Application of Deep Learning in Pharmaceutical Processes: Monitoring, Diagnosis and Modeling

by

Mohammad Aghaee Foroushani

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Chemical Engineering

Waterloo, Ontario, Canada, 2024

© Mohammad Aghaee Foroushani 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Prashant Mhaskar Professor, Dept. of Chemical Engineering, McMaster University
Supervisor(s):	Hector Budman Professor, Dept. of Chemical Engineering, University of Waterloo
Internal Member:	Luis Ricardez Sandoval Professor, Dept. of Chemical Engineering, University of Waterloo
Internal-External Member:	Olga Vechtomova Professor, Dept. of Management Science and Engineering, University of Waterloo
Other Member(s):	Joshua Pulsipher Professor, Dept. of Chemical Engineering, University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of contribution

For Chapter 3, the author and Piyush Agarwal implemented the documented methodologies, obtained the numerical results. Piyush Agarwal and Hector Budman were involved in the conceptualization of the work. All authors read and approved the final manuscript.

Abstract

In recent years, artificial intelligence (AI) has emerged as a transformative technology in various fields, including pharmaceutical processes. The integration of AI in pharmaceutical manufacturing offers significant potential for enhancing process monitoring, fault detection, and predictive modeling. By leveraging advanced machine learning and deep learning algorithms, AI enables the development of detailed models that can handle the complexities and nonlinearities of biopharmaceutical processes. These models facilitate real-time monitoring, allowing for the early detection of process deviations and faults, which is crucial for maintaining product quality and efficiency. Additionally, AI-driven modeling approaches provide deeper insights into the dynamic behavior of biochemical pathways, offering valuable tools for optimizing production and ensuring regulatory compliance.

The current work investigates the application of machine learning/deep learning algorithms in chemical engineering processes, with a particular focus on bio-pharmaceutical processes. Due to tight regulations, pharmaceutical operations are operated within a small range of conditions, resulting in an insufficient amount of faulty data for training a supervised process monitoring model. Furthermore, critical information about faults, such as the type of fault, the time of fault occurrence, and the magnitude of the fault, is often unavailable in bio-processes due to limited availability of informative online sensors. This lack of detailed fault data makes it impractical to train supervised models for fault detection and diagnosis, as these models require comprehensive and accurately labeled datasets to function effectively. Consequently, developing robust and reliable unsupervised models becomes essential for effective process monitoring and fault diagnosis in these scenarios. To this end, this work focuses on unsupervised learning based algorithms that do not require labeled data and instead they identify deviations from normal operating conditions. Within the framework of unsupervised learning the following aspects are tackled:

i- An unsupervised PLS-AE (Partial Least Squares-Autoencoder) model is proposed

for online batch process monitoring. Detection by this algorithm is based on a novel loss function for training which focuses on maximizing the fault detection rate. This results in more accurate process monitoring models compared to those trained with the classical loss function of minimizing the mean square of reconstruction error. It will be shown that the PLS-AE model trained with the novel loss function and utilizing dynamic upper control limits can surpass other models, including PCA, PLS, and PLS-AE models.

ii- Following the development of the PLS-AE model approach this thesis further develops unsupervised deep LSTM-AE (Long Short Term Memory Autoencoder) models which are more suitable for capturing dynamic nonlinear patterns. These models utilize the novel loss function and dynamic upper control limits, enhancing their fault detection capabilities. The effectiveness of these LSTM-AE models is demonstrated through rigorous testing on three case studies: the penicillin batch fermenter simulator, the real-world whooping cough vaccine manufacturing process at Sanofi Toronto, and the lab-scale diphtheria process at Sanofi Toronto.

iii- To address the interpretability of deep learning models, this work proposes two novel algorithms for generating contribution plots for H^2 and SPE statistics, enabling the identification of the root causes of faults. This advancement significantly improves the interpretability and diagnostic capabilities of autoencoder-based process monitoring models, making them more practical and effective for real-world applications. Furthermore, due to the unavailability of fault information in the real-world Sanofi processes, this work proposes a new metric based on the amount of violation of process monitoring metrics compared to their upper control limits to assess the effectiveness of process monitoring models. The new proposed metric can be utilized to detect low productivity batches (faulty batches) before the end of the fermentation.

iv- To address over-fitting, hybrid models are developed for process monitoring and modeling. For the process monitoring task, this work proposes a novel hybrid framework that combines deep LSTM-AE models with exact equations of process controllers. In this

framework, manipulated variables are excluded from the input and output layers of the autoencoder. To preserve their data in the process monitoring model, they are reconstructed through the controller equations fed by the reconstructed controlled variables. Removing manipulated variables from the input and output layers of the autoencoder results in a neural network model with fewer parameters to tune compared to non-hybrid autoencoders that include manipulated variables in the input and output layers. The proposed hybrid LSTM-AE models are tested on two case studies: the penicillin batch fermenter simulator and the Tennessee Eastman process under a decentralized control strategy. The proposed hybrid LSTM-AE models not only reduce the size of the model and the risk of overfitting but also achieve higher process monitoring efficacy by utilizing perfect models of controllers.

v- To further tackle the problem of over-fitting, this work proposes a novel modeling approach named Metabolic Graph Neural Network (MGNN). This approach imposes a priori knowledge about the metabolic pathways of a microorganism onto the architecture of multi-layer neural network models to represent the dynamic behavior of the cell culture. To capture the nonlinear dynamic behavior of each metabolite, a sub-neural network is considered. The input layer of each sub-neural network model is designed based on the reactions in which the metabolite is involved. This results in a smaller neural network where not all layers and units are connected, thereby reducing the risk of overfitting. The proposed hybrid modeling framework is tested on the metabolic pathway of oxidative stress in Bordetella pertussis bacteria cells. By using the a priori known metabolic network, the proposed MGNN model effectively reduces the overfitting issue compared to a fully connected network that does not use metabolic network knowledge. The MGNN exhibits a superior fit for both training and testing datasets. Additionally, the proposed MGNN is highly interpretable, as it efficiently computes the relevance of each metabolite on any other metabolite by applying gradient computation and back-propagation operations to the neural network. The model is also shown to be useful for fault detection.

In summary, by tackling typical aspects of bio-process modelling including lack of measurements, nonlinear process dynamics and data over-fitting, this work advances the application of deep learning algorithms for monitoring of industrial bio-processes.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Professor Hector Budman. Long before becoming my supervisor, he was an exceptional role model in both work ethics and research methods. His promptness and willingness to spare his invaluable time have been instrumental in helping me advance swiftly throughout my research. His passion and devotion to research have been a constant source of inspiration, motivating me during challenging times. I deeply appreciate his unwavering support throughout my graduate studies and research. I am profoundly grateful for his advice on my overall development and for his understanding and support during my personal struggles. His kindness and encouragement have been invaluable to me both academically and personally. Thank you sir.

I gratefully acknowledge Dr. Melih Tamer and Dr. Stephane Krau for their unwavering support throughout my PhD program. Their roles at Sanofi were instrumental in providing the resources and assistance necessary for my research.

Thank you to my colleagues, especially Abhishek Mishra and Xin Shen, for sharing their experiences and for being key sources of support during my PhD.

I would like to thank the Department of Chemical Engineering at the University of Waterloo for providing me with the opportunity to pursue my PhD degree, along with the best resources and a friendly atmosphere. Special thanks to all the administrative staff, particularly Judy Caron and Sarah Fischer, for their patience and assistance throughout my journey.

I would like to express my gratitude to my doctoral committee members: Prof. Peter Douglas, Prof. Nasser Mohieddin Abukhdeir and Prof. Kumaraswamy Ponnambalam, for their feedback that gave direction to my entire doctoral research and brought in threads of thought that made my research so much richer.

Without the support of my family members, this work would not have been possible. My endless gratitude goes to my parents, Abdollah and Tahereh, for bestowing their unconditional love and affection. I am also indebted to my siblings, Shirin and Ali, for their affectionate love and friendship.

Dedication

I dedicate this thesis to
my parents, Maman and Baba,
and my siblings, Shirin and Ali,
for their support and unconditional love.

I love you all dearly

Table of Contents

Examining Committee ii

Author's Declaration iii

Statement of contribution iv

Abstract v

Acknowledgements ix

Dedication x

List of Figures xv

List of Tables xix

1 Introduction 1

2 Background 10

2.1 Principal Component Analysis 11

2.2 Partial Least Squares 11

2.3 Autoencoders 12

2.4 Long Short-Term Memory 13

2.5 Long Short-Term Memory Autoencoder . . . 13
2.6 Fault detection and diagnosis[1] . . . 14
2.6.1 Fault detection . . . 14
2.6.2 Fault diagnosis . . . 16

3 A Novel Unsupervised Approach for Batch Process Monitoring using Deep Learning 20

3.1 Introduction . . . 21
3.2 Preliminaries . . . 25
3.2.1 Multiway Principal Component Analysis (MPCA) . . . 26
3.2.2 MPLS . . . 28
3.2.3 Autoencoder Neural Network (AE-NN) . . . 30
3.3 Proposed Methodology . . . 31
3.3.1 Multiway Partial Least Squares Autoencoder (MPLS-AE) . . . 31
3.3.2 Novel Objective Function for Maximizing Fault Detection Rate . . . 34
3.3.3 Average Fault Detection Rate ($\overline{\text{FDR}}$) . . . 36
3.3.4 Case study . . . 38
3.4 Results and Discussions . . . 42
3.4.1 MPCA and MPLS with J^{MPCA} and J^{MPLS} . . . 42
3.4.2 MPCA and MPLS with $J^{MPCA-MPLS,FDR}$ . . . 45
3.4.3 MPLS-AE with $J^{\mathbf{MPLS-AE}}$ . . . 50
3.4.4 MPLS-AE with $\mathbf{J}^{\mathbf{MPLS-AE,FDR}}$ . . . 53
3.5 Conclusion . . . 55

[1] Adapted from Aghaee, Mohammad, et al. "Artificial Intelligence Applications for Fault Detection and Diagnosis in Pharmaceutical Bioprocesses: A Review." doi.org/10.1016/j.coche.2024.101025

4 Unsupervised Fault Detection of Pharmaceutical Processes Using Long Short-Term Memory Autoencoders **58**

4.1 Introduction . . . 59

4.2 Preliminaries . . . 64

4.2.1 Long Short-Term Memory (LSTM) . . . 64

4.2.2 Autoencoder (AE) . . . 65

4.2.3 Batch synchronization - Indicator Variable (IV) . . . 66

4.3 Proposed Methodology . . . 67

4.3.1 Long Short-Term Memory Autoencoder (LSTM-AE) . . . 67

4.3.2 Average fault detection rate . . . 69

4.3.3 A metrics based on an average of process monitoring metrics along the batch . . . 70

4.3.4 Fault Diagnosis - Contributions of individual variables to the detection metrics . . . 71

4.3.5 Case studies . . . 74

4.4 Results and Discussions . . . 77

4.4.1 Results in IndPenSim case study . . . 78

4.4.2 Results in Sanofi case study . . . 82

4.5 Conclusion . . . 87

4.6 Testing the Proposed Methodology on Sanofi's Lab-Scale Diphtheria Process 89

5 Unsupervised Hybrid Models Integrating Deep Autoencoders and Process Controllers' Models for Enhanced Process Monitoring and Fault Detection **95**

5.1 Introduction . . . 96

5.2 Preliminaries . . . 101

5.2.1 Long short-term memory autoencoders (LSTM-AE) . . . 101

5.2.2 Long short-term memory autoencoders (LSTM-AE) . . . 101

5.2.3 Hybrid models . . . 103

5.2.4 Process monitoring metrics . . . 103
5.3 Proposed Methodology . . . 105
5.3.1 Proposed hybrid model: integration of LSTM-AE with process controllers' exact models . . . 105
5.3.2 Case studies . . . 109
5.4 Results and Discussions . . . 114
5.4.1 Results in the IndPenSim case study . . . 116
5.4.2 Results in the Tennessee Eastman case study . . . 119
5.5 Conclusion . . . 124

6 Graph Neural Network Representation of State Space Models of Metabolic Pathways 126
6.1 Introduction . . . 127
6.2 Proposed methodology . . . 130
6.2.1 Graph Neural Networks . . . 130
6.2.2 Oxidative Stress Metabolic Pathways . . . 131
6.2.3 Graph Neural Network for Oxidative Stress Metabolic Pathways . . . 132
6.3 Results and discussion . . . 135
6.4 Conclusion . . . 141

7 Conclusions and Future Work 143
7.1 Conclusions . . . 143
7.2 Future Work . . . 148

References 151

List of Figures

2.1 The flowchart of online FDD procedure. 17

3.1 Schematic of a single layer Autoencoder Neural Network (AE-NN) 31

3.2 Schematic of a Multiway Partial Least Squares Autoencoder (MPLS-AE) . 31

3.3 Flowchart for fault detection based on novel objective function 37

3.4 Summary of all model inputs and outputs recorded by IndPenSim 41

3.5 Profiles of measured variables in normal and faulty conditions. 41

3.6 Plots of T_k^2 and static control limits T_α^2 for six different faulty batches of the test dataset based on J^{MPCA} as the objective function ($\alpha = 0.01$). 43

3.7 Plots of Q_k and static control limits Q_α for six different faulty batches of the test dataset based on J^{MPCA} as the objective function ($\alpha = 0.01$). . . . 44

3.8 Plots of T_k^2 and dynamic control limits $T_{k,\alpha}^2$ of six different faulty batches of the test dataset by using J^{MPLS} as the objective function ($\alpha = 0.01$). . . . 45

3.9 Plots of Q_k and dynamic control limits $Q_{k,\alpha}$ for six different faulty batches of the test dataset based on J^{MPLS} as the objective function ($\alpha = 0.01$). . 46

3.10 Novel FDR loss function as a function of principal components for Model 1 (MPCA with static control limits) for validation dataset 46

3.11 Plots of T_k^2 and dynamic control limits $T_{k,\alpha}^2$ for six different faulty batches of the test dataset based on $J^{MPCA-MPLS,FDR}$ as the objective function for the MPLS model ($\alpha = 0.01$). 51

3.12 Plots of Q_k and dynamic control limits $Q_{k,\alpha}$ for six different faulty batches of the test dataset based on $J^{MPCA-MPLS,FDR}$ as the objective function for the MPLS model ($\alpha = 0.01$). 52

3.13 Loss of the validation dataset with MPLS-AE model using $J^{MPLS-AE}$ as the objective function to train the network. 52

3.14 Plots of H_k^2 and dynamic control limits $H_{k,\alpha}^2$ for six different faulty batches of the test dataset based on $J^{MPLS-AE,FDR}$ as the objective function to train MPLS-AE ($\alpha = 0.01$). 56

3.15 Plots of SPE_k^2 and dynamic control limits $SPE_{k,\alpha}^2$ for six different faulty batches of the test dataset based on $J^{MPLS-AE,FDR}$ as the objective function to train MPLS-AE ($\alpha = 0.01$). 56

4.1 Schematic of an autoencoder (AE) 66

4.2 A simple illustration of the LSTM-AE network structure. 68

4.3 The decoder network of the trained LSTM-AE for computing H^2 contributions. 73

4.4 A schematic of *B. pertussis* fermentation process at Sanofi, Toronto. 76

4.5 Data points of ID versus normalized Kjeldahl values at Sanofi, Toronto. . . 77

4.6 Contributions of variables in different faulty batches in IndPenSim case study. 80

4.7 The illustration of D_b versus Batch IDs using the LSTM-AE model trained by maximizing the fault detection rate for one of the folds of the validation data set. 87

4.8 Most contributed variables in Sanofi case study. 88

4.9 Average contribution of all variables over the entire fermentation time in Sanofi case study. 88

4.10 The time required to detect low productivity batches in the testing data set. 88

4.11 Diphtheria process variables at Sanofi, Toronto. 90

4.12 Final productivity of Diphtheria process at Sanofi, Toronto. 91

4.13 Process monitoring profiles for training, validation and testing datasets of Diphtheria process. 93

4.14 The illustration of Db versus Batch IDs using the trained AE model for monitoring the Diphtheria process. 94

4.15 Time required to identify low productivity batches in the Diphtheria process. 94

5.1 Schematic of LSTM cell architecture. 102

5.2 Schematic of process monitoring models: **A.** single autoencoder model, **B.** proposed hybrid autoencoder model. . . . 108
5.3 Architecture of the SPE and $SPE^{augmented}$ matrices of the proposed hybrid process monitoring model. . . . 109
5.4 Schematic of IndPenSim process. . . . 112
5.5 IndPenSim variables and their reconstruction by the trained hybrid LSTM-AE model for a validation batch. . . . 120
5.6 Plots of H^2, SPE, and their corresponding upper control limits with Method 3 for faults 4 and 5, which are associated with the controllers. . . . 121
5.7 Tennessee Eastman manipulating variables and their reconstructions by the trained hybrid LSTM-AE model for a validation run. . . . 123
5.8 Plots of H^2, SPE, and their corresponding upper control limits with Method 3 for faults IDV (2), IDV (3), and IDV (7). . . . 123

6.1 A schematic of graph neural networks (GNNs). . . . 131
6.2 An illustration of the metabolic pathways involved in oxidative stress. . . . 134
6.3 Synthetic data produced by the mechanistic model developed by Vitelli et al., 2023. . . . 134
6.4 Architecture of the metabolic graph neural network for the oxidative stress metabolic network. . . . 138
6.5 Comparison of the predicted concentrations of metabolites over time in a training batch using the developed fully connected neural network with the mechanistic model. . . . 139
6.6 Comparison of the predicted concentrations of metabolites over time in a testing batch using the developed fully connected neural network with the mechanistic model. . . . 139
6.7 Comparison of the predicted concentrations of metabolites over time in a testing batch using the developed Metabolic Graph Neural Network (MGNN) with the mechanistic model. . . . 140
6.8 The relevance of metabolites at the previous time to the change of metabolites at the current time using the MGNN model. . . . 140
6.9 The positive or negative influence of each input on the output direction; Green (Positive), Red (Negative), Yellow (Neutral) . . . 141

6.10 The contribution of different metabolites to detect the imposed fault in the media (S_{ext}) in $t = [20, 26]$ compared to their contributions in normal media. 142

List of Tables

2.1 Summary of Reviewed Papers; FD(D): Fault Detection (and Diagnosis), QP: Quality Prediction . . . 17

3.1 Description of different types of fault. . . . 40
3.2 Comparison of ($\overline{\text{FDR}}$) test accuracy for different models . . . 49
3.3 Summary of $\overline{\text{FDR}}_{\text{normal}}/\overline{\text{FDR}}_{\text{abnormal}}$ for different faults in the test dataset by using different models . . . 49
3.4 Summary of $\overline{\text{FDR}}_{\text{normal}}/\overline{\text{FDR}}_{\text{abnormal}}$ for different faults for Model 5 (test accuracy) . . . 50
3.5 Summary of the average detection time for different models (averages are on 600 batches (testing data)) . . . 55

4.1 Description of different types of fault, [Agarwal et al., 2022]. . . . 75
4.2 Comparison of $\overline{FDR}\%$ (training/validation/test) in different models. . . . 79
4.3 Results of the Sanofi case study. . . . 85
4.4 Results of monitoring the Diphtheria batch process at Sanofi, Toronto. . . . 92

5.1 Details of various fault types in the IndPenSim case study. . . . 111
5.2 Parameters of the PID controllers for pH and temperature control in the IndPenSim process. . . . 112
5.3 Controlled and manipulated variables of Tennessee Eastman plant process. 114
5.4 Details of various fault types in the TE case study. . . . 115
5.5 Parameters of the PI controllers in the Tennessee Eastman plant process. . 116

5.6 Summary of the performance of three different methods for monitoring the IndPenSim process. 119

5.7 Summary of the performance of three different methods for monitoring the Tennessee Eastman process. 124

6.1 Comparison of the Metabolic Neural Network model (MGNN) and the fully connected neural network (FCNN). 138

Chapter 1

Introduction

Optimization of bioprocesses is crucial for ensuring the quality, efficiency, and safety of final pharmaceutical products. Fault detection and diagnosis, collectively referred to as online process monitoring, are key tasks for ensuring process optimality by offering the means to identify, mitigate, and prevent deviations that could compromise product integrity or/and production efficacy [Aghaee et al., 2024a]. Traditional fault detection methods often utilize rule-based approaches or multivariate statistical techniques like principal component analysis (PCA) [Wise et al., 1990] and partial least squares (PLS) [MacGregor et al., 1994]. Although these methods can be effective in certain contexts, they often face challenges when dealing with the complexity and variability inherent in bioprocesses [Mouzakitis, 2013]. Pharmaceutical processes are mostly carried out in batch or fed-batch modes to minimize contamination risks [Kumar et al., 2020], resulting in operations that cover a wide range of operating conditions and exhibit highly nonlinear behavior, unlike continuous processes that typically operate near a specific steady state thus allowing for linear modeling approximations.

In recent years, there has been substantial emphasis on improving global methodolo-

gies for real-time process monitoring, especially after the Food and Drug Administration (FDA) introduced the Process Analytical Technology (PAT) guidance [Darrow et al., 2020] that encourages the use of this technology. Artificial intelligence, with its capability to autonomously learn hierarchical data representations, has transformed fault detection and diagnosis across numerous industries [Swami et al., 2023]. Due to their ability to identify complex nonlinear patterns from extensive datasets, deep learning based methods have resulted in significant improvements in precision and reliability of fault detection systems in pharmaceutical bioprocesses. These applications involve training deep learning models, such as convolutional neural networks (CNNs) [Albawi et al., 2017], recurrent neural networks (RNNs) [Salehinejad et al., 2017], or hybrid architectures [Kilicarslan et al., 2021], that were applied to large datasets containing various process variables and operation details. Despite ongoing research in this area, the pharmaceutical industry's adoption of deep learning techniques is still in its early stages. This thesis aims to address this gap by investigating the application of advanced deep learning techniques for process monitoring, diagnosing process faults and for biosystems' modeling.

For the task of fault detection and diagnosis, four case studies were investigated in the current work: the whooping cough vaccine manufacturing process at Sanofi Toronto (the industrial collaborator for the current work), the lab-scale process for Diphtheria at Sanofi Toronto, the Tennessee Eastman process, and an industrial-scale simulator of the Penicillin batch fermentation process. For the task of modeling, the oxidative stress metabolic pathway in *Bordetella pertussis* fermentation was considered to develop a novel metabolic graph neural network for modeling its dynamic behavior [Aghaee et al., 2024b]. The use of simulator case studies of the Tennessee Eastman process and the Penicillin batch process were motivated by the need to proof the effectiveness of the proposed methodologies based on full a priori knowledge of the faults entering the system. Then, the same methodologies were applied to real-world industrial processes currently operated at Sanofi-Pasteur (Toronto) where the faults were not known.

Bioprocesses are highly sensitive to fluctuations in operating conditions such as temperature, pH, oxygen, and nutrient levels. Changes in these conditions will result in deviations from normal operation thus leading to faulty operation. While minor disturbances can often be managed by standard controllers like PIDs, significant deviations typically require operator intervention or even a temporary shutdown for detailed inspection. Consequently, in this work, a fault is referred to any process variation that drives the process variables beyond their normal ranges thus motivating process interventions or the need to search for the source of the fault [Gunther et al., 2007].

Therefore, defining the normal operation of a bioprocess in terms of acceptable ranges of process variables is essential for implementing a fault detection algorithm. This normal operation must consider both productivity and quality. However, the challenge lies in the lack of precise measurements of these properties, making it difficult to accurately differentiate between normal and faulty states. For instance, productivity evaluations using techniques like ELISA, HPLC, and other titer analyses may be highly inaccurate due to sensor noise thus compromising the effectiveness of detection algorithms based on such measurements. Also, the source of faults is often unknown in industrial processes which precludes the calibration of models that relate faults to quality variables [Lim et al., 2023]. These challenges often motivates the choice of unsupervised learning techniques over supervised ones [Hastie et al., 2009]. Unsupervised learning methods involve models that identify patterns and structures in data without the need for labeled samples that define normal operation, such as the desired quantity or quality of the product. Thus, unsupervised methods aim to derive meaningful representations or features from the input data itself. In contrast, supervised learning models are based on the relation between measured inputs, which may include known faults, with labeled outputs.

A significant issue in the Pertussis Vaccine Manufacturing process for whooping cough at Sanofi is the inconsistency in the production of Pertactin (PRN), one of the vaccine's

key antigens. Similarly, the lab-scale Diphtheria process faces variability problems in its final fermentation yield. To reduce this variability, it is necessary to identify its root cause using the available measurements. The data for this study is primarily drawn from Sanofi's extensive historical database, covering the manufacturing of the whooping cough vaccine and the lab-scale batch process of Diphtheria over several years of process operation. This data encompasses frequent measurements from upstream fermentation processes, including dissolved oxygen levels, aeration rates, and agitation rates. Despite the abundance of process data from Sanofi, detailed mathematical models of their processes are not available. Another challenge is that the sources of variability in Sanofi's processes have not been fully identified, unlike simulator case studies where these faults are evidently available a priori. Lack of knowledge about the faults rules out the development of supervised learning algorithms for these processes thus motivating the use of unsupervised learning algorithms to detect abnormal operations.

A significant additional challenge in fault detection with deep learning models is overfitting due to insufficient training data for calibration of large models with noisy data. Overfitting leads to inaccurate predictions for unseen data, affecting fault detection accuracy. Conventional methods to mitigate overfitting include reducing model size or augmenting the dataset. However, smaller models may fail to capture process intricacies, and data collection can be costly [Yu and Zhang, 2023]. Gray-box or hybrid models combine mechanistic and empirical model-based techniques to enhance predictive capabilities and reduce reliance on training data [Pistikopoulos et al., 2021]. These hybrid models integrate mechanistic and empirical components either in serial or parallel architectures [Fontes and Budman, 2017]. However, the performance improvement of a hybrid model-based approach over a purely empirical approach largely depends on the accuracy of the available mechanistic model. This problem is particularly relevant to pharmaceutical systems where models are often inaccurate due to the variability of biochemical processes.

Moreover, overfitting can be further understood in terms of the discrepancies between the process and the model, which can originate from two main sources: noise and model structure error. In an ideal scenario, where the model structure exactly matches that of the process—although this is practically unattainable—an infinite amount of data would enable the model to learn the precise parameters, with noise being averaged out over time. However, in real-world settings, model structure error is inevitable. This error, which is often non-Gaussian and not mean-centered, prevents noise from being completely averaged out, even with an infinite dataset. This is particularly significant when dealing with nonlinear models, where the formalization of these concepts is far more complex than in linear systems, making it difficult to establish clear theoretical boundaries for overfitting prevention.

Thus, while the development of advanced architectures and hybrid models offers ways to improve accuracy, managing overfitting requires balancing model complexity with data quality and amount. Additionally, enhancing the quality of the mechanistic models, where possible, is critical to the success of hybrid approaches, especially in complex and variable systems like pharmaceutical processes.

The thesis is organized into chapters dealing with the different challenges of unsupervised fault detection that were identified above.

In Chapter 3, we evaluated the effectiveness of unsupervised deep learning process monitoring algorithms compared to traditional multivariate statistical methods such as PCA and PLS by developing an innovative deep learning autoencoder (AE), termed PLS-AE. This model was specifically designed for monitoring the simulator benchmark of penicillin batch fermentation. Autoencoders are generally recognized as the nonlinear extension of the unsupervised PCA algorithm because they compress the input space into a lower-dimensional space, referred to as hidden variables, through the encoder component. Subsequently, the decoder component reconstructs the input space utilizing these hidden variables. Since the encoder compresses the data into a lower dimensional space, the hidden variables in the

AE model can be viewed as equivalent to the principal components in conventional PCA and PLS algorithms.

However, unlike PCA and PCA-AE models, the proposed PLS-AE model requires an output variable similar to its linear counterpart. To address this, we employed process time as a dummy output variable for training the PLS-AE model. This model not only includes encoder and decoder components, as in standard AE models, but also incorporates an additional neural network component parallel to the decoder. This additional component is responsible for reconstructing the dummy variable of process time. Additionally, we proposed a novel loss function for training the model that directly maximizes the fault detection rate, contrasting with the classical loss function of minimizing the mean square error of reconstruction errors. For the task of process monitoring and fault detection, this work employs dynamic metrics of H^2 and SPE, which are equivalent to Hotelling's T^2 and $Q-$statistics in conventional PCA and PLS algorithms. In Chapter 3, we demonstrate that deep learning models, due to their ability to capture greater nonlinearity and dynamics of the process, exhibit superior performance in monitoring the penicillin benchmark case. The models are trained using a novel loss function aimed at maximizing fault detection rate and employing dynamic process monitoring metrics thus outperforming traditional methods that minimize reconstruction errors but do not focus on the detection task.

Building on the idea of the detection oriented loss, Chapter 4 proposes unsupervised multi-layer long short-term memory autoencoder (LSTM-AE) models for monitoring nonlinear pharmaceutical batch processes. The motivation for developing LSTM-AE models for process monitoring stems from their potential to better capture the nonlinearity and dynamics of the data. This methodology was tested on both the industrial-scale penicillin simulator and the whooping cough vaccine manufacturing process at Sanofi Toronto. Consistent with 3, the proposed model was trained using two different loss functions: minimizing the mean square error (MSE) between the input and reconstructed data, and

maximizing the average fault detection rate (FDR) in the training dataset.

As previously noted, in most real-world processes, labeled datasets and information regarding faults are typically unavailable thus making unsupervised algorithms the method of choice for process monitoring. Due to the lack of strategies for diagnosing faults using unsupervised deep learning process monitoring models, this chapter proposes two algorithms for obtaining contribution plots for the diagnosis of faults. For the industrial case study, where the faults are not known a priori, the contribution plots are found to be a valuable tool for identifying possible sources of faults. Furthermore, a semi-supervised procedure has been proposed for the industrial study to select the normal process region for training the model.

Two metrics are also presented to evaluate the performance of the proposed methodology: one for the simulator case study, where fault knowledge is available, and one for the industrial case study, where fault knowledge is not available a priori. In the simulator case study, since fault knowledge is available, the fault detection rate is used to evaluate the performance of the methodology by counting the number of samples correctly identified as normal or abnormal. However, for cases where fault information is unavailable, such as the industrial case study in Chapter 4, this metric cannot be calculated. Instead, we proposed a novel metric based on the magnitude of the cumulative violations of process monitoring upper limits (H^2 and SPE). The proposed unsupervised LSTM-based algorithms demonstrate a clear improvement in accuracy compared to linear methods or nonlinear techniques that do not explicitly account for dynamic behavior.

To address the problem of overfitting mentioned above, Chapter 5 introduces a novel serial hybrid process monitoring model that integrates unsupervised deep LSTM-AE models, known for capturing process nonlinearity as shown in Chapter 4, with the exact models of process controllers to enhance fault detection. The key innovation is the integration of an exact mechanistic component, which only represents the process controllers that are

perfectly known a priori. This approach significantly reduces the number of deep learning parameters and avoids the addition of model error that would occur if other mechanistic equations were used.

The proposed hybrid fault detection model trains the LSTM-AE model under normal operating conditions by utilizing a loss function that minimizes the combination of the mean square error (MSE) between input and reconstructed samples and the MSE related to the control actions reconstructed from the exact controllers' equations. The proposed methodology is tested on two case studies for fault detection: the industrial-scale penicillin fermentation simulator and the Tennessee Eastman (TE) process benchmark, using no fault information during model development to demonstrate the unsupervised learning approach. This chapter demonstrates that the proposed hybrid LSTM-AE fault detection model not only enhances process monitoring efficacy but also reduces the risk of overfitting due to having fewer parameters.

Chapter 6 further tackles the problem of overfitting by proposing a novel hybrid modelling approach referred to as metabolic graph neural networks (MGNNs). This approach is applied for reducing over-fitting in modeling dynamic metabolic pathways as compared to alternative modelling approaches such as conventional kinetic modeling (KM), dynamic flux balance analysis (DFBA), and purely machine learning/deep learning approaches. The proposed approach is also demonstrated for fault detection and diagnosis problems. The methodology is motivated by the potential of hybrid models to combine mechanistic knowledge with empirical approaches, aiming to reduce model parameters and mitigate overfitting risks.

The specialized architecture of the MGNN consists of several sub-neural networks representing metabolites, and the overall MGNN model is constrained by a priori biochemical information about metabolic reactions. This constrained architecture offers several benefits: i. it avoids overfitting compared to fully interconnected networks that lack a priori

reaction network information, ii. it is easier to tune with current deep learning algorithms than stoichiometry-based methods like DFBA, which require complex nonlinear programming (NLP) solutions, iii. it does not require predefined nonlinear kinetic constraints, such as Monod or Hill functions, and iv. it provides explainability by linking neurons to metabolites and interconnections to fluxes. The methodology is tested on the oxidative stress metabolic network of *B. pertussis*, a bacterium used in whooping cough vaccine production at Sanofi, Toronto.

Following the above, the novel contributions of this work can be summarized as follows:
1. Developed unsupervised process monitoring approaches using machine learning, deep learning, and hybrid deep learning models.

2. Developed unsupervised algorithms for diagnosing process faults using unsupervised deep autoencoder models.

3. Proposed novel loss function for training process monitoring models towards maximizing fault detection rate.

4. Proposed a new metric for assessing the effectiveness of process monitoring models when fault information is unavailable.

5- Tackle the problem of over-fitting by two different hybrid approaches: i- a hybrid model that combine a deep learning auto-encoder with exact controller equations, ii- a novel metabolic graph neural network models for modeling metabolic pathways.

6- The proposed approaches were validated with actual industrial (whooping cough and diphtheria vaccine manufacturing) or laboratory scale data (oxidative stress flask experimental data).

Chapter 2

Background

This chapter provides a comprehensive overview of the key methodologies and techniques used in this thesis. I review several fundamental tools, including Principal Component Analysis (PCA) and its extension Multiway Principal Component Analysis (MPCA), Partial Least Squares (PLS) and its extension Multiway Partial Least Squares (MPLS), Autoencoders (AE), Long Short-Term Memory (LSTM) networks, and Long Short-Term Memory Autoencoders (LSTM-AE). Each section will delve into the conceptual underpinnings, applications, and relevance of these techniques to process monitoring and fault detection in batch processes. For more in-depth information and detailed mathematical formulations, references to specific sections in subsequent chapters are provided. Additionally, Section 2.6 presents insights and findings from one of my published papers, reviewing the application of artificial intelligence algorithms for fault detection and diagnosis in pharmaceutical processes [Aghaee et al., 2024a].

2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique that projects high-dimensional, correlated data onto a lower-dimensional orthogonal space while preserving a significant level of variance [Wise and Gallagher, 1996]. This projection helps in simplifying the dataset by capturing the most significant features in fewer dimensions. PCA is typically used in industrial practice to detect abnormal conditions by analyzing the variance in the data. Key metrics such as Hotelling's T^2 and Q-statistic (sum of squares of residuals) are employed to identify deviations from normal operation [Kourti and MacGregor, 1996].

Multiway Principal Component Analysis (MPCA) extends PCA to handle three-dimensional data arrays, making it suitable for batch processes. In MPCA, the data initially structured into a three-dimensional array is unfolded into a two-dimensional matrix, and then PCA is applied. This approach helps in monitoring batch processes by capturing the variability across different batches and detecting faults.

For a detailed explanation of PCA and MPCA, including mathematical formulations and process monitoring techniques, the reader is referred to Section 3.2.1.

2.2 Partial Least Squares

Partial Least Squares (PLS) is a regression technique that projects both input variables $\mathbf{X}$ and output variables $\mathbf{Y}$ onto lower-dimensional spaces. Then, the aim is to maximize the covariance between the projections of $\mathbf{X}$ and $\mathbf{Y}$, making it useful for scenarios where the goal is to predict outcomes based on input data. For example, PLS helps in detecting faults by identifying the relationship between process variables and output responses.

Multiway Partial Least Squares (MPLS) extends PLS to handle three-dimensional data arrays occurring in batch processes, i.e. two dimensions for process variables at each time

for different batches and the third dimension is time. Similar to standard PLS, MPLS estimates the projections of both input and output variables and maximizes the covariance between the projections. This approach is beneficial in scenarios where both input and output data are available. Also, in the absence of relevant output variables, a dummy output variable such as time interval during the batch can be used to aid the modeling process. Similar to MPCA, fault detection based on MPLS is also based on metrics like Hotelling's T^2 and Q statistics.

For a comprehensive discussion on PLS and MPLS, including mathematical formulations and fault detection methods, the reader is referred to Section 3.2.2.

2.3 Autoencoders

AE is an unsupervised learning model designed to extract underlying patterns in data and reduce its dimensionality. It consists of three main components: the encoder, the decoder, and the embedding layer (latent space). The encoder compresses the input data into a reduced latent space, while the decoder reconstructs the input data from this latent space. This process helps for filtering noise while capturing the most significant features of the data in a lower-dimensional representation.

While the encoder maps the input vector to a latent space using non-linear activation functions, the decoder reconstructs back the inputs from the latent space. The model is trained by minimizing the reconstruction error, which is the difference between the input data and the reconstructed data. The minimization is done with respect to the weights and biases of the network to accurately capture the underlying structure of the data.

For a detailed explanation of AEs, including mathematical formulations, training methods, and illustrative figures, the reader is referred to Section 3.2.3 and Section 4.2.2.

2.4 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) designed to capture both long-term and short-term dependencies in sequential data. LSTMs are particularly effective for tasks involving time-series data due to their ability to address the issues of gradient vanishing and long-term memory loss that affect traditional RNN architectures. Each LSTM unit contains a cell state that retains information over time, along with three gates (input, forget, and output) that regulate the flow of information. These gates control how much of the past information is remembered, how much of the current input is accepted, and how much of the cell state is outputted at each time step [Hochreiter and Schmidhuber, 1997]. For more detailed explanations and illustrations, refer to section 4.2.1 and section 5.2.1.

2.5 Long Short-Term Memory Autoencoder

Long Short-Term Memory Autoencoders (LSTM-AEs) extend the concept of autoencoders to sequential data by incorporating LSTM units. These models are particularly effective for reconstructing nonlinear dynamically correlated data, making them suitable for time-series analysis. An LSTM-AE consists of an encoder and a decoder, both composed of multiple LSTM layers. The encoder compresses input sequences into a lower-dimensional latent space, while the decoder reconstructs the original sequences from this latent space. LSTM-AEs are trained in an unsupervised manner to minimize the reconstruction error between the input sequences and their reconstructions, allowing them to capture complex temporal dependencies in the data [Hochreiter and Schmidhuber, 1997].

LSTM-AEs are particularly useful for tasks such as anomaly detection, sequence prediction, and time-series forecasting. By leveraging the capabilities of LSTM units, these autoencoders can effectively model and reconstruct sequences, making them highly suitable

for applications that involve dynamic data.

For a detailed explanation of LSTM-AEs, including mathematical formulations, and illustrative figures, the reader is referred to Section 4.3.1 and Section 5.2.1.

2.6 Fault detection and diagnosis[1]

2.6.1 Fault detection

The detection of a fault is based on a statistically significant boundary between normal and faulty data. In practical applications, data-driven fault detection typically involves several steps: i- a fault detection model is trained with data collected under normal operating conditions, ii- control limits, such as Hotelling's T^2 and mean reconstruction error $Q-$statistic, are calculated based on the model, iii- During online operation, the developed FDD model is fed new sample's data and compares its response to the established control limits and iv- if T^2 or $Q-$statistical bounds exceed the corresponding control limit, the new sample is flagged as a fault.

Multivariate statistical algorithms

Multivariate statistical algorithms are widely applied through commercial software (SIMCA [Bylesjö et al., 2006]) and are renowned for their ease of implementation. Principal component analysis (PCA) and partial least squares (PLS) are the most common multivariate statistical algorithms for batch process monitoring in industry. Typically, they utilize Hotelling's T^2 statistic and $Q-$statistic to detect faults. However, due to the nonlinear and dynamic nature of data of chemical processes, linear PCA and PLS algorithms are not accurate enough. Hence, kernel-based and dynamic-based methods such as kernel PCA,

[1] Adapted from Aghaee, Mohammad, et al. "Artificial Intelligence Applications for Fault Detection and Diagnosis in Pharmaceutical Bioprocesses: A Review." doi.org/10.1016/j.coche.2024.101025

kernel PLS, dynamic PCA, and dynamic PLS have been developed to enhance performance. Also, due to the prevalent batch or semi-batch nature of pharmaceutical processes, multiway-PCA (MPCA) and multiway-PLS (MPLS) methods are commonly utilized in practice [Agarwal et al., 2022]. Additionally, nonlinear PCA and nonlinear PLS algorithms, which are based on autoencoder neural network structures, have also been utilized for batch process monitoring to capture the dynamic and nonlinear aspects of the process [Agarwal et al., 2022, Aghaee et al., 2023]. The following section provides more details about these algorithms.

Deep learning algorithms

Deep Neural Networks (DNNs) have the ability to uncover inherent nonlinear and dynamic patterns and hierarchical structures often resulting in superior performance than traditional machine learning techniques such as multivariate statistical methods.

Different novel neural network architectures have been proposed that are particularly useful for fault detection in nonlinear dynamic problems, e.g. multi-layer neural networks, autoencoders (AE), and recurrent neural networks (RNN), such as Long Short-Term Memory (LSTM) [Aghaee et al., 2023] and gated recurrent units (GRU) [Ji et al., 2023]. Specifically, convolutional neural networks (CNN) have proven adept at capturing process nonlinearity with heightened accuracy [Hematillake et al., 2022]. Generative adversarial networks (GAN) have emerged as a tool for addressing data scarcity by generating realistic synthetic data for training [Li et al., 2021a, Yan et al., 2020, Abbasi et al., 2022, Chandrashekar et al., 2023, Li et al., 2021b].

Hybrid models

While neural networks have high interpolating capability, they are less accurate at generalizing behaviour, i.e. predicting outputs for input values that are very different from those

used for training. Gray-box or hybrid models combine both mechanistic and data-driven modeling techniques to improve predictive capabilities and reduced reliance on training data [Pistikopoulos et al., 2021]. These hybrid models integrate mechanistic and data-driven components either in serial or parallel architectures [Fontes and Budman, 2017]. Mechanistic models describe fundamental conservation laws of the process. By partially relying on fundamental governing laws, hybrid models facilitate the estimation of unmeasured process states and enable the augmentation of datasets, thereby enhancing the performance of process monitoring models [Cabaneros Lopez et al., 2021]. For example, Physics Informed Neural Networks (PINN) is a special type of hybrid model that is trained to satisfy the governing physical laws of the process [Cuomo et al., 2022].

2.6.2 Fault diagnosis

Statistical fault detection charts serve to signal anomalies in a system based on predefined bounds without elucidating the underlying cause. Conversely, fault diagnosis methodologies aim to pinpoint the origin of faults, employing either supervised or unsupervised algorithms. However, supervised approaches necessitate comprehensive labeled dataset, a rarity in practical scenarios. Moreover, information about sources of faults is often scarce in real-world processes. Then, unsupervised algorithms can be used to construct contribution plots that help to interpret and hypothesize potential fault sources. In this regard, two innovative algorithms for generating contribution plots within deep learning process monitoring models have been introduced in [Aghaee et al., 2023]. Additionally, AI-based models like AEs in process monitoring tasks can utilize hypothesis testing to assess the differences between the latent space representations of normal and abnormal samples [Li et al., 2023a, Antwarg et al., 2021]. This integration of statistical methods within AI approaches is expected to bolster robustness in model evaluation and validation steps. A summary of recently proposed FDD models for pharmaceutical processes employing machine learning, deep learning, or hybrid approaches is presented in Table 2.1. Also, Figure

2.1 presents a flowchart diagram outlining the development of fault detection and diagnosis models. The fault detection model is developed using historical batches in offline mode. New samples are labeled as normal or faulty. Faulty samples trigger the diagnosis model to identify the fault source.

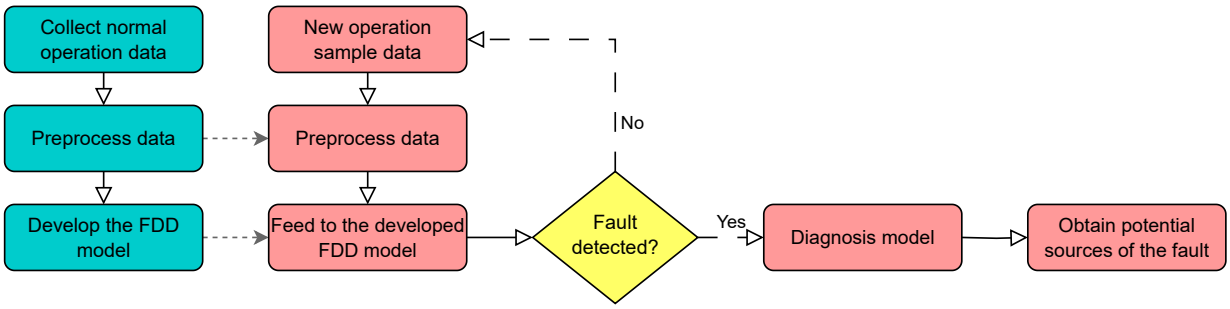


Figure 2.1: The flowchart of online FDD procedure.

Table 2.1: Summary of Reviewed Papers; FD(D): Fault Detection (and Diagnosis), QP: Quality Prediction

Model	Purpose	Results	Ref.
Principal component analysis (PCA)	**FDD in** a pharmaceutical drug manufacturing	Addressing batch process challenges: drifting; overlapping; imbalanced data.	Zeberli et al. [2021]
MPCA, MPLS MPLS-AE	**FDD in** a penicillin batch process	The MPLS-AE model shows superior accuracy compared to others.	Agarwal et al. [2022]
Long short-term memory autoencoder (LSTM-AE)	**FDD in** a penicillin batch process; Sanofi *B. pertussis* vaccine manufacturing process	Exhibits a clear accuracy improvement over linear or nonlinear techniques that do not explicitly consider dynamic behavior.	Aghaee et al. [2023]
Multiway Kernel Entropy Independent Component Analysis (MKEICA)	**FDD in** a penicillin batch process; an actual bioprocess	To address the challenges of nonlinear and non-Gaussian traits in batch processes and diagnose faults using nonlinear contribution plots.	Peng et al. [2020a]
Affinity propagation broad learning system (AP-BLS)	**FD in** a penicillin batch process	To develop a multi-stage monitoring framework integrating AP and BLS, surpassing AP-PCA, AP-ICA, AP-LSTM.	Peng et al. [2020b]
Supervised deep convolutional autoencoder (CNN-AE)	**FDD in** a penicillin batch process	The CNN-AE model significantly outperformed the PLS-DA model in the optimal integration of FDD with process design.	Hematillake et al. [2022]
Canonical variate analysis (CVA)	**FDD in** a granulation process	Addressing nonlinearity in dynamic processes for future automated FDD.	Quatrini et al. [2020, 2022]

Multiphase enhanced high-order information extraction (MEHOIE)	**FD in** a penicillin batch process; a microbial bioprocess	Addressing feature extraction challenges with a higher number of samples than features, surpassing PCA and ICA.	Chunhao et al. [2020]
Autoencoders, K-means clustering and multi-input neural networks	**FDD in** a rolling-elements bearings in an industrial pharmaceutical process	To propose a scalable framework for automated prediction of degradation stages in rolling-element bearings, offering reliable predictions.	Juodelyte et al. [2022]
Differential recurrent neural network (DRNN)	**FD and QP in** a penicillin batch process	Addressing batch process challenges: different initial conditions; uneven batch lengths; multiphase feature extraction.	Ji et al. [2023]
Overcomplete broad learning system (OBLS)	**FD in** a penicillin batch process; a real-world fermenter	Addressing the challenge of non-Gaussian and nonlinearity in data, outperforming traditional PCA and ICA.	Peng and RuiWei [2021]
Batch-wise LSTM-AE	**FD in** a penicillin batch process	Capturing features within and between batches, surpassing AE and Sub-PCA.	Ren and Ni [2020]
Weak Embryo Detection Network (WEDNet)	**FDD of** weak and live embryos in egg-based vaccines	Surpassing prior models in processing image time, as well as in the detection and classification of weak embryos.	Chen et al. [2020a]
Electrical resistance tomography (ERT)	**FDD in** a reactive crystallization process	Real-time monitoring of $CaCO_3$ reactive crystallization using a single ERT sensor.	Aghajanian et al. [2021]
Hybrid model: PLS-EKF	**FDD in** a penicillin batch process	Improving FDD with augmented datasets of measured and estimated process states.	Destro et al. [2020]
Hybrid model: PLS-EKF	**FD in** a crystallization batch process	PLS-EKF trained with augmented dataset outperforms standard PLS.	Ghosh et al. [2021]
Stochastic hybrid process model (SHPM)	**FDD in** a penicillin batch process	The proposed SHPM model surpasses Hidden Markov Model (HMM).	Windmann [2022]
Ensemble Models: Random Fourier feature analysis (RFFA)	**FD in** a penicillin batch process	Lower computational load compared to traditional kernel methods, handling large datasets and nonlinear local behaviors.	Deng and Du [2021]
Multiway Laplacian autoencoder (MLAE)	**FD in** a penicillin batch process; a real-world E. coli fermenter	Contrary to traditional AEs, the MLAE model captures both the local structure and deviations among batches.	Gao et al. [2020]
Adversarial autoencoder and K-Nearest Neighbor (AAE-KNN)	**FD in** a penicillin batch process	Outperforms traditional methods such as KPCA and KICA in monitoring minor magnitude and non-Gaussian faults.	Li et al. [2021b]
Sequential phase division (SPD)	**FD in** a penicillin batch process	Monitoring batch multiphase processes with synchronized uneven-length batches.	Zhang et al. [2023a]
Two-dimensional LSTM variational auto-encoder (2DLSTM-VAE)	**FDD in** a penicillin batch process	Outperforms complex models like VAE, cVAE, and LSTM-Encoder in addressing nonlinear dynamics and online FDD.	Liu et al. [2023a]
Siamese deep neighbourhood preserving embedding network (SDeNPE)	**FD in** a penicillin batch process	Outperforming MPCA, NPE, KNPE, SNPE, and DeNPE models despite limited runs and nonlinearity challenges.	Liu et al. [2024a]
Multi-way concurrent locally weighted projection regression	**FD and QP in** a penicillin simulator	Addressing batch process challenges: different operation conditions; multiphase and nonlinearity characteristics.	Zhang et al. [2023b]
Convolutional block attention module convolutional neural network (CBAM)-CNN	**QP in** two penicillin fermentation processes	Addressing batch process challenges: different initial conditions; multiphase characteristics; nonlinearity, providing better generalization performance.	Zhao et al. [2023]
Multi-stage fusion regression network (MSFRN)	**QP in** a penicillin batch process	Addressing the oversight of correlations among stages, leading to poor quality prediction in batch processes.	Yao et al. [2023]

For real-time prediction of quality variables and to capture system dynamics within and between batches, especially for dealing with uneven batch lengths, RNN-based models are recommended. CNN-based models are more suitable for capturing multi-phase nonlinear processes and offer better generalization performance.

Chapter 3

A Novel Unsupervised Approach for Batch Process Monitoring using Deep Learning

Overview[1]

Process monitoring is an important tool used to ensure safe operation of a process plant and to maintain high quality of end products. The focus of this chapter is on unsupervised Statistical Process Control (SPC) of batch processes using Deep Learning (DL). A DL architecture referred as Multiway Partial Least Squares Autoencoder (MPLS-AE) is proposed and trained using a genetic optimization algorithm with a novel objective function that directly maximizes the average fault detection rate ($\overline{\text{FDR}}$). The efficacy of the proposed method is demonstrated on an industrial scale Penicillin process. Comparisons of

[1]Adapted from Agarwal, Piyush, et al. "A Novel Unsupervised Approach for Batch Process Monitoring using Deep Learning." doi.org/10.1016/j.compchemeng.2022.107694

the proposed algorithm with linear Multiway Principal Component Analysis (MPCA) and Multiway Partial Least Squares (MPLS) based fault detection (FD) algorithm, trained with the same objective as used by the DL model, demonstrates the superiority of the deep learning based approach. The use of dynamic control limits significantly improves the detection rates for both the linear and DL models.

3.1 Introduction

Batch fermenters are one of the most common unit operations used in modern industries for the manufacturing of pharmaceuticals, biotechnological products, semiconductors etc. Efficient process monitoring is crucial to operate these processes safely and to keep the critical states of the process, such as temperature, pressure, pH, etc. within their optimal ranges of operation. Furthermore, accurate process monitoring model will drive necessary corrective actions to maintain safe operation and optimal productivity. Online fault detection (FD) is a technique to monitor optimal operation and safety of the process. Numerous FD algorithms have been developed in literature so far. FD algorithms can be broadly classified into three different classes according to the type of model used for FD: (i) knowledge-based, (ii) model-based and (iii) data-based methods. Data-based approaches are generally preferred since they do not require accurate mechanistic models of the process which are often difficult to obtain. For example, biochemical processes are difficult to model by first principles equations due to insufficient understanding about the metabolic behaviour of micro-organisms used in the system. Data-based FD algorithms can be either supervised or unsupervised. In the supervised learning approach, a model is trained with labeled samples corresponding to normal or faulty operation [Agarwal et al., 2019, 2021, 2020] whereas, the unsupervised learning approaches learn patterns directly from unlabeled data. Thus, the unsupervised approach does not require data that is labeled with faulty or normal operation status information. From an industrial point of view, this is an attractive option since most data in an industrial setting is obtained during normal operation and

faulty data may not be available or may be insufficient. Hence, in this work, we pursue an unsupervised learning approach for statistical process monitoring.

Statistical Process Control (SPC) algorithms are either unsupervised, e.g. principal component analysis (PCA) [Wise et al., 1990], independent component analysis (ICA) [Kano et al., 2003], or supervised, e.g. partial least squares (PLS) [MacGregor et al., 1994], qualitative trend analysis (QTA) [Maurya et al., 2005], and Fisher discriminant analysis [Chiang et al., 2000]. While PLS based approaches generally require an output label indicating fault or normal status associated to each sample for model training, the method can still be used in unsupervised learning by using a dummy variable, instead of labeled samples, as an output that is indicative of the accumulation of species during the batch operation [Chen and Liu, 2002]. Many commercial applications of PLS for FD in bio-pharmaceutical batch processes use the fermentation time (also known as timestamp) as an output for unsupervised FD learning tasks when labeled outputs are not available [Bylesjö et al., 2006]. However, other indicative variable measurements such as biomass can be used but are often not available at each time interval. Algorithms based on PCA or PLS and its variants (eg. MPCA, MPLS, etc.) have been extensively demonstrated in fault detection of industrial fermentation processes [Lennox et al., 2001, Ündey et al., 2003, Kourti, 2005, Chiang et al., 2006, Goldrick et al., 2017]. The strength of these algorithms lie in their ability to extract features that are most informative about the input space (PCA and MPCA) and that are most informative about the output (PLS and MPLS) by compressing a high dimensional input space and input-output space respectively into a lower dimensional latent variable space that can be used for FD. However, these algorithms are generally based on linear decomposition techniques and fail to accurately extract non-linear information when applied to nonlinear processes. Although less common in industrial practice, nonlinear extensions of PCA and PLS methods have also been reported. These are based on the use of nonlinear kernel based transformations of either the input variables for PCA or the inner relation between the projections of the input

and output variables respectively. The relative advantages of neural networks over kernel based methods have been reported in different theoretical studies. For example, Ghorbani et al. (2020) [Ghorbani et al., 2020] showed that kernel methods suffer from a curse of dimensionality in learning nonlinear target functions with low dimensional projections. For example, the representation of one sigmoidal function requires the use of many Gaussian kernels for accurate description. Hence, to capitalize on this advantage of neural networks over kernel methods, a deep learning neural network (NN) architecture is selected for the formulation of FD scheme in the current work.

Deep learning models extract non-linear features effectively and have been widely studied in the literature for developing different FD methodologies. Various deep learning architectures such as convolutional neural networks (CNN) [Chadha et al., 2019], recurrent neural networks (RNN) [Zhang et al., 2019a, Ren and Ni, 2020], and autoencoders neural networks (AE-NNs) [Cheng et al., 2019, Park et al., 2019, Yin et al., 2020] have been utilized in process monitoring. In particular, autoencoders (AE) are deep neural networks with the same number of units in the input and output layers with one or more low-dimensional hidden layers. AE-NNs can be intuitively understood as non-linear PCA in terms of their ability to compress the input space into a latent variable space. These networks consists of three parts, namely encoder, decoder and the embedding. Encoder compresses the input space to produce embedding, then the decoder reconstructs the input from embedding. It is trained in an unsupervised fashion usually by minimizing a reconstruction loss function [Yan et al., 2016, Yu and Zhao, 2019]. Denoising autoencoders and contractive autoencoders are variants of AE-NNs that had been applied to monitor a continuous chemical process by tracking the H^2 statistic metric [Yan et al., 2016]. In another work, a two-dimensional deep correlated representation learning (2D-DCRL) has been proposed to monitor a penicillin batch fermenter [Jiang et al., 2019] where an AE model is implemented to extract the relations between the process variables. Finally, the canonical correlation analysis (CCA) of 2D matrices is used to capture the dynamic fea-

tures between batches. Two common metrics, Hotelling T^2 and Q-statistics, have been used to detect faults with AE. In another application, a multi-way Laplacian autoencoder (MLAE) has been proposed to monitor batch fermentation processes [Gao et al., 2020]. In comparison with traditional AEs, the MLAE considers the local structure of the normal process data and the stochastic deviations among batches are captured by the Laplacian matrix of the regularization term which improves the performance of the process monitoring model. In Chen et al. (2020), to capitalize on the feature extraction ability of AE-NNs, a one-dimensional convolutional autoencoder (1D-CAE) had been applied for monitoring a penicillin fermentation process [Chen et al., 2020b]. The 1D-CAE shows a better performance than typical DNNs due to its ability to extract features in the data. Although the above applications of AE-NNs were effective for FD, these methods were trained with a loss function as input reconstruction error but do not explicitly consider the fault detection performance. In contrast, in the current work, we propose a novel objective function for training the AE-NNs that directly considers and improves the average fault detection rate ($\overline{\text{FDR}}$).

Specifically, the FD problem in the current work proposes an AE architecture (Multiway Partial Least Squares Autoencoder) that is trained with a novel objective function, tailored for detecting faults efficiently through an unsupervised learning approach. The idea is to learn the distribution of normal batches in order to differentiate them from the faulty batches. The algorithm involves two main steps: i- the MPLS-AE is trained to explain the variation of process variables with respect to an average batch trajectory and also predict the output variable simultaneously. ii- the control limits are estimated both on learned latent features and residuals using kernel density estimation at each time interval.

In order to demonstrate the advantages of the proposed MPLS-AE methodology for fault detection, systematic comparisons are conducted with Multiway Principal Component Anaysis (MPCA) and Multiway Partial Least Squares (MPLS) on an industrial scale

Penicillin Simulator. To assess the advantage of the novel loss function over traditional loss function, both MPLS and MPLS-AE are compared with and without the $\overline{\text{FDR}}$-based objective function respectively. Both the MPLS and MPLS-AE models are trained using the time of fermentation as the dummy output variable. Another important contribution of the work is to assess whether the consideration of the dummy (indicative) output variable for process monitoring, e.g. fermentation batch age, enhances fault detection accuracy.

The paper is organized as follows. The mathematical background of process monitoring with MPCA, MPLS and a brief description of AE-NNs model are introduced in Section 3.2. Section 3.3 presents the proposed method for FD, the novel loss function and description of the penicillin process simulator used in the case study. The application of the proposed method to a Penicillin batch process and comparisons with the traditional methods are presented in Section 4 followed by conclusions in Section 5.

3.2 Preliminaries

This section briefly reviews the fundamentals of MPCA, MPLS and AE-NNs and its application in FD. For each of these methods, the dataset is divided into 3 parts namely training, validation and test dataset. The general idea is to estimate model parameters using the training set while the validation set is used to tune the hyper-parameters of each model. For example, the validation set is used to select the number of principal components for both MPCA and MPLS and to select the optimal hyper-parameters for deep learning models such as number of neurons, number of layers, learning rate, batch-size etc. Finally, the efficacy of the method is evaluated using the test dataset.

3.2.1 Multiway Principal Component Analysis (MPCA)

Multiway PCA is an extension of PCA that operates on data organised in a three dimensional array. For a batch process this array is $\mathbf{X}_{3D} \in \mathbb{R}^{k \times b \times j}$ where k is number of samples in each batch b and j is the number of measured process variables. PCA is performed on an unfolded $\mathbf{X} \in \mathbb{R}^{kb \times j}$ matrix where $n = kb$ is the total number of training samples. PCA projects the high dimensional correlated input space $\mathbf{X}$ onto a lower dimensional orthogonal space while preserving the variance [Wise and Gallagher, 1996]. The samples are mean centered and scaled to unit variance prior to applying PCA. The decomposition is defined as follows:

$$\mathbf{X} = \sum_{k=1}^{K} \mathbf{t}_k \mathbf{p}_k^T + \mathbf{E} = \mathbf{T}\mathbf{P}^T + \mathbf{E} \tag{3.1}$$

$$s.t. \quad \mathbf{P}^T\mathbf{P} = I \tag{3.2}$$

where $\mathbf{T} \in \mathbb{R}^{n \times K}$, $\mathbf{P} \in \mathbb{R}^{j \times K}$ and $\mathbf{E} \in \mathbb{R}^{n \times j}$ denote the matrices of scores, loadings and residuals respectively. K is the number of principal components that are retained based on a relevant objective function by using the validation dataset. The product $\mathbf{TP^T}$ describes the process variability. Two statistical metrics Hotelling T^2 and Q-statistic are used for process monitoring. T^2 measures the variability of each sample [Kourti and MacGregor, 1996] calculated as follows :

$$T^2_{new} = \mathbf{t}_{new} \frac{(\mathbf{T}\mathbf{T}^T)^{-1}}{n-1} \mathbf{t}^T_{new} \tag{3.3}$$

where n is the number of normal samples and $\mathbf{t}_{new}$ is the score for a new observation calculated as follows:

$$\mathbf{t}_{new} = \mathbf{x}_{new}\mathbf{P}(\mathbf{P}^T\mathbf{P})^{-1} \tag{3.4}$$

An abnormal observation, i.e. an observation denoting a faulty condition, can be detected if T^2_{new} exceeds the control limit T^2_α [Lee et al., 2004a] which can be calculated as follows:

$$T^2_\alpha = \frac{K(n-1)}{n-K} F_{K,n-K,\alpha} \tag{3.5}$$

where $F_{K,n-K,\alpha}$ is the F-distribution with n and $n-K$ degrees of freedom and alpha is the confidence limit. Abnormal samples can also be detected using the sum of squares of residuals (SPE) or Q-statistic [Jackson and Mudholkar, 1979] calculated as follows:

$$\begin{aligned} Q_{new} &= \mathbf{e}_{new}\mathbf{e}^{\mathbf{T}}_{new} \\ \mathbf{e}_{new} &= \mathbf{x}_{new} - \mathbf{t}_{new}\mathbf{P}^{\mathbf{T}} \end{aligned} \tag{3.6}$$

Since these residuals follow a chi-squared distribution (χ^2), the control limit for Q-statistic [Jackson and Mudholkar, 1979] is defined as follows:

$$\begin{aligned} Q_\alpha &= \theta_1 \Big[\frac{z_\alpha (2\theta_2 h_0^2)^{0.5}}{\theta_1} + 1 + \frac{\theta_2 h_0 (h_0 - 1)}{\theta_1^2} \Big]^{\frac{1}{h_0}} \\ \mathbf{V} &= \frac{\mathbf{E}\mathbf{E}^{\mathbf{T}}}{n-1} \\ \theta_i &= trace(\mathbf{V}^i); i = 1, 2, 3 \\ h_0 &= 1 - \frac{2\theta_1\theta_3}{3\theta_2^2} \end{aligned} \tag{3.7}$$

where $\mathbf{V}$ is the co-variance matrix of $\mathbf{E}$ and z_α denotes a normal deviate with significance level α. The metrics reviewed above are commonly referred to as static control limits since they assumed that the input data at all time intervals follows the same statistical distribution. Furthermore, the calculation of T^2_α and Q_α assumes that the input data corresponding to normal operation follows a Gaussian distribution which may not be accurate in practical problems. To correct for this, a kernel density estimation is used in the proposed work as explained in Section 3.1.

Online process monitoring can be carried out in two steps: i- Historical data records of normal operation samples are split between training and validation datasets. Pre-processing of datasets is carried out as described above by using the mean and variance of the training dataset. Training dataset is used to estimate scores $\mathbf{T}$ and loadings $\mathbf{P}$ of the MPCA model while the validation dataset is used to select the optimal number of principal components K using a relevant criterion. In this work, the number of principal components are selected based on two different objectives: a- the commonly used criterion of minimizing the reconstruction error (or a pre-determined threshold of explained variance) and b- maximization of fault detection rate in the validation dataset. The latter will be shown to be superior in the case study in Section 4.2. Thereafter, the control limits T^2_α and Q_α are evaluated using Eq 3.5 and 3.7. ii- Each newly acquired sample is mean-centered and normalized based on the training set mean and variance. T^2_{new} and Q_{new} of the new sample are evaluated using Eq 3.3 and 3.6 and abnormality is determined if any one of these metrics exceeds its control limits.

3.2.2 MPLS

Similar to MPCA, MPLS is an extension of traditional PLS for handling three dimensional data arrays. It estimates the projection of both input $\mathbf{X}$ and output variables $\mathbf{Y}$ onto a lower dimensional space by finding multidimensional directions in the input space such that it maximizes the covariance between $\mathbf{X}$ and $\mathbf{Y} \in \mathbb{R}^{n \times p}$. MPLS extracts the scores of input and output data such that the covariance between inputs and outputs is maximized

as follows:

$$\mathbf{X} = \sum_{k=1}^{K} \mathbf{t_k} \mathbf{p_k^T} + \mathbf{E} = \mathbf{T}\mathbf{P}^\mathbf{T} + \mathbf{E}$$

$$\mathbf{Y} = \sum_{k=1}^{K} \mathbf{u_k} \mathbf{q_k^T} + \mathbf{F} = \mathbf{U}\mathbf{Q}^\mathbf{T} + \mathbf{F}$$

$$\text{s.t.} \quad \max \quad \mathbb{E}\Big\{ \Big(\mathbf{T} - \bar{\mathbf{T}} \Big) \Big(\mathbf{U} - \bar{\mathbf{U}} \Big) \Big\}$$

$$\implies \quad \max \quad \text{corr } (\mathbf{T}, \mathbf{U}) \times \sqrt{\mathbf{T}^T\mathbf{T}} \times \sqrt{\mathbf{U}^T\mathbf{U}} \tag{3.8}$$

where $\mathbf{T} \in \mathbb{R}^{n \times K}$ and $\mathbf{U} \in \mathbb{R}^{n \times K}$ contain the score vectors of input and output variables and $\mathbf{P} \in \mathbb{R}^{j \times K}$ and $\mathbf{Q} \in \mathbb{R}^{p \times K}$ contain the loading vectors of input and output variables respectively. $\mathbf{E} \in \mathbb{R}^{n \times j}$ and $\mathbf{F} \in \mathbb{R}^{n \times p}$ are the residuals. In a supervised learning approach an MPLS model can be trained for FD with both input data and output labels where the latter are related to the faulty and normal status of each sample. The challenge in formulating an unsupervised FD MPLS algorithm is that output labels or output measurements at each time interval are not available for training. To address this challenge, current commercial applications consider an dummy output variable to be predicted by the MPLS model. For example, the fermentation time has been used in bioreactor applications as an dummy output variable for training the MPLS model. The rationale for using MPLS with such dummy variable is that it may result in a more informative representation about the normal operation boundary with a smaller number of latent variables as compared to MPCA. A smaller number of latent variables is expected to result in less over-fitting of measurement noise. The use of the fermentation time as a dummy variable is also biochemically motivated by the fact that the cumulative biomass and productivity are approximately proportional to the fermentation time. The role of the dummy variable towards FD accuracy will be explicitly investigated later in the case study. Similar to MPCA, to detect abnormal samples, Hotelling T^2 and Q (Q_x)-statistics are evaluated and compared to the control limits using Eqs. 3.3-3.7.

3.2.3 Autoencoder Neural Network (AE-NN)

A traditional AE-NN is a neural network model composed of three parts: encoder, decoder and embedding, as shown in Figure 3.1. An AE-NN is trained in an unsupervised fashion to extract underlying patterns in the data and to reduce the dimensionality of the input data. The encoder is trained to compress the input data onto a reduced latent space defined by a hidden layer and the decoder decompresses back the hidden layer outputs into the reconstructed inputs. Let us consider $\mathbf{x}_i$ as the input vector to an AE-NN, then the operation performed by the encoder for a single hidden layer between the input variables to the latent space $\mathbf{z} \in \mathbb{R}^{d_z}$ variables (latent variables) for time sample i ($\mathbf{x}_i$) is as follows:

$$\mathbf{z}_i = f_e(\mathbf{W}_e \mathbf{x}_i + \mathbf{b}_e) \tag{3.9}$$

where f_e is a chosen non-linear activation function for the encoder, $\mathbf{W}_e \in \mathbb{R}^{d_z \times j}$ is an encoder weighting matrix, and $\mathbf{b}_e \in \mathbb{R}^{d_z}$ is a bias vector. The decoder reconstructs the input variables from the feature or latent space $\mathbf{z}_i \in \mathbb{R}^{d_z}$ according to the following operation:

$$\hat{\mathbf{x}}_i = f_d(\mathbf{W}_d \mathbf{z}_i + \mathbf{b}_d) \tag{3.10}$$

where f_d is a chosen activation function for the decoder, $\mathbf{W}_d \in \mathbb{R}^{j \times d_z}$ and $\mathbf{b}_d \in \mathbb{R}^j$ is a decoder weighting matrix and a bias vector respectively. The 'tanh' function is used for both transforming the inputs into the latent variables and for reconstructing back the inputs from the latent variables. The AE-NN is trained based on the following minimization problem:

$$\min_{W_e, W_d, b_e, b_d} \frac{1}{n} \Big(\sum_{s=1}^{n} \big(\mathbf{x}_s - \hat{\mathbf{x}}_s \big)^2 \Big) \tag{3.11}$$

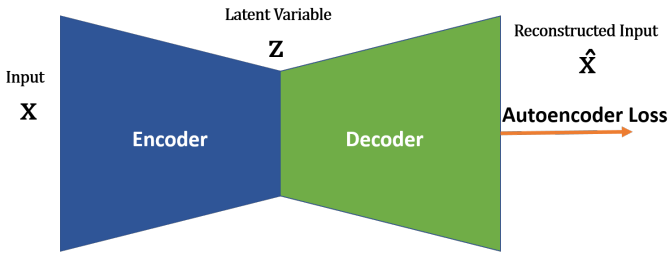


Figure 3.1: Schematic of a single layer Autoencoder Neural Network (AE-NN)

where n is the number of samples.

3.3 Proposed Methodology

3.3.1 Multiway Partial Least Squares Autoencoder (MPLS-AE)

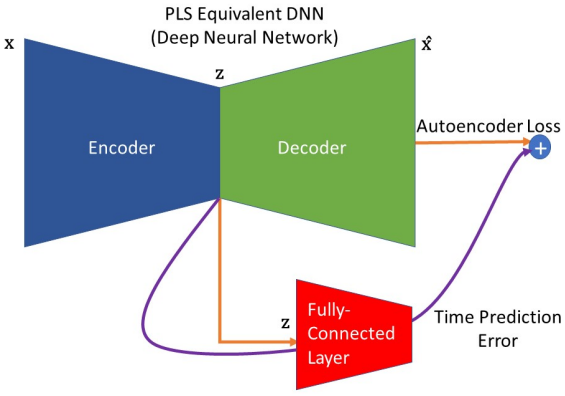


Figure 3.2: Schematic of a Multiway Partial Least Squares Autoencoder (MPLS-AE)

Motivated by the MPLS algorithm, we propose an equivalent Deep Learning architecture that can be trained to simultaneously predict output $\mathbf{Y}$ along with reconstructing inputs $\mathbf{X}$. Since maximizing the covariance between scores $\mathbf{T}$ and $\mathbf{U}$ in MPLS (refer Eq.

3.8) implies maximizing three simultaneous objectives i.e. i- best explanation of inputs $\mathbf{X}$ (given by $\mathbf{T}^T\mathbf{T}$), ii- best explanation of outputs $\mathbf{Y}$ (given by $\mathbf{U}^T\mathbf{U}$) and iii- greatest relationship between $\mathbf{X}$ and $\mathbf{Y}$ (given by correlation between $\mathbf{T}$ and $\mathbf{U}$) in Eq. 3.8. These three objectives are equivalent to minimizing a weighted sum of the input reconstruction error (best explanation of inputs) and the output variable prediction errors (best explanation of outputs) according to Eq. 3.12. Minimizing the objective function along with the weighting factor λ in Equation 12 is equivalent to maximizing correlation between inputs and outputs as is done in PLS. It should be noticed that λ lies between $(0, 1)$ in order to give higher importance to prediction of output variables.

The key idea is to attach a fully connected layer to the bottleneck layer of the AE-NN as shown in Figure 3.2 to predict the indicative output (dummy) variables. This is referred henceforth in this study as Multiway Partial Least Squares Autoencoder (MPLS-AE) NN. In this work, we consider fermentation time ($\mathbf{t}$) as the indicative dummy variable as considered in commercial MPLS applications for FD [Bylesjö et al., 2006]. The MPLS-AE model is trained based on the following loss function:

$$J^{\mathbf{MPLS-AE}} = \frac{1}{n}\Big(\sum_{s=1}^{n}\big(t_s - \hat{t}_s\big)^2 + \lambda\sum_{s=1}^{n}\big(\mathbf{x}_s - \hat{\mathbf{x}}_s\big)^2\Big) \tag{3.12}$$

where n, $\mathbf{x}_s$, $\hat{\mathbf{x}_s}$, t_s and $\hat{t_s}$ are the number of training samples, an input sample, reconstructed input sample, output (dummy) variable and predicted output (dummy) variable respectively. However, the MPLS-AE architecture can be trained by other user-defined loss functions for specific purposes. In this work we propose a novel objective function specifically tailored for FD application as described in the next sub-section.

Detection of abnormality with the MPLS-AE model is based on the use of H^2 and SPE metrics instead of the metrics T^2 and Q that are used with the linear methods (MPCA and

MPLS). H^2 represents the variability of the latent variables with respect to their historical means as follows:

$$H^2_{k,b} = \sum_{d_z=1}^{R} \frac{(z^{deepest}_{k,b,d_z} - z^{mean}_{k,d_z})^2}{z^{variance}_{k,d_z}} \tag{3.13}$$

where $H^2_{k,b}$ is H^2 of the k^{th} sample in the b^{th} batch, $z^{deepest}_{k,b,d_z}$ is the d_z^{th} dimension of the latent variables of the k^{th} sample in b^{th} batch, z^{mean}_{k,d_z} and $h^{variance}_{k,d_z}$ are the mean and variance of the d_z^{th} dimension of the k^{th} samples in all training batches, respectively and R is the number of units in the bottleneck layer of MPLS-AE NN model. SPE is the metric used to measure the residual error between reconstructed input sample and the original input sample and is calculated as follows:

$$SPE_{k,b} = \sum_{d_z=1}^{R} (\mathbf{x}^b_{k,d_z} - \hat{\mathbf{x}}^b_{k,d_z})^2 \tag{3.14}$$

where $SPE_{k,b}$ is the SPE of the k^{th} sample in the b^{th} batch, $\mathbf{x}^b_{k,d_z}$ and $\hat{\mathbf{x}}^b_{k,d_z}$ is the d_z^{th} dimension of the k^{th} sample in the b^{th} batch for input and reconstructed input respectively.

Since the input data may not follow a Gaussian distribution, a kernel density estimation (KDE) [Parzen, 1962a] procedure is used to obtain control limits H^2_α and SPE_α metrics. Further, we relax an earlier assumption made for the MPCA and MPLS based FD model that the distribution of input data does not change with time. Instead, both H^2_α and SPE_α are allowed to change with time based on the current estimate of KDE and thus they will be referred to henceforth as dynamic control limits $H^2_{k,\alpha}$ and $SPE_{k,\alpha}$. To calculate these limits, we use Gaussian kernels to estimate the probability distribution function of univariate data ($\mathbf{H}^2_k$ and $\mathbf{SPE}_k$) for each time interval k as follows:

$$F(H_k^2) = \sum_{i=1}^{b} G\Big(\frac{H_k^2 - H_i^2}{b_w}\Big) \tag{3.15}$$

$$G(x_i;\sigma) \propto exp\Big(-\frac{{x_i}^2}{2\sigma^2}\Big) \tag{3.16}$$

where $F(H_k^2)$ is the probability density function, $G(x_i;\sigma)$ is the Gaussian kernel and b_w is a bandwidth parameter that controls the smoothness of the distribution and it is selected from the Silverman algorithm [Silverman, 2018]. The $H_{k,\alpha}^2$ for a pre-specified significance level α is then calculated using the equation below:

$$1 - \alpha = \int_{\min \mathbf{H}_{k,train}^2}^{H_{k,\alpha}^2} F(H_k^2)\ dH_k^2 \tag{3.17}$$

Similar procedure (Eqs. 3.15-3.17) can be followed to evaluate the residual control limits $SPE_{k,\alpha}$.

3.3.2 Novel Objective Function for Maximizing Fault Detection Rate

In previous studies on process monitoring with MPCA and MPLS models, the objective function to be minimized for selecting the number of principal components, i.e. hyper-parameters in MPCA and MPLS, is either the mean square error (MSE) of input reconstruction (refer Eq. 3.18) in MPCA or the MSE of output prediction (refer Eq. 3.19) in

MPLS.

$$J^{MPCA} = \frac{1}{n}\sum^{b}\sum^{k}\left(x_k - \hat{x}_k\right)^2 \tag{3.18}$$

$$J^{MPLS} = \frac{1}{n}\sum^{b}\sum^{k}\left(\mathbf{t}_k - \hat{\mathbf{t}}_k\right)^2 \tag{3.19}$$

where $\hat{\mathbf{x}}_k$ is a reconstructed input sample and $\hat{\mathbf{t}}_k$ is a predicted indicative output (dummy) variable. However, since maximizing the $\overline{\text{FDR}}$ is our main objective, we propose an alternative objective function for model training.

It should be emphasized that since all the samples used for training of an unsupervised learning model correspond to normal operation, $T^2_{k,train}$ of each sample at time 'k' should be less than the associated control limit i.e. $T^2_{k,\alpha}$ (Dynamic control limits). An objective function that is most relevant for fault detection should measure the number of misdetected samples, i.e. normal samples detected as abnormal (also known as false alarm rate (FAR)) and abnormal samples detected as normal, according to the dynamic control limits proposed above. Hence, a fault detection motivated objective function $J^{MPCA-MPLS,FDR}$ is proposed for selecting number of principal components in MPCA and MPLS as follows:

$$J^{MPCA-MPLS,FDR} = \frac{1}{n}\sum^{b}\sum^{k}\left(\left[T^2_{k,train} - T^2_{k,\alpha}(t)\right] > 0\right)||\ \left(\left[Q_{k,train} - Q_{k,\alpha}\right] > 0\right) \tag{3.20}$$

where $T^2_{k,train}$ and $T^2_{k,\alpha}$ are a function of scores and nPCs K. These hyper-parameters are selected using Eq 3.20 for both MPCA and MPLS models. Thus, the nPCs that results in the minimum value of $J^{MPCA-MPLS,FDR}$ in the validation dataset are selected for the final model.

Similarly, to incorporate FDR explicitly in the loss function for training MPLS-AE model, the following loss function is proposed in this study:

$$J^{MPLS-AE,FDR} = \frac{1}{n}\sum^{b}\sum^{k}\left(\left[H^2_{k,train} - H^2_{k,\alpha}\right] > 0\right)||\ \left([SPE_{k,train} - SPE_{k,\alpha}] > 0\right) \quad (3.21)$$

The above loss function (Eq. 3.21) evaluates the number of miss-detected samples with the MPLS-AE model. Hence the loss function is directly related to the ($\overline{\text{FDR}}$) which is a function of $H^2_{k,train}$ and $Q_{k,train}$ and consequently these are a function of latent variables z (refer Eqs. 3.13 and 3.14). The proposed methodology for fault detection is schematically described in Figure 3.3.

3.3.3 Average Fault Detection Rate ($\overline{\text{FDR}}$)

Thereafter the average fault detection rate ($\overline{\text{FDR}}$) can be evaluated based on the average detection rates of normal and abnormal samples for MPCA and MPLS using metric M_1 and metric M_2 for MPLS-AE as follows:

$$M_1 : \begin{cases} \overline{FDR}_{normal} & = \frac{1}{n_{normal}}\sum_{k=1}^{n_{normal}}\sum^{b}\left([T^2_{k,b} - T^2_{k,\alpha}] \leq 0\right)\ \&\&\ \left([Q_{k,b} - Q_{k,\alpha}] \leq 0\right) \\ \overline{FDR}_{abnormal} & = \frac{1}{n_{abnormal}}\sum_{k=1}^{n_{abnormal}}\sum^{\mathbf{b}}\left([T^2_{k,b} - T^2_{k,\alpha}] > 0\right)\ ||\ \left([Q_{k,b} - Q_{k,\alpha}] > 0\right) \end{cases} \quad (3.22)$$

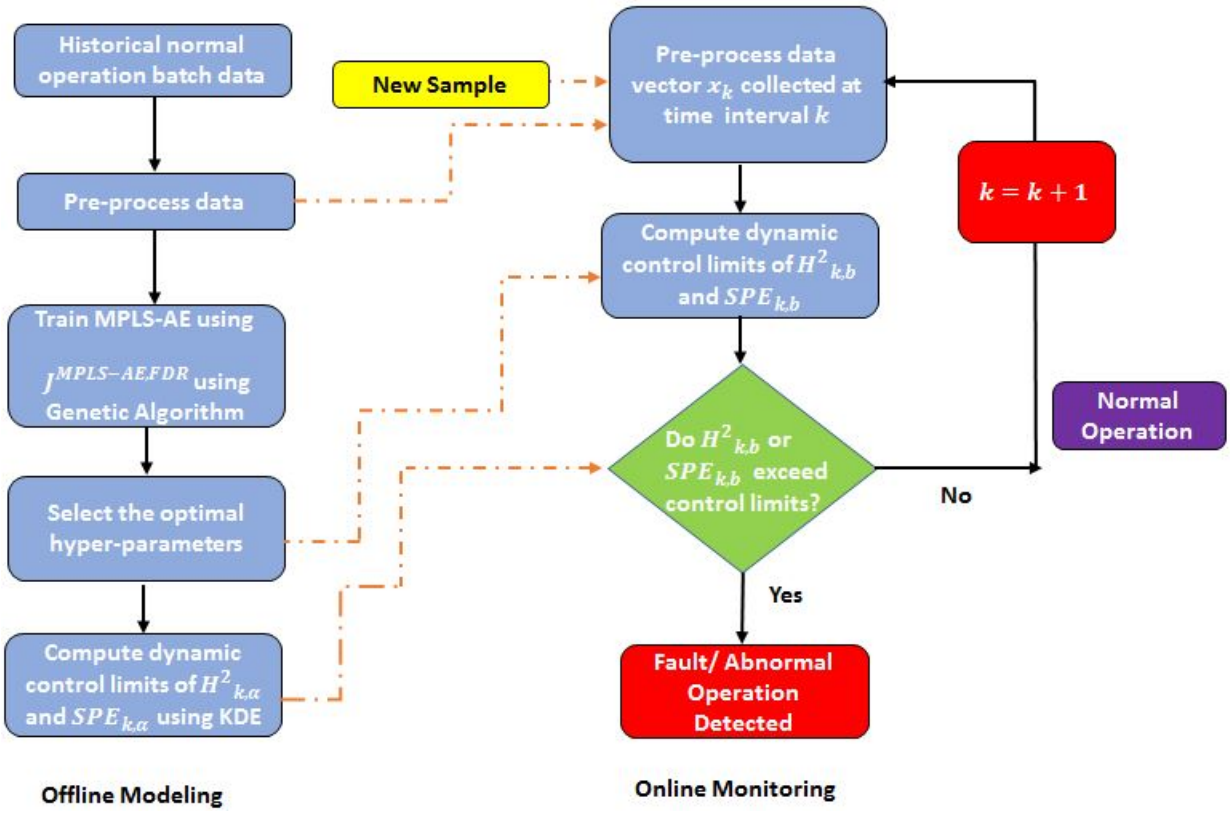


Figure 3.3: Flowchart for fault detection based on novel objective function

$$M_2 : \begin{cases} \overline{FDR}_{normal} & = \frac{1}{n_{normal}} \sum_{k=1}^{n_{normal}} \sum^{b} \left([H^2_{k,b} - H^2_{k,\alpha}] \leq 0 \right) \;\&\&\; \left([SPE_{k,b} - SPE_{k,\alpha}] \leq 0 \right) \\ \overline{FDR}_{abnormal} & = \frac{1}{n_{abnormal}} \sum_{k=1}^{n_{abnormal}} \sum^{\mathbf{b}} \left([H^2_{k,b} - H^2_{k,\alpha}] > 0 \right) \;||\; \left([SPE_{k,b} - SPE_{k,\alpha}] > 0 \right) \end{cases} \tag{3.23}$$

$$\overline{FDR} = \frac{n_{normal}\overline{FDR}_{normal} + n_{abnormal}\overline{FDR}_{abnormal}}{n_{normal} + n_{abnormal}} \tag{3.24}$$

where n_{normal} is the number of normal operation samples, $n_{abnormal}$ is the number of samples with faults, b is the total number of normal batches in the training dataset and $H^2_{k,\alpha}$ and $SPE_{k,\alpha}$ are the control bounds $H^2_{k,\alpha}$ and $SPE_{k,\alpha}$ of the k^{th} sample, respectively. A normal operation sample is detected correctly if both $T^2_{k,b}$ or $H^2_{k,b}$ and $Q_{k,b}$ or $SPE_{k,b}$ are below their respective control limits. Otherwise the sample will be incorrectly detected as abnormal. On the other hand, abnormal samples are correctly detected if either $H^2_{k,b}$ or $SPE_{k,b}$ or both exceed their control limits in case of MPLS-AE and $T^2_{k,b}$ or $Q_{k,b}$ or both exceed their control limits in case of MPCA and MPLS. Overall $\overline{FDR}$ is calculated as the weighted sum of $\overline{FDR}_{normal}$ and $\overline{FDR}_{abnormal}$ (refer Eq. 3.24).

Note: $J^{MPLS-AE,FDR}$ is trained with a GA (genetic optimization algorithm) using the solution from $J^{MPLS-AE}$ as the initial guess to speed up the search. This implies that the architecture for both the models are same.

3.3.4 Case study

The IndPenSim code [Goldrick et al., 2019] that simulates batch fermentation of Penicillin is used for the case study. A schematic of the process is shown in Figure 3.4. The figure also shows the available off-line and online measurements, manipulated variables and

controlled variables. The simulator is available at www.industrialpenicillinsimulation.com. The total biomass inside the bioreactor is calculated for 4 separate regions: actively growing regions (A_0), non-growing regions (A_1), degenerated regions (A_2) and autolysed regions (A_3) [Goldrick et al., 2015].
Growing regions (A_0):

$$\begin{aligned}
\frac{dA_0}{dt} &= r_b - r_{diff} - \frac{F_{in}A_0}{V} \\
\frac{dA_1}{dt} &= r_e - r_b + r_{diff} - r_{deg} - \frac{F_{in}A_1}{V} \\
\frac{dA_2}{dt} &= r_{deg} - r_a - \frac{F_{in}A_2}{V} \\
\frac{dA_3}{dt} &= r_a - \frac{F_{in}A_3}{V}
\end{aligned} \tag{3.25}$$

Total biomass ($X_{biomass}$):

$$X_{biomass} = A_0 + A_1 + A_2 + A_3 \tag{3.26}$$

where r_a, r_b, r_e, r_P, r_{diff}, r_{deg}, F_{in} and V are the rates of autolysis, branching, extension, differentiation, degeneration, total flow in and fermenter volume respectively. The dynamic balances describing production formation and substrate consumption are as follows:
Product formation (P):

$$\frac{dP}{dt} = r_P - r_h - \frac{F_{in}P}{V} \tag{3.27}$$

Substrate consumption (S):

$$\frac{dS}{dt} = -Y_{S/X}r_e - Y_{S/X}r_b - m_s r_m - Y_{S/P}r_P + \frac{F_S C_S}{V} + \frac{F_{oil}C_{oil}}{V} - \frac{F_{in}S}{V} \tag{3.28}$$

where r_P, r_h and r_m denote the rates of product formation, hydrolysis and maintenance, respectively, and m_s is the substrate maintenance term. Also, $Y_{S/X}$ and $Y_{S/P}$ indicate the

substrate yield coefficients of biomass and penicillin respectively, and F_S, F_{oil}, C_S, and C_{oil} are the sugar and soybean oil feed rate and concentrations, respectively.

A dataset of total 700 batches is generated using the simulator. The total fermentation time of each batch is 230 hours (1150 time samples for each batch) and involves 25 measured input variables. Six different types of faults are considered in this case study. Different faults occur at different time interval during a batch with varied magnitude as described in Table 3.1. The dynamic profiles of these faults (except for fault number 6 which is a combination of all the other faults) are shown in Figure 3.5. The dataset includes 100 normal operation batches, i.e. batches without fault, and 100 batches for each type of fault. Batches with normal operation are then divided between training and validation sets. 80 normal batches are used to train the MPCA, MPLS and MPLS-AE models and 20 batches are used as the validation dataset to obtain the optimal hyper-parameters for each FD model. The remaining 600 batches are used as the test dataset. First, the three dimensional training data array ($\mathbf{X}_1 \in \mathbb{R}^{1150\times25\times80}$) is unfolded to a matrix as described in Section 2.1 and 2.2 with dimensions $\mathbf{X} \in \mathbb{R}^{92000\times25}$. A row in $\mathbf{X}$ represents a sample data with 25 variables (columns). After unfolding, the training set samples are mean centered and normalized by the corresponding variance for the MPCA and MPLS models and normalized between 0 and 1 using min-max normalization for MPLS-AE model.

Table 3.1: Description of different types of fault.

Fault No.	**Fault**	**Magnitude**	**Time (hr)**
1	disturbance in aeration flow rate (L/hr)	20, 20	[20, 24], [100, 110]
2	disturbance in vessel back pressure (bar)	2, 2	[100, 104], [200, 230]
3	disturbance in substrate feed rate (L/hr)	2, 20, 20	[20, 30], [76, 92], [200, 214]
4	disturbance in base flow rate (L/hr)	5, 10	[80, 84], [140, 160]
5	disturbance in coolant flow rate (L/hr)	2	[70, 90]
6	all of the above faults		

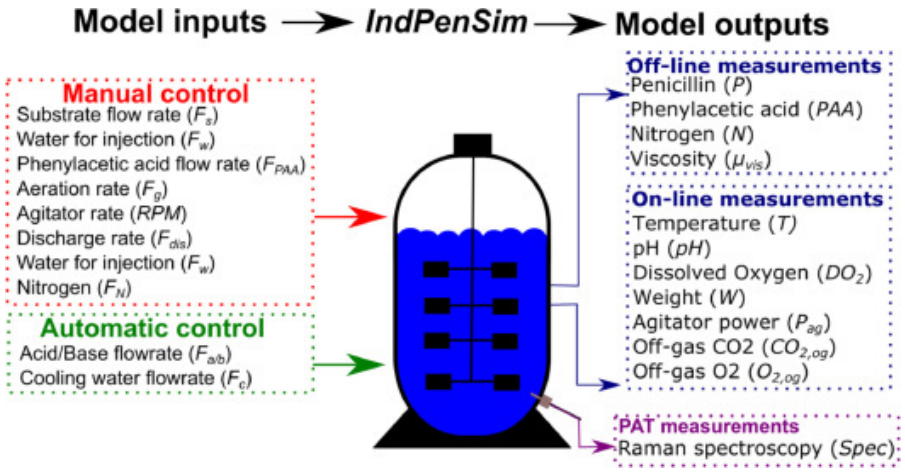


Figure 3.4: Summary of all model inputs and outputs recorded by IndPenSim

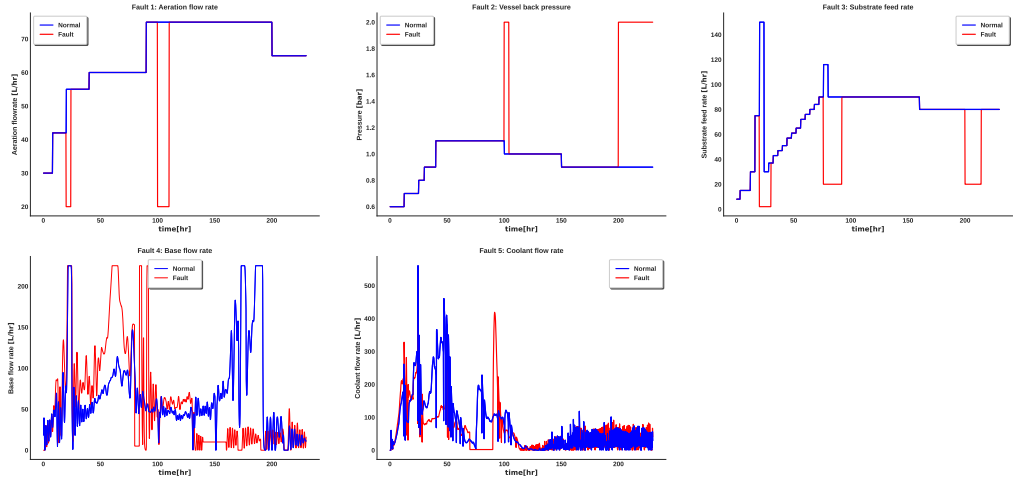


Figure 3.5: Profiles of measured variables in normal and faulty conditions.

3.4 Results and Discussions

The three FD methods:MPCA, MPLS and MPLS-AE are applied to the Industrial Penicillin Simulator (IndPenSim) and are compared in terms of $\overline{\text{FDR}}$ accuracy, use of static or dynamic control limits and use of the different proposed objective functions for selecting optimal hyper-parameters. The training, validation and test sets are the same for the three methods. As discussed in the previous section, the training dataset is used for calibration of model parameters, the validation set is used to find the optimal hyper-parameters and the test dataset is used to compare the performance of different methods with different objective functions (refer Section 3.3.2).

3.4.1 MPCA and MPLS with J^{MPCA} and J^{MPLS}

First, the linear MPCA and MPLS are compared to each other in terms of $\overline{\text{FDR}}$ accuracy (refer Table 2, Model No. 1-4). As mentioned previously in Section 3, MPLS uses the fermentation time as an indicative output (dummy) variable. An important first goal of this comparison is to investigate whether the use of the indicative output variable enhances the $\overline{\text{FDR}}$ accuracy of MPLS versus MPCA that does not use this output variable. Further, we also compare the use of static versus dynamic control limits with respect to average detection rate.

Static control limits T^2_α and Q_α

Following the methodology presented in Section 3.2.1, the nPCs K of MPCA or MPLS are determined by evaluating the traditional objective function reported in other studies (refer Eq. 3.18 and 3.19) on validation set. The nPCs that result in the lowest validation errors are 16 and 7 for the MPCA and MPLS models respectively (Model No. 1 and 2 in Table 3.2,). Figs. (3.6)-(3.7) show the performance of the MPCA and MPLS models with

static control limits in 6 batches of the test dataset that corresponds to 6 different faults. MPLS performs better than MPCA for detecting faults with FD test accuracy of 78.26% and 75.48% for MPCA. For these two models (Model No 1 and 2), the $\overline{\text{FDR}}$ of the normal and abnormal samples in different faulty parts of the test dataset are presented in Table 3.3. It should also be noticed that the nPCs in MPLS is significantly lower as compared to the nPCs of MPCA resulting in superior $\overline{\text{FDR}}$ for the test dataset. This result hints at the merit of including the output indicative variable, i.e. the fermentation time, in MPLS since it results in a smaller number of significant latent vectors.

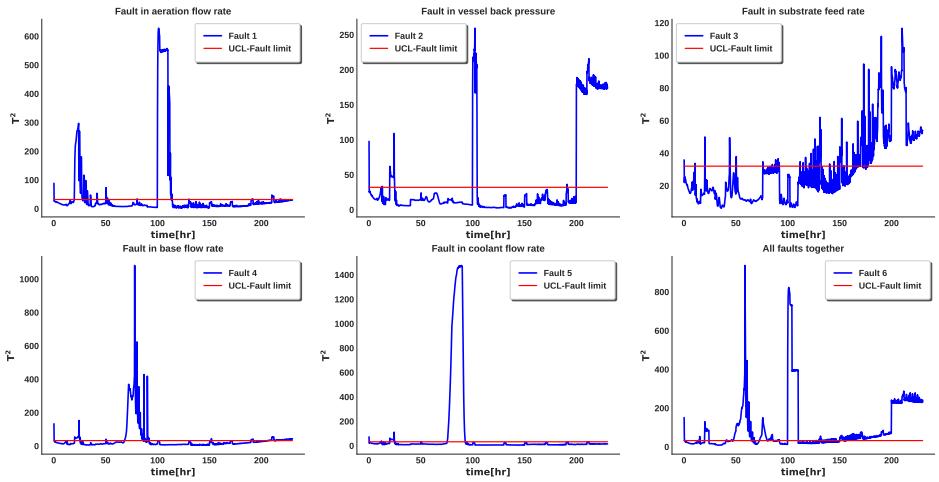


Figure 3.6: Plots of T_k^2 and static control limits T_α^2 for six different faulty batches of the test dataset based on J^{MPCA} as the objective function ($\alpha = 0.01$).

Dynamic control limits $T_{k,\alpha}^2$ and $Q_{k,\alpha}$

To improve the $\overline{\text{FDR}}$ results, dynamic control limits $T_{k,\alpha}^2$ and $Q_{k,\alpha}$ were used instead of the static control limits used above. Gaussian kernel density functions were fitted to the $T_{k,train}^2$ and $Q_{k,train}$ of the training dataset at each time interval k to estimate the

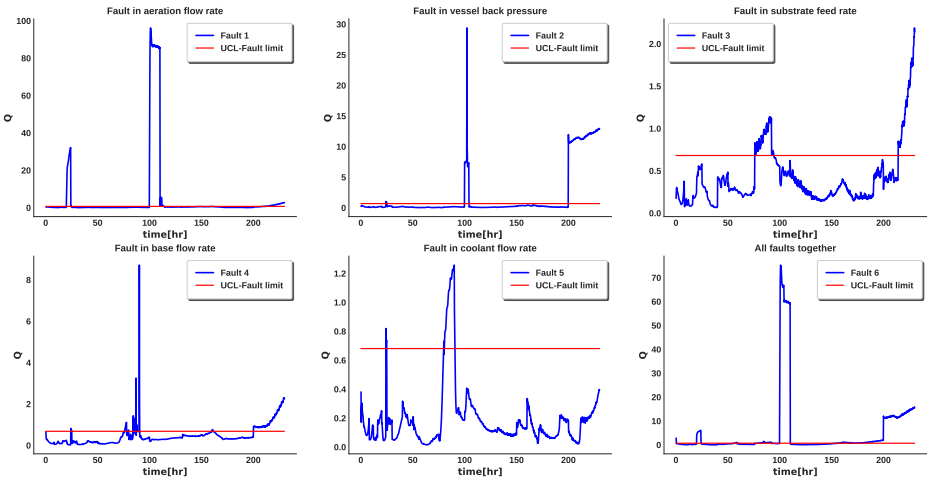


Figure 3.7: Plots of Q_k and static control limits Q_α for six different faulty batches of the test dataset based on J^{MPCA} as the objective function ($\alpha = 0.01$).

control limits $T^2_{k,\alpha}$ and $Q_{k,\alpha}$ with a pre-specified significance level $\alpha = 0.01$. nPCs K are selected using the objective function commonly used as described in section 3.4.1. It was observed that the number of principal components are same for both MPCA and MPLS as obtained with static limits. However, there is a significant increase in the $\overline{\text{FDR}}$, 83.20% with MPCA and 84.09% with the MPLS FD model (refer Model No. 3 and 4 in Table 3.2). This demonstrates that the use of dynamic control limits significantly increase the $\overline{\text{FDR}}$. The improvements in $\overline{\text{FDR}}$ accuracy with dynamic bounds arises from the ability of these bounds to deal with the model nonlinearity. Since MPCA and MPLS are linear models, following linearization arguments it is evident that different linear models should be used to approximate the nonlinear process behaviour at each time interval. Different linear models translate into different control bounds at different time intervals. At the same time, the superiority of MPLS as compared to MPCA is maintained also with dynamic control limits since the nPCs of MPLS are less as compared to MPCA. The overall superiority of MPLS as compared to MPCA motivated the development of a MPLS equivalent deep learning

architecture (MPLS-AE) to capitalize on the advantages of MPLS and the ability of AE to deal with nonlinear behaviour.

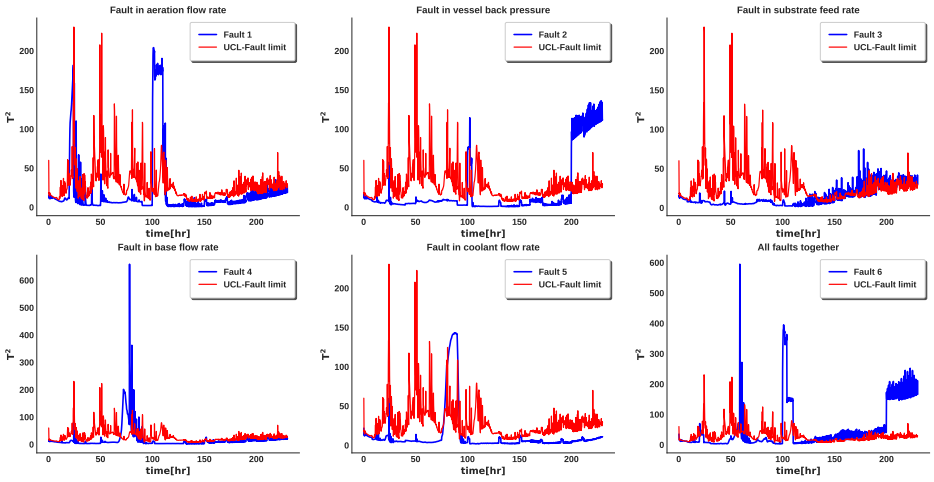


Figure 3.8: Plots of T_k^2 and dynamic control limits $T_{k,\alpha}^2$ of six different faulty batches of the test dataset by using J^{MPLS} as the objective function ($\alpha = 0.01$).

3.4.2 MPCA and MPLS with $J^{MPCA-MPLS,FDR}$

The focus of this subsection is on evaluating the performance of novel objective function $J^{MPCA-MPLS,FDR}$, proposed in this work, for linear MPCA and MPLS FD model and compare these with the traditional objective function. Then, we also compare the performance of FD models based on $J^{MPCA-MPLS,FDR}$ with static and dynamic control limits.

Static control limits T_α^2 and Q_α

Eqs. (3.3)-(3.7) were used to evaluate T_k^2, Q_k, T_α^2, and Q_α. The number of principal components K for both MPCA and MPLS were selected by evaluating the objective function

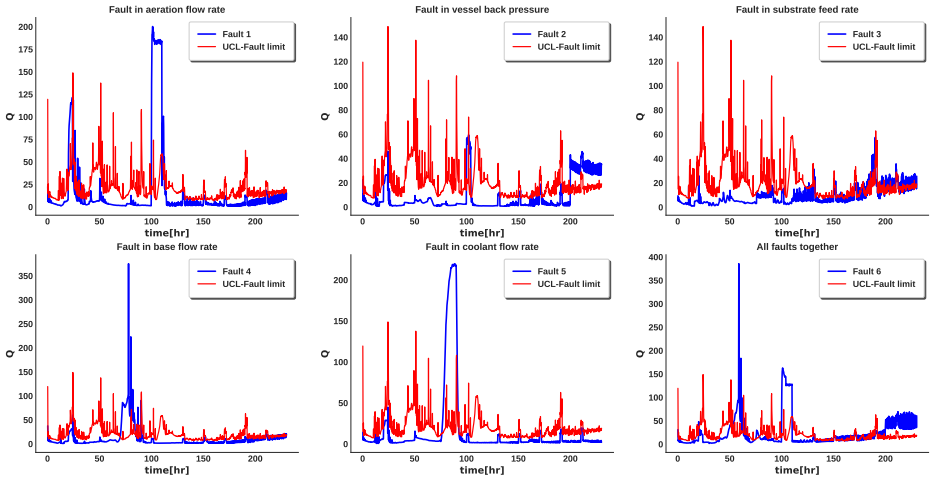


Figure 3.9: Plots of Q_k and dynamic control limits $Q_{k,\alpha}$ for six different faulty batches of the test dataset based on J^{MPLS} as the objective function ($\alpha = 0.01$).

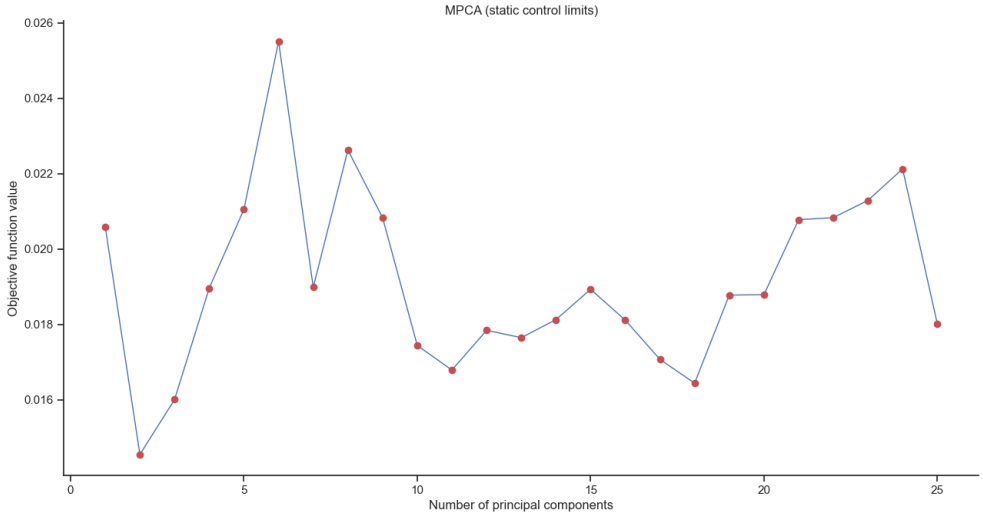


Figure 3.10: Novel FDR loss function as a function of principal components for Model 1 (MPCA with static control limits) for validation dataset

$J^{MPCA-MPLS,FDR}$ on validation dataset. It is observed (presented in Table 3.2) that 2 PCs were found to be optimal in order to minimize the validation error for both the MPCA and MPLS models. In this case, the MPCA and MPLS models achieve 81.83% and 81.99% $\overline{\text{FDR}}$ test accuracy respectively. This confirms our earlier argument that the $\overline{\text{FDR}}$ accuracy is highly related to the nPCs and the over-fitting of noise. The comparison of both the models with their counterparts in terms of $\overline{\text{FDR}}$ test accuracy, i.e the comparison between MPCA model with J^{MPCA} (Model No. 1; 16 PCs) versus $J^{MPCA-MPLS,FDR}$ (Model No. 5; 2 PCs) and MPLS model with J^{MPLS} (Model No. 2; 7 PCs) versus $J^{MPCA-MPLS,FDR}$ (Model No. 6; 2 PCs) are presented in Table 3.3. The MPCA model (Model No. 5; 2 PCs) based on the novel objective function performs better than the MPCA model (Model No. 1; 16 PCs) based on the commonly used objective function. It was also observed that Model No. 1 is better for faults 3, 4, 5 and 6 faults. Conversely, Model No. 5 has overall better performance (weighted average fault detection rate $\overline{FDR}$)and has lower false alarm rate (by about 10%). Moreover, the results are consistent in terms of the higher $\overline{FDR}$ for MPLS models as compared to MPCA FD models. It should be noticed that the resulting improvements are in the overall weighted fault detection rate and have significantly low false alarm rates but not necessarily on detection of individual faults since the latter will require supervised learning (see Table 2). For example, Table 3.3 shows that Model 5 achieves a higher weighted average test accuracy ($\overline{FDR}$, refer Table 3.2) but lower detection accuracy for individual faults.

On the other hand, it is also possible with our approach to enhance the individual fault detection rate while maintaining a similar or better $\overline{FDR}_{normal}$. The plot in Figure 3.10 shows the loss function obtained with Model 5 as a function of the number of principal components (nPCs). It can be seen that there are multiple local minima of the loss function at different nPCs 7, 11, and 18. Improvement in the individual fault detection rate ($\overline{FDR}_{abnormal}$) can be achieved by maintaining similar false alarm rate using the novel FDR objective function by choosing a different minima from the one used, preferably a minima

corresponding to nPC similar to the one used with the traditional objective function (nPC = 16, refer Table 2 in the manuscript). Individual $\overline{FDR}_{normal}$ and $\overline{FDR}_{abnormal}$ is estimated for the different local minima for individual faults in Table 3.4. It can be seen that for different local minima, both the overall average fault detection test accuracy ($\overline{FDR}$) and the individual fault detection accuracy of different faults consistently increases while maintaining similar false alarm rates. It should be noticed that different nPCs have higher $\overline{FDR}$ and none of these was obtained for the MPCA model with the traditional objective function. The MPCA model (Model 5) with FDR objective function improves both the false alarm rates as well as fault detection rates for 18 nPCs with a weighted average fault detection rate $\overline{FDR}$ of 82.53% (refer Table 3.4).

Dynamic control limits $T^2_{k,\alpha}$ and $Q_{k,\alpha}$

Based on the novel objective function $J^{MPCA-MPLS,FDR}$ with dynamic control limits $T^2_{k,\alpha}$ and $Q_{k,\alpha}$ (Model No. 7 and 8), the optimal number of nPCs was 18 and 22 for MPCA and MPLS model respectively. The use $J^{MPCA-MPLS,FDR}$ with dynamic control limits resulted in detection rates of 85.62% for MPCA and 85.72% for MPLS. This is consistent with the result obtained with static control limits. It is also important to note that MPCA FD model with dynamic control limits (Model No. 5) detect faults better than MPCA with static control limits (Model No. 5). The MPLS model based on the novel objective function $J^{MPCA-MPLS,FDR}$ (Model No. 8) performs better than the FD model with the traditional objective function J^{MPLS} (Model No. 4) in most of the time segments of the test dataset. The performance of the best linear FD model (MPLS Model No. 8), where the proposed objective function $J^{MPCA-MPLS,FDR}$ was used to select the nPCs along with the use of dynamic control limits is illustrated in Figs. (3.11)-(3.12).

Table 3.2: Comparison of ($\overline{\text{FDR}}$) test accuracy for different models

Model No.	Objective function	Model type	control limits	nPCs	$\overline{\text{FDR}}$ %(training/validation/test)
1	J^{MPCA}	MPCA	static	16	93.03/93.17/75.48
2	J^{MPLS}	MPLS	static	7	94.24/94.28/78.26
3	J^{MPCA}	MPCA	dynamic	16	97.47/97.91/83.20
4	J^{MPLS}	MPLS	dynamic	7	98.02/97.84/84.09
5	$J^{MPCA-MPLS,FDR}$	MPCA	static	2	93.92/94.18/81.83
6	$J^{MPCA-MPLS,FDR}$	MPLS	static	2	94.47/94.55/81.99
7	$J^{MPCA-MPLS,FDR}$	MPCA	dynamic	18	97.49/98.16/85.62
8*	$J^{MPCA-MPLS,FDR}$	MPLS	dynamic	22	97.47/98.20/85.72
9	$J^{MPLS-AE}$	MPLS-AE	dynamic	15	98.04/98.66/87.21
10*	$J^{MPLS-AE,FDR}$	MPLS-AE	dynamic	15	98.37/98.84/88.64

*Best linear and non-linear models are highlighted with red colour

Table 3.3: Summary of $\overline{\text{FDR}}_{\text{normal}}/\overline{\text{FDR}}_{\text{abnormal}}$ for different faults in the test dataset by using different models

Model No.	Fault No.					
	1	2	3	4	5	6
1	78.55/100.00	91.68/100.00	70.86/72.71	89.23/78.44	89.81/72.62	60.63/80.86
2	83.54/100.00	93.23/100.00	76.22/24.93	90.84/75.34	92.26/67.58	67.19/69.74
3	86.10/100.00	96.43/100.00	60.70/73.40	95.08/72.97	95.98/72.62	53.24/83.88
4	86.99/99.99	96.75/99.63	75.42/20.63	95.39/75.72	96.42/67.83	63.65/70.88
5	82.05/99.98	93.44/100.00	78.36/24.28	91.33/69.75	92.73/68.56	67.80/60.65
6	82.62/99.98	93.64/100.00	78.67/24.36	91.59/69.02	92.99/68.32	67.74/59.76
7	84.35/100.00	96.45/100.00	73.34/74.80	94.76/72.55	96.38/72.51	63.24/82.63
8	86.33/100.00	97.18/100.00	79.42/45.52	95.25/67.61	97.09/72.82	67.98/71.42
9	91.67/100.00	98.41/100.00	70.80/97.07	97.12/66.25	97.83/48.30	54.25/92.37
10	91.73/100.00	98.25/100.00	70.38/96.63	97.15/68.67	97.60/48.33	54.38/92.15

Table 3.4: Summary of $\overline{\text{FDR}}_{\text{normal}}/\overline{\text{FDR}}_{\text{abnormal}}$ for different faults for Model 5 (test accuracy)

nPC	**Fault No.**						$\overline{\textbf{FDR}}$ %
	1	**2**	**3**	**4**	**5**	**6**	
7	81.78/100.0	91.86/100	76.85/27.88	89.13/81.68	90.3/69.05	66.67/67.79	81.98
11	83.69/100.00	91.60/100.00	73.28/54.18	89.17/78.30	90.78/72.38	63.95/73.10	82.05
18	77.26/100.00	91.57/100.00	73.48/79.33	89.45/77.83	91.49/72.42	63.67/82.17	82.53

3.4.3 MPLS-AE with $J^{\mathbf{MPLS-AE}}$

In this part of the study, we employ the nonlinear MPLS-AE FD model trained using the traditional objective function $J^{MPLS-AE}$ that minimizes the weighted sum of the reconstruction error and the prediction error of the fermentation time (dummy variable). It should be noted that we have formulated a NN architecture (MPLS-AE) based on prediction of an indicative variable (refer Section 3.1) since the latter was found, from the comparisons above, to consistently enhance the $\overline{\text{FDR}}$. The hyper-parameters, such as number of layers, number of neurons in each layer, learning rate, weights etc. are selected using validation data. The weights of the network are obtained for each set of hyper-parameters with the training data and the validation data is then used to compare networks with different hyper-parameters to select the best set among them. The hyper-parameter search is implemented using the Keras-tuner in Python. To perform this search, a grid of hyper-parameters is defined, for example number of encoder layers = [1,2,3], number of neurons units for each of these layers ranging from 2 to 100, learning rate = [1e^{-1},2e^{-1},3e^{-1}, 1e^{-2}], range of weights in the objective function 0.01 to 1, etc. Then, the Keras-tuner trains the model using different combinations of these hyper-parameters values and the averaged validation accuracy is evaluated at every epoch. The models are trained with a few epochs in the start and the selected models with high validation accuracy are chosen to be trained for more epochs with a early stopping technique. The best run with highest validation accuracy and the combination of hyper-parameters for the run are used to evaluate the test accuracy. The loss function as a function of the number of latent variables of the hidden

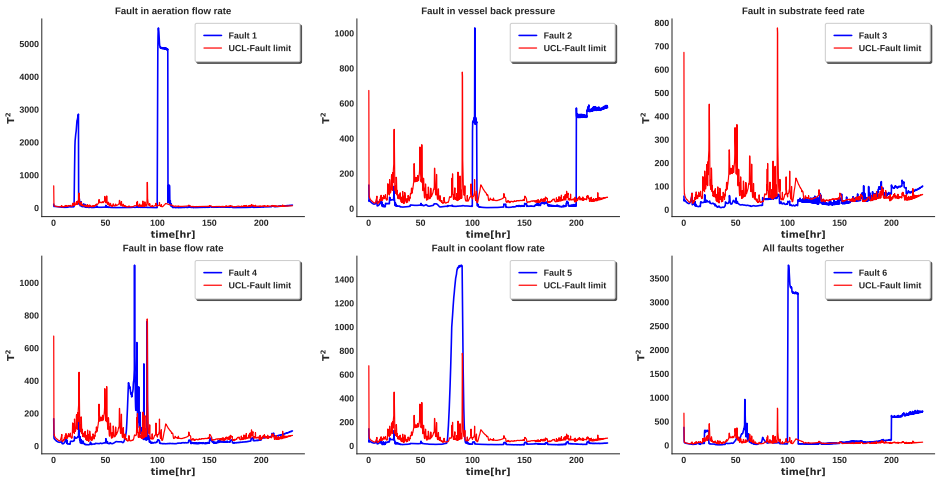


Figure 3.11: Plots of T_k^2 and dynamic control limits $T_{k,\alpha}^2$ for six different faulty batches of the test dataset based on $J^{MPCA-MPLS,FDR}$ as the objective function for the MPLS model ($\alpha = 0.01$).

layer of the MPLS-AE network is shown in Fig. (3.13). This figure indicates that the loss function does not significantly decreased beyond 15 latent variables (optimal number of units). Similar to the procedure described in section 3.4.1, the dynamic control limits $H_{k,\alpha}^2$ and $Q_{k,\alpha}$ can be obtained by estimating the distribution of $\mathbf{H}_{k,train}^2$ and $\mathbf{Q}_{k,train}$ at each time interval. A $\overline{\text{FDR}}$ test accuracy of 87.21% is obtained by the MPLS-AE model using $J^{MPLS-AE}$ as the loss function. Model No. 9 in Table 3.3 shows the $\overline{\text{FDR}}_{\text{normal}}/\overline{\text{FDR}}_{\text{abnormal}}$ for different faulty batches in the test dataset.

It should be noticed that the FDR for MPLS-AE is higher than all the other linear models. This can be attributed to the ability of deep learning models to extract non-linear features.

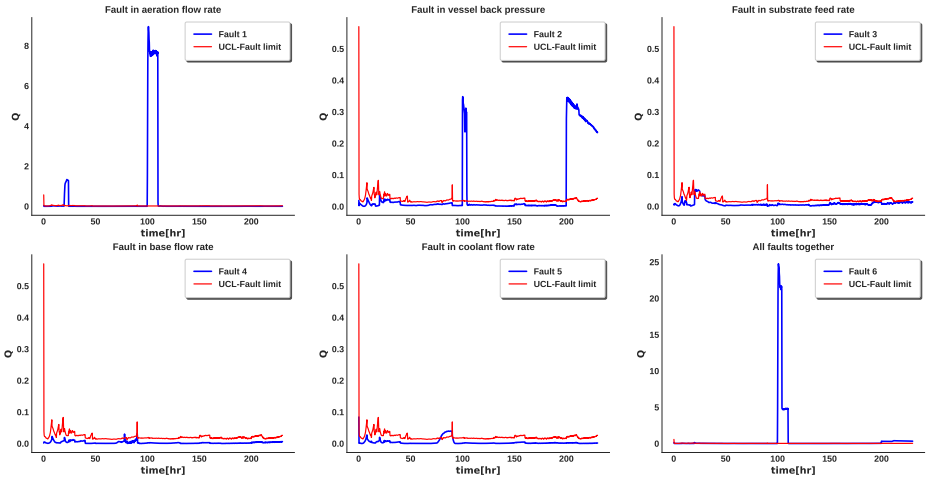


Figure 3.12: Plots of Q_k and dynamic control limits $Q_{k,\alpha}$ for six different faulty batches of the test dataset based on $J^{MPCA-MPLS,FDR}$ as the objective function for the MPLS model ($\alpha = 0.01$).

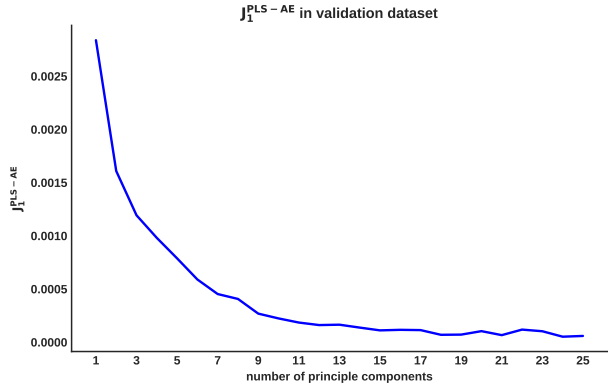


Figure 3.13: Loss of the validation dataset with MPLS-AE model using $J^{MPLS-AE}$ as the objective function to train the network.

3.4.4 MPLS-AE with $\mathbf{J^{MPLS-AE,FDR}}$

Finally, we evaluate the performance of MPLS-AE trained with the proposed loss function to maximize the $\overline{\text{FDR}}$ given by Eq. (3.21) in combination with dynamic control limits. In order to train MPLS-AE with the proposed objective function, it is essential to evaluate the $H^2_{k,\alpha}$ and $SPE_{k,\alpha}$ by estimating the distribution of $\mathbf{H}^2_k$ and $\mathbf{SPE}_k$ through KDE. Additionally, we need to evaluate the cumulative distribution function for a pre-specified significance level α (refer Eq. 3.17). A Genetic Algorithm (GA) was used for training the NN. GA is an evolutionary algorithm used to search for the best individual (here, each individual corresponds to weights of MPLS-AE) from a larger set of individuals (termed as 'population'). Each individual of the population is subsequently ranked on the basis of a metric, conventionally termed as 'fitness'. The purpose of this metric is to assess the goodness of fit of the predictions made by the model. We use the novel objective function in this work as a fitness function for training MPLS-AE. The ranked population is then operated upon by genetic operators (selection, mutation and crossover), resulting in a new population. This loop runs for a specified number of generations which must be input by the user at the start of the optimization, along with the number of individuals in a population (population size). After the specified number of generations, the best set of candidate solution are obtained. We emphasize here that the initial guess for the GA optimization is the solution obtained in the previous section, i.e. with the model trained with the traditional objective function. This is done to speed up the solution search. While this may restrict the optimization result to a neighborhood of the initial guess, it was sufficient to show improvements due to the use of the new FDR oriented objective even with the same model architecture (see Section 4.3).

Figs. (3.14)-(3.15) show the performance of MPLS-AE model with dynamic control limits and $J^{MPLS-AE,FDR}$ for the test dataset. A 88.64% (Model No. 10 in Table 3.2) $\overline{\text{FDR}}$ accuracy is obtained with the MPLS-AE model that use $J^{MPLS-AE,FDR}$ thus corrob-

orating the importance of using this objective to improve detection. In comparison with the Model No. 9, both MPLS-AE models have a close process monitoring performance in different types of faulty batches in the test dataset. However, in general, as shown in Table 3.2, the MPLS-AE trained with the novel objective function $J^{MPLS-AE,FDR}$ results in a higher $\overline{\text{FDR}}$. In comparison with the best linear model i.e. the MPLS model with dynamic control limits (Model No. 8), the MPLS-AE model provides a significant improvement in most parts of the test dataset. For example, in normal samples of fault number 1 and 2 batches, the MPLS-AE achieves 5% and 1% higher $\overline{\text{FDR}}$ accuracy as compared to the linear MPLS model, respectively. MPLS-AE model is 54% more effective in detecting fault 3 than MPLS.

While Table 3.2 also shows that the average fault detection rate $\overline{FDR}$ of Model 10 is higher than the one obtained with Model 9, the detection rates of individual faults (abnormal samples) do not always improve with Model 10 as compared to Model 9. However, it should be noted that the network architecture used for Model 10 with the new objective is the same as that used for Model 9 i.e., the hyper-parameters of the network are same for Models 9 and 10. While the hyper-parameters of Model 10 could be optimized to further minimize the loss and possibly improve detection rate of individual faults, the focus was on demonstrating that even with the same architecture the model with the FDR objective function (Model 10) results in higher average weighted test accuracy ($\overline{FDR}$) as compared to model 9 that uses the traditional training objective.

To further compare the models, the fault detection time for all models was calculated and is shown in Table 3.5. The fault detection time is estimated as the time-sample difference between the fault occurrence and fault detection (i.e. T_k^2 or Q_k and H_k^2 or SPE_k in case of the non-linear model, violate their corresponding thresholds). The detection time is calculated for 600 batches (testing data) for different faults and only the averages

are reported for brevity. It is observed that the average fault detection time in the MPLS-AE based models (Models 9 and 10) have average fault detection time closer to the mean of detection time obtained for all the other models (Models 1-8). Thus, the improved detection rates with the NN based models are not achieved at the expense of detection times.

Table 3.5: Summary of the average detection time for different models (averages are on 600 batches (testing data))

Model	**Average detection time**
1	8.07
2	12.43
3	7.20
4	11.12
5	13.45
6	13.47
7	7.31
8	9.14
9	10.52
10	10.24

3.5 Conclusion

In this work, a novel objective function is proposed for unsupervised batch process monitoring. Both linear and non-linear fault detection models were employed and compared in terms of fault detection capabilities. A Multiway Partial Least Squares Autoencoder (MPLS-AE) NN architecture was also designed to include an indicative (dummy) output variable, i.e. batch age, to further enhance the average fault detection rates. The hyper-parameters of FD models were selected by the minimization of two different objective functions: i- a traditionally used objective function involving reconstruction errors and ii- a novel objective function that is explicitly dependent on the fault detection rate. We demonstrated that the use of the proposed objective function provides a significant improvement

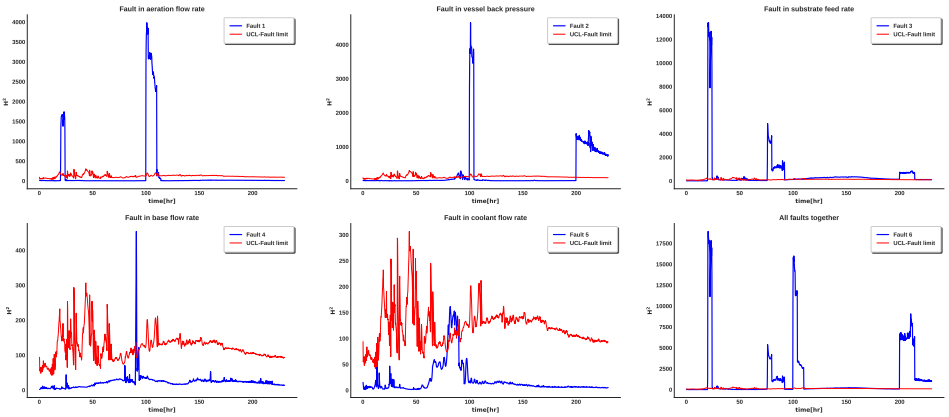


Figure 3.14: Plots of H_k^2 and dynamic control limits $H_{k,\alpha}^2$ for six different faulty batches of the test dataset based on $J^{MPLS-AE,FDR}$ as the objective function to train MPLS-AE ($\alpha = 0.01$).

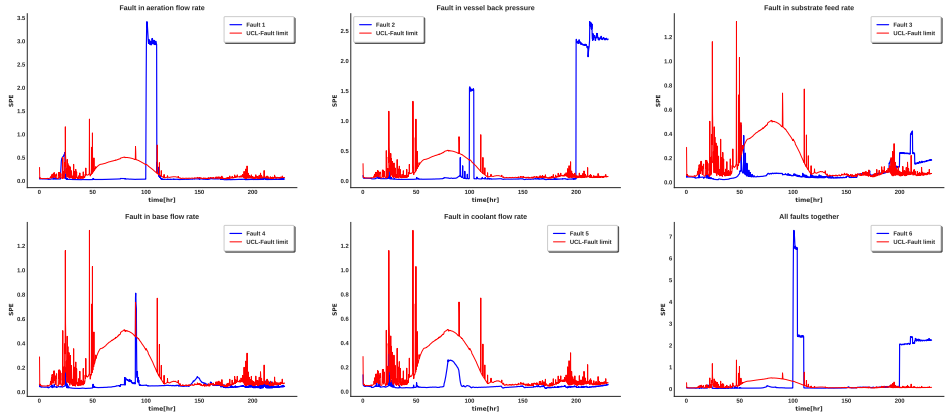


Figure 3.15: Plots of SPE_k^2 and dynamic control limits $SPE_{k,\alpha}^2$ for six different faulty batches of the test dataset based on $J^{MPLS-AE,FDR}$ as the objective function to train MPLS-AE ($\alpha = 0.01$).

in fault detection accuracy for both linear and nonlinear models but the improvement is larger with the nonlinear MPLS-AE. The key reason behind this improvement is that the metrics used for evaluating detection rates are functions of latent variables and these depend on the objective function used for training. When the objective function is explicitly related to the detection rate, the resulting latent variables are most informative about the faults. Both static and dynamic statistical control limits were applied with all methods. A case study of industrial penicillin batch dataset suggests that the use of dynamic control limits result in significant improvements in detection for all methods. Furthermore, it was also found that the proposed MPLS-AE which is trained by the minimization of the novel objective function along with dynamic control limits performs better than the best MPLS model. This confirmed the ability of MPLS-AE to tackle nonlinear system dynamics.

Chapter 4

Unsupervised Fault Detection of Pharmaceutical Processes Using Long Short-Term Memory Autoencoders

Overview[1]

In this chapter unsupervised multi-layer Long Short-Term Memory Autoencoders (LSTM-AEs) models are proposed for monitoring non-linear batch processes. The methodology is demonstrated for a simulation-based study of an industrial-scale penicillin process and for an industrial vaccine manufacturing process using production data. The LSTM-AE model was trained with two different loss functions: minimizing mean square error (MSE) between

[1]Adapted from Aghaee, Mohammad, et al. "Unsupervised Fault Detection of Pharmaceutical Processes Using Long Short-Term Memory Autoencoders." doi.org/10.1021/acs.iecr.3c00995

the input and reconstructed data, and maximizing the average fault detection rate ($\overline{FDR}$) in the training data set. Two algorithms are also proposed for obtaining contribution plots for the diagnosis of faults. For the industrial study where the faults are not known a priori, the contribution plots are found to be a valuable tool for identifying possible sources of faults. The proposed unsupervised algorithms exhibit clear improvement in accuracy over linear methods or nonlinear techniques that do not explicitly account for dynamic behaviour.

4.1 Introduction

Due to strict regulations, pharmaceutical manufacturers face significant challenges for achieving high productivity while maintaining normal process states. Variability in raw materials and other operational disturbances in the upstream and downstream processing can cause the process to deviate from normal operating conditions. It is important to note that such deviations, in conjunction with the requirement to maintain process inputs within bounds often lead to reduced productivity.

Pharmaceutical processes are mostly operated in batch or fed-batch modes to reduce contamination risks. Nonlinear behaviour is a salient characteristic of batch processes since the system evolves through a wide range of operating conditions as compared to continuous processes where the system is generally operated in the neighborhood of a specific steady state. In current industrial practice, batch processes are commonly monitored and controlled using statistical process control (SPC) techniques [MacGregor and Kourti, 1995, Martin and Morris, 1996, Ge and Song, 2012].

When a mechanistic nonlinear model of the process is available, faults can be identified and detected by utilizing probability based techniques [Du et al., 2015]. In the absence of nonlinear mechanistic models, machine learning and deep learning techniques can be used for monitoring of nonlinear systems.

There have been numerous studies on the use of deep learning models for process monitoring of nonlinear processes. A variety of deep learning architectures have been employed, including convolutional neural networks (CNN) [Chadha et al., 2019], recurrent neural networks (RNN) [Zhang et al., 2019a, Ren and Ni, 2020], and autoencoder neural networks (AEs) [Cheng et al., 2019, Park et al., 2019, Yin et al., 2020].

Fault detection models are generally grouped in two categories according to the measured variables that are used for model calibration. Models that are calibrated based on both inputs and faults are considered as supervised whereas models that only use input variables for calibration are considered unsupervised. In pharmaceutical processes, such as the ones considered in the current work, the faults are often unknown a priori and thus unsupervised monitoring methods are the only option for monitoring them.

AEs are a type of unsupervised learning neural network designed for analyzing nonlinear processes that learn by reconstructing the original inputs as accurately as possible [Gao et al., 2020, Le et al., 2015]. These AEs architecture are considered as unsupervised since they do not use any information on the faults and they are solely calibrated based on the inputs. The AE neural network is composed of two serially connected networks referred to as encoder and decoder respectively. The role of the encoder is to compress the input space into a lower dimensional space known as latent (or hidden) variables. Then, the decoder's function is to reconstruct the input space from the latent variables. Thus AEs can be conceptualized as a nonlinear principal component analysis (PCA) decomposition following their ability to compress the larger input space into a smaller latent variables' space [Agarwal et al., 2022, Yu and Zhao, 2019].

Two types of multiway partial least square autoencoders (MPLS-AE) for batch processes were recently reported by Agarwal et al. [Agarwal et al., 2022], one trained to minimize the mean square error (MSE) between the input data and the reconstructed data, and the other trained to maximize the average fault detection rate ($\overline{FDR}$). These MPLS-AE were applied to the Industrial Penicillin Simulator (IndPenSim) case study [Agarwal et al.,

2022]. These MPLS-AE models provided significantly higher detection rates as compared to linear MPCA and MPLS models for the IndPenSim case study. Furthermore, the results demonstrated that models which are trained by maximising ($\overline{FDR}$) provide a higher level of process monitoring efficacy as compared to models that were calibrated to minimize the reconstruction error. It was further shown in the study of Agarwal et al., that dynamic (varying) upper control limits are significantly more effective at monitoring processes than fixed upper control limits.

In an additional study a batch-wise LSTM-AE model was proposed for monitoring nonlinear batch processes [Ren and Ni, 2020] that captures both between- and within-batch dynamic features simultaneously in nonlinear batch processes by using sliding through batch-wise input sequences. In that study, dynamic upper control limits of process monitoring metrics, H^2 and SPE, were used to monitor the Penicillin Simulator (PenSim) benchmark. A set of samples collected at sample time k from different batches is fed to the model to reconstruct the sample time k in the b^{th} batch. In other words, the goal was to capture the dynamic correlation between sample time k of the current batch b and the same sample time in a few prior batches. Thus, in that study the "Long Short Term Memory (LSTM)" feature of the network was used to capture a dynamic correlation between different batches. It should be also noticed that the algorithm in [Ren and Ni, 2020] was trained based on the minimization of the reconstruction error. It will be shown in the current study that the use of detection rate oriented objective function to train the LSTM-AE results in significantly better results as compared to the use of the reconstruction error for model calibration.

While dynamic correlation between successive batches may be suitable for batch processes that experience slowly occurring external or seasonal disturbances, such correlation may not necessarily occur in many pharmaceutical processes since raw materials may change very frequently thus ruling out a dynamic correlation between successive batches. For example, one of the case studies used in the current work considers a fermentation

process for the production of antigens of the whooping cough vaccine where for each batch the raw materials may be different than for the previous batch. Hence, in the current study, an AE-LSTM structure that is based on a sequence of samples from the same batch different from the ones reported above had to be developed for fault detection.

One distinctive advantage of statistical process control monitoring techniques is their ability to generate contribution plots that inform what is the relative contribution of the input variables to a particular fault. This information is instrumental for making predictions of possible faults entering the system and for suggesting ways for mitigating the effects of these faults on the system. Thus, obtaining the contribution of process variables are essential after detecting the fault for restoring the system to normal operation [Vedam and Venkatasubramanian, 1999, Misra et al., 2002, Luo et al., 2017, Wu and Zhao, 2018]. To the knowledge of the authors, the generation of contribution plots based on AE-LSTM models has not been reported as yet.

All unsupervised process monitoring methods are based on the premise that the ranges of input values defining the region of normal operation are known a priori. Then, the model is trained using data corresponding to normal operation whereas faulty operation can be detected by assessing the statistical significance of the deviations of new measured inputs from inputs that correspond to normal operation [Nomikos and MacGregor, 1994, Dong and McAvoy, 1996]. If the process has been optimized, the normal operating region is assumed to be a neighborhood of the mean values of controlled and manipulated variables. However, in some situations, as shown for one of the case studies in the current work, the region of normal operation is not clearly defined a priori and the mean values of the manipulated and controlled variables do not necessarily correspond to normal operation. This can happen when the set-points in the controlled variables have not been optimized with respect to the productivity and the disturbances entering the process. Such situations have not been specifically studied in the literature and they must be resolved in order to train the unsupervised detection algorithm to detect deviations from normal operation.

It will be shown in one of the case studies that generating a region of normal operation requires a particular semi-supervised approach based on limited measurements of quality variables. In semi-supervised learning a limited amount of labeled data is used for training.

To address the limitations that were identified from previous studies as reviewed above, the present work proposes an unsupervised time-wise Long Short-Term Memory Autoencoder (LSTM-AE) network to identify and diagnose batch process faults.

The novel contributions related to the proposed approach are: i- development of an unsupervised LSTM-AE model for fault detection. ii- generation of contribution plots based on the hidden variables of the LSTM-AE model. iii- development of an approach to choose a region of normal operation for unsupervised fault detection. iv- validation of the algorithm for a simulator based case study and for an industrial case study.

The proposed LSTM-AE algorithm is composed of three elements: i- training of the LSTM-AE model using successive samples from normal batches within a finite time horizon and based on a fault detection oriented objective function (if the region of normal batch operation is not known a priori a semi-supervised approach based on partial knowledge of quality variables is used to defined such region), ii- Use of Gaussian kernel density estimation at each sampling time to determine upper control limits on hidden variables and detecting faults based on these limits, and iii- Generation of contribution plots for each process monitoring metric are generated based on the latent variables to identify the underlying cause(s) of faults.

In the present study, the effectiveness of the proposed methodology in detecting and diagnosing faults is evaluated on two case studies: the simulation-based IndPenSim case study [Goldrick et al., 2015, 2019] and ii- a real-world process of manufacturing *B. pertussis* vaccine at the Sanofi Toronto facility [Zavatti et al., 2021]. The simulation study was motivated by the fact that the faults are known although that knowledge is not available to the unsupervised detection algorithm. Knowledge of the faults permit to accurately assess the ability of the network to detect the faults thus testing the performance of the

proposed algorithm. On the other hand, for the second industrial case study, the faults are not known and thus the performance of the detection algorithm can only be assessed by validation. An additional challenge in the second case study is that the region of normal operation was not available and it has to be determined before the detection algorithm could be developed.

4.2 Preliminaries

4.2.1 Long Short-Term Memory (LSTM)

LSTM is a particular type of recurrent neural network (RNN) that describes long- and short- term effects of past events [Hochreiter and Schmidhuber, 1997] on current events. RNNs in general and LSTM in particular are designed to resolve the gradient vanishing and long-term memory loss problems that affect other types of neural network architectures. Besides the hidden state, a LSTM unit has a cell state to remember past values. A LSTM unit also contains three gates referred to as the input, forget and output gates to regulate the flow of information of accepted values from the current input, past cell state and the current cell state respectively. Each component of the LSTM unit is updated according to the following equations:

$$\begin{cases} i_t & = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t & = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t & = tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t & = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t & = f_t * c_{t-1} + i_t * g_t \\ h_t & = o_t * tanh(c_t) \end{cases} \tag{4.1}$$

where h_t , c_t are the hidden state and cell state at time t respectively, i_t, f_t and o_t are the input, forget and output gate, g_t is the cell gate, σ is the sigmoid activation function, * is the Hadamard product.

4.2.2 Autoencoder (AE)

As shown in Figure 4.1, AEs are traditionally composed of three parts: encoders, decoders and embeddings. An AE is trained in an unsupervised manner to extract patterns from data and reduce its dimensionality. In the encoder, the input data is compressed onto a reduced latent space, and the decoder reconstructs back the inputs using the outputs of the hidden layer. Consider x_k as the input vector for sample time k to an AE, then for a single hidden layer to convert the input variables to the latent space elements $h \in \mathbb{R}^{n_h}$ (latent variables) for time sample k (x_k) the encoder performs the following operation:

$$\mathbf{h}_k = f_e(\mathbf{W}_e \mathbf{x}_k + \mathbf{b}_e) \tag{4.2}$$

where f_e, $\mathbf{W}_e \in \mathbb{R}^{n_h \times f}$ and $\mathbf{b}_e \in \mathbb{R}^{n_h}$ are the nonlinear activation function, the weighting matrix, and the bias vector of the encoder neural network. Then, the decoder reconstructs the input variables from the hidden variables h_k as per the following operation:

$$\hat{\mathbf{x}}_k = f_d(\mathbf{W}_d \mathbf{h}_k + \mathbf{b}_d) \tag{4.3}$$

where f_d, $\mathbf{W}_d \in \mathbb{R}^{f \times n_h}$ and $\mathbf{b}_d \in \mathbb{R}^{n_h}$ are the nonlinear activation function, the weighting matrix, and the bias vector of the decoder neural network respectively. To train the LSTM-AE model, the traditional approach is to minimize the reconstruction error, i.e. the mean square error (MSE) between the input and reconstructed data, as described below:

$$\theta = \arg\min_{\theta} \sum_{k=1}^{n} ||\hat{x}_k - x_k|| \tag{4.4}$$

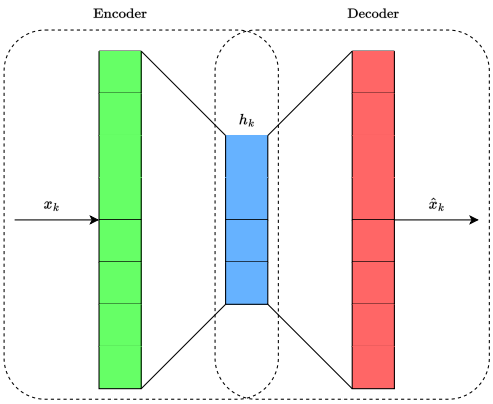


Figure 4.1: Schematic of an autoencoder (AE)

where n is the number of samples and $\theta = \{W_e, W_d, b_e, b_d\}$ is the set of AE's parameters.

4.2.3 Batch synchronization - Indicator Variable (IV)

Due to process variability, the batches may have different duration and they may progress at different speed in terms of cell growth or amount of secreted product. The need to compare batches of different duration that progress at different rates motivates the introduction an 'indicator variable' that is informative about the degree of completion or progress of the fermentation process. For example, an indicator variable that is defined as equal to the sampling time divided by the overall duration of the batch will show the "maturity" or "percent completion" of the batch. Then, the progress of other process variables are assessed in the model relative to the progress of the indicator variable. Variables other than normalized sampling times may also serve as indicator variables. Since the indicator variable describe a degree of completion of the process it must be monotonic, e.g $iv(tk) > iv(tk-1)$ (for a monotonic increase pattern of the indicator variable iv). Different indicator variables may be defined for different time intervals along the batch. For example, during fed-batch operation, the volume of the fermenter can serve as indicator variable since it increases monotonically thus describing the progress of the fermentation process towards

completion. For that purpose, the process can be divided into phases (time segments) to identify an indicator variable that is most suitable for each phase where such suitability can be decided on the basis of model validation. The use of indicator variables has been reported for many industrial or theoretical researches such as an industrial semi-batch emulsion polymerization reactor [Kourti et al., 1996] or a fed-batch penicillin production benchmark process simulator [Ündey et al., 2004]. It is also worth mentioning that for the purpose of online detection of a new batch and since the exact duration is not known in the priori, the indicator variable is based on the average duration of all the batches in the training data set.

4.3 Proposed Methodology

4.3.1 Long Short-Term Memory Autoencoder (LSTM-AE)

The inputs to the LSTM-AE network are a time-wise moving window of samples at sample time x_k. The network encodes these inputs through multiple LSTM layers into a lower dimensional latent space and subsequently, it reconstructs back the inputs through multiple layers of the decoder part of the LSTM-AE. As shown in Figure 4.2, we denote the sample of b^{th} batch at time k as $x^b(k) \in \mathbb{R}^J$. The input sequence into the proposed LSTM-AE and to be reconstructed back by the decoder for sample $x^b(k)$ is denoted as $X_k^b = \{x^b(k-w+1), ..., x^b(k)\} \in \mathbb{R}^{w \times J}$, in which w is the length of time-wise moving window. For the initial $w-1$ time intervals, the input sequence is initialized to $X_k^b = \{x^b(1), ..., x^b(k)\} \in \mathbb{R}^{k \times J}$. The LSTM unit of each layer operates on the input sequence X_k^b or the hidden state sequence from the last layer and calculates the following hidden state according to Eq. 4.1. After forward-propagation of the input sequence layer by layer, the deepest hidden state, $h^{deepest}(k)$, shown by the blue colored unit in the Figure 4.2, is fed as input to the decoder.

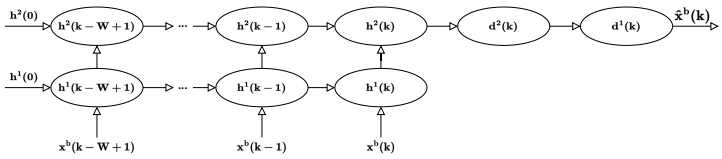


Figure 4.2: A simple illustration of the LSTM-AE network structure.

The decoder is trained to regenerate the sample x_k^b, of the input sequence X_k^b as follows:

$$\hat{x}^b(k) = f_d(h^{deepest}(k)) \tag{4.5}$$

where $\hat{x}^b(k)$ and f_d are the reconstructed sample time k in b^{th} batch and the multi LSTM layers function of the decoder network respectively. The parameter set of the LSTM-AE network, θ, can be jointly estimated by solving the following optimization problem:

$$\theta = \arg\min_{\theta} \sum_{b} \sum_{k} ||\hat{x}^b(k) - x^b(k)|| \tag{4.6}$$

Detection rate oriented loss function used for LSTM training

Since our primary objective is to maximize the $\overline{FDR}$, we propose an alternative objective function for model training to the reconstruction error loss function that is traditionally used for network training. However, for the purpose of fault detection, the most relevant objective function for training the LSTM-AE model is to maximize the number of correctly detected samples in the training data set. Moreover, due to the fact that all samples used to train an unsupervised learning model correspond to normal operation, the calculated process monitoring metrics should be below their associated upper control limit at each sampling time. Accordingly, to train the weights of LSTM-AE model, we are considering a loss function which is explicitly tailored to the $\overline{FDR}$ objective by enforcing the upper

control limits as follows:

$$\theta = \arg\max_{\theta} \sum_{b} \sum_{k} \left[H^2_{b,k} \leq H^2_{k,\alpha}\right] \; and \; \left[SPE_{b,k} \leq SPE_{k,\alpha}\right] \tag{4.7}$$

where $H^2_{b,k}$ and $SPE_{b,k}$ are the process monitoring metrics at sample time k in the b^{th} batch of the training data set, and $H^2_{k,\alpha}$ and $SPE_{k,\alpha}$ are their associated upper control limits with a significance level of α that can be calculated from Eq. 4.8. In Eq. 4.8 $F(M_k)$ is the probability density function, $G(x_i;\sigma)$ is the Gaussian kernel and b_w represents a bandwidth parameter, which is selected from the Silverman algorithm [Silverman, 2018]. The loss function (Eq. 4.7) trains the LSTM-AE model to detect the normal samples by enforcing the process monitoring metrics (H^2 and SPE) which are a function of the hidden variables h and input reconstruction error.

$$\begin{cases} F(M_k) = \sum_{i=1}^{b} G\left(\frac{M_k - M_i}{b_w}\right); M \in \{H^2, SPE\} \\ G(x_i;\sigma) \propto exp\left(-\frac{{x_i}^2}{2\sigma^2}\right) \\ 1 - \alpha = \int_{\min \mathbf{M}_{k,train}}^{M_{k,\alpha}} F(M_k)\, dM_k \end{cases} \tag{4.8}$$

4.3.2 Average fault detection rate

To quantify the performance of the algorithm, the average fault detection rate ($\overline{FDR}$) can be evaluated based on the average detection rates of normal ($\overline{FDR}_n$) and abnormal samples ($\overline{FDR}_a$) as follows:

$$\begin{cases} \overline{FDR} & = \frac{\overline{FDR}_n + \overline{FDR}_a}{(n_n + n_a)B} \\ \overline{FDR}_n & = \sum_{k=1}^{n_n} \sum_{b=1}^{B} \left[H^2_{k,b} \leq H^2_{k,\alpha}\right] \; and \; \left[SPE_{k,b} \leq SPE_{k,\alpha}\right] \\ \overline{FDR}_a & = \sum_{k=1}^{n_a} \sum_{b=1}^{B} \left[H^2_{k,b} > H^2_{k,\alpha}\right] \; or \; \left[SPE_{k,b} > SPE_{k,\alpha}\right] \end{cases} \tag{4.9}$$

where n_n, n_a and B are the number of normal samples, the number of abnormal samples and the total number of normal batches in a particular data set, respectively. A sample is detected correctly as in normal operation if both its corresponding $H^2_{k,b}$ and $SPE_{k,b}$ values are below their upper control limits. Otherwise, if the sample corresponds to normal operation but its $H^2_{k,b}$ and $SPE_{k,b}$ exceeds one or both of the limits, normal operation will be falsely detected as abnormal. In contrast, abnormal samples will correctly be detected as abnormal if either $H^2_{k,b}$ or $SPE_{k,b}$ or both exceed their control limits. Conversely, abnormal samples will be falsely detected as normal if their $H^2_{k,b}$ and $SPE_{k,b}$ do not exceed the upper control limits. Finally, the overall $\overline{FDR}$ is obtained from the weighted sum of $\overline{FDR}_n$ and $\overline{FDR}_a$. It should be noticed that $\overline{FDR}$ is a quantity that can be calculated accurately only when we know with certainty that a fault has occurred. Certainty about the fault is only possible in simulation studies where the occurrence of the fault is known but it is not possible in actual industrial problems when the occurrence of a fault is not known. For the latter situation an alternative quantity to $\overline{FDR}$ is proposed to quantify the performance of the fault detection algorithm.

4.3.3 A metrics based on an average of process monitoring metrics along the batch

As details about the faults such as their type, their time of occurrence, and their magnitude are often not available must be determined (measurable) in real-world processes, the $\overline{FDR}$ cannot be accurately quantified to assess the performance of the algorithm. For example, in our second case study, the faults are not available and the decision whether a situation is normal or abnormal is based on measurements of the final batch productivity. Since product is continuously accumulating in the batch reactor, the final productivity is related to the evolution of the batch from beginning to end rather than on events at a particular time point along the batch. In such case, to assess the occurrence of a fault and the effectiveness of the process monitoring model, it is not relevant to look at violation of

upper control limits at a particular time point. Instead, we propose an alternative measure that is equal to the average distance between the process monitoring metrics and their upper control limits over the entire batch duration. This measure permits us to assess both the occurrences of faults along a batch as well as the process productivity. The process will be identified as a low productive process if D_b violates its threshold $D_b^{threshold}$. This metric is formulated as follows:

$$\begin{cases} D_b = \sum_k \left[\frac{H_{k,b}^2}{H_{k,\alpha}^2} + \frac{SPE_{k,b}}{SPE_{k,\alpha}}\right] \; if \; H_{k,b}^2 > H_{k,\alpha}^2 \; or \; SPE_{k,b} > SPE_{k,\alpha} \\ D_b > D_b^{threshold} : \text{faulty batch/low productivity} \\ D_b \leq D_b^{threshold} : \text{normal batch/high productivity} \end{cases} \tag{4.10}$$

where D_b refers to the sum of all the ratios of the process monitoring metrics and their upper control limits where the index k refers to a sample along the batch for which the metrics exceed their upper control limits. In principle, since D_b is the average of ratios between each metric to its upper bound, $D_b^{threshold}$ should be one which corresponds to each metric being equal to its control limit. However, in practice, a value slightly larger than one was used to account for the noise in the measurement of productivity and for modelling errors. The actual value of the threshold $D_b^{threshold}$ was found based on a validation procedure.

4.3.4 Fault Diagnosis - Contributions of individual variables to the detection metrics

Statistical control charts provide an alarm when an abnormal situation is encountered, but do not provide any further information about the cause of the abnormality [Miller et al., 1998]. Therefore, it has been proposed in the literature to calculate the contributions of each measured variable to the fault detection statistics. Then, it is possible to extract information about the faults from these contributions that could serve to hypothesize about different sources of faults. While contributions of measured variables to the metrics used

for detection have been reported for multivariate statistical methods such as PCA, such plots have not been reported for LSTM-AE unsupervised learned models such as the one used in the current study. The calculations of the contributions of the measured variables to each of the proposed metrics $H^2_{k,b}$ or $SPE_{k,b}$ used in the current work are detailed below.

Contributions of variables to H^2

In a Principal Component Analysis (PCA) model, the contribution of each variable to the Hotteling's T^2 can be calculated based on loading, score vectors, and eigenvalues of the input data covariance matrix [Mnassri et al., 2009]. In contrast to PCA models, neural network models cannot be mathematically expressed as a superposition of products of scores by loading vectors. Thus, classical approaches cannot be used to determine the contribution of H^2 to each variable. It is noteworthy that H^2 is somehow equivalent to the T^2 in a linear PCA model. For AE models, the variables in the hidden layer of the network can be interpreted as the latent variables of the model since the hidden layer is a lower dimensional representation of the high dimensional input layer of the network. In the current study a numerical algorithm 1 is proposed for finding the contribution to H^2 from each variable. Where n_k, n_u, n_b, n_f and B are the number of samples in a batch, number of

Algorithm 1 Finding variables' contributions to H^2

1: Train the LSTM-AE network and find the hidden variables of the training data set, $h_{train} \in \mathbb{R}^{n_k,n_u,n_b}$,
2: Obtain the average of h_{train} from the batch index, $\bar{h}_{train} \in \mathbb{R}^{n_k,n_u}$,
3: **for** batch $b \in \{1, ..., B\}$ **do**
4: $\quad \Delta h_b = h_b - \bar{h}_{train}$,
5: $\quad$ Input Δh_b to the decoder network of the trained AE model and calculate the decoder output ΔX_b (see Figure 4.3); $\Delta X_b \in \mathbb{R}^{n_k,n_f}$,
6: **end for**
7: Calculate $\Delta X = \{\Delta X_b | \forall b \in \{1, ..., B\}\}$; $\Delta X \in \mathbb{R}^{n_k,n_f,B}$,
8: Obtain the average of ΔX from the batch index, $\overline{\Delta X} \in \mathbb{R}^{n_k,n_f}$,
9: Generate a bar chart from $|\overline{\Delta X}|$ as the contributions of each variable to H^2.

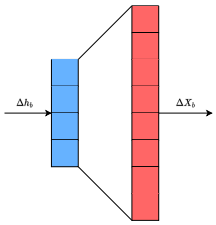


Figure 4.3: The decoder network of the trained LSTM-AE for computing H^2 contributions.

units in the deepest layer of the AE model, number of training batches, number of variables and number of testing batches, respectively. The $|\overline{\Delta X}|$ indicates the H^2-contribution of process variables over time. Moreover, the H^2-contribution for the entire duration of the process can be determined by averaging $|\overline{\Delta X}|$ from the index of time.

Contributions of individual variables to SPE

The algorithm 2 illustrates how to determine the contribution of each variable to SPE. These contributions are directly related to the reconstruction error of the LSTM-AE model, i.e. the differences between the inputs and the reconstructed inputs. The $|\overline{E}|$ indicates the

Algorithm 2 Finding variables' contributions to SPE

1: **for** batch $b \in \{1, ..., B\}$ **do**
2: Input $X_b \in \mathbb{R}^{n_k, n_f}$ to the trained AE model,
3: Save $E_b = \hat{X}_b - X_b$; $E_b \in \mathbb{R}^{n_k, n_f}$,
4: **end for**
5: $E = \{E_b | \forall b \in \{1, ..., B\}\}$; $E \in \mathbb{R}^{n_k, n_f, B}$,
6: Obtain the average of E from the batch index, $\overline{E} \in \mathbb{R}^{n_k, n_f}$,
7: Generate a bar chart from $|\overline{E}|$ as the contributions of each variable to SPE.

SPE-contribution of process variables over time. Moreover, the SPE-contribution over the entire duration of the batch can be determined by averaging $|\overline{E}|$ from the index of time.

4.3.5 Case studies

Industrial Penicillin Simulator (IndPenSim)

This case study is conducted with the IndPenSim code [Goldrick et al., 2019] that simulates the batch fermentation of penicillinwww.industrialpenicillinsimulation.com. The simulator is used to generate a data set of 700 batches in total. The total fermentation time of each batch is 230 hours (1150 time samples per batch) and includes 25 measured input variables. There are six different types of faults considered in this case study. Different types of faults may occur at different time intervals with varying magnitudes, as shown in Table 4.1. A complete data set is comprised of 100 batches of normal operation, as well as 100 batches for each type of fault. Only batches with normal operation are used for model calibration corresponding following the proposed unsupervised learning approach. These normal operation batches are divided into training and validation sets. An LSTM-AE model is trained using 80 normal batches, and the network hyperparameters are determined using the remaining 20 batches. As a test data set, we use the remaining 600 batches. To train the LSTM-AE model, the training set samples are normalized between 0 and 1 using min-max normalization. A key goal in this case study is to test the accuracy of the methodology which can be checked at each time interval during the batch since the faults are available from the simulator. Although the faults are generally not available in actual industrial problems such as the second case study, the results are used to test the actual false alarm rate FAR at each time interval.

Bordetella pertussis vaccine manufacturing process at Sanofi

We consider an industrial case study involving *B. pertussis* fermentation for the production of antigens that constitute the cellular whooping cough vaccine. Figure 4.4 illustrates a schematic of the Sanofi process. A batch fermentation consists of two parallel fermenter trains (train 1: $\{F_1, F_2, F_3\}$ and train 2: $\{F_4, F_5, F_6\}$). The $20L$ fermenters (F_1 and F_4)

Table 4.1: Description of different types of fault, [Agarwal et al., 2022].

fault in	magnitude	time (hr)
aeration flow rate (L/hr)	20, 20	[20, 24], [100, 110]
vessel back pressure (bar)	2, 2	[100, 104], [200, 230]
substrate feed rate (L/hr)	2, 20, 20	[20, 30], [76, 92], [200, 214]
base flow rate (L/hr)	5, 10	[80, 84], [140, 160]
coolant flow rate (L/hr)	2	[70, 90]
all of the above faults		

which also known as seed fermenters are grown for approximately 22 to 25 hours. At the end of the seed fermentation, the material of F_1 (or F_4) will be moved to the $200L$ fermenter F_2 (or F_5) which is referred to as intermediate fermenter. Intermediate fermentation is also complete within approximately 22 to 25 hours. After completion, the culture of the intermediate fermenters F_2 and F_5 are transferred to $2000L$ production fermenters F_3 and F_6 respectively. The production fermenters have the longest duration compared to the smaller fermenters taking 56 hours to completion. The production fermentation is divided into two stages: i. batch and ii. fed-batch. Component Pertussis Supplement is added to the production fermenters when the initial supply of nutrients is exhausted, as indicated by a noticeable decrease in oxygen consumption, which is also indicated, by a decrease in airflow and agitation. Component Pertussis Supplement is fed into the fermenter for the remaining duration of the fermentation process. Finally, the cultures of four production fermenters (two F_3s and two F_6s) are combined together and final protein concentration is measured in the last step by a Kjeldahl method which provides a protein (antigen component of the vaccine) level indicator. In this work, we focus on the unsupervised process monitoring of larger F_3 and F_6 fermenters. The historical data of upstream and down-stream data that are used in this work are collected by Sanofi company at Toronto. The database includes 65 sets of four fermenters $\{F_3^1, F_6^1, F_3^2, F_6^2\}$ collected for the last three years. In the Sanofi fermentation process, not only F_3 (or F_6) fermenters may be run for different time duration as compared to other sets of F_3 and F_6 runs, but also the

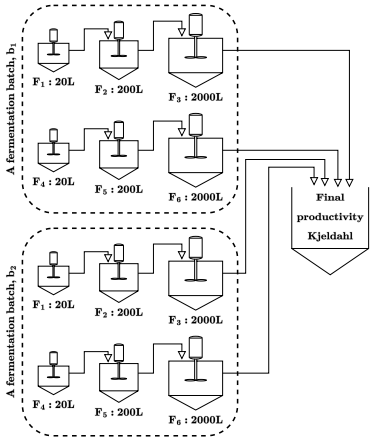


Figure 4.4: A schematic of *B. pertussis* fermentation process at Sanofi, Toronto.

duration of a F_3 fermenter can be different from the F_6 fermenter that was run in parallel. Like most of real-world data, there are some missing data in the data set. The missing data are completed by using linear interpolation. The occurrence of uneven batch lengths (durations) is tackled by using indicator variables approach as explained in section 4.2.3. Since the IndPenSim is conducted with a simulator, all details about the faults are known at each time interval during the batch, although they are not used in model development and training. As a result of these details, we are able to evaluate FDR and distinguish between normal batches and abnormal batches. However, in the Sanofi industrial case study, there is no information about the faults both in terms of their time of occurrence and their nature or cause. Thus, in this case study we are unable to evaluate $\overline{FDR}$ to quantify the performance of the detection algorithm. Furthermore, in this case study in contrast with the previous one, because the faults are not known the batches that should be assumed normal and used for training cannot be clearly identified. Therefore, to establish a region of normal operation and identify normal batches to be used for model training we had to rely on a final productivity measurement (Kjeldahl test) of a that is performed after the purification of the products of two F_3/F_6 fermenters, i.e. the total product from 4 fermenters, as the metric of normal/abnormal batches. It should be emphasized that

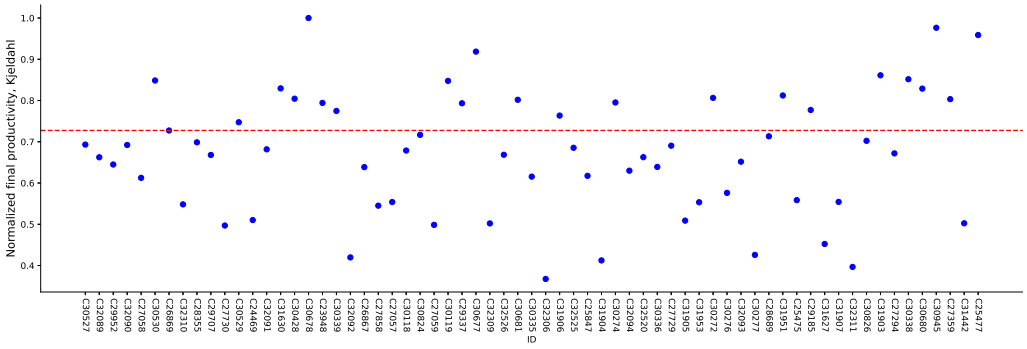


Figure 4.5: Data points of ID versus normalized Kjeldahl values at Sanofi, Toronto.

the time averages of the measured variables for all the batches correspond to a relatively low productivity based on communication with Sanofi personnel. Thus, the means of the process variables cannot be used to define normal operation as done for processes that are considered to be in control. According to the experts of Sanofi, a normalized Kjeldahl (productivity) value above 0.73 should be considered normal operation for two F_3/F_6 pairs of fermenters and below that value should be considered faulty operation. Figure 4.5 shows the IDs of 65 pairs of batches versus their corresponding Kjeldahl values. Due to the impossibility to calculate average $\overline{FDR}$ as in the first case study where the faults were known, to assess the effectiveness of the process monitoring model in this case study, Eq. 4.10 is used.

4.4 Results and Discussions

Two LSTM-AE models were trained to monitor the processes for each of the two case studies based on two different optimization problems: i- minimization of the MSE between the input data and the reconstructed data, ii- maximization of the $\overline{FDR}$ in the training data set. Additionally, contribution plots have been provided for diagnosing the primary root(s) of the faults. It should be noticed that the overall data set is divided into 3 subsets: i- a training data set that is used to train the models and to obtain the weights of the

network, ii- a validation data set that is used to obtain the optimal hyperparameters, and iii- a testing data set that is used to assess the effectiveness of the proposed methodology in monitoring the case studies. There are a number of hyperparameters in the model including the number of layers, the number of units in each layer, the learning rate, and the time horizon window (W). These parameters are tuned to obtain the extreme of the loss function on the validation data set. Python's Keras-tuner is used to implement the hyperparameter search. To perform this search, a grid of hyperparameters is defined, for example number of encoder layers = [1,2,3], number of units within layers = [2, 4, 8, 16], learning rate = [1e^{-1},2e^{-1},3e^{-1}, 1e^{-2}] and W = [1, 2, 4, 8, 16, 32, 64]. Some of these hyperparameters have been removed from the search due to overfitting issues (such as a large number of units) or insufficient memory for storing large variables (such as a large number of W).

4.4.1 Results in IndPenSim case study

Two LSTM-AE models of the same architecture are trained to monitor the IndPenSim case study with two different loss functions. Based on the validation procedure explained above, the models' hyper-parameters were as follows: layer unit numbers of 25, 16, 8, 8, 16, 25, learning rate of 0.01, and time horizon window of 4. The two models are used to calculate the $\overline{FDR}$ accuracy for the 6 faults that were considered. Then, the two LSTM-AE models are compared to the most efficient AE model presented in our previous paper and to a linear dynamic principal component analysis (DPCA) in Table 4.2. Based on the elbow method [Joshi and Nalwade, 2013], 20 principal components are selected for the DPCA model. It should be noticed that the AE model in our previous study was static, i.e. it only operated on data at a current time point and not along a time horizon as done in the current work. As shown in Table 4.2 there was a significant improvement in $\overline{FDR}$ for the test data set for all nonlinear AE models in comparison to linear DPCA models. Both LSTM-AE models outperform the MPLS-AE model which has been trained with the

Table 4.2: Comparison of $\overline{FDR}$% (training/validation/test) in different models.

No.	Model type	Loss function	$\overline{\text{FDR}}$%
1	DPCA	Classical loss function, Eq. 4.6	96.98/ 97.63/ 83.89
2	MPLS-AE*	Novel loss function, Eq. 4.7	98.37/ 98.84/ 88.64
3	LSTM-AE	Classical loss function, Eq. 4.6	98.45/ 98.77/ 91.08
4	LSTM-AE	Novel loss function, Eq. 4.7	98.87/ 98.89/ **93.86**

*Best developed model in [Agarwal et al., 2022].

novel loss function in our previous work, as shown in Table 4.2. This comparison indicates that by capturing the dynamics of the process the LSTM-AE models result in a significant improvement on the performance of monitoring models as compared to models that do not capture dynamic behaviour. Furthermore, the model trained to maximize the fault detection rate resulted in significant improvement in testing as compared to the model trained to minimize the reconstruction error.

Subsequently, based on the best LSTM-AE model, i.e. the model based on a novel loss function, contributions plots were generated with the Algorithms 1 and 2 based on the process monitoring metrics H^2 and SPE. Since this case study is based on a simulator and thus the occurrences of the faults are known, the contribution plots can be interpreted separately. Our goal was to verify whether the contribution plots are informative about the variables that are most correlated to each particular fault. Figure 4.6A illustrates the contributions of five most relevant variables obtained for batches with fault no. 1, which is a disturbance in the aeration flow rate. Since the IndPenSim simulator state-space model is available we could actually identify which variables are most affected by disturbances in aeration rate and verify whether the contributions show the importance of these variables. According to Eq. 4.11, following disturbances in aeration flow rates, significant contributions in off-gas CO_2 ($O_{2,out}$) and dissolved CO_2 ($CO_{2,d}$) are expected. Furthermore, the effect of changes in aeration rates on dissolved oxygen and off-gas mounts can also be readily predicted from physical understanding of the process. If the simulator would not be available, significant contributions for dissolved oxygen and off-gas flow rate

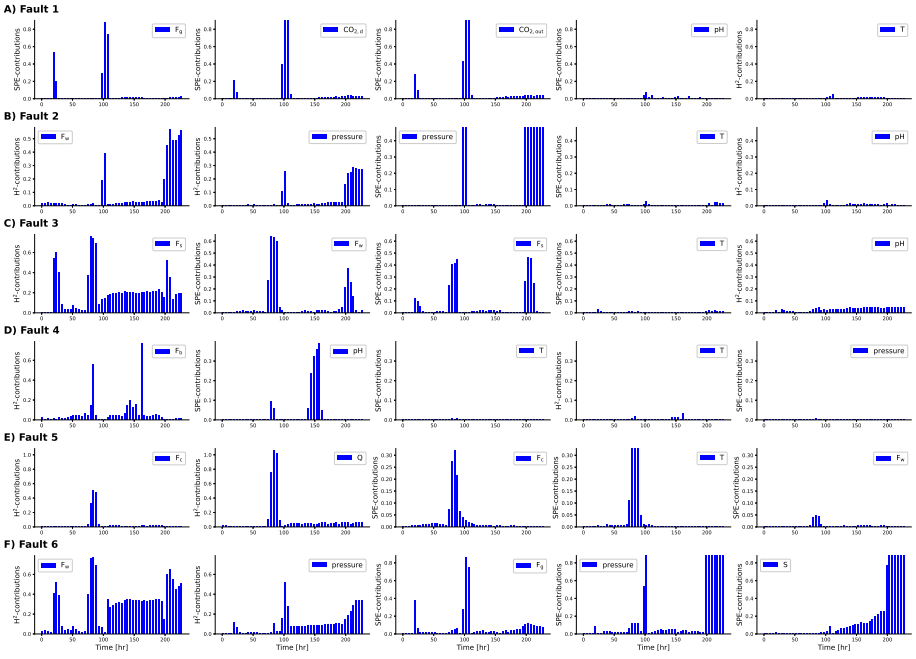


Figure 4.6: Contributions of variables in different faulty batches in IndPenSim case study.

would be still indicative of aeration rates changes a a possible source of this fault.

$$\begin{cases} \frac{dCO_{2,out}}{dt} & = \frac{22.4(Q_{g,in}CO_{2,in} - Q_{g,out}CO_{2,out} + CER)}{29V_g} \\ Q_{g,out} & = \frac{Q_{g,in}N_{2,in}}{1 - O_{2,out} - CO_{2,out}/100} \\ Q_{g,in} & = \frac{60 \times 1000 \times 32}{22.4} F_g \\ CER & = \frac{44}{22.4} F_g \left(\frac{CO_{2,out}N_{2,in}}{1 - O_{2,out} - CO_{2,out}} - CO_{2,in} \right) \end{cases} \tag{4.11}$$

Based on the contribution plots for fault 2 shown in Figure 4.6B, significant changes in the water flow rate (F_w) appear as the most important contribution related to this fault. The back pressure in the simulator is given by Eq. 4.12 indicating that the fermenter pressure is a function of volume which is a function of process input flow rates. Since all other input variables are controlled in closed loop, the input flow rate F_w is the only input variable that can be varied to maintain the pressure at its set-point thus explaining the

significance of its contribution.

$$\begin{cases} P = \frac{P_{bottom} - P_{top}}{log(\frac{P_{bottom}}{P_{top}})} \\ P_{bottom} = 1 + P_{back} + \rho_b h_b \\ P_{top} = 1 + P_{back} \\ h_b = \frac{V}{\pi R^2}(1 - \epsilon) \\ \frac{dV}{dt} = F_s + F_b + F_a + F_w + \frac{1000 F_{discharge}}{\rho_b} - F_{evp} + F_{paa} \end{cases} \tag{4.12}$$

Fault 3 corresponds to changes in substrate feed rate. According to the contribution plots for fault 3 shown in Figure 4.6C, significant changes have also been observed in the water flow rate (F_w). These changes can be explained by Eq. 4.13. The F_w stream can be varied in order to maintain the substrate concentration at its set-point in response to a disturbance in the substrate feed rate.

$$\begin{cases} \frac{dS}{dt} & = -Y_{S/X} r_e - Y_{S/X} r_b - m_s r_m - Y_{S/P} r_P + \frac{F_s c_s}{V} + \frac{F_{oil} c_{oil}}{V} - \frac{F_{in} S}{V} \\ F_{in} & = F_s + F_b + F_a + F_w - F_{evp} + F_{paa} \end{cases} \tag{4.13}$$

A significant change in pH and the base flow rate (F_b) has also been detected following the disturbance in the base flow rate (fault 4) as seen in the contribution plots for fault 4, Figure 4.6D. Base flow rate is responsible for maintaining pH at its set point, so a large change in pH is logically expected.

$$\begin{cases} Basic: \frac{dpH}{dt} = -\gamma_1(r_{b0} + r_{e1} + r_{d1} + r_{d4} - r_p + m_{pH} X) - \gamma_2 pH_{dis} - B - pH \\ Acidic: \frac{dpH}{dt} = \gamma_1(r_{b0} + r_{e1} + r_{d1} + r_{d4} + r_p + m_{pH} X) + \gamma_2 pH_{dis} - pH \\ pH_{dis} = F_s + F_{oil} + F_b + F_a + F_{discharge} + F_w \end{cases} \tag{4.14}$$

Additionally, as shown in Figure 4.6E for fault 5 that corresponds to disturbances in coolant flow rate, considerable changes have been observed in variables T, Q and F_c, which are

the temperature, heat exchanged and coolant flow rate respectively. Equations 4.15in the simulator model also support the significance of these contributions for fault 5:

$$\begin{cases} \frac{dT}{dt} = \frac{1000}{C_{pw}\rho_b V}\frac{dQ}{dt} \\ \frac{dQ}{dt} = \frac{1}{1000}F_s\rho_{feed}C_{ps}(T_f - T) + \frac{1}{1000}F_w\rho_w C_{pw}(T_w - T) \\ -\frac{1}{1000}F_{evp}\rho_b C_{pw} - \frac{1}{1000}F_{evp}\rho_w\Delta H_{evp} + Q_{exn} + W_s - Q_{con} \\ -\frac{\alpha_1}{1000}\frac{F_c^{\beta_T+1}(T-T_{c,in})}{\frac{F_c}{1000}+0.5\alpha_1(\frac{F_c}{1000})^{\beta_T}\rho_b C_{ps}} - \frac{\alpha_1}{1000}\frac{F_h^{\beta_T+1}(T-T_h)}{\frac{F_h}{1000}+0.5\alpha_1(\frac{F_h}{1000})^{\beta_T}\rho_b C_{ps}} \\ Q_{con} = U_{jacket}A_c(T - T_{air}) \end{cases} \tag{4.15}$$

As expected, fault 6 that consists of the combination of all previous faults as shown in Figure 4.6F, results in contributions from most of the process variables. In summary, we have shown that the contributions plots correctly identify the variables that are most contributing to each particular fault. Since a detailed dynamic simulator was available, which will not be the case in most industrial applications, we were able to verify our contributions by relating them to the model equations. This example also illustrated that even when a detailed model is not available, basic physical understanding of the process can be used to refer the variables' contributions back to the actual sources of the faults.

4.4.2 Results in Sanofi case study

The main challenges in this case study in contrast with the previous case study are: i- a simulator model is not available and thus the sources and occurrences of faults are not known and ii- the region of normal operation is not defined a priori and iii- the mean values of the measured variables cannot be used to define a nominal normal operating condition since the process is the process is not considered to be in control. In other words, because the process is still exhibiting large variability and it has not been optimized in terms of productivity, there are many batches that exhibit low productivity which cannot be considered as normal operation. Since the productivity is the priority for this process,

the LSTM-AE models has been trained in this case study using batches that have high productivity. The goal is then to identify the abnormal batches will correspond to low productivity. As illustrated in Figure 4.5, batches with normalized Kjeldahl values approximately greater than 0.73 are considered to be high productivity batches. This limit was defined based on what is considered as a minimal acceptable productivity for this process. A total of 22 high productivity batches were available based on this limit. Then, following a seven-fold cross-validation procedure, the high productivity batches that resulted in minimal reconstruction error or maximal fault detection rate for the validation data set with the LSTM-AE model were used for model validation. Since this procedure for selecting the normal batches relies on labelled data, i.e. final batch productivity, the approach for this case study should be referred as *semi-supervised*. The total number of normal batches selected based on this validation procedure was 18 and the corresponding batches are indicated by blue dots in Figure 4.7. The hyper-parameters of the models used in the validation procedure were as follows: layer unit numbers of 40, 16, 4, 4, 16, 40, learning rate of 0.01 and time horizon window of 4 samples (8 minutes). It should be emphasized that while labelled data was used for selecting normal batches no labelled data was used for abnormal batches that correspond to low productivity, i.e. normalized Kjeldahl values less than 0.73. Those abnormal batches are expected to be detected by the algorithm on the testing set.

Since the occurrence of the faults is not known for this process, to both monitor the process and to quantify the performance of the monitoring scheme we use the concept of average detection rate over the batch based on the distance D_b as described in Section 4.3.3. In this case, while we can detect faults at each time interval based on violations of the upper control limits, we cannot evaluate how accurate is the detection accuracy based on $\overline{FDR}$ since the occurrence of the fault is not known.

A summary of the results of average fault detection rate $\overline{FDR}$ and the average violations of process monitoring metrics (D_b) is provided in Table 4.3 and shown in Figure 4.7. The

results indicate that the LSTM-AE model based on the average fault detection rate is superior to the LSTM-AE trained with the reconstruction error. The LSTM-AE model, as a nonlinear model, is as expected significantly superior to the linear DPCA model. Table 4.3 indicates that there is a significant correlation between Kjeldahl's final productivity and the average violation of process monitoring metrics (D_b). As shown in Figure 4.7, the values of D_b in training and validation data sets (high-productivity batches) are slightly above 1.0 which means that in these batches process monitoring metrics do not violate their upper control limits significantly. Therefore, for evaluation of the method on the testing data set, the upper limit of D_b was considered and hyper-parameter that was chosen as the maximum D_b in the validation data set. Using this limit, 86.05% of low-productive batches are correctly identified as abnormal batches. Moreover, it should be emphasized that the average D_b, i.e. the average of values of the metric D_b over different time intervals along the batch, may violate the upper limit $D_b^{threshold}$ much before the end of the batch. Thus, as shown in Figure 4.10, the proposed method provides an opportunity to detect a low productivity batch before its completion and thus it opens future opportunities to implement earlier corrective measures. The fact that the LSTM-AE model based on the average detection rate results in large detection accuracy, i.e. 86.05% of abnormal batches are correctly detected, it confirms the impact of the fermentation step on the overall productivity of the process as compared to the influence of the downstream steps. It should be noticed that the Kjeldahl test is performed after the fermentation broth is purified though 16 purification steps. While the variability in Kjeldahl values was expected to be correlated to both the fermentation (upstream) and the purification (downstream) phases of the manufacturing process, the result presented here reveals that the fermentation phase has the most impact on final productivity. This result is very important since it indicates which part of the process is exposed to the most significant disturbances/faults and where the efforts should be focused for overall process improvement.

Contribution plots were subsequently generated for the LSTM-AE model that was trained based on the maximization of average fault detection rate. The variables that

Table 4.3: Results of the Sanofi case study.

No.	Model type	Loss function	$\overline{\text{FDR}}$%	D_b accuracy
1	DPCA	Eq. 4.6	73.88/ 75.06/ None	88.89/ 100.0/ 81.39
2	LSTM-AE	Eq. 4.6	90.01/ 98.89/ None	94.44/ 100.0/ 83.72
3	LSTM-AE	Eq. 4.7	91.12/ 99.11/ None	94.44/ 100.0/ 86.05

contributes the most to the detection metrics, shown in Figure 4.8, were investigated to identify possible source of faults that result in low-productivity batches. Consistent with a previous study conducted with supervised learning on data collected at a different time period [Agarwal, 2022], airflow, temperature (T), seal temperature (SealT) and pH (before the acid control kicks in) were found significant with respect to the detection metrics that were based on the final productivity of the process. Airflow changes between low productivity (abnormal) batches as compared to high productivity (normal) batches is important after 20 hours of operation. Since airflow is a manipulated variable to control the dissolved oxygen, changes in airflow imply that the low productivity batches require different aeration at a later time period than the high productivity ones. For instance, if the low productivity batches have a longer lag phase of growth, they will require less aeration at the beginning and more aeration later when growth is significant. A longer lag phase is perfectly consistent with a lower productivity batch since the cells do not secret during this phase. Slower growth has been identified as a cause of low productivity in bench scale bioreactor experiments conducted for this process [Zavatti, 2014]. Since the cells continuously secrete product (antigen) during the batch, a slower growth will result in a smaller amount of product being accumulated in the fermentation supernatant. A slower growth may also explain the initial changes in pH which show a significant contribution during the first period of the process. Slower growth will result in a smaller production of CO2 for the low productivity batches which result in lower levels of dissolved carbonic acid (pH=4.1), i.e. higher pH in low productivity batches. Other influential variable is the seal temperature (sealT), which has also been observed in previous research. Two temperatures are measured, one at the wall of the vessel and another (sealT) close to the impeller shaft.

While the temperature in the vessel is controlled in closed loop, the seal temperature is not controlled. Thus, in cases of temperature probe calibration problems, these will not be reflected in the vessel temperature since it is maintained at a constant set point by closed loop control. On the other hand, since the sealT is not controlled in closed loop it can serve as an efficient indicator of changes in the vessel temperature. The seal temperature is expected to be between the temperature inside of the bioreactor and the surrounding temperature, since the seal temperature is not controlled in closed loop, the variation in the seal temperature is informative about the fermentation. Productivity has been found by Sanofi in lab scale studies to be very sensitive to small changes in temperature of less that 1 degree C.

This study also identified two new important variables that were not identified in previous studies by Sanofi: pressure (P) and tank weight (Wt). Pressure is correlated with airflow, and thus changes in airflow, which is also found to deviate significantly for abnormal batches, result in corresponding changes in pressure. In addition, we found that the initial tank weight varied by almost 10% as compared to high productivity (normal) batches, which led us to identify tank weights as major contributors to final productivity. Further inspection of the process operational procedure reveals that at the beginning of the process and before inoculation the vessel is filled with growth media and the vessel is sterilized in situ with steam. It was also found that the sterilization times were variable which is expected to result in different evaporation rates thus explaining different tank volumes as shown by the contribution plots. Changes in liquid (water) volume are expected to result in changes in concentrations of main aminoacids and main media components. In a recent small bioreactor experiment for this process it was shown that higher concentrations of glutamate [Wellen and Thompson, 2010, Wang and Levin, 2009] that will occur for smaller liquid volumes resulted in longer lag phase due to glutamate induced inhibition.

It is also imperative to note that all other variables are also relevant in some degree to final productivity, although their contributions are not as significant as those shown in

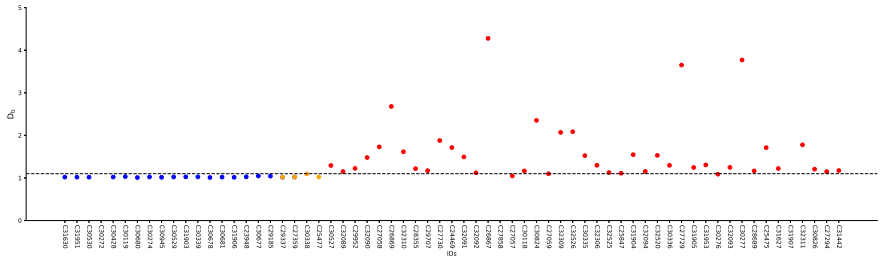


Figure 4.7: The illustration of D_b versus Batch IDs using the LSTM-AE model trained by maximizing the fault detection rate for one of the folds of the validation data set.

Figure 4.8. The average of variable contributions over the entire process time is shown in Figure 4.9. Based on the contributions plots obtained for Sanofi case study, all significant variables are identified with the SPE metric except for the seal temperatures.

4.5 Conclusion

This study proposes LSTM-AE models for monitoring the IndPenSim process and the Sanofi whooping cough vaccine manufacturing process. Two different loss functions are proposed for training: i- minimizing MSE and, ii- maximizing the average fault detection rate $\overline{FDR}$. Online monitoring of the process is accomplished by comparing the H^2 and SPE statistics of the current sample with the corresponding control limits. It is shown that the use of a loss function that is directly related to the fault detection rate provides significant improvement over the reconstruction error based loss function that was traditionally used to train auto-encoder models. Furthermore, it is demonstrated in the case studies that the use of LSTM based models provide significant improvements in detection accuracy as compared to both linear dynamic models (DPCA) and static autoencoders which do not account for dynamics. Two algorithms were developed to obtain contribution plots for the process monitoring metrics H^2 and SPE that can be used for diagnosing faults unsupervised. For the penicillin study since the mathematical model was available, it was possible to verify the ability of the contribution plots to identify the variables that are most corre-

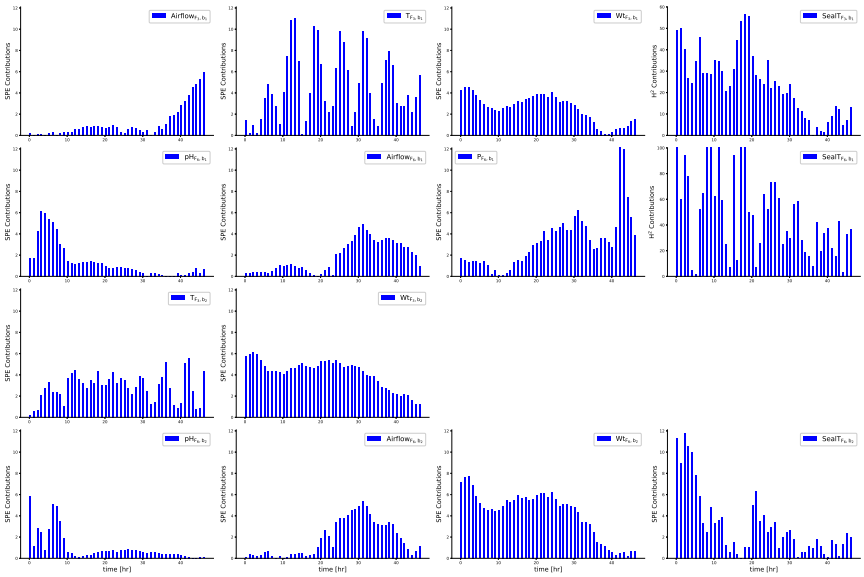


Figure 4.8: Most contributed variables in Sanofi case study.

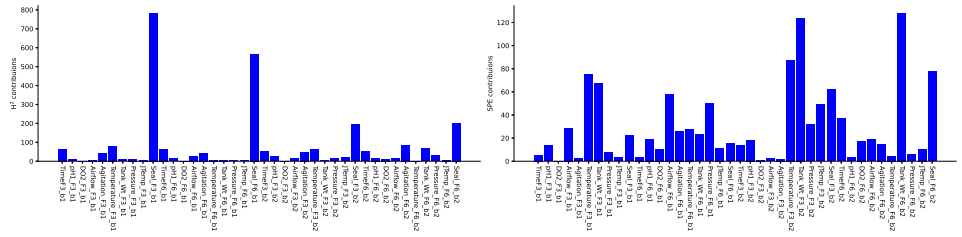


Figure 4.9: Average contribution of all variables over the entire fermentation time in Sanofi case study.

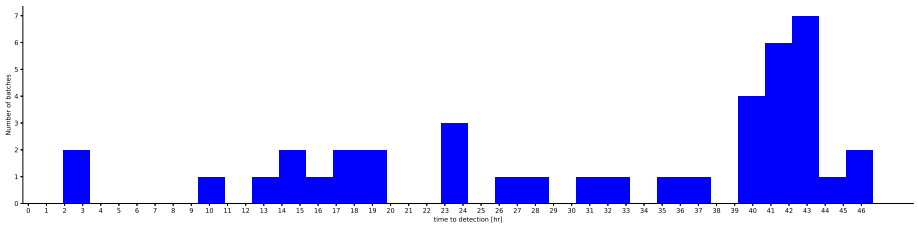


Figure 4.10: The time required to detect low productivity batches in the testing data set.

lated to each one of the faults. For the industrial case study where knowledge about the faults is not available, the contribution plots provided information about possible sources of faults and these findings were found to be in agreement with previous experimental observations. For the industrial case study, two main challenges for the application of an unsupervised learning approach were finding a region of normal operation and testing the fault detection accuracy when the occurrences and sources of faults is not known a-priori. The choice of the region of normal operation required a semi-supervised approach that used final productivity measurements of high productivity batches. For testing the detection accuracy, an average of the distances between the monitoring metrics and the upper control limits over time was used. An additional benefit of monitoring detection based on productivity was identifying the fermentation step of the manufacturing process as the one that impacts the most the final productivity as compared to other manufacturing steps.

4.6 Testing the Proposed Methodology on Sanofi's Lab-Scale Diphtheria Process

The proposed methodology has also been tested on Sanofi's lab-scale diphtheria process. However, since this section is not included in the published paper it is added as an Appendix [Aghaee et al., 2023].

The production of the diphtheria vaccine involves a meticulously controlled batch fermentation process. This begins with the cultivation of *Corynebacterium diphtheriae* in a bioreactor containing a precisely formulated nutrient medium. The bacteria are grown under tightly regulated conditions of temperature, pH, and oxygenation. During this process, the bacteria produce diphtheria toxin, which is the crucial antigen for the vaccine. After a pre-defined incubation period, the bacterial culture is harvested, and the toxin is extracted. The toxin is subsequently detoxified using formaldehyde, converting it into a non-toxic yet immunogenic form known as a toxoid. The toxoid undergoes further purification and is

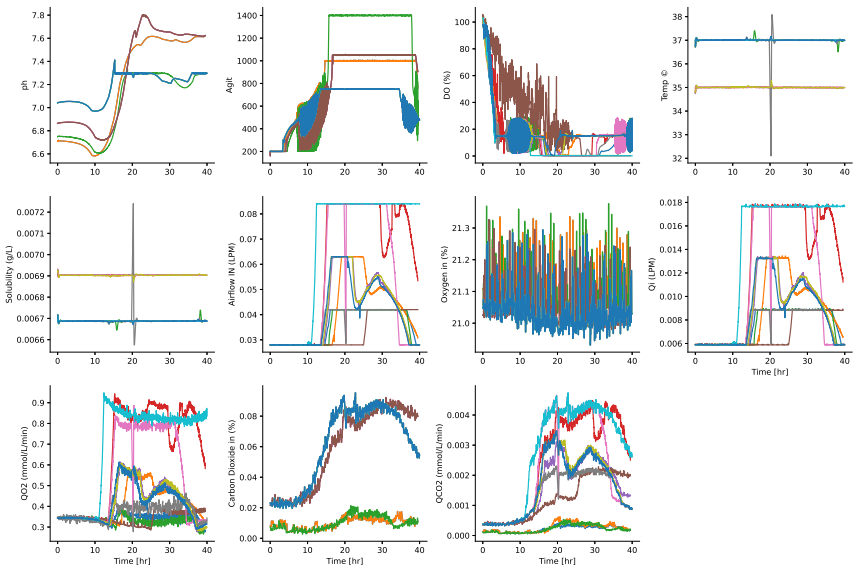


Figure 4.11: Diphtheria process variables at Sanofi, Toronto.

then combined with adjuvants and preservatives to enhance its stability and immunogenic efficacy. This results in the final vaccine preparation, which is subjected to rigorous quality control testing before being approved for distribution.

At Sanofi's laboratory, 11 process variables, including dissolved oxygen, solubility, temperature, agitation, and airflow, are recorded every minute. These variables are essential for maintaining optimal conditions during the fermentation process. The process profiles from a single batch, are illustrated in Figure 4.11.

Similar to the Sanofi's whooping cough vaccine manufacturing process described in earlier sections of this chapter, the process exhibits significant variability in final productivity, which has been measured by LF (Lateral Flow) antigen measurement. Figure 4.12 shows the wide range of final productivity for 24 batches, based on data provided by Sanofi. In consultation with expert Sanofi staff, batches with LF productivity above 190 are considered normal (golden) batches, while those with lower LF productivity are classified as low productivity (faulty) batches. Among the 11 normal batches, 9 are randomly selected

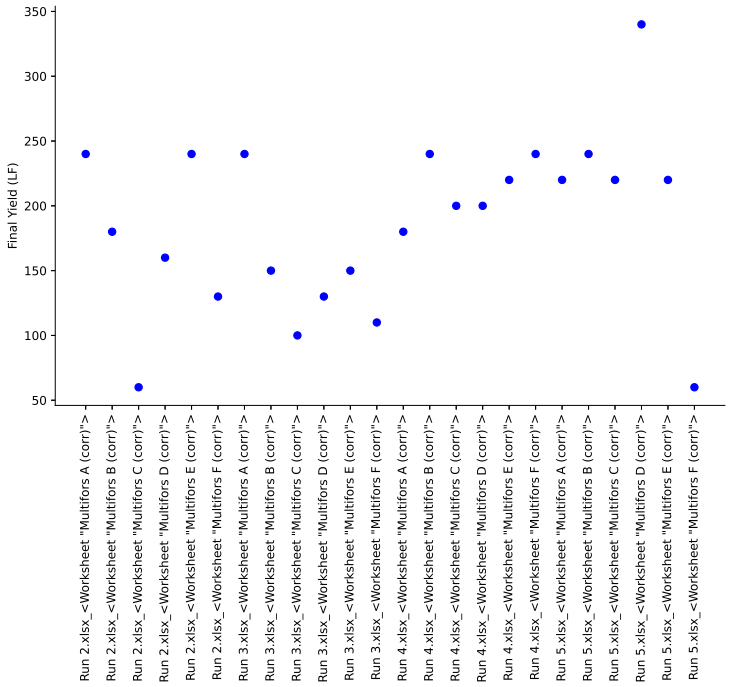


Figure 4.12: Final productivity of Diphtheria process at Sanofi, Toronto.

to form the training dataset, while the remaining 2 batches are used as the validation dataset for hyperparameter tuning. In this case study, due to the limited amount of data, a simple multi-layer autoencoder has been trained to monitor the process instead of using LSTM-AE models, which have significantly more parameters to tune. After tuning the hyperparameters, the architecture of the Autoencoder (AE) was set with layers comprising 12, 6, and 12 units, respectively. The primary hyperparameter considered in this study was the number of hidden units, with options set at 2, 4, 6, or 8 units for the hidden layer. This selection was crucial for determining the model's capacity to effectively encode and reconstruct the input data while avoiding significant over-fitting. Python's Keras-tuner is used to implement the hyperparameter search.

The classical loss function, as shown in equation 4.6, is used to train the AE model. Following this, equation 4.8 is employed to obtain process monitoring metrics, namely

Table 4.4: Results of monitoring the Diphtheria batch process at Sanofi, Toronto.

Dataset	LF	Faulty samples (%)	D_b accuracy
Training	236	15.84	0.255
Validation	230	18.04	0.301
Testing	135	60.355	1.844

H^2 and SPE. Due to the unavailability of detailed fault information in this case study, the cumulative metric, D_b also used for the Pertussis process, is utilized to assess the effectiveness of the methodology. Also, the percentage of faulty samples across different datasets is presented in Table 4.4, along with the corresponding D_b accuracy.

As shown in Table 4.4, the D_b accuracy of the training and validation datasets are of the same magnitude, as both consist solely of high productivity or normal batches. However, since the testing dataset includes only low productivity or faulty batches, the D_b accuracy and the percentage of detected faulty samples in the testing dataset are relatively higher than the D_b values for training and validation. These differences demonstrate the model's heightened sensitivity in identifying faults. The process monitoring metrics, H^2 and SPE, along with their corresponding upper control limits (UCL), are illustrated for all batches across different datasets in Figure 4.13. By comparing the metrics against their UCLs, it is possible to identify deviations and potential faults, thereby ensuring that the production process remains within acceptable operational conditions and maintaining the high quality of the diphtheria vaccine batches.

As shown in Figure 4.14, using the proposed D_b metric and setting its threshold, $D_b^{threshold}$, as the maximum of D_b values in both the training and validation datasets results in correctly detecting all the low productivity batches, except for one batch. Furthermore, Figure 4.15 demonstrates the capability of the methodology to determine the required time for detecting low productivity batches. This detection time is defined as the point at which the cumulative metric of D_b exceeds its threshold, $D_b^{threshold}$. This metric provides an indication of process deviations which could be used in the future for inferring the source of

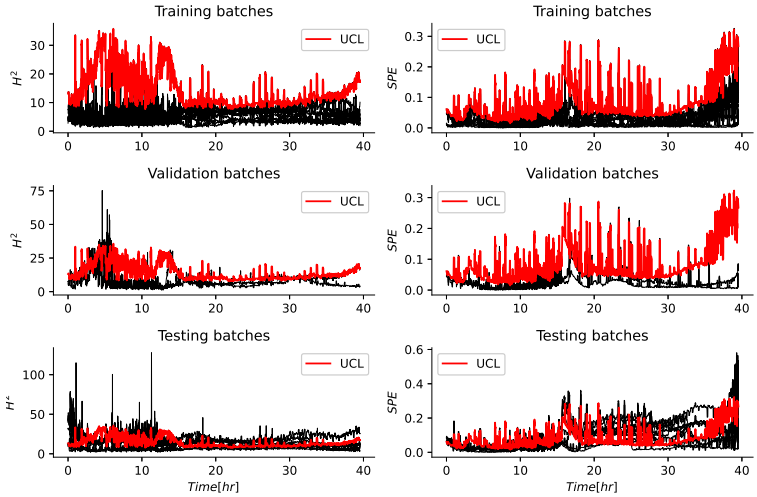


Figure 4.13: Process monitoring profiles for training, validation and testing datasets of Diphtheria process.

faults or for implementing potential corrective actions.

The results of this study illustrate the effectiveness of a multi-layer AE model in monitoring and maintaining the quality of the diphtheria batch process at Sanofi's Toronto facility. By leveraging 11 critical process variables, the AE model successfully differentiates between high productivity (normal) and low productivity (faulty) batches. The novel metric D_b, combined with classical process monitoring metrics H^2 and SPE, demonstrated a high sensitivity for identifying faults.

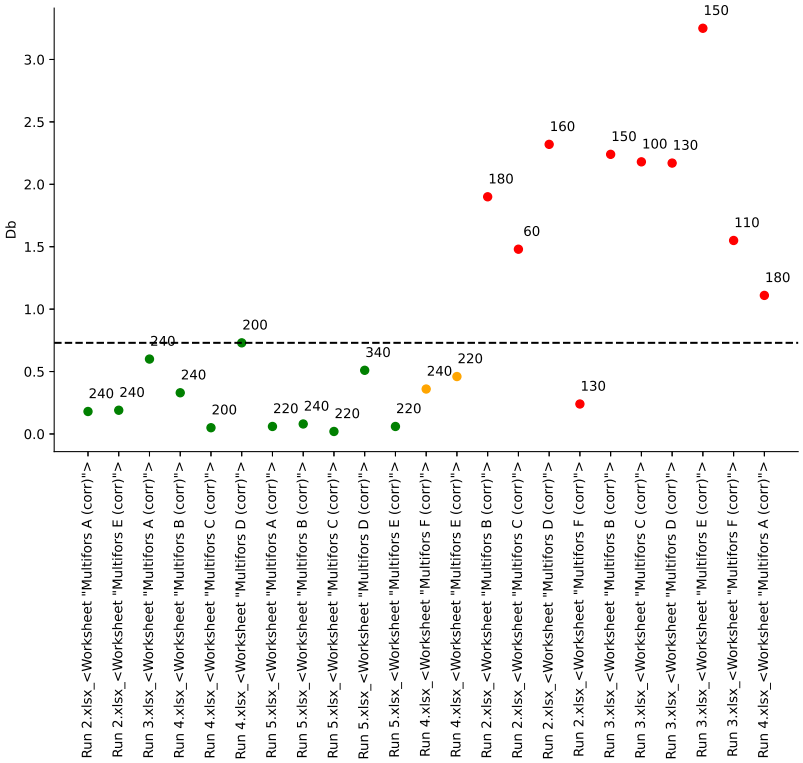


Figure 4.14: The illustration of Db versus Batch IDs using the trained AE model for monitoring the Diphtheria process.

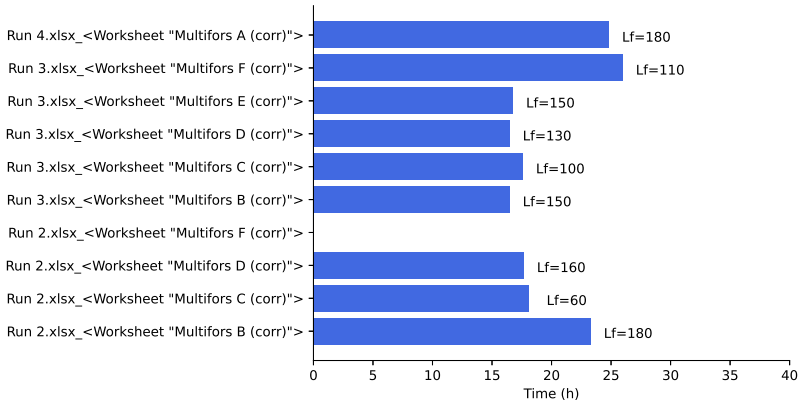


Figure 4.15: Time required to identify low productivity batches in the Diphtheria process.

Chapter 5

Unsupervised Hybrid Models Integrating Deep Autoencoders and Process Controllers' Models for Enhanced Process Monitoring and Fault Detection

Overview[1]

This chapter introduces a novel hybrid process monitoring model that integrates long short-term memory autoencoders with process controllers' models. The parameters of the hybrid

[1]Adapted from Aghaee, Mohammad, et al. "Unsupervised Hybrid Models Integrating Deep Autoencoders and Process Controllers' Models for Enhanced Process Monitoring and Fault Detection." doi.org/10.1021/acs.iecr.4c01980

model are optimized by minimizing a novel loss function, which combines the mean square error (MSE) between controlled variables and their reconstructions from the LSTM-AE model, along with the MSE of manipulated variables and their reconstructions obtained with the numerically implemented and exactly a priori known controller equations. The effectiveness of the proposed method is evaluated on the benchmark of an industrial-scale penicillin process as a batch case study and the Tennessee Eastman plant process under a decentralized control strategy as a continuous case study. A comparative analysis of the proposed hybrid model with an equivalent non-hybrid LSTM-AE model, which does not utilize process controllers' equations, highlights the superiority of the proposed hybrid monitoring model in fault detection. These improvements result from the use of an LSTM-AE network with fewer parameters thus making it less susceptible to overfitting.

5.1 Introduction

Effective online process monitoring is crucial for ensuring the safe and efficient operation of chemical processes, as it guarantees that process variables remain within optimal ranges. On the other hand, variability in raw materials and operational disturbances can lead to deviations from these optimal conditions, ultimately causing diminished process performance. The deployment of intelligent fault detection techniques represents a strategic approach to real-time process monitoring, aimed at maintaining process optimality. Fault detection methods can be classified into two primary categories based on the utilized model: i. mechanistic models, and ii. empirical models. While mechanistic model-based methods are based on models derived from first principles, empirical model-based approaches are generally preferred due to the challenges in obtaining mechanistic models. For instance, modeling of pharmaceutical processes by first-principles equations is challenging owing to an incomplete understanding of the metabolic behavior of microorganisms and the inherent nonlinear behaviour especially in batch or fed-batch operations. Empirical model-based fault detection algorithms can be further divided into supervised and unsupervised meth-

ods. In supervised learning, a model is trained using labeled samples of normal or faulty operation provided that knowledge about the occurring fault is available during model training. On the other hand, in the absence of such knowledge, unsupervised learning methods derive patterns directly from unlabeled data.

Artificial intelligence, with its ability to autonomously extract hierarchical representations from data, has significantly revolutionized fault detection and diagnosis across various industrial domains. Deep learning models, known for their effectiveness in extracting nonlinear features, have been extensively investigated for the development of diverse fault detection methodologies [Aghaee et al., 2024a]. Particularly in scenarios where conventional nonlinear mechanistic models are lacking, machine learning and deep learning techniques have served to develop empirical model-based methods for monitoring nonlinear systems. Autoencoders (AEs) have been widely employed in process monitoring, characterized by an equal number of units in their input and output layers, typically incorporating one or more low-dimensional hidden layers [Zhang and Qiu, 2022, Agarwal et al., 2022, Essien and Giannetti, 2020]. They may be operated in an unsupervised manner, solely relying on input data for calibration. Consisting of an encoder and a decoder, AEs sequentially compress the input space into a lower-dimensional latent variable space and then reconstruct the input space from these latent variables. Consequently, AEs can be conceptually likened to nonlinear principal component analysis (PCA) due to their ability to compress the larger input space into a smaller latent variable space.

The ability of recurrent neural networks (RNNs) to effectively model the nonlinear dynamics inherent in high-dimensional process data has facilitated the development of empirical model-based fault detection algorithms based on RNN-based models [Amini and Zhu, 2022, Liu et al., 2023b, Qi et al., 2023]. Various variational autoencoder (VAE) models have been extensively evaluated for monitoring industrial processes, particularly on the Tennessee Eastman process benchmark [Zhu et al., 2021]. The results from these studies demonstrate that recurrent VAE models, including LSTM-VAE and GRU-VAE, are well-

suited for industrial process monitoring tasks and are thus recommended for such applications. In a recent study, unsupervised multi-layer long short-term memory autoencoder (LSTM-AE) models designed for monitoring nonlinear batch processes were introduced [Aghaee et al., 2023]. These models were evaluated using both a penicillin simulator case study and real-world data from *B. pertussis* vaccine manufacturing at Sanofi. The findings indicate that the proposed LSTM-AE models offer significant improvements in accuracy compared to both linear and nonlinear approaches that do not explicitly address dynamic behavior.

One significant challenge in fault detection model development is over-fitting due to the limited availability of sufficient data for training large-scale models capable of accurately replicating the nonlinear dynamic behavior inherent in industrial processes. Over-fitting will result in inaccurate prediction for input data that was not used during training thus impacting fault detection accuracy. Conventional methods to mitigate over-fitting include reducing model size by decreasing the number of parameters or augmenting the dataset. However, smaller models may struggle to adequately capture process intricacies, while gathering more data could be impractical due to cost constraints [Yu and Zhang, 2023].

While deep learning neural networks demonstrate high interpolating capabilities, they often struggle with accurately generalizing behavior, particularly when predicting outputs for input values significantly different from those used during training. Gray-box or hybrid models combine mechanistic and empirical model-based techniques to enhance predictive capabilities and reduce reliance on training data [Pistikopoulos et al., 2021]. These hybrid models integrate mechanistic and empirical-based components either serially or in parallel architectures [Fontes and Budman, 2017]. Mechanistic models encapsulate the fundamental conservation laws governing the process. By leveraging these fundamental laws partially, hybrid models facilitate the estimation of unmeasured process states and augment datasets, thus improving the performance of process monitoring models [Cabaneros Lopez et al., 2021]. For example, Narayanan et al. [Narayanan et al., 2020] introduce a serial hybrid

model that combines partial least squares (PLS) with an extended mechanistic model-based Kalman filter to enhance the monitoring of a mammalian cell culture process. This model improves monitoring by utilizing an augmented dataset to train the PLS model, integrating both estimated and measured process variables.

The present work is inspired by existing serial grey box models, which combine partial knowledge of the process with empirical models to enhance process monitoring. However, the mechanistic component of the proposed hybrid model are the controllers' equations that are exactly known a priori in contrast to mechanistic models traditionally employed in hybrid approaches that may have error.

However, two key challenges are identified for fault detection studies that used hybrid models:

i- most of the hybrid process monitoring models utilize linear machine learning models such as PCA or PLS as the empirical model-based component of the hybrid model [Destro et al., 2020, Narayanan et al., 2020, Fontes and Budman, 2017, Li et al., 2023b, Hassanpour et al., 2024].

ii- the ability of a hybrid model-based approach to improve performance in comparison with a purely empirical- approach largely depends on the accuracy of the available mechanistic model.

To address these identified limitations in prior research, this study introduces a novel serial hybrid process monitoring model. This hybrid model integrates unsupervised deep LSTM-AE models, proficient in capturing process nonlinearity, with the process controllers' exact models to enhance fault detection. The key innovation lies in integrating into the hybrid model a mechanistic part, i.e. controllers' equations, that is exact because the process controllers are perfectly known a priori. The use of the controllers' equations results in a significant reduction in the number of deep learning parameters while avoiding the introduction of model error that would result if other mechanistic equations were used. Furthermore, the use of a numerical implementation of the controller equations allows

for the consideration of any type of controller such as PID with anti-windup, constrained MPC and nonlinear controllers. The distinction between our previous work [Aghaee et al., 2023] and the current study is that the hybrid approach proposed here, in contrast with our earlier study, does not use the manipulated variables in the autoencoder. Instead, the manipulated variables are reconstructed from the controlled variables reconstructed in the autoencoder with the exact controllers' equations thus reducing the autoencoder size and the risk of overfitting. The proposed fault detection framework comprises two main components: i- training the LSTM-AE model under normal operating conditions, utilizing a loss function that minimizes the (weighted) mean square error (MSE) between input and reconstructed samples, along with the MSE in predicting control actions, ii- Establishing statistical upper control limits for fault detection by employing Gaussian kernel density estimation at each time-step.

The effectiveness of the proposed methodology in fault detection is then evaluated on two case studies: the industrial-scale penicillin fermentation simulator benchmark (IndPenSim) [Goldrick et al., 2015, 2019], and the Tennessee Eastman (TE) process benchmark under a decentralized control strategy [Ricker, 1996, Andersen et al., 2022]. In both case studies, fault information, including type, magnitude, and time range, has not been utilized during model development to illustrate the effectiveness of the proposed unsupervised learning-based approach.

The chapter is structured as follows: Section 5.2 presents the mathematical background of process monitoring with deep LSTM-AE and hybrid models and a concise overview of process monitoring metrics employed to assess model effectiveness. Section 5.3 introduces the novel hybrid modeling approach for monitoring that combines perfect process controllers' exact equations and the LSTM-AE model. The effectiveness of the proposed methodology on both case studies is presented in Section 5.4, followed by conclusions in Section 5.5.

5.2 Preliminaries

5.2.1 Long short-term memory autoencoders (LSTM-AE)

5.2.2 Long short-term memory autoencoders (LSTM-AE)

An Autoencoder (AE) consists of two serially connected neural networks: an encoder and a decoder. Trained in an unsupervised manner, an AE can extract patterns from data and compress them into a smaller dimensional latent variable space. Within the encoder, the input sample data $x_k \in \mathbb{R}^j$ is compressed onto a reduced latent space, $h \in \mathbb{R}^l$, while the decoder reconstructs the inputs, $\hat{x}_k \in \mathbb{R}^j$, using the outputs of the hidden layer. Here, j denotes the number of variables, and l denotes the number of hidden units in the deepest layer of the AE. Traditional autoencoders have shown promise by efficiently learning non-dynamic data, but their applicability to dynamically correlated data has been limited.

LSTM-AEs represent a specialized class of AEs specifically tailored for reconstructing nonlinear dynamically correlated data. Unlike conventional AEs that rely on simple neural network units, LSTM-AEs use long short-term memory (LSTM) units, which are capable of modeling temporal dependencies inherent in dynamic data. As illustrated in Figure 5.1, an LSTM unit comprises three crucial gates: the input gate (i_t), the forget gate (f_t), and the output gate (o_t), each tasked with regulating the flow of particular information within the network. These gates govern the acceptance and retention of relevant information from the current input (x_t), the previous hidden state (h_{t-1}), and the current cell state (c_{t-1}). Specifically, the input gate (i_t) determines the extent to which new data should be incorporated, the forget gate (f_t) decides which past data to discard, and the output gate (o_t) controls the distribution of data towards subsequent layers. Mathematically, the LSTM unit's dynamics are governed by a set of equations, where each component is

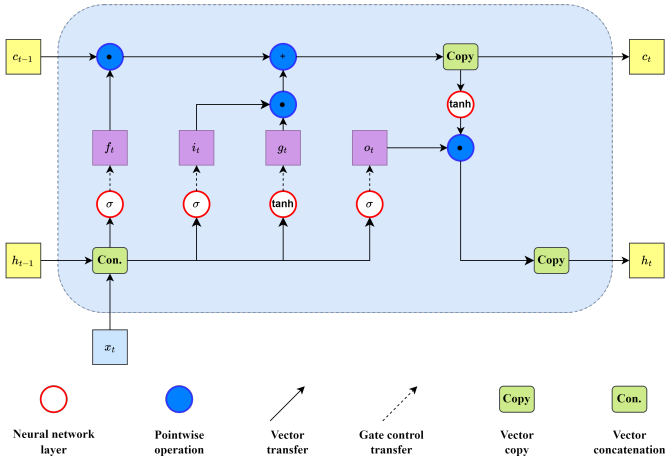


Figure 5.1: Schematic of LSTM cell architecture.

updated iteratively over time according to the following expressions:

$$\begin{cases} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot tanh(c_t) \end{cases} \tag{5.1}$$

Here, h_t and c_t represent the hidden state and cell state at time step t, respectively, i_t, f_t and o_t denote the input, forget, and output gates, respectively, while g_t stands for the cell gate. The sigmoid function (σ) serves as the activation function and the Hadamard product ($\odot$) denotes element-wise multiplication. These equations explain the intricate dynamics that can be captured by LSTM-AEs, enabling them to effectively model and reconstruct complex nonlinear dynamic data patterns. Additional information about LSTM units can be found in the original reference papers [Hochreiter and Schmidhuber, 1997].

5.2.3 Hybrid models

The proliferation of data in industrial settings has made empirical model-based approaches increasingly appealing for process monitoring. Conversely, first principles models, also known as mechanistic models, capture process variable interactions based on physical principles such as mass, energy, and momentum balances. Hybrid models provide a trade-off of both first principles and empirical-based methodologies, leveraging the strengths of each approach [Psichogios and Ungar, 1992]. Empirical models are known to have good interpolation properties within the region of data used for model calibration but are poor at generalizing behaviour beyond the region of data used for training. Thus, the combination of empirical models with mechanistic models is expected to improve the generalization properties. Also, the hybrid model will generally have significantly fewer parameters as compared to the purely empirical model since the mechanistic part of the model is expected to provide a more compact description of the correlations among variables as compared to a neural network of similar accuracy. For instance, a single neural network model needs many units with *tanh*, *relu* or other basic activation functions to be able to construct some relatively simple nonlinear correlations between the process states, e.g. Monod equation, or even linear ones, e.g. many *sigmoid* units are required to reconstruct a linear function. The use of a smaller number of parameters results in diminished overfitting decreased sensitivity to data noise, and enhanced generalization [Mavrovouniotis and Chang, 1992]. However, the success of a hybrid model-based fault detection strategy will largely depend on the accuracy of the available mechanistic equations. If the mechanistic model error is very high, its use may degrade the performance of the hybrid model.

5.2.4 Process monitoring metrics

Sensitivity, specificity, and balanced accuracy serve as metrics for evaluating the effectiveness of process monitoring models when fault information is available. Sensitivity, also

referred to as the true positive rate, measures the model's ability to accurately detect faulty samples. Conversely, specificity, or the true negative rate, indicates the model's capability to correctly identify normal samples. These metrics are defined as follows:

$$\begin{cases} Balanced \;\; accuracy = \frac{Sensitivity+Specificity}{2} \\ Sensitivity = \frac{1}{n_a B} \sum_{k=1}^{n_a} \sum_{b=1}^{B} [H_{k,b}^2 > H_{k,\alpha}^2] \; or \; [SPE_{k,b} > SPE_{k,\alpha}] \\ Specificity = \frac{1}{n_n B} \sum_{k=1}^{n_n} \sum_{b=1}^{B} [H_{k,b}^2 \leq H_{k,\alpha}^2] \; and \; [SPE_{k,b} \leq SPE_{k,\alpha}] \end{cases} \tag{5.2}$$

Here, n_n and n_a represent the number of normal and abnormal samples within a single batch or continuous run. B denotes the total number of batches or continuous runs. Similar to linear models such as PCA and PLS, which utilize Hotelling's T^2 and Q-statistics for monitoring [MacGregor et al., 1994], AE-based models employ equivalent statistical metrics, H^2 and SPE, for monitoring purposes. These statistical metrics can be calculated as follows:

$$H_{k,b}^2 = \sum_{d_h=1}^{R} \frac{(h_{k,b,d_h}^{deepest} - h_{k,d_h}^{mean})^2}{h_{k,d_h}^{variance}} \tag{5.3a}$$

$$SPE_{k,b} = \sum_{j=1}^{n} (x_{k,b,j} - \hat{x}_{k,b,j})^2 \tag{5.3b}$$

Here, $h^{deepest}k, b, d_h$ represents the d_h^{th} element of the hidden variables in the deepest layer of the encoder part of the AE model at time k of the b^{th} dataset. $x_{k,b,j}$ and $\hat{x}_{k,b,j}$ denote the j^{th} process variable of the input sample and its corresponding reconstructed variable, respectively, for the b^{th} dataset. The upper control limits, represented by ${H^2}_{k,\alpha}$ for H_k^2 and $SPE_{k,\alpha}$ for SPE_k, are established at a significance level of α for each sampling time through kernel density estimation (KDE) [Parzen, 1962b], as the input data may not follow a Gaussian distribution, as described in Equation 5.4. Here, $F(M_k)$ represents

the probability density function, $G(x_i;\sigma)$ is the Gaussian kernel, and b_w is the bandwidth parameter determined through the Silverman algorithm [Silverman, 2018].

$$\begin{cases} F(M_k) = \sum_{i=1}^{b} G\Big(\frac{M_k - M_i}{b_w}\Big); M \in \{H^2, SPE\} \\ G(x_i;\sigma) \propto exp\Big(-\frac{{x_i}^2}{2\sigma^2}\Big) \\ 1-\alpha = \int_{\min(M_{k,train})}^{M_{k,\alpha}} F(M_k)\ dM_k \end{cases} \tag{5.4}$$

A sample is correctly detected as corresponding to normal operation if both its corresponding $H^2_{k,b}$ and $SPE_{k,b}$ values fall below their respective upper control limits. Conversely, if a sample corresponds to normal operation but either or both of its $H^2_{k,b}$ and $SPE_{k,b}$ values exceed their limits, the sample will be erroneously classified as abnormal. On the other hand, abnormal samples are correctly identified as such if either $H^2_{k,b}$ or $SPE_{k,b}$, or both, exceed their control limits. Conversely, abnormal samples will be misclassified as normal if neither $H^2_{k,b}$ nor $SPE_{k,b}$ surpass their respective upper control limits.

5.3 Proposed Methodology

5.3.1 Proposed hybrid model: integration of LSTM-AE with process controllers' exact models

In this section, we introduce a novel hybrid process monitoring model that combines the deep LSTM-AE model with process controllers' models. Additionally, we outline the procedure used for evaluating the proposed methodology. The proposed methodology exploits the exact correlations between manipulated and controlled variables enforced by the process controllers. This will be shown to enhance the effectiveness of process monitoring models while resulting in a smaller AE neural network model, which is less prone to overfitting. Typically, all measured variables including manipulated and controlled variables

are directly fed into AE models. Then, the AE is trained to minimize the MSE between input and reconstructed variables as outlined in Figure 5.2A. In contrast, for the novel hybrid model proposed in this study, only the open-loop variables and controlled variables are directly fed into the AE model, while the closed-loop manipulated variables are reconstructed using the controller equations based on the reconstructed controlled variables. In other words, the manipulated variables are eliminated from the input and output layer of the LSTM-AE model as shown in Figure 5.2 resulting in a smaller model in terms of the number of parameters needed to reconstruct the inputs to the LSTM-AE. To preserve the information contained in the manipulated variables for process monitoring, the loss function of the LSTM-AE model incorporates the MSE between these variables and their reconstructions obtained through the controllers' equations in addition to the MSE of LSTM-AE variables. It should be emphasized that both the controllers' structures, e.g. PID, model predictive control (MPC), ratio, etc., and the controller tuning parameters are always available a priori to the process engineer. The motivation for using the exact controllers' equations in combination with a LSTM-AE is that reconstructing the linear or nonlinear dynamic relations between manipulated and controlled variables by the LSTM-AE alone may require larger network interconnections.

In the proposed hybrid model, autoencoders are integrated with process controllers by incorporating the process controller equations into the loss function of the autoencoder. This raises the question of how TensorFlow, which relies on automatic differentiation, handles the differentiation of the process controller equations, especially when using a custom loss function. In TensorFlow, "custom loss" functions are defined by explicitly specifying how the loss is computed using the model outputs. TensorFlow then uses automatic differentiation (back-propagation) to calculate the gradients of the loss with respect to the model weights. For this to work, the operations within the custom loss function, including the process controller equations, must be differentiable. Most commonly used controllers, such as PI and PID controllers without saturation, are based on continuous mathematical models that are differentiable. As a result, their inclusion in the loss function does

not introduce non-differentiable components, allowing TensorFlow to compute the necessary gradients by autograd operation. However, in cases where the controller equations involve non-differentiable elements, smoothing techniques or differentiable approximations are employed by default to calculate gradients.

A comparison between the architectures of the proposed hybrid AE model and traditional AE models is depicted in Figure 5.2. The proposed loss function combines the MSE between controlled variables and their reconstructions from the LSTM-AE model, along with the MSE of manipulated variables and their reconstructions obtained through the controllers' equations, as given by the following equations:

$$J_1 = \sum_{j=1}^{m} \left[u_j^{OL} - \hat{u}_j^{OL}\right]^2 + \sum_{r=1}^{p} \left[y_k^{OL} - \hat{y}_k^{OL}\right]^2 + \sum_{r=1}^{n} \left[y_r^{CL} - \hat{y}_r^{CL}\right]^2 \tag{5.5a}$$

$$J_2 = \sum_{r=1}^{n} \left[u_r^{CL} - \hat{u}_r^{CL}\right]^2 = \sum_{r=1}^{n} \left[u_r^{CL} - f_r^{controller}(\hat{y}_r^{CL})\right]^2 \tag{5.5b}$$

$$J = J_1 + J_2 \tag{5.5c}$$

Here, m, n, and p represent the numbers of open-loop input variables, i.e. all inputs other than manipulated variables used in closed-loop, controlled variables, and open-loop input variables reconstructed by the decoder, respectively. Examples of input variables that are not used for closed-loop control are feed rates of raw materials, composition of raw materials, etc. y_k^{OL} and $\hat{y}_k^{OL}$ represent the k^{th} output variable and its decoder reconstructed counterpart, while y_r^{CL} and $\hat{y}_r^{CL}$ represent the r^{th} closed-loop controlled variable and its reconstruction, respectively. Similarly, u_j^{OL} and $\hat{u}_j^{OL}$ denote the j^{th} open-loop input variable and its decoder reconstruction, while u_r^{CL} and $\hat{u}_r^{CL}$ denote the r^{th} closed-loop manipulated variable and its reconstruction by the controller's equations. $f_r^{controller}$ indicates the controller's equations relating the predicted r^{th} manipulated variable ($\hat{u}_r^{CL}$) and the controlled variable value reconstructed by the decoder ($\hat{y}_r^{CL}$). It is important to note that,

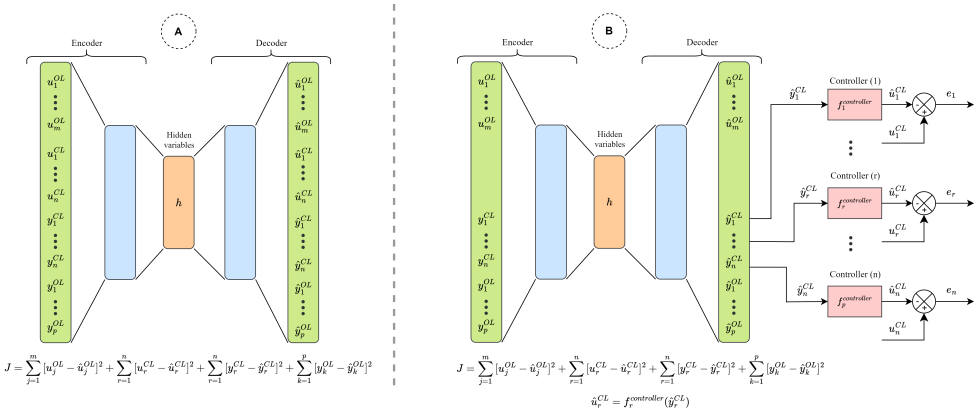


Figure 5.2: Schematic of process monitoring models: **A.** single autoencoder model, **B.** proposed hybrid autoencoder model.

as shown in the Eqs. 5.5a-5.5c, the objective function J_2 incorporates the controller equation and the reconstructed output, $f_r^{controller}(\hat{y}_r^{CL})$, which follows from the predictions in J_1, thereby resulting in a hybrid serial model. The controller equations $f_r^{controller}(\hat{y}_r^{CL})$ are implemented numerically thus permitting implementation of any controller scheme. For example, in the Tennessee Eastman case study we have implemented PI with anti-windup and ratio controllers. If an advanced controller such as constrained MPC is used, the required MPC optimization algorithm can be numerically implemented as a function (e.g., Python function code) outside the AE network model and the numerical results can be used in the loss function.

The hybrid model employs the statistical metrics of H^2 and SPE for process monitoring reviewed in the previous section. The H^2 metric is affected by the controller equations, as the loss function used to train the LSTM-AE includes both the reconstruction error of the LSTM-AE as well as the error between the measured and the manipulated variable predicted with the controller equation. As for the SPE metric, which encompasses reconstruction errors, it is modified with respect to a standard LSTM-AE to incorporate the error in the reconstruction of the manipulated variable between the measured value and the value predicted by the controllers' equations. To do so, this work proposes $SPE^{augmented}$,

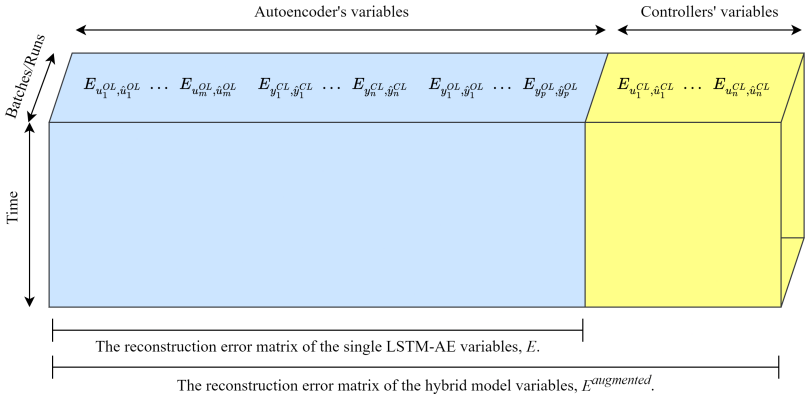


Figure 5.3: Architecture of the SPE and $SPE^{augmented}$ matrices of the proposed hybrid process monitoring model.

which is computed from the augmented reconstruction error matrix. This matrix includes the reconstruction error of the closed-loop manipulated variables in addition to the reconstruction error of the LSTM-AE model. Figure 5.3 illustrates the structure of the reconstruction error matrices E and $E^{augmented}$, where $E \in \mathbb{R}^{k \times b \times (m+n+p)}$ corresponds to SPE, and $E^{augmented} \in \mathbb{R}^{k \times b \times (m+2n+p)}$ corresponds to $SPE^{augmented}$. Here, k represents the number of sampling times, and b represents the number of batches or runs for continuous processes. After training the hybrid LSTM-AE model and constructing the matrix $E^{augmented}$, equations 5.3a and 5.3b can be used to compute H^2 and $SPE^{augmented}$ for each batch or run. Finally, the equation for FDR (Eq. 5.2) can be reformulated using H^2 and $SPE^{augmented}$, instead of the standard H^2 and SPE, to assess the effectiveness of the proposed hybrid model-based FD algorithm.

5.3.2 Case studies

To assess the generality of the approach, it is tested for two case studies involving batch (Industrial-scale penicillin fermentation) and continuous (Tennessee Eastman Problem) operations respectively. Nonlinear behaviour is especially significant in batch operations

since the process variables undergo large changes during the batch. Thus, the application of the proposed approach to batch serves to illustrate the ability of the algorithm to cope with nonlinearity. On the other hand, continuous operations will generally involve a richer dynamic behaviour as the process is exposed to faults (disturbances) entering at different times. In this case, the application of the proposed approach to a continuous process serves to illustrate the ability of the LSTM-AE to deal with dynamically correlated data. Also, the Tennessee Eastman problem involves several PI with anti-windup and ratio controllers, and thus, by accounting for the exact controller equations, the proposed hybrid model-based approach has the potential for significant improvements in performance.

Industrial-scale penicillin fermentation simulator (IndPenSim)

The simulator of IndPenSim, available at www.industrialpenicillinsimulat ion.com, has been widely utilized to assess numerous fault detection algorithms for monitoring batch processes [Agarwal et al., 2022, Jia et al., 2023, Aghaee et al., 2023, Liu et al., 2024b]. IndPenSim simulates a 100,000L bioreactor, modeling several process variables of the penicillin chrysogenum fermentation such as dissolved oxygen, viscosity, temperature, pH, and dissolved carbon dioxide. Furthermore, the model incorporates the influences of nitrogen and phenylacetic acid concentrations on biomass and penicillin production rates. A schematic of the IndPenSim process is depicted in Figure 5.4. The simulator code is used to generate in-silico data from 700 batches, comprising 100 normal batches and 600 faulty batches (100 batches per fault). Each batch dataset comprises 25 measured variables, encompassing both controlled and manipulated variables, collected over a duration of 230 hours (equivalent to 1150 timesteps). Information regarding the six different types of faults is outlined in Table 5.1. Normal operation batches are subsequently split into training and validation sets. Among the 100 normal batches, 80 are designated for training the hybrid model, while 20 are reserved for validation to fine-tune the hyperparameters of the LSTM-AE model. The remaining 600 batches are utilized as the testing dataset. Each dataset is represented

Table 5.1: Details of various fault types in the IndPenSim case study.

Fault No.	**Fault**	**Faulty value**	**Time range**
1	Disturbance in aeration flow rate (L/hr)	20, 20	[20, 24], [100, 110]
2	Disturbance in vessel back pressure (bar)	2, 2	[100, 104], [200, 230]
3	Disturbance in substrate feed rate (L/hr)	2, 20, 20	[20, 30], [76, 92], [200, 214]
4	Disturbance in base flow rate (L/hr)	5, 10	[80, 84], [140, 160]
5	Disturbance in coolant flow rate (L/hr)	2	[70, 90]
6	All of the above faults		

by a three-dimensional matrix, $X \in \mathbb{R}^{k \times b \times j}$, where k is the number of samples in a batch dataset, b is the number of batches for that particular dataset, and j is the number of process variables in the dataset. The first step involves unfolding the three-dimensional dataset into a two-dimensional format, denoted as $X^{unfolded} \in \mathbb{R}^{kb \times j}$. Subsequently, normalization of all datasets is carried out using the minimum and maximum values for each variable present in the training dataset.

The IndPenSim process has two PID controllers with the clipping strategy for controlling pH and temperature (T) of the fermenter by manipulating the base flow rate (F_b) and coolant flow rate (F_c), respectively. The structure of the two PID controllers and their corresponding parameters are presented in Equation 5.6 and Table 5.2, respectively.

$$\begin{aligned} u_k = \min \Bigg(\max \Bigg(& u_{k-1} + K_c(e_k - e_{k-1}) + \frac{K_c T_s}{\tau_I} e_k \\ & - \frac{K_c \tau_D}{T_s}(y_k - 2y_{k-1} + y_{k-2}), u_{low} \Bigg), u_{high} \Bigg) \\ e_k = y^{sp} - y_k & \end{aligned} \tag{5.6}$$

where u_k, y^{sp}, and T_s are the control action at time k, the setpoint of controlled variable y, and the sampling time (0.2 hours), respectively.

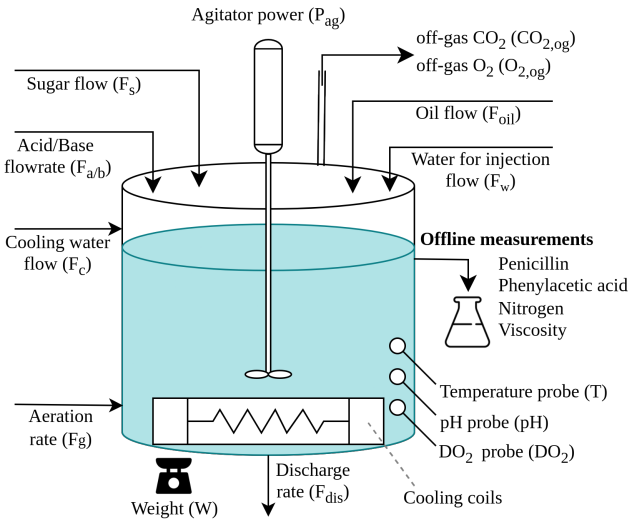


Figure 5.4: Schematic of IndPenSim process.

Table 5.2: Parameters of the PID controllers for pH and temperature control in the IndPenSim process.

Loop	**Controlled variable (y)**	**Manipulated variable (u)**	$\mathbf{K_c}$	$\tau_\mathbf{I}$**(min)**	$\tau_\mathbf{D}$**(min)**	$\mathbf{y^{sp}}$
1	pH	$F_b \in [0, 225]$	0.08	4e-5	8	6.5
2	T (K)	$F_c \in [0, 1500]$	-300	1.6	0.005	298

Tennessee Eastman (TE) process under decentralized control strategy

In order to evaluate the efficacy of the proposed methodology across a larger-scale system featuring multiple controllers, we opted to utilize the TE process employing Ricker's control strategy. The TE process has been widely used to assess various empirical model-based methods for monitoring and detecting faults in continuous processes [Shams et al., 2010, Heo and Lee, 2019, Lau et al., 2013, Chiang et al., 2000, Lomov et al., 2021, Kong and Ge, 2021]. This plant comprises various unit operations, including a vapor-liquid separator, reactor, stripper, recycle compressor, and condenser. Through a series of chemical reactions, four gaseous reactants (A, B, C, and D) give rise to two liquid product streams (G and H)

alongside a by-product (F). This system encompasses 41 measured outputs, each subject to inherent measurement noise [Ricker, 1996]. Among these outputs, 19 pertain to composition measurements obtained from gas chromatographs, sampled at two different rates and characterized by pure delays. Table 5.3 provides the details of TE process variables, covering both measurement and manipulated variables. Additionally, the system encompasses 12 manipulated variables, comprising 11 valves and the agitation speed of the reactor. The data generator GUI is freely accessible at https://github.com/dtuprodana/TEP. The Simulink file is utilized to generate 10 normal runs for the training dataset, 3 normal runs for the validation dataset, and 28 runs representing various faulty scenarios (1 run per fault). Faults 3, 5, 9, 15, 16, 21, 22, 23, and 28 are widely reported to be difficult to detect due to their relatively minor deviations in process variables compared to normal operating conditions [Reinartz et al., 2021]. Detection of these faults has been addressed with specialized approaches such as CUSUM-PCA [Shams et al., 2010] and hierarchical approaches. Details regarding all the faults are provided in Table 5.4. Each simulation run has a duration of 20 hours and consists of 40,001 sample points, with a sampling time (T_s) of 5e-4 hours. Various implementations of PI controllers are employed in the Simulink file, with each presented by the following Equations 5.7a-5.7b. Detailed information regarding the controllers is also provided in Table 5.5. As evident in the setpoint column (y^{sp}) of Table 5.5, the setpoints of control loops 1-7 are dependent on the controlled variable of control loop 8. This complex control structure is referred to as a ratio control strategy in the work by Ricker et al. [Ricker, 1996]. Additionally, the setpoint of control loop 17 depends on the controlled variable of loop 11. Additional details about the process model, Ricker's control strategy, and the GUI of the Simulink file can be found in the original papers [Downs and Vogel, 1993, Ricker, 1995, 1996, Bathelt et al., 2015, Andersen et al., 2022]. The hybrid model to be used in this work for fault detection considers explicitly all these control strategies within the model.

Table 5.3: Controlled and manipulated variables of Tennessee Eastman plant process.

Variable Name	**Variable Number**	**Unit**	**Variable Name**	**Variable Number**	**Unit**
A feed (stream 1)	XMEAS (1)	kscmh	Feed %F	XMEAS (28)	mol%
D feed (stream 2)	XMEAS (2)	kgh^{-1}	Purge %A	XMEAS (29)	mol%
E feed (stream 3)	XMEAS (3)	kgh^{-1}	Purge %B	XMEAS (30)	mol%
A and C feed (stream 4)	XMEAS (4)	kscmh	Purge %C	XMEAS (31)	mol%
Recycle flow	XMEAS (5)	kscmh	Purge %D	XMEAS (32)	mol%
Reactor feed rate (stream 6)	XMEAS (6)	kscmh	Purge %E	XMEAS (33)	mol%
Reactor pressure	XMEAS (7)	kPa guage	Purge %F	XMEAS (34)	mol%
Reactor level	XMEAS (8)	%	Purge %G	XMEAS (35)	mol%
Reactor temperature	XMEAS (9)	°C	Purge %H	XMEAS (36)	mol%
Purge rate (stream 9)	XMEAS (10)	kscmh	Product %D	XMEAS (37)	mol%
Product separator temperature	XMEAS (11)	°C	Product %E	XMEAS (38)	mol%
Product separator level	XMEAS (12)	%	Product %F	XMEAS (39)	mol%
Product separator pressure	XMEAS (13)	kPa guage	Product %G	XMEAS (40)	mol%
Product separator underflow (stream 10)	XMEAS (14)	m^3h^{-1}	Product %H	XMEAS (41)	mol%
Stripper level	XMEAS (15)	%	D Feed flow	XMV(1)	kgh^{-1}
Stripper pressure	XMEAS (16)	kPa guage	E Feed flow	XMV(2)	kgh^{-1}
Stripper underflow (stream 11)	XMEAS (17)	m^3h^{-1}	A Feed flow	XMV(3)	kgh^{-1}
Stripper temperature	XMEAS (18)	°C	A and C Feed flow	XMV(4)	kgh^{-1}
Stripper steam flow	XMEAS (19)	kgh^{-1}	Compressor recycle valve	XMV(5)	%
Compressor work	XMEAS (20)	kW	Purge valve	XMV(6)	%
Reactor cooling water outlet temperature	XMEAS (21)	°C	Separator pot liquid flow	XMV (7)	m^3h^{-1}
Separator cooling water outlet temperature	XMEAS (22)	°C	Stripper liquid product flow	XMV (8)	m^3h^{-1}
Feed %A	XMEAS (23)	mol%	Stripper steam valve	XMV (9)	%
Feed %B	XMEAS (24)	mol%	Reactor cooling water flow	XMV (10)	m^3h^{-1}
Feed %C	XMEAS (25)	mol%	Condenser cooling water flow	XMV (11)	m^3h^{-1}
Feed %D	XMEAS (26)	mol%	Agitator speed	XMV (12)	%
Feed %E	XMEAS (27)	mol%			

$$u_k = min(max(u_{k-1} + K_c(e_{k-1} - e_{k-2}) + \frac{K_c T_s}{\tau_I} e_{k-1}, u_{low}), u_{high}) \tag{5.7a}$$

$$u_k = u_{k-1} + K_c(e_k - e_{k-1}) + \frac{K_c T_s}{\tau_I} e_k \tag{5.7b}$$

5.4 Results and Discussions

For each of the two case studies, two models were trained: (i) a non-hybrid LSTM-AE model (*Method 1*), and (ii) a hybrid LSTM-AE model. From the trained hybrid LSTM-AE model, two process monitoring approaches were developed: one using H^2 and SPE (*Method 2*), and another using H^2 and $SPE^{augmented}$ (*Method 3*) as process monitoring metrics. Therefore, three different process monitoring methods were compared for each

Table 5.4: Details of various fault types in the TE case study.

Fault No.	**Fault description**	**Time range**	**Fault type**
IDV (1)	Stream 4 A/C ratio (with constant B content)	[5, 20]	Step change
IDV (2)	Stream 4 composition of B (with constant A/C ratio)	[5, 20]	Step change
IDV (3)	Temperature stream 2	[5, 20]	Step change
IDV (4)	Reactor cooling water temperature	[5, 20]	Step change
IDV (5)	Condenser cooling water temperature	[5, 20]	Step change
IDV (6)	Stream 1 loss	[5, 20]	Step change
IDV (7)	Stream 4 reduced availability	[5, 20]	Step change
IDV (8)	Stream 4 composition of A, B, and C	[5, 20]	Random variation
IDV (9)	Temperature stream 2	[5, 20]	Random variation
IDV (10)	Temperature stream 4	[5, 20]	Random variation
IDV (11)	Reactor cooling water temperature	[5, 20]	Random variation
IDV (12)	Condenser cooling water temperature	[5, 20]	Random variation
IDV (13)	Reaction kinetics	[5, 20]	Slow drift
IDV (14)	Reactor cooling water valve	[5, 20]	Sticking
IDV (15)	Condenser cooling water valve	[5, 20]	Sticking
IDV (16)	Variation coefficient of steam supply	[5, 20]	Unknown
IDV (17)	Variation coefficient of reactor heat transfer	[5, 20]	Unknown
IDV (18)	Variation coefficient of condenser heat transfer	[5, 20]	Unknown
IDV (19)	Unknown	[5, 20]	
IDV (20)	Unknown	[5, 20]	
IDV (21)	Temperature stream 1	[5, 20]	Random variation
IDV (22)	Temperature stream 3	[5, 20]	Random variation
IDV (23)	Stream 1 pressure	[5, 20]	Random variation
IDV (24)	Stream 2 pressure	[5, 20]	Random variation
IDV (25)	Stream 3 pressure	[5, 20]	Random variation
IDV (26)	Stream 4 pressure	[5, 20]	Random variation
IDV (27)	Reactor cooling water pressure	[5, 20]	Random variation
IDV (28)	Condenser cooling water pressure	[5, 20]	Random variation

case study in terms of specificity, sensitivity, and balanced accuracy. The dataset was divided into three subsets: (i) a training dataset used for model training and weight optimization, (ii) a validation dataset used to fine-tune hyperparameters of the LSTM-AE and to evaluate the methodology's performance on normal processes, and (iii) a testing dataset used to evaluate the methodology's effectiveness in monitoring faulty scenarios. Several hyperparameters, such as the number of layers, units per layer, and learning rate, were tuned to optimize the loss function on the validation dataset. The search for hyperparameters' values was implemented using Python's Keras-tuner within a defined grid of parameters which included the number of encoder layers, the number of units within

Table 5.5: Parameters of the PI controllers in the Tennessee Eastman plant process.

Loop	**Controlled variable (y)**	**Manipulated variable (u)**	$\mathbf{K_c}$	$\tau_\mathbf{I}$**(min)**	$\mathbf{y^{sp}}$	**Eq.**
1	XMEAS (1)	XMV (3) $\in [0, 100]$	0.01	0.001	$F_p r_1$	5.7a
2	XMEAS (2)	XMV (1) $\in [0, 100]$	1.6e-6	0.001	$F_p r_2$	5.7a
3	XMEAS (3)	XMV (2) $\in [0, 100]$	1.8e-6	0.001	$F_p r_3$	5.7a
4	XMEAS (4)	XMV (4) $\in [0, 100]$	0.003	0.001	$F_p r_4$	5.7a
5	XMEAS (10)	XMV (6) $\in [0, 100]$	0.01	0.001	$F_p r_5$	5.7a
6	XMEAS (14)	XMV (7) $\in [0, 100]$	4e-4	0.001	$F_p r_6$	5.7a
7	XMEAS (17)	XMV (8) $\in [0, 100]$	4e-4	0.001	$F_p r_7$	5.7a
8	XMEAS (17)	$F_p \in [-30, 30]$	3.2	2	22.89	5.7a
9	XMEAS (15)	Ratio in loop 7, $r_7 \in [0, 100]$	-2e-4	200	50	5.7a
10	XMEAS (12)	Ratio in loop 6, $r_6 \in [0, 100]$	-1e-3	200	50	5.7a
11	XMEAS (8)	SepTempSP[a], $\in [0, 120]$	0.8	60	65	5.7a
12	XMEAS (7)	Ratio in loop 5, $r_5 \in [0, 100]$	-1e-4	20	2800	5.7a
13	XMEAS (40)	$E^b_{adj} \in [-\inf, \inf]$	-0.4	100	53.8	5.7a
14	yA[c]	Ratio in loop 1, r_1	2e-4	60	63.1373	5.7b
15	yAC[d]	$r_1 + r_4$	3e-4	120	51	5.7b
16	XMEAS (9)	XMV (10) $\in [0, 100]$	-8.0	7.5	122.9	5.7a
17	XMEAS (11)	XMV (11) $\in [0, 100]$	-4.0	15.0	SepTempSP	5.7a

[a] SepTempSP is the setpoint of the separator temperature (Setpoint of loop 17).
[b] E_{adj} is the adjustment to the molar feed rate of E.
[c] $y_A = 100\text{XMEAS}(23)/y_{AC}$
[d] $y_{AC} = \text{XMEAS}(23) + \text{XMEAS}(25)$

layers, and learning rates.

5.4.1 Results in the IndPenSim case study

Two LSTM-AE models were trained to monitor the IndPenSim case study, each utilizing a distinct loss function corresponding to either a non-hybrid LSTM-AE model (*Method 1*) or a hybrid LSTM-AE model. As mentioned earlier, two process monitoring models have been developed from the hybrid LSTM-AE model: *Method 2* and *Method 3*. Following the validation procedure outlined above, the hyperparameters of the non-hybrid LSTM-AE model were set as follows: layer unit numbers of 18, 12, 18, and a learning rate of 0.01. In order to ensure a fair comparison between the hybrid LSTM-AE model and the non-hybrid LSTM-AE model (*Method 1*), the number of units in the hidden layers and the learning rate were set to be identical for both models. It will be shown below that

while the number of hidden units is kept the same for the non-hybrid and hybrid models, the hybrid model will require significantly fewer input and output units as compared to the non-hybrid model. Subsequently, both models were evaluated in terms of sensitivity, specificity, and balanced accuracy for both the validation dataset and the testing dataset, which encompassed six fault scenarios. Figure 5.5, illustrates the hybrid LSTM-AE model's ability to reconstruct process variables for one validation batch. Specifically, the initial 16 variables undergo direct reconstruction from the encoder part of the LSTM-AE model, whereas the last two variables (coolant flow rate (F_c) and base flow rate (F_b)) are reconstructed via the controller correlation, utilizing the reconstructed pH and temperature ($\hat{pH}$ and $\hat{T}$), respectively. A summary of the performance of three different methods for monitoring the IndPenSim process in terms of these metrics is provided in Table 5.6. Overall, the hybrid model incorporating H^2 and $SPE^{augmented}$ (*Method 3*) demonstrates superior performance, achieving a balanced accuracy of 82.93% in the testing datasets, compared to the non-hybrid LSTM-AE (*Method 1*) with a balanced accuracy of 81.21% and the hybrid model utilizing only H^2 and SPE (*Method 2*) with a balanced accuracy of 79.52%, for monitoring all six different types of faults. Compared to *Method 1*, *Method 3* exhibits significant improvements in balanced accuracy for faults 3, 4, and 5, with increases of over 7%, over 21%, and 1.7%, respectively. It is found that this significant improvement is mostly because of the improvement in sensitivity, which is the ability of the model to correctly identify faulty samples. For example, fault 3, which poses a particular challenge due to its similarity to normal operation conditions compared to other faults, exhibited an improvement of over 13% in sensitivity when utilizing the proposed hybrid model (*Method 3*) instead of the non-hybrid LSTM-AE model. Fault 4, associated with a base flow rate disturbance as detailed in Table 5.1, exhibits the most substantial enhancement in sensitivity. This improvement surpasses 43% compared to the non-hybrid LSTM-AE model, attributable to the utilization of the proposed hybrid monitoring model. Notably, this enhancement is observed due to one of the two controllers in this case study leveraging the base flow rate to regulate the pH of the fermenter. Fault 5, which pertains to the coolant

flow rate and its impact on the temperature controller, demonstrates a slight improvement of over 3% when employing the hybrid model in comparison to the non-hybrid LSTM-AE model. Figure 5.6 presents the process monitoring metrics H^2 and SPE for faults 4 and 5, which are associated with the controllers. In summary, the most significant improvement is observed in correctly detecting faulty samples, with a slight enhancement in correctly identifying normal samples. Furthermore, the hybrid model, requiring 2922 parameters, is more compact than the non-hybrid LSTM-AE model, which involves 2800 parameters. This reduction in the number of parameters not only diminishes the risk of over-fitting but also reduces the requirement for substantial training data and computational resources. Additionally, the comparison between the two hybrid models highlights the advantage of incorporating the proposed $SPE^{augmented}$, which includes the error associated with the reconstructions of control actions, over SPE alone, which solely considers the reconstruction error associated with the LSTM-AE variables. In the hybrid model architecture, the metric of H^2 is directly influenced by the network's latent variables which are trained based on the use of the controllers' related reconstruction errors during training. In the case of $SPE^{augmented}$, its information is directly related to the controllers' equations since it includes the error between the measured manipulated variables and the reconstructed manipulated variables using the controllers' equations. It is important to notice that the exact PID controllers including their saturation limits are numerically implemented outside the neural network architecture. As shown in Figure 5.5, in the Penicillin case study, F_b and F_c reach their saturation levels when at zero. Both PID controllers are numerically implemented to account for hard bounds in their functions and to compensate for such bounds with the clipping strategy. While certain faults exhibit better detectability based on H^2, others are more discernible via SPE. The reason some faults yield superior results without the inclusion of $SPE^{augmented}$ is due to H^2 playing a more significant role in their detection as compared to SPE. Conversely, for faults that rely more on SPE, employing $SPE^{augmented}$ instead of solely relying on SPE leads to improved detection outcomes.

Table 5.6: Summary of the performance of three different methods for monitoring the IndPenSim process.

	Sensitivity: True Positive Rate			Specificity: True Negative Rate			Balanced Accuracy		
Dataset	**Method 1**[a]	**Method 2**[b]	**Method 3**[c]	**Method 1**	**Method 2**	**Method 3**	**Method 1**	**Method 2**	**Method 3**
Training	NaN	NaN	NaN	97.63	97.93	97.25	97.63	97.93	97.25
Validation	NaN	NaN	NaN	97.91	97.78	97.84	97.91	97.78	97.84
Testing: Fault (1)	100.0	100.0	100.0	82.60	81.81	84.18	91.30	90.91	92.09
Testing: Fault (2)	100.0	84.28	100.0	96.24	94.97	96.37	98.12	89.62	98.19
Testing: Fault (3)	38.03	38.52	51.72	64.32	64.37	65.92	51.17	51.47	58.82
Testing: Fault (4)	51.84	80.19	94.89	95.26	95.0	95.43	73.55	87.81	94.94
Testing: Fault (5)	68.95	68.73	72.25	96.11	89.24	96.23	82.48	79.09	84.18
Testing: Fault (6)	81.06	77.11	81.16	45.86	48.52	49.95	63.46	62.81	65.56
Testing: Average	75.89	72.33	80.63	82.26	80.93	83.38	81.21	79.52	82.93

[a] Method 1: non-hybrid LSTM-AE with $\{H^2, SPE\}$ monitoring metrics.
[b] Method 2: Hybrid LSTM-AE with $\{H^2, SPE\}$ monitoring metrics.
[c] Method 3: Hybrid LSTM-AE with $\{H^2, SPE^{augmented}\}$ monitoring metrics.

5.4.2 Results in the Tennessee Eastman case study

We developed two LSTM-AE models for monitoring the TE process: (i) a non-hybrid LSTM-AE model (*Method 1*) and (ii) a hybrid LSTM-AE model (*Methods 2 and 3*). We also investigate the benefits of utilizing the proposed $SPE^{augmented}$ metric compared to using SPE alone. For this case study, the architecture of the non-hybrid LSTM-AE model is configured with layers consisting of 50, 32, and 50 units, with a learning rate set to 0.1. The hybrid LSTM-AE architecture reduces the model's parameters from 20,594 to 19,145, representing a reduction of slightly over 7%.

Similar to the previous section, we compared the performance of all three methods in terms of sensitivity, specificity, and balanced accuracy. Figure 5.7 depicts the reconstruction performance of the proposed hybrid LSTM-AE model across the manipulating variables of the TE process for a validation run not included in the training dataset. The reconstruction of the first 9 variables relies on the controllers' correlations, as detailed in Table 5.5, while the remaining variables are reconstructed using the decoder network of the model. Table 5.7 provides a comparison of the three methods employed in monitoring the TE process. Notably, both the training and validation datasets show high specificity rates across all methods, indicating a robust ability to identify normal operating conditions. However, distinctions between the methods become apparent during the testing phase.

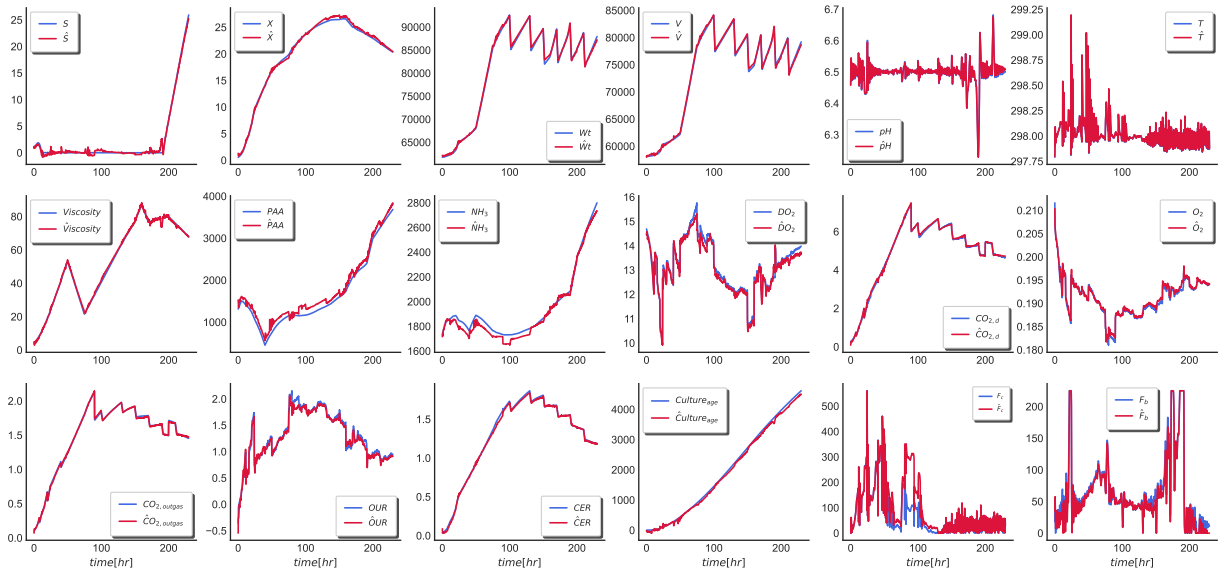


Figure 5.5: IndPenSim variables and their reconstruction by the trained hybrid LSTM-AE model for a validation batch.

For instance, in the testing scenarios involving fault injections, *Method 3* consistently outperforms both *Method 1* and *Method 2* in terms of sensitivity, which measures the model's ability to accurately detect faults. *Method 3*, the hybrid LSTM-AE model incorporating $SPE^{augmented}$, demonstrates superior performance in detecting various fault types, showcasing its effectiveness in distinguishing between normal and faulty conditions.

As previously stated, 28 testing runs were generated using the TE simulator code, employing identical initial conditions and the same seed number used for introducing noise to the process model. Each fault occurred after 5 hours from the start of the simulation and persisted until the simulation ended at 20 hours. Since specificity is an indicator of detecting normal samples as normal and the operation is normal for the first 5 hours only, each method results in the same specificity rate for all 28 faulty scenarios. In terms of specificity that reflects the model's ability to accurately detect normal samples, *Method 3* (hybrid LSTM-AE model with $SPE^{augmented}$) surpasses *Method 1* (non-hybrid LSTM-AE

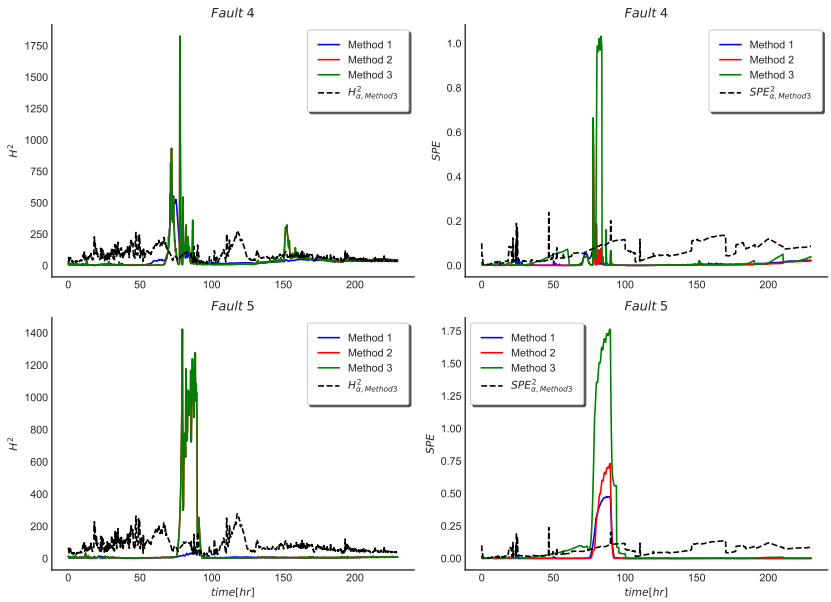


Figure 5.6: Plots of H^2, SPE, and their corresponding upper control limits with Method 3 for faults 4 and 5, which are associated with the controllers.

model) by 5.25% and *Method 2* (hybrid LSTM-AE model with SPE alone) by 8.41%.

Significant improvements are observed by utilizing *Method 3* compared to both *Method 1* and *Method 2* for faults IDV (4), IDV (7), IDV (24), and IDV (27). These instances highlight the effectiveness of the hybrid LSTM-AE model with the loss function $SPE^{augmented}$ in detecting and responding to specific fault scenarios, resulting in enhanced performance metrics compared to alternative monitoring methods.

For example, for Fault IDV (4) that involves a substantial disturbance in the process, *Method 3* exhibits a significantly higher sensitivity of 94.89%, significantly outperforming both *Method 1* and *Method 2*. This notable improvement emphasizes the advantage of incorporating $SPE^{augmented}$ into the monitoring framework, as it enables the model to capture and respond to deviations in control actions that may precede or indicate the onset of faults. By leveraging the additional information provided by $SPE^{augmented}$, the hybrid model demonstrates superior fault detection capabilities, thereby enhancing overall process

reliability and safety. Similarly, in Fault IDV (7), IDV (24), and IDV (27), *Method 3* consistently achieves higher sensitivity accuracy compared to *Method 1* and *Method 2*. These results reflect the robustness and efficacy of the hybrid LSTM-AE model with $SPE^{augmented}$ in identifying anomalous events and deviations from normal operation, even in complex and dynamic industrial environments.

Indeed, the enhanced sensitivity accuracy of the hybrid LSTM-AE model with $SPE^{augmented}$ is particularly notable in cases where anomalous events are relatively rare, such as in fault IDV (3), IDV (5), IDV (9), IDV (15), IDV (16), IDV (21), IDV (22), IDV (23), and IDV (28). These scenarios represent instances where the system experiences deviations from normal operation that occur infrequently but still have a significant impact on the process. In such cases, the hybrid model's incorporation of $SPE^{augmented}$ proves advantageous due to its ability to capture relatively small deviations in the error between the measured and reconstructed control actions, providing additional information that enhances the model's sensitivity to anomalous events. By considering not only the reconstruction error of process variables but also the error associated with control actions, the model becomes more adept at identifying deviations that may be indicative of impending faults or abnormal conditions. This heightened sensitivity accuracy is essential in industrial settings where the occurrence of rare events can have significant implications for process efficiency, product quality, and safety. By enabling early detection and intervention in such scenarios, the hybrid LSTM-AE model with $SPE^{augmented}$ offers enhanced predictive capabilities. Figure 5.8 depicts the process monitoring metrics H^2 and SPE for faults IDV (2), IDV (3), and IDV (7) across the three methods. Another benefit of the proposed hybrid LSTM-AE model with $SPE^{augmented}$, as shown in Figure 5.8, is that the SPE metric of *Method 3* ($SPE^{augmented}$) exhibits a more significant violation of the upper control limit when the faults are injected. This indicates greater observability of abnormal events.

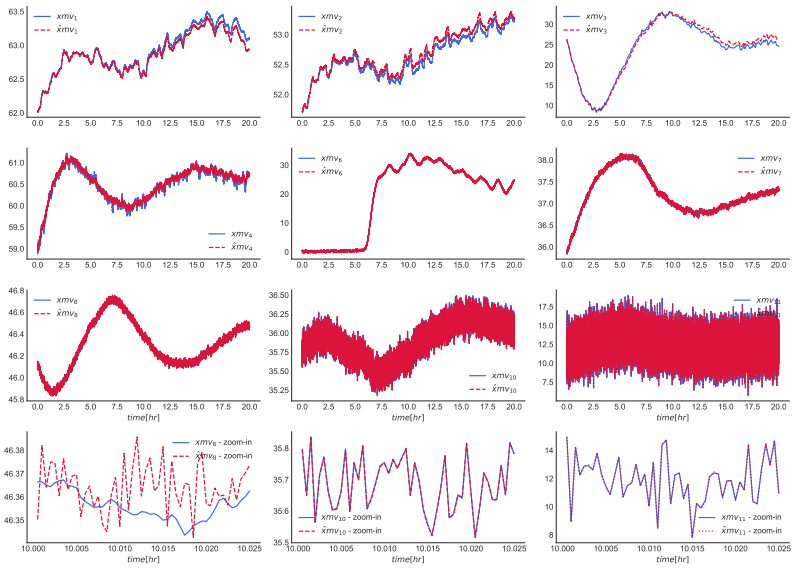


Figure 5.7: Tennessee Eastman manipulating variables and their reconstructions by the trained hybrid LSTM-AE model for a validation run.

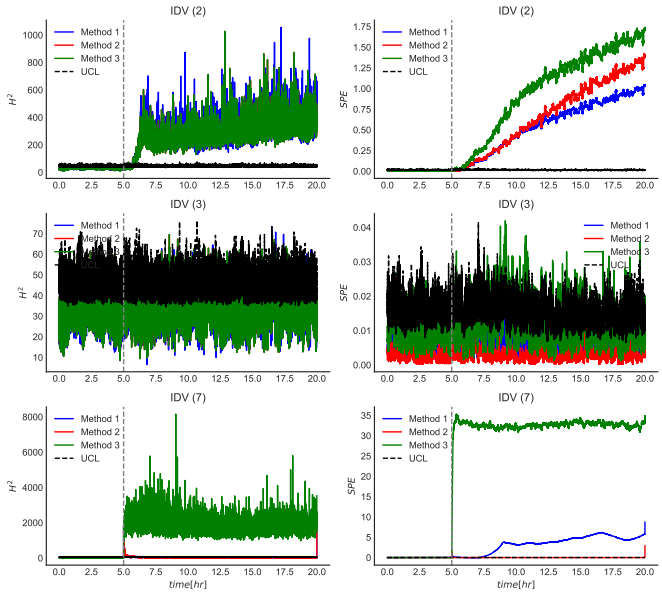


Figure 5.8: Plots of H^2, SPE, and their corresponding upper control limits with Method 3 for faults IDV (2), IDV (3), and IDV (7).

Table 5.7: Summary of the performance of three different methods for monitoring the Tennessee Eastman process.

	Sensitivity: True Positive Rate			**Specificity: True Negative Rate**			**Balanced Accuracy**		
Dataset	**Method 1**[a]	**Method 2**[b]	**Method 3**[c]	**Method 1**	**Method 2**	**Method 3**	**Method 1**	**Method 2**	**Method 3**
Training	NaN	NaN	NaN	85.84	85.32	86.36	85.84	85.32	86.36
Validation	NaN	NaN	NaN	85.77	85.20	85.98	85.77	85.20	85.98
Testing: IDV (1)	99.64	99.46	99.76	89.28	86.12	94.53	94.46	92.79	97.16
Testing: IDV (2)	98.58	95.75	98.87	89.28	86.12	94.53	93.93	90.94	96.70
Testing: IDV (3)	23.09	15.14	29.21	89.28	86.12	94.53	56.18	50.63	61.87
Testing: IDV (4)	68.91	12.0	100.0	89.28	86.12	94.53	79.10	49.06	97.26
Testing: IDV (5)	9.08	4.44	13.01	89.28	86.12	94.53	49.18	45.28	53.77
Testing: IDV (6)	100.0	99.96	100.0	89.28	86.12	94.53	94.64	93.04	97.27
Testing: IDV (7)	87.50	33.54	100.0	89.28	86.12	94.53	88.39	59.83	97.26
Testing: IDV (8)	94.97	95.54	96.0	89.28	86.12	94.53	92.12	90.33	95.26
Testing: IDV (9)	17.29	10.44	29.29	89.28	86.12	94.53	53.28	48.28	61.91
Testing: IDV (10)	78.52	81.05	84.66	89.28	86.12	94.53	83.90	83.59	89.59
Testing: IDV (11)	98.23	91.38	98.15	89.28	86.12	94.53	93.76	88.75	96.34
Testing: IDV (12)	45.17	27.01	53.85	89.28	86.12	94.53	67.23	56.56	74.19
Testing: IDV (13)	82.68	80.47	85.42	89.28	86.12	94.53	85.98	83.29	89.98
Testing: IDV (14)	99.70	99.72	99.89	89.28	86.12	94.53	94.49	92.92	97.21
Testing: IDV (15)	13.45	5.47	13.81	89.28	86.12	94.53	51.36	45.79	54.17
Testing: IDV (16)	12.03	2.99	13.85	89.28	86.12	94.53	50.65	44.55	54.19
Testing: IDV (17)	90.36	89.41	90.65	89.28	86.12	94.53	89.82	87.76	92.59
Testing: IDV (18)	81.71	81.48	82.67	89.28	86.12	94.53	85.49	85.38	88.60
Testing: IDV (19)	82.01	79.0	97.99	89.28	86.12	94.53	85.65	82.56	96.26
Testing: IDV (20)	91.05	88.57	91.29	89.28	86.12	94.53	90.16	87.34	92.91
Testing: IDV (21)	12.49	4.72	12.65	89.28	86.12	94.53	50.88	45.42	53.59
Testing: IDV (22)	17.63	9.98	31.78	89.28	86.12	94.53	53.45	48.05	63.15
Testing: IDV (23)	10.37	5.05	14.32	89.28	86.12	94.53	49.83	88.12	54.43
Testing: IDV (24)	85.69	81.52	97.45	89.28	86.12	94.53	87.48	83.82	95.99
Testing: IDV (25)	91.82	30.77	92.02	89.28	86.12	94.53	90.55	58.44	93.27
Testing: IDV (26)	78.48	16.81	80.45	89.28	86.12	94.53	83.88	51.46	87.49
Testing: IDV (27)	70.76	65.77	91.34	89.28	86.12	94.53	80.02	75.95	92.93
Testing: IDV (28)	13.64	6.47	14.82	89.28	86.12	94.53	51.46	46.29	54.67
Testing: Average	62.67	50.50	68.33	89.28	86.12	94.53	82.63	77.22	87.98

[a] Method 1: Non-hybrid LSTM-AE with $\{H^2, \text{SPE}\}$ monitoring metrics.
[b] Method 2: Hybrid LSTM-AE with $\{H^2, \text{SPE}\}$ monitoring metrics.
[c] Method 3: Hybrid LSTM-AE with $\{H^2, \text{SPE}^{\text{augmented}}\}$ monitoring metrics.

5.5 Conclusion

This study introduces hybrid LSTM-AE models that leverage process controllers' exact equations for monitoring both IndPenSim and TE processes. For the hybrid model, we proposed two loss functions; one that does not include the reconstruction error of the manipulated variables and an augmented loss function that includes that reconstruction error. Then, we compare three methods regarding their sensitivity to faults, specificity, and balanced accuracy: (i) single LSTM-AE with H^2 and SPE, (ii) hybrid LSTM-AE with H^2 and SPE, and (iii) hybrid LSTM-AE with H^2 and $SPE^{augmented}$. Our findings demonstrate that employing the hybrid LSTM-AE architecture using H^2 and $SPE^{augmented}$

significantly improves process monitoring efficacy across all metrics, particularly sensitivity as compared to methods that rely on a non-hybrid model or a hybrid model with SPE without augmentation.

This improvement primarily stems from the fact that certain faults are strongly correlated to controlled variables' closed-loop responses. Incorporating these correlations into the loss function of the model enhances the detection of these specific faults. Moreover, the utilization of $SPE^{augmented}$, computed from the augmented reconstruction error matrix that combines the reconstruction errors of both LSTM-AE variables and manipulating variables, demonstrates superior efficacy in detecting various types of faults compared to using SPE alone, which is derived solely from the reconstruction error matrix of LSTM-AE variables.

In the IndPenSim case study, faults 4 and 5, which are linked to the controllers, exhibit greater detectability when employing the hybrid LSTM-AE model with $SPE^{augmented}$. Similarly, in the TE process, the hybrid process monitoring approach not only demonstrates higher accuracy in detecting significantly large faults but also exhibits greater efficacy in cases where anomalous events are relatively rare, e.g. faults 3 or 5 in the TE case study. Additionally, the hybrid LSTM-AE model features fewer parameters to tune compared to a non-hybrid LSTM-AE model with the same number of hidden layers and units. This reduction in AE network's units with the corresponding reduction in training parameters reduces the risk of overfitting and lowers the computational costs.

Chapter 6

Graph Neural Network Representation of State Space Models of Metabolic Pathways

Overview[1]

In this chapter a novel Metabolic Graph Neural Network (MGNN) model is proposed for simulating the dynamic behavior of metabolites involved in oxidative stress metabolic pathways in a bacterial cell culture. The developed MGNN model is trained and validated with in-silico data generated from the mechanistic model. By using the a priori known metabolic network, the proposed MGNN model effectively reduces the overfitting issue as compared to a fully connected network that does not uses the metabolic network knowledge. The MGNN exhibits a superior fit for both training and testing datasets. The proposed MGNN

[1]Adapted from Aghaee, Mohammad, et al. "Graph Neural Network Representation of State Space Models of Metabolic Pathways." doi.org/10.1016/j.ifacol.2024.08.380

is highly interpretable since it efficiently computes the relevance of each metabolite on any other metabolite by applying gradient computation and back-propagation operations to the neural network. The proposed model is also shown to be useful for fault detection.

6.1 Introduction

With the development of improved computer technology and the availability of large amounts of biological data, accurate modeling of biochemical processes has become increasingly important to leveraging these data to predict biological systems' behavior.

The traditional kinetic modelling (KM) approach that has been widely used for modelling dynamic bio-processes is based on the formulation of mass balances for key metabolites. For example, [Lei et al., 2001] proposed a continuous mechanistic model for pyruvate metabolism in *saccharomyces cerevisiae* fermentation. The model presents the dynamic of eight key metabolites of pyruvate metabolism in *saccharomyces cerevisiae* with 12 reactions which all were modelled based on Michaelis–Menten kinetics with respect to a given substrate and with a first order dependency on the active biomass pool. These kinetic models rely on explicit functional relationships connecting the rate of change of metabolites and the enzyme kinetics involved in each reactions. Michaelis–Menten kinetics is the most commonly used kinetic rate expression although the true mechanistic kinetic rate law for each specific reaction is unknown a priori for most enzymes. Because each such kinetic expression involves a number of parameters, overall KMs require a large number of parameters that must be calibrated via nonlinear optimization based on generally noisy and scarce data. The problem becomes more pronounced in large metabolic networks due to the large number of metabolites and reactions involved. Generally, the structure of these models is only loosely based on prior knowledge about the interactions between metabolites.

Dynamic metabolic flux modelling, especially the dynamic flux balance analysis (DFBA) is an alternative approach that uses detailed knowledge of the metabolic network of re-

actions. [Mahadevan et al., 2002] described two different formulations of DFBA, which incorporate rate of change of flux constraints. The resulting DFBA models were able to describe the dynamics of diauxic growth of *Escherichia coli* on glucose and acetate as well as the dynamics of key metabolites' concentrations. The DFBA model of [Mahadevan et al., 2002] consists of four ordinary differential equations for extracellular glucose, acetate, oxygen and biomass. DFBA are generally formulated by a constrained optimization problem where a biological objective, e.g. growth rate is maximize subject to stoichiometric and other constraints. Although this approach has the potential to reduce the number of model parameters as compared to the traditional kinetic modelling approach, stoichiometric constraints alone are not sufficient to fully describe dynamic behaviour as they ignore enzyme kinetics limitations. To address this problem, additional kinetic rate constraints are added to the constrained optimization problem. Since these additional constraints involve tuning parameters, a crucial challenge in DFBA is to identify a minimal number of kinetic constraints to fit data while avoiding over-fitting. This search often requires the solution of mixed integer optimization problems that are difficult to solve. Also, DFBA and KM are less suitable for monitoring due to model error unless the latter will be accounted for as in e.g. [Du et al., 2015].

As an alternative to the aforementioned KM and DFBA approaches, data-driven models such as machine learning models can be used when large datasets and computing power. [Costello and Martin, 2018] developed a machine learning algorithm to predict pathway dynamics in an automated fashion. For example, a Python code called TPOT (Tree-based Pipeline Optimization Tool) which relies on a Genetic optimization algorithm is used to automatically select the most suitable model among different candidates and to choose among different preprocessing algorithms that are included in the scikit-learn Python package. This method was tested on Limonene pathways and compared to the Michaelis-Menten kinetic model with 10 ordinary differential equations. However, the results exhibited considerable offset with respect to experimental data. The lack of fit was explained by the occurrence of highly nonlinear behaviour combined with scarce data for training.

Deep learning (DL) models can be used to effectively reconstruct highly nonlinear trajectories but there is always the possibility of data overfitting due to the high number of parameters in DL models. In case of over-fitting, this can be reduced by either increasing the amount of data or by pruning the neural network model to decrease the number of parameters. Data is generally limited and pruning of the network requires the use of empirical thresholds that are generally difficult to determine in the presence of noise. In this work we propose an alternative DL modelling approach where a priori knowledge about the metabolic network stoichiometry is used to constraint the structure of the neural network thus eliminating the need for empirical pruning methods.

This paper introduces Metabolic Graph Neural Networks (MGNN) as a novel approach for dynamic prediction of metabolites' concentrations and for detection and diagnosis of faults. In this special neural networks, the neurons contain information about metabolites and the network architecture is constrained based on a priori biological information about the network of metabolic reactions. The use of such constrained architecture is expected to lead to several benefits: i- overfitting of data will be avoided as compared to a fully interconnected network that does not use a priori information about the network of reactions, ii- by capitalizing on currently available powerful deep learning algorithms, this MGNN model is easier to tune as compared to other stocihiometry based approaches such as DFBA that require the solution of a challenging NLP, iii- the nonlinear form of the kinetic constraints in DFBA, e.g. Monod, Hill and other, must be decided a priori whereas MGNN does not require such choice , iv- the resulting MGNN can provide explainability since the neurons are associated to metabolites and thus the interconnection signals are associated to fluxes. The proposed methodology is tested on the oxidative stress metabolic network of B. Pertussis which is a bacteria used for the manufacturing of the whooping cough vaccine. The model is compared with a dynamic mechanistic model developed by [Vitelli et al., 2023] for the same biochemical system.

6.2 Proposed methodology

6.2.1 Graph Neural Networks

Processes involved a variety of phenomena that can be understood in terms of the relationships between the elements. A set of elements, and the connections between them, can naturally be expressed as a graph. In recent years, neural networks that operate on graph data (called graph neural networks, or GNNs) have been developed for a wide range of applications, such as antibacterial discovery [Stokes et al., 2020] or for quantum chemistry [Gilmer et al., 2017].

In the graph based modelling approach, the nodes of a graph represent objects or concepts, and the edges represent their relationships or interactions. For example, in the graph description of metabolic pathways, the nodes in the graph represent the metabolites and the connections between the nodes represent the reactions.

Typically, biochemical systems exhibit highly nonlinear behaviour due to the nonlinear kinetics regulating individual reactions among metabolites. On the other hand the connections between nodes in neural networks are also regulated by nonlinear activation functions but these can only represent simple nonlinear behaviour such as sigmoidal or semi-linear behaviour (Relu functions). Hence, single connections between two neurons can not capture the full nonlinear dependencies of the biochemical process. Instead, we propose in this work the use of one or more layers of neurons to connect between two neurons that are associated to individual metabolites. In addition to nonlinearity, the operation of bioprocesses in batch or perfusion modes, requires consideration of dynamic behaviour. To address the expected nonlinear dynamic behaviour a graph neural network model is proposed in this study as shown in Figure 6.1. This particular neural network is used for predicting state x_i at current time $k+1$ which is fed by those states which are connected to state x_i according to the graph edges at time k.

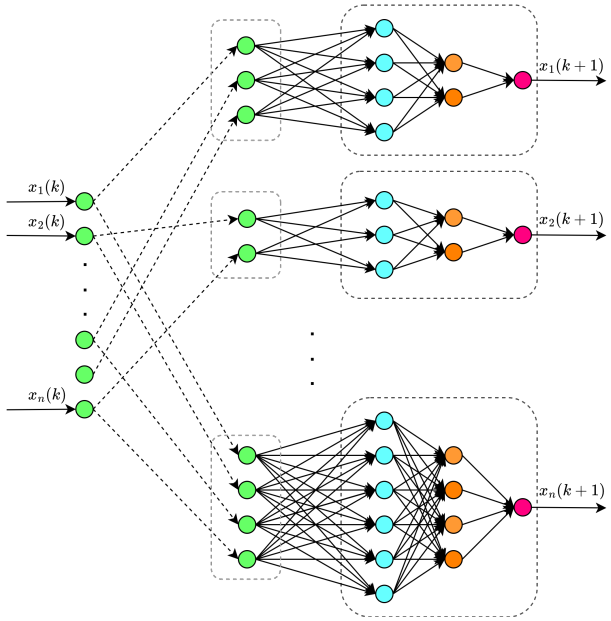


Figure 6.1: A schematic of graph neural networks (GNNs).

6.2.2 Oxidative Stress Metabolic Pathways

Oxidative stress metabolism in *Bordetella pertussis* fermentations involves a complex network of reactions occurring among various key molecules. At the heart of this process lies NADPH (Nicotinamide Adenine Dinucleotide Phosphate), a critical cofactor that plays a central role in countering the damaging effects of reactive oxygen species (ROS). When B. pertussis encounters ROS in the host environment, NADPH becomes crucial for maintaining redox balance and protecting the bacterium from oxidative harm. The motivation for focusing on the mechanism of oxidative stress is that it was found in previous studies by our group to be highly correlated with low productivity of antigens required in the formulation of the whooping cough vaccine.

NADPH serves as an essential reactant (co-factor) in various enzymatic reactions. It is involved in the reduction of NADP+ (the oxidized form of NADPH) into its reduced form, NADPH, through specific enzymatic reactions, [Vitelli et al., 2023]. In this context,

NADPH helps activate antioxidative enzymes like superoxide dismutase and catalase, which are crucial for neutralizing ROS, i.e. superoxide radicals and hydrogen peroxide into less harmful molecules, thus preventing cellular damage.

In response to oxidative stress, *Bordetella pertussis* shifts its metabolism to emphasize reactions related to generation and quenching of ROS. NADPH is a co-factor for anti-oxidative reactions catalyzed by superoxide dismutase and catalase. By diverting energy away from anabolic processes, e.g. cell growth, the bacterium uses NADPH to quench ROS.

This metabolic adaptation helps *Bordetella pertussis* strike a balance between growing and staying alive in the challenging host environment. By using NADPH and its antioxidative enzymes, the bacterium can thrive and successfully establish itself within the respiratory system despite the presence of harmful ROS.

6.2.3 Graph Neural Network for Oxidative Stress Metabolic Pathways

In this study, we have developed a MGNN model to represent the dynamic behavior of oxidative stress metabolic pathways. Figure 6.2 illustrates the pathways in the metabolic network of oxidative stress in *Bordetella pertussis* bacteria. Metabolites S_{ext}, S, A, B, C and A_{ext} represent extracellular glutamate, intracellular glutamate, NADPH, ROS, $NADP^+$ and extracellular NADPH respectively. A metabolic neural network is developed based on the presented metabolic network in order to predict the concentrations of metabolites over time. Also, a fully connected neural network is also developed for comparison with the metabolic neural network model in terms of performance and number of parameters. The training data have been synthetically generated using the mechanistic model developed by [Vitelli et al., 2023] which was fully trained and validated with experimental data. In practice, the proposed MGNN could be also directly trained with data. In this study, in

silico data for 20 batches are generated by considering different initial conditions corresponding to perturbations around the experimental initial conditions that were used in the actual experiments. The data was split into 3 data sets to be used for training, validation and testing respectively. The training set is used to train the weights of the network for a particular set of hyper-parameters, the validation set is used to find a best set of hyperparameters and the testing set is used to test the accuracy with data that is not used for neither training or validation. Hence, the overall calibration of the model involves successive iterations between training and validation steps from which a best combination of weights and hyperparameters is found. In total, fifteen batches were used in the training process, two batches in the validation process, and three batches in the testing process. The total fermentation time of each batch is 60 hours (120 samples per batch) and includes the 6 metabolites presented above plus biomass. Figure 6.3 shows the in silico dynamic profiles of 6 metabolites and biomass concentrations obtained with the mechanistic model in [Vitelli et al., 2023] during the *Bordetella pertussis* fermentation.

It is also worth mentioning that the reliability of the parameters in the MGNN is highly dependent on the quality of the data and the underlying model assumptions. In theory, if the model were perfect and an infinite amount of data were available, the parameters would be perfectly identified. However, in practice, the kinetic rates used in the model—such as those proposed [Vitelli et al., 2023]—are not necessarily true or universally applicable. The rates in that work were based on specific assumptions and experimental conditions, which may not fully capture the complexity of metabolic processes. Furthermore, the limited amount of experimental data available for developing the mechanistic model constraints the accuracy and generalizability of the MGNN parameters. As a result, while the MGNN provides valuable insights, it is important to acknowledge that the reliability of its parameters is influenced by the limitations of the data and assumptions used in its development.

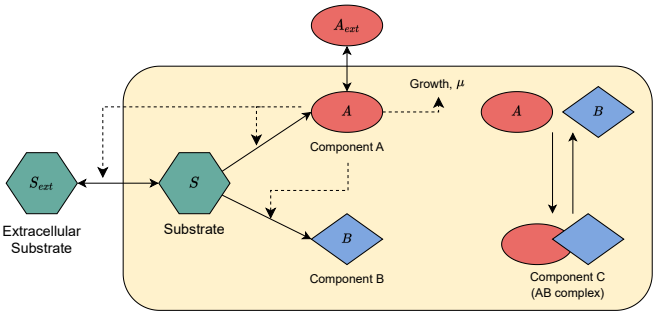


Figure 6.2: An illustration of the metabolic pathways involved in oxidative stress.

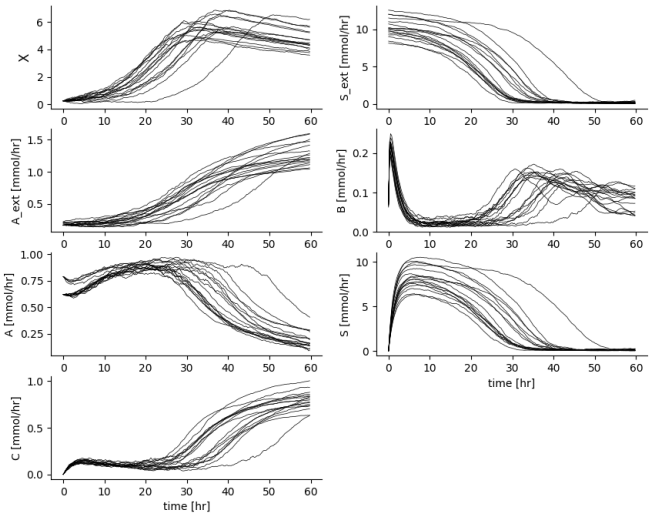


Figure 6.3: Synthetic data produced by the mechanistic model developed by Vitelli et al., 2023.

6.3 Results and discussion

Two different deep learning models were trained to predict metabolites' concentrations over time: i- the MGNN model, and ii- the associated fully connected neural network model. The hyperparameters considered for these models are: the number of layers, the number of units within each layer, and the learning rate. Using the validation data set, these parameters are tuned to minimize the loss function. The hyperparameter search is implemented using Python's Keras-tuner. To perform this search, a grid of hyperparameters is defined, for the number of hidden layers for correctly capturing nonlinear dynamics of metabolites' behavior = [1, 2, 3], number of neurons for each layer = [1, 2, 4, 8], and learning rate = [0.1, 0.01, 0.001, 0.0001]. Figure 6.4 illustrates the architecture of the metabolic graph neural network model that was developed for predicting the concentrations of different metabolites during oxidative stress in *Bordetella pertussis* fermentation. Figures 6.5 and 6.6 compare the dynamics of metabolite concentrations between the fully connected neural network and the mechanistic model in a training and in a testing batch respectively. As depicted in these figures, the fully connected neural network demonstrates a good fit for the training batch but a notably poor fit for the testing batch, indicating an overfitting issue in the fully connected neural network model. As previously mentioned, overfitting is caused by the high number of parameters in the fully connected neural network. However, Figure 6.7 exhibits a very good fit for the testing batch, addressing the overfitting issue due to the model's significantly lower number of parameters. A comparison between the MGNN model and the corresponding fully connected neural network model in terms of the mean square error is shown in Table 6.1.

Physical explainability (interpretability) of the network is a desired property of deep learning models. Also, the proposed MGNN model has the ability to quantify the effect of input variables on the outputs by back-propagation operations through the network. To interpret the MGNN we consider the gradients of the time derivative of each metabolite's concentration with respect to each metabolite's concentration that can be directly

obtained from the network from the difference between the metabolite's concentrations at two consecutive times k and k+1. The absolute values of the gradients are averaged over the fermentation duration to quantify the average impact of each metabolite on any other metabolite. Also, if the signs of the gradients are considered, the actual positive or negative correlation among metabolites can be inferred.

Figure 6.8 illustrates the absolute value of the impact of each metabolite on each other metabolite in the metabolic network. The figure accurately demonstrates that metabolites not connected to a specific metabolite in the metabolic graph exhibit no contributions in the change of that specific metabolite. Figure 6.9 also illustrate the positive or negative impact of various metabolites on the change of a particular metabolite.

For example, the first row of Figures 6.8 and 6.9, shows a positive impact from metabolites X (biomass), A (NADPH), and S (glutamate) on the change in biomass concentration which aligns with the mechanistic ODE equation developed for the dynamics of this metabolite:

$$\begin{cases} \frac{dX}{dt} = (\mu - D)X \\ \mu = \alpha F_A A \end{cases} \tag{6.1}$$

For A_{ext} (NADPH extracellular), Figure 6.9 displays a positive effect from X and A, in accordance with the ODE equation below:

$$\frac{dA_{ext}}{dt} = (K_m + D)AX \tag{6.2}$$

Eq. 6.2, indicates that the change dA_{ext} has a direct positive correlation with variables A and X. Correspondingly, according to the metabolic graph, A_{ext} either converts into intracellular A or accumulates outside the cell, aligning with the negative influence of A_{ext} on its own alteration, as determined by the MGNN model.

Similarly, Figure 6.8 and 6.9 shows positive influences of metabolites A, B, C, and X, as well and negative influence of S on the change of metabolite A (NADPH). These obser-

vations can be verified with the mechanistic ODE equation describing the concentration dynamics of this metabolite:

$$\frac{dA}{dt} = F_A A - k_p AB + k_m C - \mu A - K_m A \tag{6.3}$$

A (NADPH), S (glutamate), C ($NADP^+$), and X (biomass) contribute to the increase in B, while the consumption of metabolite B in producing C and degradation of B result in its decrease.

$$\frac{dB}{dt} = F_B A - k_p AB + k_m C - \mu B - d_B B \tag{6.4}$$

Regarding metabolite C ($NADP^+$), Figures 6.8 and 6.9 depict a positive influence from metabolites A, B, C, and S, and a negative influence from X. This observation can be again verified with the mechanistic model concerning the dynamics of this metabolite.

$$\begin{cases} \frac{dC}{dt} = k_p AB - k_m C - \mu C \\ F_A = \frac{(\frac{v}{1+S/K_{si}})S}{KX+S}(\frac{S}{K_t+S})^{1.5} \\ \mu = \alpha F_A A \end{cases} \tag{6.5}$$

Regarding metabolite S_{ext} (extracellular glutamate), only the intracellular glutamate (S) has the potential to positively impact the increase in S_{ext} concentration, as it can only be transported to the exterior of the cell. The concentrations of metabolites S_{ext}, X, and A at the previous time lead to a decrease in the concentration of S_{ext}. This can also be corroborated from the metabolic graph (Figure 6.2) and from the mechanistic model (Eq. 6.6):

$$\frac{dS_{ext}}{dt} = -\frac{\mu X}{Y_{X/S}} \tag{6.6}$$

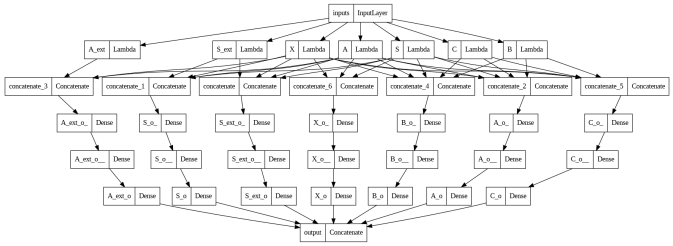


Figure 6.4: Architecture of the metabolic graph neural network for the oxidative stress metabolic network.

Table 6.1: Comparison of the Metabolic Neural Network model (MGNN) and the fully connected neural network (FCNN).

	MGNN	FCNN
MSE in the training dataset	3.6157e-04	8.2261e-05
MSE in the testing dataset	4.1534e-04	1.379e-01
Number of parameters	575	2247
Overfitting problem	No	Yes

Furthermore, the mechanistic model ODE (Eq. 6.7) confirms that the metabolites X, S, and A have positive impact and metabolite S at the previous time has a negative impact on the change of S.

$$\frac{dS}{dt} = -F_A A - F_B A + \alpha A(S_{ext} - S) - \mu S \tag{6.7}$$

An interesting application of the MGNN model is for fault detection. To assess this capability, a change (fault) was intentionally introduced into the media (S_{ext}) during the time interval $t = [20, 26]$ by increasing the concentration of this metabolite by 10. Based on an online fluorescence probe currently available in our lab, we assumed that the concentrations of biomass (X) and extracellular NADPH (A_{ext}) can be measured online. The MGNN model, initialized with the same conditions as normal batches, was simulated while forcing the predicted values of biomass (X) and extracellular NADPH (A_{ext}) to be equal to the measured values. Using the contribution plots' algorithms developed in [Aghaee et al.,

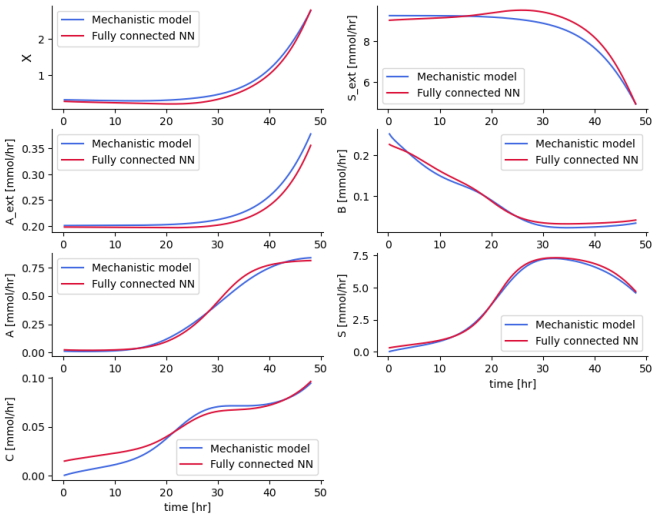


Figure 6.5: Comparison of the predicted concentrations of metabolites over time in a training batch using the developed fully connected neural network with the mechanistic model.

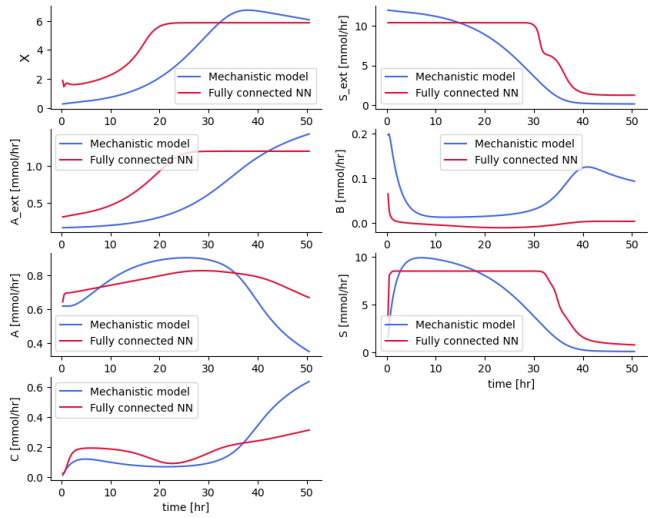


Figure 6.6: Comparison of the predicted concentrations of metabolites over time in a testing batch using the developed fully connected neural network with the mechanistic model.

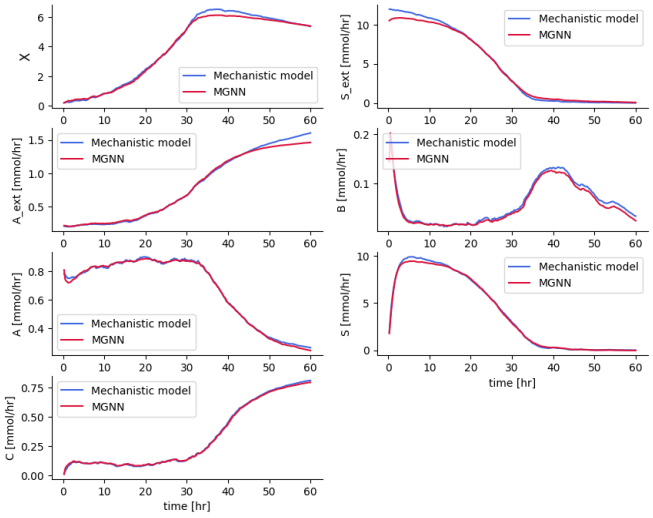


Figure 6.7: Comparison of the predicted concentrations of metabolites over time in a testing batch using the developed Metabolic Graph Neural Network (MGNN) with the mechanistic model.

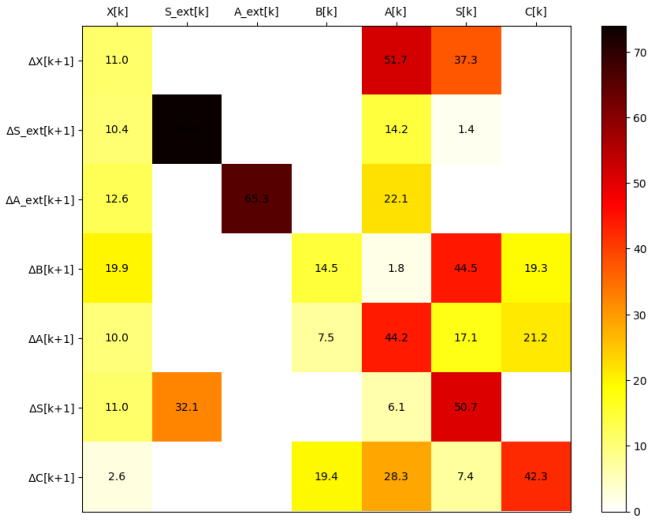


Figure 6.8: The relevance of metabolites at the previous time to the change of metabolites at the current time using the MGNN model.

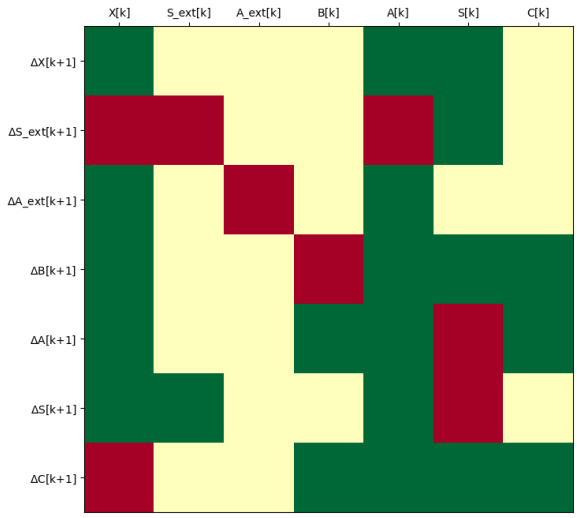


Figure 6.9: The positive or negative influence of each input on the output direction; Green (Positive), Red (Negative), Yellow (Neutral)

2023], the difference between the response of the MGNN model for the faulty batch and the average response for normal batches was back-propagated through the MGNN model to determine the contribution of different metabolites, as illustrated in Figure 6.10. These plots clearly show the detection of the change in nutrient concentration and its impact on the other metabolites.

6.4 Conclusion

This study proposes a novel Metabolic Graph Neural Network (MGNN) as a modelling tool for simulating the dynamic behavior of metabolites within a metabolic network. By incorporating prior knowledge of metabolic graph pathways, a neural network model with a significantly lower number of parameters is obtained thus leading to less over-fitting of data as compared to a fully connected NN that does not use prior metabolism knowledge. The developed MGNN model results in significantly better accuracy as compared to the

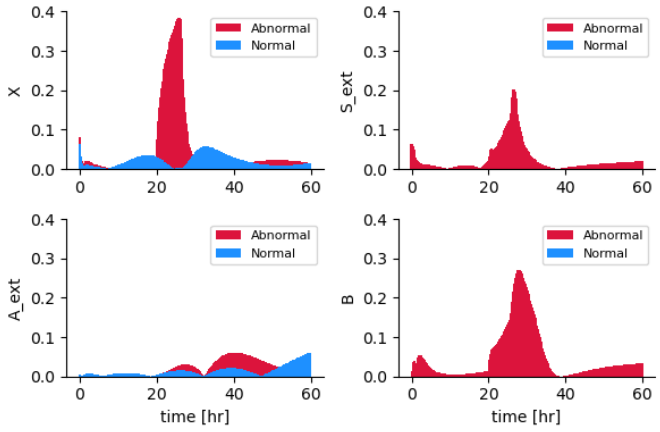


Figure 6.10: The contribution of different metabolites to detect the imposed fault in the media (S_{ext}) in $t = [20, 26]$ compared to their contributions in normal media.

fully connected in terms of the mean square errors.

Furthermore, the MGNN model is highly explainable since it comprises multiple sub-neural network models, each representing the dynamics of a specific metabolite. In contrast, the fully connected neural network model cannot be physically interpreted.

Moreover, the methodology can be used to generate a relevance matrix that reveals the influence of different metabolites on the change of a specific metabolite. The relevance matrix correctly predicts (positive or negative) the correlations among metabolites as verified by the mechanistic model that was used to generate the in silico data and by the metabolic graph associated with oxidative stress phenomena. The derivation of MGNN is found to be significantly faster than the development of a mechanistic model since the MGNN. While the mechanistic model requires a priori decisions regarding the kinetic expressions relating different metabolites, the MGNN does not require such a priori assumptions regarding the forms of the nonlinear kinetic terms relating the metabolites thus simplifying the development of the model. The proposed model is also shown to be useful for fault detection applications.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis proposes new approaches for the application of deep learning algorithms in chemical engineering with emphasis on bio-pharmaceutical processes. Different aspects of unsupervised learning based monitoring were tackled including diagnosis of process faults, detection with dynamic data and use of hybrid models to reduce over-fitting.

In Chapter 3, we investigated the effectiveness of three fault detection (FD) methods: Multiway Principal Component Analysis (MPCA), Multiway Partial Least Squares (MPLS), and Multiway Partial Least Squares Autoencoder (MPLS-AE) when applied to the Industrial Penicillin Simulator (IndPenSim). Our analysis primarily focused on comparing the fault detection rate (FDR) of these methods with varying control limits and objective functions for optimal hyper-parameter selection. The results demonstrated that MPLS generally outperformed MPCA in terms of FDR accuracy, both with static and dynamic control limits. Specifically, MPLS achieved an FDR of 78.26% with static limits compared to MPCA's 75.48%, and with dynamic limits, MPLS attained 84.09% versus MPCA's 83.20%.

The superiority of MPLS over MPCA can be attributed to the inclusion of the fermentation time as an indicative output variable, which resulted in a lower number of significant latent vectors and higher FDR accuracy. Furthermore, the use of dynamic control limits significantly improved FDR accuracy for both methods due to the increased ability to capture model nonlinearity and process variations over time. The FDR improvements observed with dynamic limits were significant (83.20% for MPCA and 84.09% for MPLS), as compared to the results obtained with static limits.

The use of a novel objective function that is directly oriented to the fault detection task, $J^{MPCA-MPLS,FDR}$, further enhanced the fault detection capabilities of both MPCA and MPLS models. The application of this objective function led to a considerable reduction in the number of principal components (nPCs) required, minimizing overfitting and improving FDR accuracy. For instance, using the novel objective function with static control limits, both MPCA and MPLS required only 2 PCs to achieve 81.83% and 81.99% FDR, respectively. It was shown that the models that were trained based on the traditional loss function required more PCs and resulted in lower FDRs thus highlighting the efficacy of our proposed objective function in optimizing model performance.

The best results were obtained with the MPLS-AE model, which combines the strengths of MPLS and Autoencoders to handle nonlinearities more effectively. The MPLS-AE model with the novel objective function and dynamic control limits achieved the highest FDR of 88.64%. This illustrates the potential of deep learning architectures in enhancing fault detection systems. Overall, our findings indicate that MPLS and its deep learning variant MPLS-AE, particularly when combined with dynamic control limits and a detection oriented objective function, offer superior fault detection capabilities in industrial process monitoring applications.

In Chapter 4, two LSTM-AE models were trained and evaluated for monitoring processes in two distinct case studies: IndPenSim and Sanofi. Following the results of the previous chapter, the models were optimized based on two criteria: minimizing MSE and

maximizing a detection oriented $\overline{FDR}$. The results demonstrated significant improvements in fault detection accuracy when using LSTM-AE models compared to traditional DPCA and previously developed AE models.

In the IndPenSim case study, the LSTM-AE model that was trained to maximize $\overline{FDR}$ achieved the highest fault detection rate, with a very high $\overline{FDR}$ of 93.86% on the test data set, outperforming both the DPCA model and the static AE model from previous research. This improvement highlights the advantage of capturing dynamic process behaviors. Contribution plots further validated the algorithm's ability in identifying relevant variables for each fault, enhancing the interpretability and diagnostic capabilities of the LSTM-AE approach.

The Sanofi case study presented additional challenges, such as lack of knowledge of faults that are available in simulators' studies and undefined regions of normal operation. Despite these challenges, the LSTM-AE models effectively identified low productivity batches, demonstrating the model's robustness in semi-supervised settings. The LSTM-AE model trained to maximize $\overline{FDR}$ correctly identified 86.05% of low-productivity batches as abnormal, confirming the critical impact of the fermentation step viz a viz downstream steps on overall process productivity. The significant correlation between the Kjeldahl productivity measure and the average violation of process monitoring metrics (D_b) underscores the model's ability to provide early fault detection, opening the possibility for implementing corrective actions.Overall, the findings emphasize the superiority of LSTM-AE models in fault detection and process monitoring, particularly when dynamic behaviors are crucial.

The methodology was further tested on Sanofi's lab-scale diphtheria process. The production of the diphtheria vaccine involves a meticulously controlled batch fermentation process. Despite Sanofi's efforts, variability in final productivity, measured by LF (Lateral Flow) antigen measurement, presented a significant challenge. Based on 11 process variables and the use of a multi-layer autoencoder (AE) model, the methodology demonstrated a high sensitivity in identifying low productivity batches. As illustrated in the study, the

AE model, combined with the novel cumulative D_b metric, effectively differentiated between high productivity (normal) and low productivity (faulty) batches. Being based on a cumulative record of statistical bound violations, the D_b metric provided a precise and timely indication of process deviations. The results indicated a consistent D_b accuracy between the training and validation datasets, with heightened performance in the testing dataset, underscoring the model's robustness. The monitoring metrics, H^2 and SPE, along with their upper control limits, facilitated the identification of deviations and potential faults, ensuring that the production process remained within acceptable operational conditions.

Chapter 5 presents a comparative analysis of non-hybrid and hybrid LSTM-AE models for process monitoring in two case studies: IndPenSim and the Tennessee Eastman (TE) process. The hybrid model is made of a combination of an empirical component corresponding to an LSTM-AE and the exact controllers' equations as the mechanistic component. Three different process monitoring methods were evaluated: a non-hybrid LSTM-AE model (Method 1), a hybrid LSTM-AE model with H^2 and SPE monitoring metrics (Method 2), and a hybrid LSTM-AE model with H^2 and $SPE^{augmented}$ monitoring metrics (Method 3). While the SPE loss involves only the reconstruction error of the LSTM-AE, the $SPE^{augmented}$ loss is comprised of both the reconstruction errors in the LSTM-AE and the reconstruction errors between the measured and predicted manipulated variables (control actions) with the controllers' equations. The results demonstrate that the hybrid LSTM-AE model incorporating H^2 and $SPE^{augmented}$ (Method 3) consistently outperforms the other methods in terms of balanced accuracy, sensitivity, and specificity in both case studies. Specifically, Method 3 achieved a balanced accuracy of 82.93% in the IndPenSim case study and demonstrated superior fault detection capabilities in the TE process, particularly for faults IDV (4), IDV (7), IDV (24), and IDV (27).

In the IndPenSim case study, Method 3 also showed significant improvements in detecting faults 3, 4, and 5, with enhancements in sensitivity of over 13%, 43%, and 3%,

respectively. The reduction in model parameters from 2800 in the non-hybrid LSTM-AE to 2922 in the hybrid model further underscores the efficiency of the hybrid approach, reducing the risk of overfitting and the need for extensive training data and computational resources.

For the TE process, the hybrid LSTM-AE model with $SPE^{augmented}$ exhibited significant improvements in sensitivity accuracy, particularly for fault scenarios involving substantial process disturbances.

Chapter 6 demonstrates the efficacy of the Metabolic Graph Neural Network (MGNN) model in predicting metabolites' concentrations over time under oxidative stress conditions in the fermentation of Bordetella pertussis. By comparing the MGNN model with a fully connected neural network, we have illustrated the superior performance of the MGNN in mitigating overfitting issues and providing more accurate predictions on testing data. The detailed examination of the hyperparameter tuning, achieved through Python's Keras-tuner, underscores the robustness of the MGNN's architecture.

The MGNN model's ability to interpret and quantify the influence of each metabolite on others is a significant advancement, providing a deeper understanding of the metabolic network's dynamics. This interpretability is crucial for validating the model's predictions against the mechanistic ODE models, as evidenced by the alignment of the MGNN's results with the mechanistic equations for various metabolites.

Additionally, the study explores the application of the MGNN model in fault detection. By introducing a fault in the nutrient concentration and analyzing the model's response, the MGNN demonstrated its capability to detect and quantify the impact of such changes. This ability is crucial for real-time monitoring and control of fermentation processes, potentially leading to improved process efficiency and product quality.

Overall, the MGNN model presents a promising tool for both predictive modeling and interpretability in metabolic networks, offering significant advantages over traditional neural network models. Model calibration with the MGNN approach was found to be much

simpler as compared to other biochemical modelling approaches currently used. Furthermore, the application of MGNN in fault detection further highlights its potential in industrial biotechnology applications. Future work could focus on extending this approach to other larger metabolic networks and integrating real-time data for continuous process monitoring and optimization.

7.2 Future Work

The research presented in this thesis has demonstrated the potential of deep learning algorithms in enhancing process monitoring, fault diagnosis, and modeling within the pharmaceutical industry. However, the presented approaches have also opened several avenues for further investigation. The following outlines some potential directions for future work:

1. **Transfer Learning for Cross-Process Applications:**
   Investigate the use of transfer learning techniques to apply models trained on one pharmaceutical process, e.g. MGNN, to other processes or products. This could enhance the versatility and applicability of the developed methodologies across different production lines.

2. **Incorporation of Advanced Process Control (APC) Strategies:**
   Explore the integration of advanced process control strategies with deep learning models. This could involve developing hybrid frameworks that combine APC techniques with LSTM-AE and MGNN models to optimize both process monitoring and control actions in real-time.

3. **Enhancement of Interpretability and Explainability:**
   Develop methods to enhance the interpretability and explainability of deep learning models, particularly in the context of fault diagnosis and metabolic pathway analysis. This could include creating more advanced contribution plots and leveraging

explainable AI techniques. This could further help industry to identify the sources of faults, e.g. deviations in media composition.

4. **Expansion to Multi-Omics Data:**
   Extend the application of deep learning models to incorporate multi-omics data (e.g., genomics, proteomics, and metabolomics) for a more comprehensive understanding of biopharmaceutical processes. This would require the development of multi-modal deep learning architectures capable of handling diverse data types.

5. **Robustness to Unseen Faults and Anomalies using Generative Adversarial Networks (GANs):**
   Investigate methods to enhance the robustness of fault detection models to detect unseen faults and anomalies. This involves developing techniques for anomaly detection in high-dimensional spaces and exploring the use of GANs to generate synthetic process data for training and testing fault detection models. GANs can be used to create realistic fault scenarios, augmenting the available data and improving model robustness. Additionally, GANs can simulate rare fault conditions that may not be present in historical data, providing a comprehensive approach to fault simulation.

6. **Extension to Continuous Processes:**
   While the current research focuses on batch processes, future work could extend these methodologies to continuous processes, which are increasingly prevalent in the pharmaceutical industry. This would involve adapting the models to handle continuous data flows and different process dynamics.

7. **Application of Transformer Architectures:**
   Investigate the application of transformer architectures for process monitoring and fault diagnosis. Transformers, known for their capability to handle sequential data, could be used to model long-term dependencies and improve the accuracy of fault detection in dynamic pharmaceutical processes. Their suitability for parallel com-

putation and for capturing global context could lead to significant advancements in process monitoring.

8. **Adding of Measurements for Important Process Variables:** In process monitoring and fault detection within pharmaceutical processes, improving accuracy may not necessarily require increasing the complexity of the machine learning architectures. Limitations in accuracy may arise due to lack of sufficiently informative data about process productivity. For example, incorporating online productivity indicators or measuring critical biochemical markers such as reactive oxygen species (ROS) can significantly enhance the model's ability to detect anomalies and faults. These additional data points capture essential process dynamics and early indicators of potential issues, allowing for more accurate and timely fault detection. By enriching the dataset with such relevant variables, it may be possible to achieve higher accuracy and robustness in process monitoring.

9. **Integration with Digital Twins:**
   Explore the integration of deep learning models with digital twins of pharmaceutical processes. Digital twins provide a virtual representation of the physical process, and combining them with advanced monitoring and fault diagnosis models could lead to more predictive and proactive process management and maintenance.

References

Mohammad Aghaee, Abhishek Mishra, Stephane Krau, Ibrahim Melih Tamer, and Hector Budman. Artificial intelligence applications for fault detection and diagnosis in pharmaceutical bioprocesses: a review. *Current Opinion in Chemical Engineering*, 44:101025, 2024a.

Barry M Wise, NL Ricker, DF Veltkamp, and Brittany R Kowalski. A theoretical basis for the use of principal component models for monitoring multivariate processes. *Process control and quality*, 1(1):41–51, 1990.

John F MacGregor, Christiane Jaeckle, Costas Kiparissides, and M Koutoudi. Process monitoring and diagnosis by multiblock pls methods. *AIChE Journal*, 40(5):826–838, 1994.

Alexandros Mouzakitis. Classification of fault diagnosis methods for control systems. *Measurement and Control*, 46(10):303–308, 2013.

Ashish Kumar, Isuru A Udugama, Carina L Gargalo, and Krist V Gernaey. Why is batch processing still dominating the biologics landscape? towards an integrated continuous bioprocessing alternative. *Processes*, 8(12):1641, 2020.

Jonathan J Darrow, Jerry Avorn, and Aaron S Kesselheim. Fda approval and regulation of pharmaceuticals, 1983-2018. *Jama*, 323(2):164–176, 2020.

Arti Swami, Swarupa Hatolkar, Ms Aishwarya Patil, Ms Kajal Dhumal, and Aarti Shastri. Process analytical technology (pat): Enhancing quality and efficiency in pharmaceutical development and production. *Korean Journal of Physiology and Pharmacology*, 27(4): 394–408, 2023.

Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

Serhat Kilicarslan, Mete Celik, and Şafak Sahin. Hybrid models based on genetic algorithm and deep learning algorithms for nutritional anemia disease classification. *Biomedical Signal Processing and Control*, 63:102231, 2021.

Mohammad Aghaee, Stephane Krau, Melih Tamer, and Hector Budman. Graph neural network representation of state space models of metabolic pathways. *IFAC-PapersOnLine*, 2024b.

Jon C Gunther, Jeremy S Conner, and Dale E Seborg. Fault detection and diagnosis in an industrial fed-batch cell culture process. *Biotechnology progress*, 23(4):851–857, 2007.

Seung Ji Lim, Moon Son, Seo Jin Ki, Sang-Ik Suh, and Jaeshik Chung. Opportunities and challenges of machine learning in bioprocesses: Categorization from different perspectives and future direction. *Bioresource Technology*, 370:128518, 2023.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. *The elements of statistical learning: Data mining, inference, and prediction*, pages 485–585, 2009.

Jianbo Yu and Yue Zhang. Challenges and opportunities of deep learning-based process fault detection and diagnosis: a review. *Neural Computing and Applications*, 35(1): 211–252, 2023.

Efstratios N Pistikopoulos, Ana Barbosa-Povoa, Jay H Lee, Ruth Misener, Alexander Mitsos, Gintaras V Reklaitis, Venkat Venkatasubramanian, Fengqi You, and Rafiqul Gani. Process systems engineering–the generation next? *Computers & Chemical Engineering*, 147:107252, 2021.

Cristiano Hora Fontes and Hector Budman. A hybrid clustering approach for multivariate time series–a case study applied to failure analysis in a gas turbine. *ISA transactions*, 71:513–529, 2017.

Barry M Wise and Neal B Gallagher. The process chemometrics approach to process monitoring and fault detection. *Journal of Process Control*, 6(6):329–348, 1996.

Theodora Kourti and John F MacGregor. Multivariate spc methods for process and product monitoring. *Journal of quality technology*, 28(4):409–428, 1996.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Max Bylesjö, Mattias Rantalainen, Olivier Cloarec, Jeremy K Nicholson, Elaine Holmes, and Johan Trygg. Opls discriminant analysis: combining the strengths of pls-da and simca classification. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 20(8-10):341–351, 2006.

Piyush Agarwal, Mohammad Aghaee, Melih Tamer, and Hector Budman. A novel unsupervised approach for batch process monitoring using deep learning. *Computers & Chemical Engineering*, 159:107694, 2022.

Mohammad Aghaee, Stephane Krau, Melih Tamer, and Hector Budman. Unsupervised fault detection of pharmaceutical processes using long short-term memory autoencoders. *Industrial & Engineering Chemistry Research*, 62(25):9773–9786, 2023.

Cheng Ji, Fangyuan Ma, Jingde Wang, and Wei Sun. Profitability related industrial-scale batch processes monitoring via deep learning based soft sensor development. *Computers & Chemical Engineering*, 170:108125, 2023.

Dylan Hematillake, Daniele Freethy, Jacob McGivern, Chris McCready, Piyush Agarwal, and Hector Budman. Design and optimization of a penicillin fed-batch reactor based on a deep learning fault detection and diagnostic model. *Industrial & Engineering Chemistry Research*, 61(13):4625–4637, 2022.

Yuxuan Li, Zhangyue Shi, Chenang Liu, Wenmeng Tian, Zhenyu Kong, and Christopher B Williams. Augmented time regularized generative adversarial network (atr-gan) for data augmentation in online process anomaly detection. *IEEE Transactions on Automation Science and Engineering*, 19(4):3338–3355, 2021a.

Ke Yan, Adrian Chong, and Yuchang Mo. Generative adversarial network for fault detection diagnosis of chillers. *Building and Environment*, 172:106698, 2020.

Maryam Abbasi, Beatriz P Santos, Tiago C Pereira, Raul Sofia, Nelson RC Monteiro, Carlos JV Simões, Rui MM Brito, Bernardete Ribeiro, José L Oliveira, and Joel P Arrais. Designing optimized drug candidates with generative adversarial network. *Journal of cheminformatics*, 14(1):40, 2022.

Leena Chandrashekar, K Namratha, Padmapriya Mohan, Raxna Bopanna PJ, and Aditi Kulkarni. Discovering novel pharmaceutical molecules with generative adversarial networks. In *2023 10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, pages 378–383. IEEE, 2023.

Zeyu Li, Peng Chang, Kai Wang, and Pu Wang. The batch process fault monitoring using adversarial auto-encoder and k-nearest neighbor rule. In *2021 IEEE 10th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 41–48. IEEE, 2021b.

Pau Cabaneros Lopez, Isuru A Udugama, Sune T Thomsen, Christian Roslander, Helena Junicke, Miguel M Iglesias, and Krist V Gernaey. Transforming data to information: A parallel hybrid model for real-time state estimation in lignocellulosic ethanol fermentation. *Biotechnology and bioengineering*, 118(2):579–591, 2021.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92 (3):88, 2022.

Zhong Li, Yuxuan Zhu, and Matthijs Van Leeuwen. A survey on explainable anomaly detection. *ACM Transactions on Knowledge Discovery from Data*, 18(1):1–54, 2023a.

Liat Antwarg, Ronnie Mindlin Miller, Bracha Shapira, and Lior Rokach. Explaining anomalies detected by autoencoders using shapley additive explanations. *Expert systems with applications*, 186:115736, 2021.

Anicia Zeberli, Sara Badr, Christian Siegmund, Markus Mattern, and Hirokazu Sugiyama. Data-driven anomaly detection and diagnostics for changeover processes in biopharmaceutical drug product manufacturing. *Chemical Engineering Research and Design*, 167: 53–62, 2021.

Chang Peng, Ding Chunhao, and Zhao Qiankun. Fault diagnosis of microbial pharmaceutical fermentation process with non-gaussian and nonlinear coexistence. *Chemometrics and Intelligent Laboratory Systems*, 199:103931, 2020a.

Chang Peng, RuiWei Lu, Olivia Kang, and Wang Kai. Batch process fault detection for multi-stage broad learning system. *Neural Networks*, 129:298–312, 2020b.

Elena Quatrini, Xiaochuan Li, David Mba, and Francesco Costantino. Fault diagnosis of a granulator operating under time-varying conditions using canonical variate analysis. *Energies*, 13(17):4427, 2020.

Elena Quatrini, Francesco Costantino, Xiaochuan Li, and David Mba. Fault detection, diagnosis, and prognosis of a process operating under time-varying conditions. *Applied Sciences*, 12(9):4737, 2022.

Ding Chunhao, Chang Peng, and Olivia Kang. Enhanced high-order information extraction for multiphase batch process fault monitoring. *The Canadian Journal of Chemical Engineering*, 98(10):2187–2204, 2020.

Dovile Juodelyte, Veronika Cheplygina, Therese Graversen, and Philippe Bonnet. Predicting bearings degradation stages for predictive maintenance in the pharmaceutical industry. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3107–3115, 2022.

Chang Peng and Lu RuiWei. Process monitoring of batch process based on overcomplete broad learning network. *Engineering Applications of Artificial Intelligence*, 99:104139, 2021.

Jiayang Ren and Dong Ni. A batch-wise lstm-encoder decoder network for batch process monitoring. *Chemical Engineering Research and Design*, 164:102–112, 2020.

Lei Chen, Lei Geng, Zenglai Gao, Shoujun Li, Xubin Fu, Weishi Wang, Yuanbo Feng, Shuncong Wang, Yue Li, Yajun Fang, et al. An innovative algorithm to process imaging data for detection of weak chick embryos in vaccine production. In *2020 5th International Conference on Universal Village (UV)*, pages 1–8. IEEE, 2020a.

Soheil Aghajanian, Guruprasad Rao, Vesa Ruuskanen, Radosław Wajman, Lidia Jackowska-Strumillo, and Tuomas Koiranen. Real-time fault detection and diagnosis of caco3 reactive crystallization process by electrical resistance tomography measurements. *Sensors*, 21(21):6958, 2021.

Francesco Destro, Pierantonio Facco, Salvador Garcia Munoz, Fabrizio Bezzo, and Massimiliano Barolo. A hybrid framework for process monitoring: Enhancing data-driven methodologies with state and parameter estimation. *Journal of Process Control*, 92: 333–351, 2020.

Debanjan Ghosh, Prashant Mhaskar, and John F MacGregor. Hybrid partial least squares models for batch processes: Integrating data with process knowledge. *Industrial & Engineering Chemistry Research*, 60(26):9508–9520, 2021.

Stefan Windmann. Data-driven fault detection in industrial batch processes based on a stochastic hybrid process model. *IEEE Transactions on Automation Science and Engineering*, 19(4):3888–3902, 2022.

Xiaogang Deng and Kunyu Du. Efficient batch process monitoring based on random nonlinear feature analysis. *The Canadian Journal of Chemical Engineering*, 2021.

Xuejin Gao, Zidong Xu, Zheng Li, and Pu Wang. Batch process monitoring using multiway laplacian autoencoders. *The Canadian Journal of Chemical Engineering*, 98(6):1269–1279, 2020.

Yan Zhang, Xiaoqiang Zhao, Jie Cao, and Yongyong Hui. Batch process monitoring based on sequential phase division multiway sparse weighted neighborhood preserving embedding. *Measurement Science and Technology*, 35(3):035704, 2023a.

Zhong Liu, Xuyang Lou, and Jinlin Zhu. Batch process monitoring with two-dimensional lstm-variational auto-encoder. In *2023 42nd Chinese Control Conference (CCC)*, pages 4945–4950. IEEE, 2023a.

Kai Liu, Xiaoqiang Zhao, Miao Mou, and Yongyong Hui. Siamese denpe network framework for fault detection of batch process. *The Canadian Journal of Chemical Engineering*, 2024a.

Yan Zhang, Jie Cao, Xiaoqiang Zhao, and Yongyong Hui. Nonlinear multiphase batch process monitoring and quality prediction using multi-way concurrent locally weighted projection regression. *Chemometrics and Intelligent Laboratory Systems*, 240:104922, 2023b.

Xiaoqiang Zhao, Benben Tuo, and Yongyong Hui. Deep learning with cbam-based cnn for batch process quality prediction. *Measurement Science and Technology*, 34(11):115123, 2023.

Hongjuan Yao, Xiaoqiang Zhao, Wei Li, and Yongyong Hui. Multi-stage fusion regression network for quality prediction of batch process. *The Canadian Journal of Chemical Engineering*, 2023.

Piyush Agarwal, Melih Tamer, M Hossein Sahraei, and Hector Budman. Deep learning for classification of profit-based operating regions in industrial processes. *Industrial & Engineering Chemistry Research*, 59(6):2378–2395, 2019.

Piyush Agarwal, Melih Tamer, and Hector Budman. Explainability: Relevance based dynamic deep learning algorithm for fault detection and diagnosis in chemical processes. *Computers & Chemical Engineering*, page 107467, 2021. ISSN 0098-1354.

Piyush Agarwal, Jorge Ivan Mireles Gonzalez, Ali Elkamel, and Hector Budman. Hierarchical deep recurrent neural network based method for fault detection and diagnosis. *arXiv preprint arXiv:2012.03861*, 2020.

Manabu Kano, Shouhei Tanaka, Shinji Hasebe, Iori Hashimoto, and Hiromu Ohno. Monitoring independent components for fault detection. *AIChE Journal*, 49(4):969–976, 2003.

Mano Ram Maurya, Raghunathan Rengaswamy, and Venkat Venkatasubramanian. Fault diagnosis by qualitative trend analysis of the principal components. *Chemical Engineering Research and Design*, 83(9):1122–1132, 2005.

Leo H Chiang, Evan L Russell, and Richard D Braatz. Fault diagnosis in chemical processes using fisher discriminant analysis, discriminant partial least squares, and principal component analysis. *Chemometrics and intelligent laboratory systems*, 50(2):243–252, 2000.

Junghui Chen and Kun-Chih Liu. On-line batch process monitoring using dynamic pca and dynamic pls models. *Chemical Engineering Science*, 57(1):63–75, 2002.

B Lennox, GA Montague, HG Hiden, G Kornfeld, and PR Goulding. Process monitoring of an industrial fed-batch fermentation. *Biotechnology and bioengineering*, 74(2):125–135, 2001.

Cenk Ündey, Sinem Ertunç, and Ali Çınar. Online batch/fed-batch process performance monitoring, quality prediction, and variable-contribution analysis for diagnosis. *Industrial & engineering chemistry research*, 42(20):4645–4658, 2003.

Theodora Kourti. Application of latent variable methods to process control and multivariate statistical process control in industry. *International Journal of adaptive control and signal processing*, 19(4):213–246, 2005.

Leo H Chiang, Riccardo Leardi, Randy J Pell, and Mary Beth Seasholtz. Industrial experiences with multivariate statistical analysis of batch process data. *Chemometrics and Intelligent Laboratory Systems*, 81(2):109–119, 2006.

Stephen Goldrick, William Holmes, Nicholas J Bond, Gareth Lewis, Marcel Kuiper, Richard Turner, and Suzanne S Farid. Advanced multivariate data analysis to determine the root cause of trisulfide bond formation in a novel antibody–peptide fusion. *Biotechnology and bioengineering*, 114(10):2222–2234, 2017.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When do neural networks outperform kernel methods? *arXiv preprint arXiv:2006.13409*, 2020.

Gavneet Singh Chadha, Monica Krishnamoorthy, and Andreas Schwung. Time series based fault detection in industrial processes using convolutional neural networks. In *IECON*

2019-45th Annual Conference of the IEEE Industrial Electronics Society, volume 1, pages 173–178. IEEE, 2019.

Xu Zhang, Yuanyuan Zou, Shaoyuan Li, and Shenghu Xu. A weighted auto regressive lstm based approach for chemical processes modeling. *Neurocomputing*, 367:64–74, 2019a.

Feifan Cheng, Q Peter He, and Jinsong Zhao. A novel process monitoring approach based on variational recurrent autoencoder. *Computers & Chemical Engineering*, 129:106515, 2019.

Pangun Park, Piergiuseppe Di Marco, Hyejeon Shin, and Junseong Bang. Fault detection and diagnosis using combined autoencoder and long short-term memory network. *Sensors*, 19(21):4612, 2019.

Chunyong Yin, Sun Zhang, Jin Wang, and Neal N Xiong. Anomaly detection based on convolutional recurrent autoencoder for iot time series. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

Weiwu Yan, Pengju Guo, Zukui Li, et al. Nonlinear and robust statistical process monitoring based on variant autoencoders. *Chemometrics and Intelligent Laboratory Systems*, 158:31–40, 2016.

Wanke Yu and Chunhui Zhao. Robust monitoring and fault isolation of nonlinear industrial processes using denoising autoencoder and elastic net. *IEEE Transactions on Control Systems Technology*, 28(3):1083–1091, 2019.

Qingchao Jiang, Shifu Yan, Xuefeng Yan, Hui Yi, and Furong Gao. Data-driven two-dimensional deep correlated representation learning for nonlinear batch process monitoring. *IEEE Transactions on Industrial Informatics*, 16(4):2839–2848, 2019.

Shumei Chen, Jianbo Yu, and Shijin Wang. One-dimensional convolutional auto-encoder-based feature learning for fault diagnosis of multivariate processes. *Journal of Process Control*, 87:54–67, 2020b.

Jong-Min Lee, ChangKyoo Yoo, and In-Beum Lee. Statistical process monitoring with independent component analysis. *Journal of process control*, 14(5):467–485, 2004a.

J Edward Jackson and Govind S Mudholkar. Control procedures for residuals associated with principal component analysis. *Technometrics*, 21(3):341–349, 1979.

Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962a. doi: 10.1214/aoms/1177704472.

Bernard W Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC Monographs on Statistics and Applied Probability. Routledge, 2018.

Stephen Goldrick, Carlos A Duran-Villalobos, Karolis Jankauskas, David Lovett, Suzanne S Farid, and Barry Lennox. Modern day monitoring and control challenges outlined on an industrial-scale benchmark fermentation process. *Computers & Chemical Engineering*, 130:106471, 2019.

Stephen Goldrick, Andrei Ştefan, David Lovett, Gary Montague, and Barry Lennox. The development of an industrial-scale fed-batch fermentation simulation. *Journal of biotechnology*, 193:70–82, 2015.

John F MacGregor and Theodora Kourti. Statistical process control of multivariate processes. *Control engineering practice*, 3(3):403–414, 1995.

EB Martin and AJ Morris. An overview of multivariate statistical process control in continuous and batch process performance monitoring. *Transactions of the Institute of Measurement and Control*, 18(1):51–60, 1996.

Zhiqiang Ge and Zhihuan Song. *Multivariate statistical process control: Process monitoring methods and applications*. Springer Science & Business Media, 2012.

Yuncheng Du, Thomas A Duever, and Hector Budman. Fault detection and diagnosis with parametric uncertainty using generalized polynomial chaos. *Computers & Chemical Engineering*, 76:63–75, 2015.

Quoc V Le et al. A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. *Google Brain*, 20:1–20, 2015.

Hiranmayee Vedam and Venkat Venkatasubramanian. Pca-sdg based process monitoring and fault diagnosis. *Control engineering practice*, 7(7):903–917, 1999.

Manish Misra, H Henry Yue, S Joe Qin, and Cheng Ling. Multivariate process monitoring and fault diagnosis by multi-scale pca. *Computers & Chemical Engineering*, 26(9):1281–1293, 2002.

Lijia Luo, Shiyi Bao, Jianfeng Mao, and Di Tang. Fault detection and diagnosis based on sparse pca and two-level contribution plots. *Industrial & Engineering Chemistry Research*, 56(1):225–240, 2017.

Hao Wu and Jinsong Zhao. Deep convolutional neural network model based chemical process fault diagnosis. *Computers & chemical engineering*, 115:185–197, 2018.

Paul Nomikos and John F MacGregor. Monitoring batch processes using multiway principal component analysis. *AIChE Journal*, 40(8):1361–1375, 1994.

Dong Dong and Thomas J McAvoy. Batch tracking via nonlinear principal component analysis. *AIChE Journal*, 42(8):2199–2208, 1996.

Vanessa Zavatti, Hector Budman, Raymond L Legge, and Melih Tamer. Evaluation of flow cytometry and chemometric models for monitoring and predicting antigen production at full-scale. *Biochemical Engineering Journal*, 175:108136, 2021.

Theodora Kourti, Jennifer Lee, and John F Macgregor. Experiences with industrial applications of projection methods for multivariate statistical process control. *Computers & chemical engineering*, 20:S745–S750, 1996.

Cenk Ündey, Eric Tatara, and Ali Çınar. Intelligent real-time performance monitoring and quality prediction for batch/fed-batch cultivations. *Journal of Biotechnology*, 108(1): 61–77, 2004.

Paige Miller, Ronald E Swanson, and Charles E Heckler. Contribution plots: a missing link in multivariate quality control. *Appl. Math. Comp. Sci.*, 8(4):775–792, 1998.

Baligh Mnassri, Bouchra Ananou, Mustapha Ouladsine, et al. Fault detection and diagnosis based on pca and a new contribution plot. *IFAC Proceedings Volumes*, 42(8):834–839, 2009.

Kalpana D Joshi and PS Nalwade. Modified k-means for better initial cluster centres. *International Journal of Computer Science and Mobile Computing*, 2(7):219–223, 2013.

Piyush Agarwal. Application of deep learning in chemical processes: Explainability, monitoring and observability. 2022.

Vanessa Zavatti. Monitoring of an antigen manufacturing process using fluorescence. Master's thesis, University of Waterloo, 2014.

Kathryn E Wellen and Craig B Thompson. Cellular metabolic stress: considering how cells respond to nutrient excess. *Molecular cell*, 40(2):323–332, 2010.

Jue D Wang and Petra A Levin. Metabolism, cell growth and the bacterial cell cycle. *Nature Reviews Microbiology*, 7(11):822–827, 2009.

Shuyuan Zhang and Tong Qiu. A dynamic-inner convolutional autoencoder for process monitoring. *Computers & Chemical Engineering*, 158:107654, 2022.

Aniekan Essien and Cinzia Giannetti. A deep learning model for smart manufacturing using convolutional lstm neural network autoencoders. *IEEE Transactions on Industrial Informatics*, 16(9):6069–6078, 2020.

Nima Amini and Qinqin Zhu. Fault detection and diagnosis with a novel source-aware autoencoder and deep residual neural network. *Neurocomputing*, 488:618–633, 2022.

Yingxiang Liu, Robert Young, and Behnam Jafarpour. Long–short-term memory encoder–decoder with regularized hidden dynamics for fault detection in industrial processes. *Journal of Process Control*, 124:166–178, 2023b.

Li Qi, Yuwei Ren, Yixian Fang, and Jinglin Zhou. Two-view lstm variational auto-encoder for fault detection and diagnosis in multivariable manufacturing processes. *Neural Computing and Applications*, 35(29):22007–22026, 2023.

Jinlin Zhu, Muyun Jiang, and Zhong Liu. Fault detection and diagnosis in industrial processes with variational autoencoder: A comprehensive study. *Sensors*, 22(1):227, 2021.

Harini Narayanan, Lars Behle, Martin F Luna, Michael Sokolov, Gonzalo Guillén-Gosálbez, Massimo Morbidelli, and Alessandro Butté. Hybrid-ekf: Hybrid model coupled with extended kalman filter for real-time monitoring and control of mammalian cell culture. *Biotechnology and Bioengineering*, 117(9):2703–2714, 2020.

Quan Li, Zhonggai Zhao, and Fei Liu. Online monitoring of penicillin manufacture based on production variables and metabolic fluxes. *Industrial & Engineering Chemistry Research*, 62(3):1480–1491, 2023b.

Hesam Hassanpour, Amir H Hamedi, Prashant Mhaskar, John M House, and Timothy I Salsbury. A hybrid clustering approach integrating first-principles knowledge with data for fault detection in hvac systems. *Computers & Chemical Engineering*, page 108717, 2024.

N Lawrence Ricker. Decentralized control of the tennessee eastman challenge process. *Journal of process control*, 6(4):205–221, 1996.

Emil B Andersen, Isuru A Udugama, Krist V Gernaey, Abdul R Khan, Christoph Bayer, and Murat Kulahci. An easy to use gui for simulating big data using tennessee eastman process. *Quality and Reliability Engineering International*, 38(1):264–282, 2022.

Dimitris C Psichogios and Lyle H Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992.

Micheal L Mavrovouniotis and S Chang. Hierarchical neural networks. *Computers & chemical engineering*, 16(4):347–369, 1992.

Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962b.

Mingwei Jia, Danya Xu, Tao Yang, Yi Liu, and Yuan Yao. Graph convolutional network soft sensor for process quality prediction. *Journal of Process Control*, 123:12–25, 2023.

Yujun Liu, Dong Ni, and Zongyi Wang. A fault-tolerant soft sensor algorithm based on long short-term memory network for uneven batch process. *Processes*, 12(3):495, 2024b.

Mohamed Bin Shams, Hector Budman, and Thomas Duever. Fault detection using cusum based techniques with application to the tennessee eastman process. *IFAC Proceedings Volumes*, 43(5):109–114, 2010.

Seongmin Heo and Jay H Lee. Statistical process monitoring of the tennessee eastman process using parallel autoassociative neural networks and a large dataset. *Processes*, 7 (7):411, 2019.

CK Lau, Kaushik Ghosh, Mohd Azlan Hussain, and CR Che Hassan. Fault diagnosis of tennessee eastman process with multi-scale pca and anfis. *Chemometrics and Intelligent Laboratory Systems*, 120:1–14, 2013.

Ildar Lomov, Mark Lyubimov, Ilya Makarov, and Leonid E Zhukov. Fault detection in tennessee eastman process with temporal deep learning models. *Journal of Industrial Information Integration*, 23:100216, 2021.

Xiangyin Kong and Zhiqiang Ge. Deep learning of latent variable models for industrial process monitoring. *IEEE Transactions on Industrial Informatics*, 18(10):6778–6788, 2021.

Christopher Reinartz, Murat Kulahci, and Ole Ravn. An extended tennessee eastman simulation dataset for fault-detection and decision support systems. *Computers & chemical engineering*, 149:107281, 2021.

James J Downs and Ernest F Vogel. A plant-wide industrial process control problem. *Computers & chemical engineering*, 17(3):245–255, 1993.

NL Ricker. Optimal steady-state operation of the tennessee eastman challenge process. *Computers & chemical engineering*, 19(9):949–959, 1995.

Andreas Bathelt, N Lawrence Ricker, and Mohieddine Jelali. Revision of the tennessee eastman process model. *IFAC-PapersOnLine*, 48(8):309–314, 2015.

Frede Lei, Morten Rotbøll, and Sten Bay Jørgensen. A biochemically structured model for saccharomyces cerevisiae. *Journal of biotechnology*, 88(3):205–221, 2001.

Radhakrishnan Mahadevan, Jeremy S Edwards, and Francis J Doyle. Dynamic flux balance analysis of diauxic growth in escherichia coli. *Biophysical journal*, 83(3):1331–1340, 2002.

Zak Costello and Hector Garcia Martin. A machine learning approach to predict metabolic pathway dynamics from time-series multiomics data. *NPJ systems biology and applications*, 4(1):1–14, 2018.

Michael Vitelli, Ibrahim Melih Tamer, Mark Pritzker, and Hector Budman. Modeling the effect of oxidative stress on bordetella pertussis fermentations. *Biotechnology Progress*, page e3335, 2023.

Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Sang Wook Choi, Julian Morris, and In-Beum Lee. Dynamic model-based batch process monitoring. *Chemical Engineering Science*, 63(3):622–636, 2008.

Gang Li, S Joe Qin, and Donghua Zhou. Geometric properties of partial least squares for process monitoring. *Automatica*, 46(1):204–210, 2010.

Svante Wold, Paul Geladi, Kim Esbensen, and Jerker Öhman. Multi-way principal components-and pls-analysis. *Journal of chemometrics*, 1(1):41–56, 1987.

Sami Romdhani, Shaogang Gong, Alexandra Psarrou, et al. A multi-view nonlinear active shape model using kernel pca. In *BMVC*, volume 10, pages 483–492. Citeseer, 1999.

Viet Ha Nguyen and Jean-Claude Golinval. Fault detection based on kernel principal component analysis. *Engineering Structures*, 32(11):3683–3691, 2010.

Kyungpil Kim, Jong-Min Lee, and In-Beum Lee. A novel multivariate regression approach based on kernel partial least squares with orthogonal signal correction. *Chemometrics and intelligent laboratory systems*, 79(1-2):22–30, 2005.

Jong-Min Lee, ChangKyoo Yoo, Sang Wook Choi, Peter A Vanrolleghem, and In-Beum Lee. Nonlinear process monitoring using kernel principal component analysis. *Chemical engineering science*, 59(1):223–234, 2004b.

Leo H Chiang, Mark E Kotanchek, and Arthur K Kordon. Fault diagnosis based on fisher discriminant analysis and support vector machines. *Computers & chemical engineering*, 28(8):1389–1401, 2004.

Melis Onel, Chris A Kieslich, Yannis A Guzman, Christodoulos A Floudas, and Efstratios N Pistikopoulos. Big data approach to batch process monitoring: Simultaneous fault detection and diagnosis using nonlinear support vector machine-based feature selection. *Computers & chemical engineering*, 115:46–63, 2018.

Sang Wook Choi, Jin Hyun Park, and In-Beum Lee. Process monitoring using a gaussian mixture model via principal component analysis and discriminant analysis. *Computers & chemical engineering*, 28(8):1377–1387, 2004.

Jie Yu and S Joe Qin. Multimode process monitoring with bayesian inference-based finite gaussian mixture models. *AIChE Journal*, 54(7):1811–1829, 2008.

Q Peter He and Jin Wang. Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes. *IEEE transactions on semiconductor manufacturing*, 20(4):345–354, 2007.

Venkat Venkatasubramanian and King Chan. A neural network methodology for process fault diagnosis. *AIChE Journal*, 35(12):1993–2002, 1989.

Hadi Shahnazari. Fault diagnosis of nonlinear systems using recurrent neural networks. *Chemical Engineering Research and Design*, 153:233–245, 2020.

Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1409–1416, 2019b.

Saïd Ladjal, Alasdair Newson, and Chi-Hieu Pham. A pca-like autoencoder. *arXiv preprint arXiv:1904.01277*, 2019.

Samir Khatir, Idir Belaidi, Roger Serra, Magd Abdel Wahab, and Tawfiq Khatir. Damage detection and localization in composite beam structures based on vibration analysis. *Mechanics*, 21(6):472–479, 2015.

A Antonelli, S Giarnetti, and F Leccese. Enhanced pll system for harmonic analysis through genetic algorithm application. In *2012 11th International Conference on Environment and Electrical Engineering*, pages 328–333. IEEE, 2012.

Michael Wetter and Jonathan Wright. Comparison of a generalized pattern search and a genetic algorithm optimization method. In *Proc. of the 8-th IBPSA Conference*, volume 3, pages 1401–1408, 2003.

H Tran-Ngoc, Samir Khatir, G De Roeck, T Bui-Tien, L Nguyen-Ngoc, and Magd Abdel Wahab. Model updating for nam o bridge using particle swarm optimization algorithm and genetic algorithm. *Sensors*, 18(12):4131, 2018.

John H Holland. Adaptation in natural and artificial systems, university of michigan press. *Ann arbor, MI*, 1(97):5, 1975.

Haitao Zhao, Shaoyuan Sun, and Bo Jin. Sequential fault diagnosis based on lstm neural network. *Ieee Access*, 6:12929–12939, 2018.

Min Zhang, Xianjun Li, and Ruiqi Wang. Incipient fault diagnosis of batch process based on deep time series feature extraction. *Arabian Journal for Science and Engineering*, 46 (10):10125–10136, 2021.

EB Martin, AJ Morris, and J Zhang. Process performance monitoring using multivariate statistical process control. *IEE Proceedings-Control Theory and Applications*, 143(2): 132–144, 1996.

George Box and Tim Kramer. Statistical process monitoring and feedback adjustment—a discussion. *Technometrics*, 34(3):251–267, 1992.

Jianbo Yu, Xing Liu, and Lyujiangnan Ye. Convolutional long short-term memory autoencoder-based feature learning for fault detection in industrial processes. *IEEE Transactions on Instrumentation and Measurement*, 70:1–15, 2020.

Sudarshan Ganesh, Qinglin Su, Nolan Pepka, Benjamin Rentz, Lucas Vann, Nima Yazdanpanah, Thomas O'Connor, Zoltan K Nagy, Gintaras V Reklaitis, et al. Design of condition-based maintenance framework for process operations management in pharmaceutical continuous manufacturing. *International journal of pharmaceutics*, 587:119621, 2020.

Carina L Gargalo, Isuru Udugama, Katrin Pontius, Pau C Lopez, Rasmus F Nielsen, Aliyeh Hasanzadeh, Seyed Soheil Mansouri, Christoph Bayer, Helena Junicke, and Krist V Gernaey. Towards smart biomanufacturing: a perspective on recent developments in industrial measurement and monitoring technologies for bio-based production processes. *Journal of Industrial Microbiology & Biotechnology: Official Journal of the Society for Industrial Microbiology and Biotechnology*, 47(11):947–964, 2020.

Milos Drobnjakovic, Roger Hart, Boonserm Kulvatunyou, Nenad Ivezic, and Vijay Srinivasan. Current challenges and recent advances on the path towards continuous biomanufacturing. *Biotechnology Progress*, 39(6):e3378, 2023.

Benjamin Bayer, Moritz von Stosch, Gerald Striedner, and Mark Duerkop. Comparison of modeling methods for doe-based holistic upstream process characterization. *Biotechnology Journal*, 15(5):1900551, 2020.

Belmiro Duarte, PM Saraiva, and CC Pantelides. Combined mechanistic and empirical modelling. *International Journal of Chemical Reactor Engineering*, 2(1), 2004.

Andrew Farlow, Els Torreele, Glenda Gray, Kiat Ruxrungtham, Helen Rees, Sai Prasad, Carolina Gomez, Amadou Sall, Jorge Magalhães, Piero Olliaro, et al. The future of epidemic and pandemic vaccines to serve global public health needs. *Vaccines*, 11(3): 690, 2023.

Lalitkumar K Vora, Amol D Gholap, Keshava Jetha, Raghu Raj Singh Thakur, Hetvi K Solanki, and Vivek P Chavda. Artificial intelligence in pharmaceutical technology and drug delivery design. *Pharmaceutics*, 15(7):1916, 2023.

Sara Gerke, Timo Minssen, and Glenn Cohen. Ethical and legal challenges of artificial intelligence-driven healthcare. In *Artificial intelligence in healthcare*, pages 295–336. Elsevier, 2020.

Prodromos Daoutidis, Jay H Lee, Srinivas Rangarajan, Leo Chiang, Bhushan Gopaluni, Artur M Schweidtmann, Iiro Harjunkoski, Mehmet Mercangöz, Ali Mesbah, Fani Boukouvala, et al. Machine learning in process systems engineering: Challenges and opportunities. *Computers & Chemical Engineering*, page 108523, 2023.

Isuru A Udugama, Carina L Gargalo, Yoshiyuki Yamashita, Michael A Taube, Ahmet Palazoglu, Brent R Young, Krist V Gernaey, Murat Kulahci, and Christoph Bayer. The role of big data in industrial (bio) chemical process operations. *Industrial & Engineering Chemistry Research*, 59(34):15283–15297, 2020.

Athanassios Kassidas, John F MacGregor, and Paul A Taylor. Synchronization of batch trajectories using dynamic time warping. *AIChE Journal*, 44(4):864–875, 1998.