identified as the prevailing operational condition of the AGRU. These calculations are represented by eqs 14 and 15.

$$y_r(x')_i = \frac{e^{(x'_i)}}{\sum_{j=1}^{k}e^{(x_j)}} \tag{14}$$

$$0 \le y_r \le 1;\ \sum_{j=1}^{k}y_j = 1 \tag{15}$$

3.5. Fine-Tuning of Sparse Autoencoder with Classification Neural Network Model.

As mentioned earlier, to optimize each layer, layerwise training/pretraining was carried out. Several studies have shown that the implementation of this approach to training deep neural networks is most likely to generate relatively low performance.[22−24] This is due to the fact that deep neural networks with sets of large initial weights are prone to generation of poor local minima, while deep neural networks with sets of small initial weights are susceptible to generation of small gradients at bottom layers. Both conditions

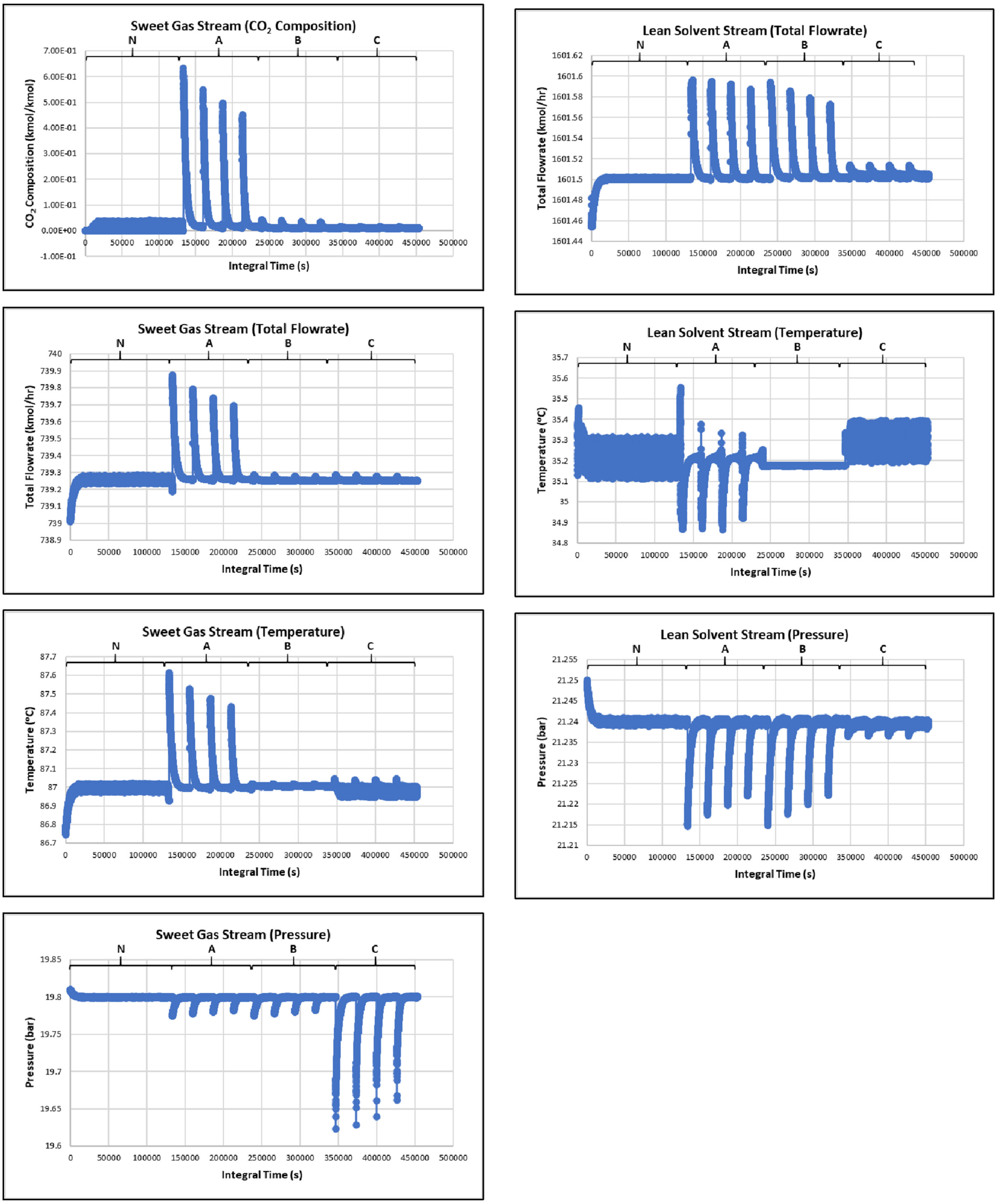


Figure 5. Dynamic behavior of sweet gas stream (left column) and lean solvent stream (right column).

may lead to deterioration in terms of the performance and applicability of deep neural networks with multiple layers.[25]

This problem can be obviated by a fine-tuning of the weights of the deep neural network.[26,27] For this purpose, scaled conjugate gradient descent was used as the backpropagation algorithm. In order to investigate the effect of performing fine-tuning on the models, a comparative analysis in terms of the fault detection and identification performance in AGRU between models with and without fine-tuning was done. Classification accuracy (%) was used as the indicator in the evaluation of the

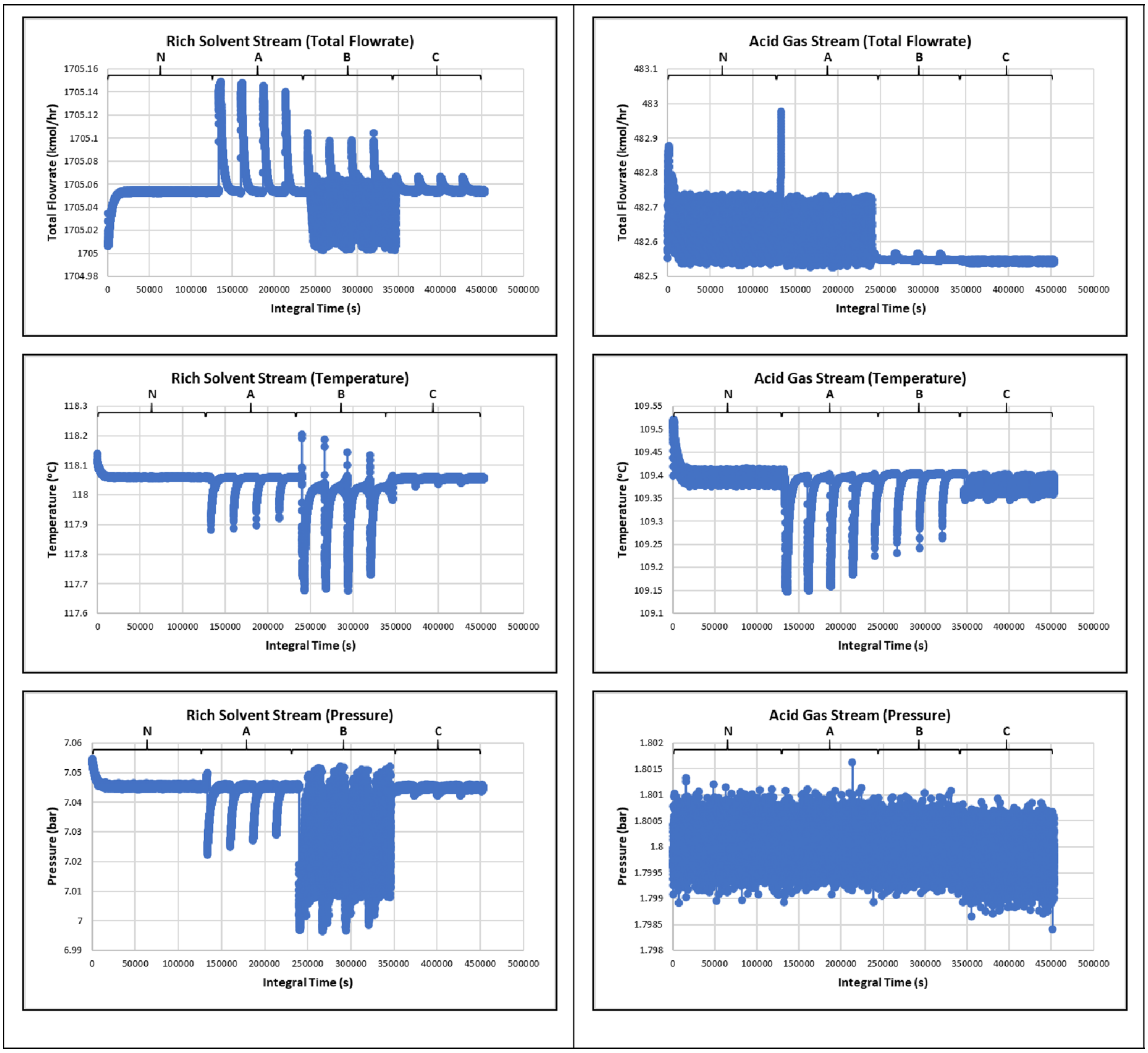


Figure 6. Dynamic behavior of rich solvent stream (left column) and acid gas stream (right column).

model performance in terms of fault detection and identification in the AGRU.

4. RESULTS AND DISCUSSION

4.1. Dynamic Behavior of Process Variables. The time series data were extracted from each of the relevant process variables as tabulated in Table 5.

The dynamic behavior in terms of time series data for each process variable, which includes the behavior during normal (N), foaming (A), damaged tray (B), and fouling (C) conditions, is presented in Figure 5a (sweet gas stream), Figure 5b (lean solvent stream), Figure 6a (rich solvent stream), and Figure 6b (acid gas stream). In Figures 5 and 6, the data for N, A, B, and C are extracted and plotted together in the same graphs to highlight the difference in the behaviors of the process variables when foaming, damaged tray, and fouling are simulated.

During dynamic process simulation, an initialization process was performed by Aspen HYSYS Dynamics whenever a fault is introduced into the system before resulting with the desired dynamic condition. Thus, the process variable behavior during the initialization stage is omitted from the time series data in order to preserve the validity of the results obtained.

As shown in Figures 5 and 6, it can be observed that the occurrence of the three faults resulted with relatively similar spike-like dynamic behavior that varied only in terms of the magnitude of the spikes. As foaming, damaged tray, or fouling occurred in the AGRU, the sour gas sweetening efficiency gradually decreased owing to the presence of undesired operating conditions for the absorption of acid gases into the solvent solution.

Hence, the CO_2 composition of the resulting sweet gas increased, resulting in poorer quality of the natural gas produced. Based on heuristic applied in typical AGRU design and operation,[17] a feedback control strategy is usually employed in order to control the CO_2 composition of the outlet sweet gas (controlled variable) by manipulating the flowrate of the inlet

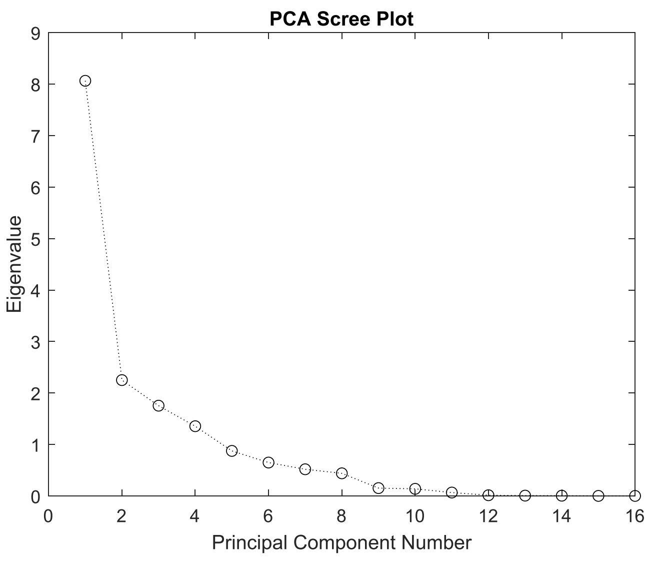


Figure 7. PCA scree plot.

lean solvent (manipulated variable) of the absorber unit. Thus, once the CO_2 composition of the sweet gas deviates from the specified setpoint value, the feedback controller will send a signal to the control valve of the lean solvent by increasing its flowrate in order to absorb more CO_2 from the natural gas, further sweetening the sour gas till the resulting sweet gas CO_2 composition is brought back to its desired value. As depicted in Figure 5, with the introduction of faults into the system, it can be observed that a sudden increment occurred in both the CO_2 composition of the sweet gas stream and total flowrate of the lean solvent stream, in accordance with the theories mentioned earlier.

The dynamic process model of the AGRU was validated with heuristic rules.[17] The time series data were subsequently used as the inputs in the machine learning models designed for fault diagnosis.

4.2. Principal Component Analysis. Four principal components were retained in the PCA model, collectively accounting for 83.5% of the total variance of the 13 variables, as shown in Figure 7. These components served as inputs or predictors in the subsequent classification model.

4.3. Autoencoders. The deep sparse autoencoder was developed with four hidden nodes in the bottleneck hidden layer (h_2), analogous with the four component PCA model. The training parameters of the deep sparse autoencoder are shown in Table 6. The same parameters were used for the shallow autoencoders (bottleneck layers only).

4.4. Classification Accuracy. Generally, validation data sets were not required when training autoencoders. However, during fine-tuning of the autoencoders, a small validation data set was used to optimize the hyperparameters of the models. This included the number of nodes in the hidden layers of the deep autoencoders, the L2 regularization coefficients, sparsity parameters, etc., and was done with a 65% training, 5% validation, and 30% test data split.

All of the results for the models in terms of classification accuracy over 20 runs are shown in Table 7 and Figure 8. As shown in Table 7 and Figure 8, the first observation is that, with the increase of feature number from one up to four, the accuracies show an increasing trend using all the methods except using deep sparse autoencoder with fine-tuning, whereas the latter has achieved near perfect performance with any number of features.

When only one feature is retained, all models involving sparse autoencoders performed markedly better than one-component

Table 6. Training Parameter of Deep Sparse Autoencoder

first hidden layer (encoder)				
training parameter	value/description			
number of hidden nodes in bottleneck layer	7	7	7	7
activation/transfer function in encoder	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)
activation/transfer function in decoder	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)
training algorithm	scaled conjugate gradient descent (trainscg)	scaled conjugate gradient descent (trainscg)	scaled conjugate gradient descent (trainscg)	scaled conjugate gradient descent (trainscg)
L2 regularizer coefficient	0.001	0.001	0.001	0.001
sparsity regularizer coefficient	16	16	16	16
desired sparsity parameter	0.25	0.2	0.1	0.2
max epochs	1000	1000	1000	1000
second hidden layer (bottleneck)				
training parameter	value/description			
number of hidden nodes in bottleneck layer	1	2	3	4
activation/transfer function in encoder	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)
activation/transfer function in decoder	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)	logistic sigmoid (logsig)
training algorithm	scaled conjugate gradient descent (trainscg)	scaled conjugate gradient descent (trainscg)	scaled conjugate gradient descent (trainscg)	scaled conjugate gradient descent (trainscg)
L2 regularizer coefficient	0.00001	0.00001	0.00001	0.00001
sparsity regularizer coefficient	16	16	16	16
desired sparsity parameter	0.15	0.1	0.1	0.2
maximum number of epochs	2000	2000	2000	2000

Table 7. Result of Classification Accuracy over 20 Runs with Various Numbers of Features[a]

Machine Learning Model Structure* $I - h_1 - h_2 - h_3 - O$, C	**Average Accuracy (%) over 20 Runs with Various Feature Numbers**			
	1	2	3	4
Deep sparse autoencoder with fine-tuning ($h_1 = 7, h_2 = [1\ 4], h_3 = 7$)	99.73	99.87	99.84	99.91
Deep sparse autoencoder without fine-tuning $h_1 = 7, h_2 = [1\ 4], h_3 = 7$)	61.45	77.17	82.98	90.73
Shallow sparse autoencoder with fine-tuning ($h_1 = 0, h_2 = [1\ 4], h_3 = 0$)	84.78	95.37	99.45	99.39
Shallow sparse autoencoder without fine-tuning ($h_1 = 0, h_2 = [1\ 4], h_3 = 0$)	40.41	74.49	86.06	89.47
Principal component analysis (1 to 4 PCs)	27.03	78.16	84.45	87.22

[a]*, All models had 13 inputs and 13 outputs (1st stage training) and the SoftMax layer had h_2 inputs and four outputs.

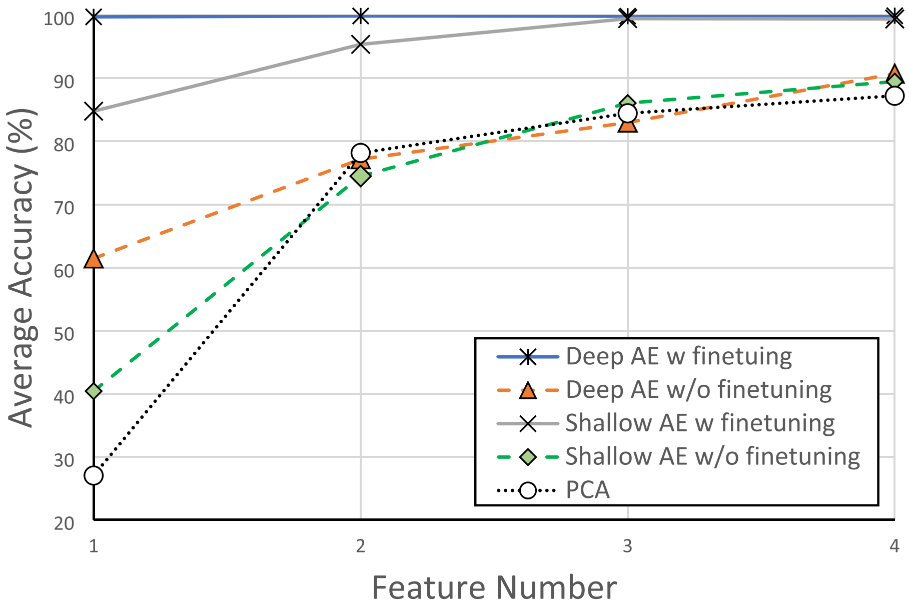


Figure 8. Result of classification accuracy over at least 20 runs with various feature numbers.

PCA. This is related to the autoencoder's ability to generate nonlinear feature representations, while PCA can only learn linear feature representations. Fine-tuning contributes to the further improvement in performance, as it facilitates the extraction of learnt features designed for optimal performance of the classifier.

Even with only one feature, the deep sparse autoencoder without fine-tuning is performing reasonably well at an average accuracy of over 60%. This indicates the potential of utilizing a deep sparse autoencoder as feature extractors in an unsupervised manner, like PCA.

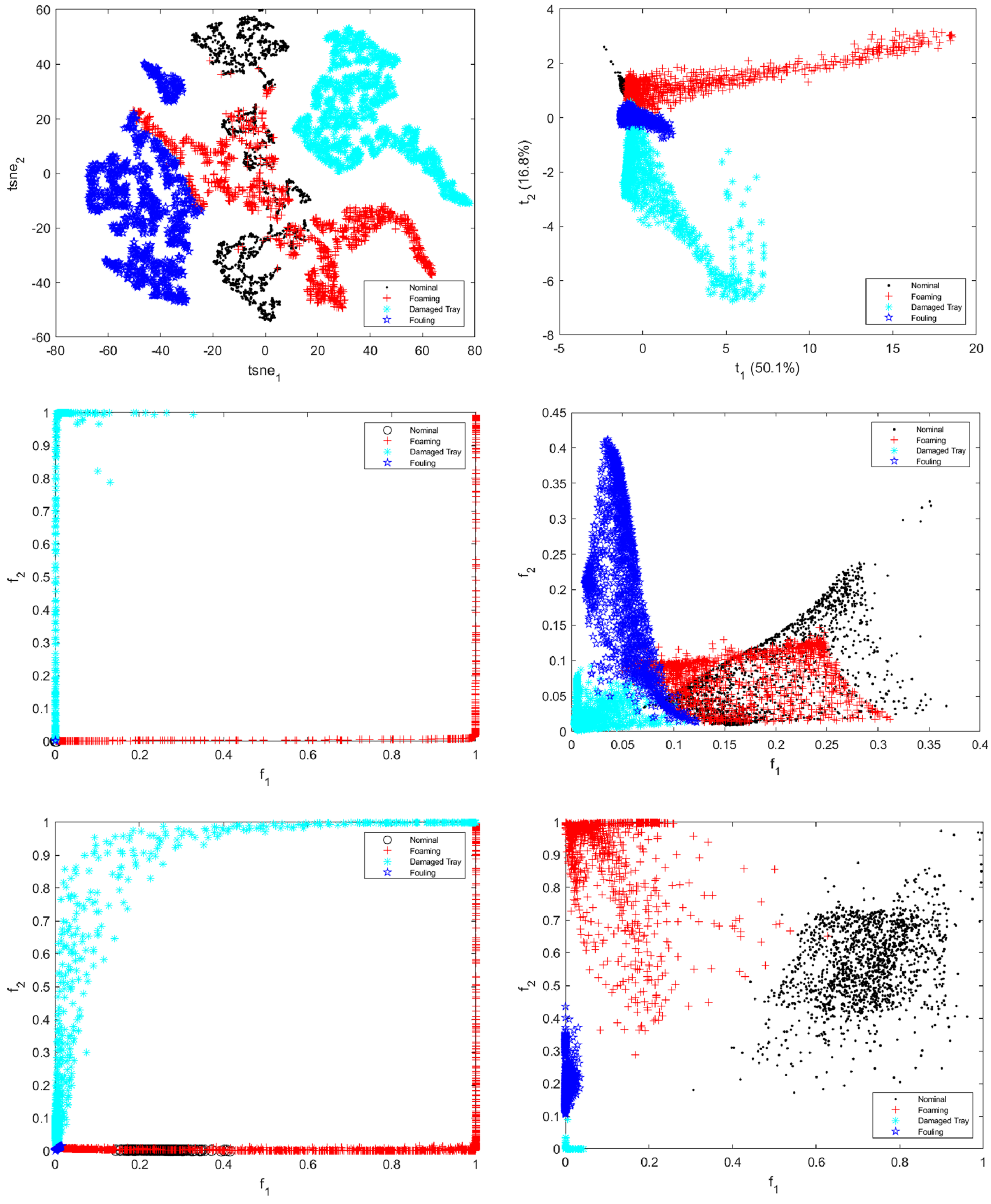


Figure 9. t-SNE (top, left), principal component (top, right), untuned two-node shallow autoencoder (middle, left), untuned two-node deep autoencoder (middle, right), fine-tuned shallow autoencoder (bottom, left), and fine-tuned deep autoencoder (bottom, right) score plot of the variables in the test data set.

With more than one feature, there is little difference in the performance between both the sparse autoencoders without fine-tuning and PCA, indicating that any of these models can be used as the alternative to one another in such cases. Meanwhile, sparse autoencoders with fine-tuning consistently performed better than PCA, with a nominal increase in accuracy of 12.2 to 17.2 percentage points. Fine-tuning during network training forces the models to learn more effective features and consequently increases the model performance in fault detection

Table 8. Hyperparameters of Support Vector Data Description Model Used to Generate Confidence Limits Shown in Figure 10

kernel type	kernel parameter	cost
Gaussian	1000	0.9

and identification in AGRU systems, in accordance with the theory discussed in Section 3.5.

4.5. Visualization and Process Monitoring. To gain further insights and qualitatively assess the discriminative power of the different models, the AGRU data can be visualized as shown in Figure 9. This figure shows the 13 plant variables projected to two dimensions with a t-SNE algorithm (top, left), a principal component model (top, right), the untuned autoencoders (middle), and the fine-tuned autoencoders (bottom). Only the deep autoencoder with fine-tuning shows nearly complete separation of all four fault classes (bottom, right) in two dimensions.

Moreover, as indicated by the plots in Figure 9, foaming is clearly the most difficult fault condition to distinguish from normal operation. Even so, the deep fine-tuned autoencoder managed to discriminate between these two conditions with a high level of confidence.

These scores can form the basis for the construction of process monitoring maps based on the future scores generated by the fine-tuned deep autoencoder, as indicated in Table 8. Confidence limits are superimposed on each of the operating regimes by use of a support vector data description (SVDD) model, the parameters of which are summarized in Table 8. This model is fitted to the first two principal component scores of the data.

SVDD models are unsupervised learning approaches,[28] in which the data are enclosed in a hypersphere. Although SVDD is a hard margin classifier, a cost parameter $C_c \in [0, 1]$ is often used in optimization to allow some observations to lie outside the enclosing hypersphere. The larger the value of C_c, the more expensive excluding observations from the hypersphere becomes.

5. CONCLUSIONS

In this study, supervised detection of faulty operating conditions in AGRUs based on the use of autoencoders with SoftMax output layers was investigated. These conditions were foaming, damaged trays, and fouling. Five different models were considered, namely, a principal component model, a shallow sparse autoencoder without fine-tuning, a shallow sparse autoencoder with fine-tuning, a deep sparse autoencoder without fine-tuning, and a deep sparse autoencoder with fine-tuning.

Discrimination between nominal operational conditions and dynamic process conditions associated with foaming were relatively difficult, and this could not be achieved by all the models considered. However, among all these models, the deep sparse autoencoder with fine-tuning always could identify all fault conditions near perfectly based on the use of as few as two or three features only. This is ideal for automated visualization and monitoring systems when fitted with appropriate confidence limits to delineate different operational regimes.

Visualization of the data was accomplished with a t-SNE score plot of all the original features, PCA score plot of the first two principal components, as well as equivalent score plots based on the autoencoder features. Visualization indicated the relative

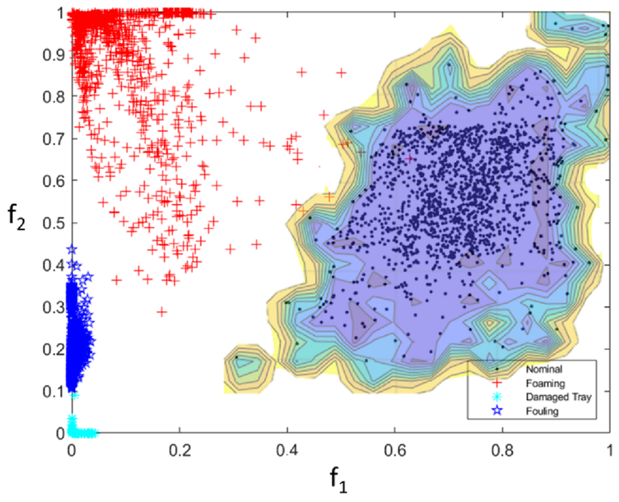


Figure 10. Process monitoring map of AGRU operation with 99% SVDD confidence limits delineating normal operation.

difficulty in distinguishing foaming from normal operating conditions. In addition, foaming had a significantly more varied effect on process operations than damaged trays, for example, a can be seen from the distribution of the data in Figure 10. However, a deep sparse autoencoder with fine-tuning can largely eliminate this kind of misclassification.

Further validation of the selected model is proposed to be carried out in future work based on real data from an industrial AGRU.

AUTHOR INFORMATION

Corresponding Author

Haslinda Zabiri − *Department of Chemical Engineering and CO2RES, Institute of Contaminant Management, Universiti Teknologi PETRONAS, Seri Iskandar 32610 Perak, Malaysia*; orcid.org/0000-0003-1821-1028; Email: haslindazabiri@utp.edu.my

Authors

Tan Kaiyun Kathlyn − *Department of Chemical Engineering, Universiti Teknologi PETRONAS, Seri Iskandar 32610 Perak, Malaysia; PETRONAS, Kuala Lumpur 50088, Malaysia*

Chris Aldrich − *Western Australian School of Mines, Curtin University, Perth 6845 WA, Australia*

Xiu Liu − *Western Australian School of Mines, Curtin University, Perth 6845 WA, Australia*

Ahmad Azharuddin Azhari Mohd Amiruddin − *Department of Chemical Engineering, Universiti Teknologi PETRONAS, Seri Iskandar 32610 Perak, Malaysia*

Complete contact information is available at: https://pubs.acs.org/10.1021/acsomega.2c08109

Funding

Collaborative Research Fund (CRF) (Cost center 015ME0-215) and YUTP-FRG (015LC0-477).

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

The authors would like to thank Universiti Teknologi PETRONAS (UTP), Malaysia and Matrix Technologies, Australia, for the facility and technical support provided to complete this work. This work is performed under the Collaborative Research Fund (CRF) (Cost center 015ME0-215), and jointly supported by YUTP-FRG (015LC0-477).

REFERENCES

(1) S., Buda; Y.M., Yak; M.F., Jais *Method for determination of dissolved hydrocarbon in amine solvent by gas chromatography* (2008) International Gas Research Conference Proceedings, 1, pp. 550−561.

(2) Paul, P.; Bhattacharyya, D.; Turton, R.; Zitney, S. E. Dynamic model-based sensor network design algorithm for system efficiency maximization. *Comput. Chem. Eng.* **2016**, *89*, 27−40.

(3) S., Pradittiamphon; S., Wongsa *Fault detection and isolation of acid gas removal units in a gas separation process using PLS, in 2016 International Conference on Instrumentation*, Control and Automation (ICA), Bandung, 2016.

(4) Hakimi, M., Omar, M.B., Ibrahim, R. 2023. Application of neural network in predicting H2S from an acid gas removal unit (AGRU) with different compositions of solvents. Sensors, 23(2), art. no. 1020, DOI: 10.3390/s23021020.

(5) Al-Sinbol, G.; Perhinschi, M. G. Development of an artificial immune system for power plant abnormal condition detection, identification, and evaluation. *International Review of Automatic Control* **2017**, *10*, 218−228.

(6) Perhinschi, M. G.; Al-Sinbol, G. Artificial dendritic cell algorithm for advanced power system monitoring. *International Review of Automatic Control* **2016**, *9*, 330−340.

(7) Al-Sinbol, G.; Perhinschi, M. G. Generation of power plant artificial immune system using the partition of the universe approach. *International Review of Automatic Control* **2016**, *9*, 40−47.

(8) Ha, D.; Ahmed, U.; Pyun, H.; Lee, C.-J.; Baek, K. H.; Han, C. Multi-mode operation of principal component analysis with k-nearest neighbor algorithm to monitor compressors for liquefied natural gas mixed refrigerant processes. *Comput. Chem. Eng.* **2017**, *106*, 96−105.

(9) Kumar, A.; Bhattacharya, A.; Flores-Cerrillo, J. Data-driven process monitoring and fault analysis of reformer units in hydrogen plants: Industrial application and perspectives. *Comput. Chem. Eng.* **2020**, *136*, No. 106756.

(10) Xiao, B.; Li, Y.; Sun, B.; Yang, C.; Huang, K.; Zhu, H. Decentralized PCA modeling based on relevance and redundancy variable selection and its application to large-scale dynamic process monitoring. *Process Saf. Environ. Prot.* **2021**, *151*, 85−100.

(11) M., Madakyaru, F., Harrou, Y., Sun, "*Monitoring Distillation Column Systems Using Improved Nonlinear Partial Least Squares-Based Strategies*," in IEEE Sensors Journal, vol. *19*, no. 23, pp. 11697−11705, 1 Dec.1, 2019, DOI: 10.1109/JSEN.2019.2936520.

(12) Figueroa Barraza, J.; Guarda Bräuning, L.; Benites Perez, R.; Morais, C. B.; Martins, M. R.; Droguett, E. L. Deep learning health state prognostics of physical assets in the Oil and Gas industry. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability.* **2022**, *236*, 598−616.

(13) Behbahani, R. M.; Jazayeri, H.; Hajmirzaee, S. Fault detection and diagnosis in a sour gas absorption column using neural network. *Chem. Eng. Technol.* **2009**, *32*, 840−845.

(14) Qi, M.; Jang, K.; Cui, C.; Moon, I. Novel control-aware fault detection approach for non-stationary processes via deep learning-based dynamic surrogate modeling. *Process Saf. Environ. Prot.* **2023**, *172*, 379−394.

(15) L., Beke, "*Contamination in amine systems*," September 2010. [Online]. Available: https://refiningcommunity.com/wp-content/uploads/2017/06/Contamination-in-Amine-Systems-Beke-BSDT-SRU-Calgary-2010.pdf. [Accessed 31 January 2020].

(16) Abbas, T.; Ghauri, M.; Rashid, Z.; Shahid, M. Dynamic simulation of sweetening process of natural gas. *Canadian Journal on Chemical Engineering & Technology* **2011**, *2*, 156−161.

(17) L., Addington, C., Ness, "*An evaluation of general "rules of thumb" in amine sweetening unit design and operation*," in Gas Processors Association (GPA) Annual Convention, Austin, 2010.

(18) A., Rea, W., Rea, "How many components should be retained from a multivariate time series PCA?," *arXiv* preprint, vol. arXiv, p. 1610.03588, **2016**.

(19) Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* **2010**, *11*, 3371−3408.

(20) Olshausen, B. A.; Field, D. J. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vis. Res.* **1997**, *37*, 3311−3325.

(21) C. M., Bishop, *Pattern recognition and machine learning*, New York: Springer, 2006.

(22) H., Drucker, R., Schapire, P., Simard, "*Improving performance in neural networks using a boosting algorithm*," in Advances in neural information processing systems 5, San Francisco, Morgan Kaufmann Publishers, 1993, pp. 42−49.

(23) Kambhatla, N.; Leen, T. K. Dimension reduction by local principal component analysis. *Neural Computation* **1997**, *9*, 1493−1516.

(24) Tenenbaum, J. B.; de Silva, V.; Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science* **2000**, *290*, 2319−2323.

(25) Hinton, G. E.; Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504−507.

(26) Hinton, G. E.; Osindero, S.; Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Computation* **2006**, *18*, 1527−1554.

(27) Y., Bengio, P., Lamblin, D., Popovici, H., Larochelle, "*Greedy layer-wise training of deep networks*," in Advances in neural information processing systems 19, vol. *19*, Cambridge, MIT Press, 2007, pp. 153−160.

(28) Tax, D. M.; Duin, R. P. *Support Vector Data Description. Machine Learning* **2004**, *54*, 45−66.