

Application of Deep Learning in Chemical Processes: Explainability, Monitoring and Observability

by

Piyush Agarwal

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, 2021

© Piyush Agarwal 2021

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: Peter Douglas
Professor, Dept. of Chemical Engineering, University of Waterloo

Internal-External Member: Kumaraswamy Ponnambalam
Professor, Systems Design Engineering, University of Waterloo

Internal Member: Nasser Mohieddin Abukhdeir
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 5, the author implemented the documented methodologies, obtained the numerical results and completed the writing of the paper. Jorge Ivan Gonzalez and Hector Budman were involved in the conceptualization of the work. All authors read and approved the final manuscript.

For Chapter 6, the author and Mohammad Aghae implemented the documented methodologies, obtained the numerical results. Author and Hector Budman were involved in the conceptualization of the work. All authors read and approved the final manuscript.

Abstract

The last decade has seen remarkable advances in speech, image and language recognition tools that have been made available to the public through computer and mobile devices' applications. Most of these significant improvements were achieved by Artificial Intelligence (AI)/ deep learning (DL) algorithms ([Hinton et al. \[2006\]](#)) that generally refers to a set of novel neural network architectures and algorithms such as long-short term memory (LSTM) units, convolutional networks (CNN), autoencoders (AE), t-distributed stochastic embedding (TSNE), etc. Although neural networks are not new, due to a combination of relatively novel improvements in methods for training the networks and the availability of increasingly powerful computers, one can now model much more complex nonlinear dynamic behaviour by using complex structures of neurons, i.e. more layers of neurons, than ever before ([Goodfellow et al. \[2016\]](#)). However, it is recognized that training of neural nets of such complex structures require a vast amount of data. In this sense manufacturing processes are good candidates for deep learning applications since they utilize computers and information systems for monitoring and control thus generating massive amount of data. This is especially true in pharmaceutical companies such as Sanofi Pasteur, the industrial collaborator for the current study, where large data sets are routinely stored for monitoring and regulatory purposes. Although novel DL algorithms have been applied with great success in image analysis, speech recognition and language translation, their applications to chemical processes and pharmaceutical processes in particular is scarce.

The current work deals with investigation of deep learning in process systems engineering for three main areas of application:

1. Developing a deep learning classification model for profit based operating regions.
2. Developing both supervised and unsupervised process monitoring algorithms.
3. Observability Analysis

It is recognized that most empirical or black-box models, including DL models, have good generalization capabilities but are difficult to interpret. For example, using these

methods it is difficult to understand how a particular decision is made, which input variable/feature is greatly influencing the decision made by the DL models etc. This understanding is expected to shed light on why biased results can be obtained or why a wrong class is predicted with a higher probability in classification problems. Hence, a key goal of the current work is on deriving process insights from DL models. To this end, the work proposes both supervised and unsupervised learning approaches to identify regions of process inputs that result in corresponding regions, i.e ranges of values, of process profit. Furthermore, it will be shown that the ability to better interpret the model by identifying inputs that are most informative can be used to reduce over-fitting. To this end a neural network (NN) pruning algorithm is developed that provides important physical insights on the system regarding the inputs that have positive and negative effect on profit function and to detect significant changes in process phenomenon. It is shown that pruning of input variables significantly reduces the number of parameters to be estimated and improves the classification test accuracy for both case studies: the Tennessee Eastman Process (TEP) and an industrial vaccine manufacturing process.

The ability to store large amount of data have permitted the use of deep learning (DL) and optimization algorithms for the process industries. In order to meet high levels of product quality, efficiency and reliability, a process monitoring system is needed. The two aspects of Statistical Process Control (SPC) are fault detection and diagnosis (FDD). Many multivariate statistical methods like PCA and PLS and their dynamic variants have been extensively used for FD. However, the inherent non-linearities in the process pose challenges while using these linear models. Numerous deep learning FDD approaches have also been developed in the literature. However, the contribution plots for identifying the root cause of the fault have not been derived from Deep Neural Networks (DNNs). To this end, the supervised fault detection problem in the current work is formulated as a binary classification problem while the supervised fault diagnosis problem is formulated as a multi-class classification problem to identify the type of fault. Then, the application of the concept of explainability of DNNs is explored with its particular application in FDD problem. The developed methodology is demonstrated on TEP with non-incipient faults. Incipient faults are faulty conditions where signal to noise ratio is small and have not been

widely studied in the literature. To address the same, a hierarchical dynamic deep learning algorithm is developed specifically to address the issue of fault detection and diagnosis of incipient faults.

One of the major drawback of both the methods described above is the availability of labeled data i.e. normal operation and faulty operation data. From an industrial point of view, most data in an industrial setting, specially for biochemical processes, is obtained during normal operation and faulty data may not be available or may be insufficient. Hence, we also develop a unsupervised DL approach for process monitoring. It involves a novel objective function and a NN architecture that is tailored to detect the faults effectively. The idea is to learn the distribution of normal operation data to differentiate among the fault conditions. In order to demonstrate the advantages of the proposed methodology for fault detection, systematic comparisons are conducted with Multiway Principal Component Analysis (MPCA) and Multiway Partial Least Squares (MPLS) on an industrial scale Penicillin Simulator.

Past investigations reported that the variability in productivity in the Sanofi's Pertussis Vaccine Manufacturing process may be highly correlated to biological phenomena, i.e. oxidative stresses, that are not routinely monitored by the company. While the company monitors and stores large amount of fermentation data it may be not be sufficiently informative about the underlying phenomena affecting the level of productivity. Furthermore, since the addition of new sensors in pharmaceutical processes requires extensive and expensive validation and certification procedures, it is very important to assess the potential ability of a sensor to observe relevant phenomena before its actual adoption in the manufacturing environment. This motivates the study of the observability of the phenomena from available data. An algorithm is proposed to check the observability for the classification task from the observed data (measurements). The proposed methodology makes use of an Supervised AE to reduce the dimensionality of the inputs. Thereafter, a criterion on the distance between the samples is used to calculate the percentage of overlap between the defined classes. The proposed algorithm is tested on the benchmark Tennessee Eastman process and then applied to the industrial vaccine manufacturing process.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Professor Hector Budman. Even before being my supervisor he was an exceptional role model in work ethics and research methods. My special thanks to him for his promptness and sparing his invaluable time to help me advance quickly through out my research. His passion and devotion to research continues to inspire me and kept me going at times. I appreciate his continuous support throughout my graduate studies and research. I am greatly indebted for his advice on my overall development.

I would like to thank the Department of Chemical Engineering, University of Waterloo, for providing me an opportunity to pursue my PhD degree and with the best of the resources and a friendly atmosphere. Special thanks to all the administrative staff and especially Judy Caron who kept her patience and have helped me through out my journey.

I would also like to express my special thanks to Professor Alexander Penlidis. I will always cherish the enjoyable times I have had while assisting him with his teaching. I can never forget his wonderful words of advice and his sense of wit that I have enjoyed in his presence. I gratefully acknowledge Dr. Melih Tamer for supporting me throughout my PhD program and for providing the resources and help me get the right information from the domain experts at Sanofi Pasteur. I will forever be thankful to my former advisor, Professor Arun Tangirala. He has been instrumental in my development as a researcher and motivated to pursue PhD in the first place.

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.

I will always cherish the warmth and affection that I have received from my present and past colleagues and friends Tharun, Mariana, Michael, Xin, Dr. Yue, Huabei, Hong-

hao, Han Wang, Shashi, Kavita, Oscar, Dr. Prasad, Manan, Dr. Manoj, Donovan, Dr. Mahshad, Shruthi, Meghana, Abhishek, Mohammad. Special thanks to Vipul Mann, my former colleague, who have been a key support during the pandemic. I would like to thank him for all engaging conversations and insightful discussions over the years. I would like to thank Vanessa for her help in understanding cell biology and her PhD work along with several discussions.

I cannot thank Changjian Li enough for being the support system. I am grateful to his advice on many critical occasions and helped me stay sane during the stay. Special thanks to Anugrah Gangrade for being ‘the constant’ through all these years (a decade at this point in time). I also thank Yash Channe and Gaurav Phule for being supportive and caring during tough times.

Without the support of my family members, this work would not have been possible. My endless gratitude towards my parents for bestowing their unconditional love and affection. I am indebted to my sister Payal for her affectionate love and friendship. Besides this, several people knowingly or unknowingly helped me in the successful completion of this work.

Dedication

*I dedicate this thesis to
my grandparents, parents and my sister
for their support and unconditional love.
I love you all dearly*

Table of Contents

List of Figures	xv
List of Tables	xx
1 Introduction	1
2 Background	7
2.1 Introduction to Artificial Neural Networks	7
2.2 Multi-Layer Perceptron Neural Network (MLP-NN)	7
2.3 Activation Functions	9
2.4 Autoencoder Neural Networks (AE-NNs)	10
2.5 Long-Short Term Memory (LSTM) Units	13
2.6 Generalization, Regularization and Dropout	15
3 Deep Learning for Classification of Profit-based Operating Regions in Industrial Processes	18
3.1 Introduction	19
3.2 Preliminaries	24
3.2.1 Long Short-Term Memory Neural Networks (LSTM-NN)	24

3.2.2	Layer-wise Relevance Propagation (LRP)	26
3.3	Proposed Methodology: Sequential Layer-Wise Relevance Propagation for Pruning (SLRPFP)	27
3.4	Results and Discussions	28
3.4.1	Case Study 1: Tennessee Eastman Process (Simulated Case Study)	28
3.4.2	Case Study 2: Industrial Vaccine Manufacturing Process	39
3.5	Conclusion	52
4	Explainability: Relevance based Dynamic Deep Learning Algorithm for Fault Detection and Diagnosis in Chemical Processes	54
4.1	Introduction	55
4.2	Preliminaries	59
4.2.1	Deep Supervised Autoencoder Classification Neural Networks (DSAE-NNs)	59
4.2.2	Dynamic Deep Supervised Autoencoder Classification Neural Networks (DDSAE-NNs)	61
4.2.3	Layer-wise Relevance Propagation (LRP)	62
4.3	Proposed Fault Detection and Diagnosis Methodology based on DSAE-NNs and DDSAE-NNs	65
4.3.1	Fault Detection Methodology	66
4.3.2	Fault Diagnosis Methodology	68
4.3.3	Proposed Methodology for FDD	69
4.4	Case Study: Tennessee Eastman Process	72
4.5	Conclusion	85

5	Hierarchical Deep LSTM for Fault Detection and Diagnosis for a Chemical Process	88
5.1	Introduction	89
5.2	Preliminaries	92
5.2.1	Deep LSTM Supervised Autoencoder Neural Network (LSTM-SAE NN)	92
5.2.2	Model Structure and Specifications	94
5.3	Hierarchical Structure	95
5.3.1	Design: Pseudo-random Binary Signal (PRBS)	98
5.4	Results and discussion	100
5.5	Conclusions	109
6	A Novel Unsupervised Approach for Batch Process Monitoring using Deep Learning	113
6.1	Introduction	114
6.2	Preliminaries	117
6.2.1	Multiway Principal Component Analysis (MPCA)	117
6.2.2	MPLS	120
6.3	Proposed Methodology	121
6.3.1	Multiway Partial Least Squares Autoencoder (MPLS-AE)	121
6.3.2	Novel Objective Function for Maximizing Fault Detection Rate	124
6.3.3	Average Fault Detection Rate ($\overline{\text{FDR}}$)	125
6.3.4	Case study	127
6.4	Results and Discussions	129
6.4.1	MPCA and MPLS with J^{MPCA} and J^{MPLS}	131
6.4.2	MPCA and MPLS with $J^{MPCA-MPLS,FDR}$	133

6.4.3	MPLS-AE with $J^{MPLS-AE}$	135
6.4.4	MPLS-AE with $\mathbf{J}^{MPLS-AE,FDR}$	137
6.5	Conclusion	140
7	Assessing Observability using Supervised Autoencoders with Application to Industrial Processes	142
7.1	Introduction	143
7.2	Preliminaries	146
7.2.1	Supervised Autoencoder Classification Neural Networks (SAE-NNs)	147
7.3	Description of Case Studies	148
7.4	Proposed Methodology	150
7.5	Results and Discussion	154
7.5.1	Effect of Reconstruction Error Loss function on classification accuracy	155
7.5.2	Degree of Classification Observability for the TEP problem	156
7.5.3	Enhancing Classification Observability	157
7.5.4	Degree of Classification Observability for the Vaccine Manufacturing Process at Sanofi Pasteur, Toronto	159
7.6	Conclusion	162
8	Conclusions and Future Work	164
8.1	Conclusions	164
8.1.1	Classification of profit-based operating conditions	164
8.1.2	Statistical Process Control and Monitoring	165
8.1.3	Evaluating observability	167
8.2	Future Work	168
	References	170

List of Figures

2.1	Multi-Layer Perceptron Neural Network	8
2.2	Traditional single layer Autoencoder Neural Network (AE-NN)	12
2.3	Schematic of a LSTM memory cell	14
3.1	Schematic of a LSTM memory cell	25
3.2	Sequence to label classification using LSTM-NN	26
3.3	Tennessee Eastman plant process (Downs & Vogel, 1993)	30
3.4	Distribution of COP values	31
3.5	Confusion Matrices for Step 1 and Step 9 of the proposed method SLRPFP. (a) Confusion Matrix for Step 1. (b) Confusion Matrix for Step 9.	32
3.6	Average relevance of input variables for different iterations (SLRPFP). (Step 1-4)	33
3.7	Average relevance of input variables for different iterations (SLRPFP). (Step 5-8)	34
3.8	Average relevance of input variables for final iteration (SLRPFP). (Step 9)	35
3.9	Confusion Matrices for unsupervised approaches (Case 1)	37
3.10	Distribution of PRN (ELISA) productivity	38
3.11	Schematic of vaccine manufacturing process at Sanofi Pasteur, Toronto, Canada	40

3.12 MLP + SLRPFP: Test dataset accuracy before (left) and after SLRPFP (right) for industrial process	43
3.13 Averaged Time-based Relevance of Aeration profile for both High and Low PRN productivity batches	44
3.14 Averaged Time-based Relevance of Agitation profile for both High and Low PRN productivity batches	45
3.15 Averaged Time-based Relevance of Jacket Temperature profile for both High and Low PRN productivity batches	46
3.16 Averaged Time-based Relevance of Seal Temperature profile for both High and Low PRN productivity batches	47
3.17 LSTM + SLRPFP: confusion matrix of test dataset before (left) and after (right) SLRPFP	48
3.18 (a) Difference in the growth profile for two fermenters; (b) Increase in agitation profiles around 18 hours because of delayed increase of growth in low PRN batches	50
3.19 Averaged Relevance of all Input Variables from the LSTM model	51
3.20 (a) Selection of avg. number of principal components in BDPCA; (b) Confusion Matrix for test dataset (BDPCA)	52
4.1 Schematic of a single layer Supervised Autoencoder Neural Network (SAE-NN)	60
4.2 Left figure: Represents forward contribution of each node to the output layer; Right figure: Represents the relevance propagation from output layer to the input layer	64
4.3 Flowchart for fault detection and diagnosis based on explainable DNN	70
4.4 Schematic: Tennessee Eastman plant process (Downs and Vogel, 1993)	72
4.5 Final Iteration: Averaged Relative Relevance Plot for Fault Detection (DDSAE Model with 2 lagged input variables as \mathbf{X}^l)	77

4.6	Confusion Matrix for Fault Classification (First Iteration: DSAE Model with 52 input variables)	80
4.7	Confusion Matrix for Fault Classification (Final Iteration: DDSAE Model with 10 lagged input variables)	81
4.8	Final Iteration: Averaged Relative Relevance Plot for Fault Diagnosis (DDSAE Model with 10 lagged input variables as \mathbf{X}^l)	82
4.9	Input variable relevance plot for Fault Diagnosis (IDV 1)	82
4.10	Relevant variables contributing to IDV(1) with nominal and abnormal profiles	83
4.11	Input variable relevance plot for Fault Diagnosis (IDV 2)	83
4.12	Relevant variables contributing to IDV(2) with nominal and abnormal profiles	84
4.13	Variable Contribution Heatmap corresponding to all faults	84
4.14	Comparison of Fault Classification rate with different methods	85
5.1	Schematic of a Deep LSTM Supervised Autoencoder Neural Network (DLSTM-SAE NN)	94
5.2	Hierarchical structure used for fault detection and diagnosis	97
5.3	Confusion Matrix for the first level model of the hierarchical structure (i.e. classification of non-incipient faults and considering incipient faults as a normal class)	101
5.4	Comparison of averaged fault classification rates (non-incipient faults only)	102
5.5	Comparison of averaged fault classification rates (all faults)	103
5.6	Confusion Matrix on test data for the second level model of the hierarchical structure: a) After adding designed PRBS signal w.r.t. fault 15 b) After adding designed PRBS signal w.r.t. fault 9 and fault 15	107
5.7	Selection of optimal time horizon for Hierarchical LSTM-SAE Level 1 model	108
6.1	Schematic of a Multiway Partial Least Squares Autoencoder (MPLS-AE) .	121

6.2	Flowchart for fault detection based on novel objective function	126
6.3	Summary of all model inputs and outputs recorded by IndPenSim	129
6.4	Profiles of measured variables in normal and faulty conditions.	130
6.5	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 ($\alpha = 0.01$).	132
6.6	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 ($\alpha = 0.01$).	133
6.7	Plot 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 ($\alpha = 0.01$).	134
6.8	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 ($\alpha = 0.01$).	135
6.9	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 ($\alpha = 0.01$).	137
6.10	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 ($\alpha = 0.01$).	138
6.11	Loss of the validation dataset with MPLS-AE model using $J^{MPLS-AE}$ as the objective function to train the network.	139
6.12	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 ($\alpha = 0.01$).	140
6.13	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 ($\alpha = 0.01$).	141
7.1	Traditional single layer Autoencoder Neural Network (AE-NN)	146

7.2	Left: Projection of input space in 2 dimensions using TSNE for non-overlapping case (Case 1), Right: Projection of input space in 2 dimensions using TSNE for overlapping case (Case 2)	148
7.3	Schematic of Tennessee Eastman plant process (Downs & Vogel, 1993) . . .	150
7.4	Distribution of Cost Of Productivities (COP)	151
7.5	Average Relevance corresponding to correctly classified samples for low overlapping case (Case 1)	154
7.6	Average Relevance corresponding to overlapping samples for Case 1	155
7.7	Confusion Matrix for Case 1 (Validation Data-set)	156
7.8	Classification Overlap Matrix (COv) for Case 1 (Validation Data-set)	157

List of Tables

2.1	Commonly used activation functions and plots	11
3.1	Profit-based defined classes for COP	31
3.2	Implementation of SLRPFP for pruning input variables in TEP using LSTM model	36
3.3	Classes for productivity of Pertactin	41
4.1	Measured and manipulated variables (from Downs and Vogel, 1993)	73
4.2	Process Faults for classification in TE Process	74
4.3	Detection Delay for different faults	77
4.4	Network Architecture and iterations for fault detection methodology	78
4.5	Network Architecture and iterations for fault diagnosis methodology	79
4.6	Comparison of Fault Detection Rate with different methods with non-incipient faults only	86
5.1	Confusion Matrix for each fault (IDV(i))	102
5.2	Comparison of Fault Detection Rate with different methods with non-incipient faults only	110
5.3	Comparison of Fault Detection rate with different methods (with all faults)	111
5.4	Ablation study for the proposed method	112

6.1	Description of different types of fault	130
6.2	Comparison of ($\overline{\text{FDR}}$) test accuracy for different models	136
7.1	Profit-based defined classes for COP	149
7.2	Degree of classification observability (C_{obs}) for Case 1 and Case 2	158
7.3	Classification Accuracy for both cases ($\mathbf{z} \in \mathbb{R}^{d_z}, d_z = 7$)	158

Chapter 1

Introduction

Artificial intelligence (AI) involves a vast range of technologies that permit computers to emulate human thinking for the purpose of solving problems. Within AI there is a smaller category of algorithms, referred to as machine learning, that includes important mathematical techniques such as multivariate statistical models, neural networks and other data driven modeling methods, that can be used to improve processes based on data. Deep Learning refers to a particular type of machine learning techniques that use Deep (multilayered) Neural Network (NN) architectures. Although neural networks (NNs) are not new, their initial buzz momentarily fizzled in the 2000s when their computational limitations were identified in a classical paper by [Minsky and Papert \[1972\]](#). That study demonstrated that NNs are limited since they require a large amount of data for training, inefficient learning and often converge to a local optimum of the loss function that it is minimized for learning. These convergence issues were due to the use of gradient based algorithms that require very good initial guesses of the network parameters to converge to global optima. The current interest in NNs stems from the significant increase of computational power over the years combined with key theoretical developments that address some of the earlier limitations of NNs. These developments are in the area of deep NNs with new architectures such as Convolutional NNs, Autoencoder NNs, Long-short term memory (LSTM) NNs ([Hochreiter and Schmidhuber \[1997\]](#)), General Adversarial Networks (GAN) ([Goodfellow et al. \[2016\]](#)), etc. Novel greedy learning approaches were developed to

train these networks where the neuron parameters for the layers of the multi-layered (deep) neural networks are learned progressively starting from the first layer (input) to the last layer (outputs) one at a time and where each successive layer is learned based on results from the previous layer. While deep learning algorithms have been applied extensively in speech, language and vision problems, there are only few reported applications in chemical processes and specifically to bio-pharmaceutical processes as considered in this work.

Three case studies were considered in the current work: the whooping cough vaccine manufacturing process at Sanofi Pasteur, (the industrial collaborator for the current work), the Tennessee Eastman process and a Penicillin batch manufacturing process. The Sanofi process motivated the topics that were investigated in the current work. Since the objectives of this work were to both develop novel methodologies as well as monitoring approaches for the Sanofi process we pursued a pragmatic approach where the proposed algorithms were first tested on a known simulator of a chemical engineering plant, the TEP and Penicillin batch process and then they were tested on data provided by Sanofi Pasteur.

A major problem of the Pertussis Vaccine Manufacturing process for whooping cough at Sanofi Pasteur, Toronto (our industrial collaborator) is the variability in production of Pertactin (PRN), one of the antigens in the vaccine. In order to lower the variability, it is required to identify its root cause with the available measurements. The data to be used in this work is primarily extracted from the extensive historical data base of Sanofi for the manufacturing of whooping cough vaccine. The data included frequent measurements of fermentation data such as dissolved oxygen, aeration rates etc. While large amounts of process data are available from Sanofi, detailed mathematical models of their processes are not available. Also, while a lot of data was available, it was not a-priori clear which data is informative about the sources of process variability. In view that deep neural networks involve many model parameters it is crucial to identify the informative inputs to avoid over-fitting of the data by the model. The ability to quantify the relevance of specific measurements on productivity and other variables of interest was also crucial in order to decide on the addition of new sensors to the process. The adoption of new sensors in a pharmaceutical manufacturing environment requires extensive validation and certification procedures. Thus, it is vital to predict the relevance of a new sensor based on preliminary

data before its adoption. Another important challenge for the Sanofi process is that the sources of variability have not been fully identified as yet. This rules out the ability to develop supervised learning algorithms for this process but motivates the application of unsupervised learning based algorithms to detect abnormal operation.

To identify inputs that are most informative about outputs of interest, we develop a NN based classification methodology in Chapter 3 for classifying inputs pertaining to different productivity regions. This algorithm identify correlations between different ranges of input variables to different ranges of productivity of PRN antigens. The problem of classifying the inputs to a process according to different corresponding regions of productivity is somewhat different from a conventional fault classification problem since the goal here is to find combinations of input variables that result in different levels of profit/cost of productivity. For example, this type of problem permits both identifying input variables that have significant impact on profit as well as for identifying regions of input conditions that will result in high profit/low productivity cost. Classification tasks have been previously carried out using linear latent variables based techniques such as Principal Component Analysis (PCA) and Partial Least Squares (PLS). However, linear models when applied to non-linear processes may not be sufficiently accurate to represent highly nonlinear dynamic behavior. Machine learning/deep learning tools have been shown to be capable to both compress the data and to more accurately capture non-linear dynamics and consequently they were utilized in this work for modelling and classification purposes.

Another important task is to derive process insights from these classification models. Although DL based models have better generalization capabilities, they are poor in interpretation abilities because of their black box nature. Using these models it is difficult to identify the root cause, to understand how particular decisions are made, which input variable/feature is greatly influencing the decision made by the DL-NN models, etc. This understanding is important to shed light on why biased results can be obtained, why a wrong class is predicted with a higher probability in classification problems etc. Towards these goals, concepts from explainable AI were explored in order to explain the NN predictions. These explanations are termed as ‘relevances’. Relevance of input variables to a

classification task is paramount to understand the process model.

Moreover, the use of ‘relevances’ helped in discarding input measurements that are not relevant or had low relevance with respect to a particular classification task. Removing these irrelevant input variables reduces the number of network connections thus reducing the model parameters and reducing model over-fitting. Deep neural networks tend to contain millions of parameters, necessitating costly computational resources in order to train and deploy them in practice, and motivating the need to develop compact networks with fewer parameters (Han et al. [2015]). A reduction in parameter count alleviates the computational burden on training and inference, making it easier to deploy high capacity models. “Pruning” techniques have been proposed to eliminate irrelevant nodes, i.e. nodes that do not contribute significantly for minimizing the loss function. Most reported pruning methods are based on eliminating nodes while minimizing a non-linear loss function thus potentially converging to local minima. Other typical pruning algorithms either proceed by gradually pruning the nodes/units of the network during training (Gale et al. [2019]; He et al. [2018]; Zhu and Gupta [2017]) or by pruning the network after training followed by a retraining period (Han et al. [2015]; Liebenwein et al. [2021]; Renda et al. [2020]). Instead, we propose in this study a novel method for pruning ‘input variables’ instead of nodes/units of the NN by an algorithm that is based on evaluation of the averaged-relative relevance of input variables. Accordingly, the proposed method is referred to as ‘relevance based pruning’ in this work. Beyond its ability of pruning input variables and the network it will also be shown that the proposed methodology provides process insights. For example, the proposed pruning method in Chapter 3 can be used to identify regions of input conditions that will result in corresponding regions of high/low profit levels or high/low quality.

In order to meet high quality and productivity of end products, it is imperative to design a process monitoring system. Numerous empirical monitoring algorithms have been developed in the literature based on linear, non-linear and DL models. Recent DL methods have shown considerable improvement over traditional methods. Using these methods it is difficult to identify the root cause of faults i.e. input variables that are most correlated to the occurrence of the faults by significantly deviating from their normal trajectories following the occurrence of the fault. Chapter 4 investigates the application of the explainability

concept to derive contribution plots from deep NNs and enhance the fault detection and diagnosis (FDD) accuracy by pruning measurements of low relevance score.

A key issue in monitoring faults in manufacturing processes is the occurrence of incipient faults that cannot be easily identified due to high overlap of signals corresponding to different faults. The detection of incipient faults is shown in Chapter 5 where a deep LSTM based method is developed that uses dynamic information of the process along the time horizon. Based on this network, a hierarchical structure is formulated by grouping faults based on their similarity into subsets of faults for detection and diagnosis. Further, an external input is designed and introduced at specific locations in the TEP for improving the low signal to noise ratio. This improves the detection and classification accuracy significantly for both incipient and non-incipient faults. Both supervised FDD approaches (Chapter 4 and Chapter 5) are tested on benchmark TEP resulting in significant improvements over other state of the art methods.

One of the major drawback of both the methods proposed in Chapter 4 and 5 is that they required labeled data for training i.e. normal operation and faulty operation data. From an industrial point of view, most data in an industrial setting such as the whooping cough vaccine manufacturing process at Sanofi Pasteur, Toronto, is obtained during normal operation and faulty data may not be available or may be insufficient. Hence, we develop a unsupervised DL approach for process monitoring where the process monitoring system utilizes normal operation historical data records in Chapter 6. It involves a novel objective function and a NN architecture that is tailored to detect the faults effectively. The idea is to learn the distribution of normal operation data and to be able to distinguish abnormal operation based on the model trained with normal operation data only. In order to demonstrate the advantages of the proposed methodology for fault detection, systematic comparisons are conducted with Multiway Principal Component Analysis (MPCA) and Multiway Partial Least Squares (MPLS) on an industrial scale Penicillin Simulator.

Past investigations reported that the variability in productivity in the Sanofi process may be highly correlated to biological phenomena, i.e. oxidative stresses, that are not

routinely monitored by the company. Thus, while the company monitors and stores large amount of fermentation data, the latter may be not be sufficiently informative about the underlying phenomena affecting the level of productivity. To improve process diagnosis, Sanofi Pasteur continuously consider the adoption of new sensors, e.g. measurement of off-gases concentrations in the fermentation process. This motivated the study of observability of phenomena from preliminary data collected with a new sensor and its ability to add information about the process. An algorithm is proposed to check the observability for the classification task from the observed data (measurements). The proposed methodology makes use of an Autoencoder to reduce the dimensionality of the inputs. Thereafter, a criterion on the distance between the samples is used to calculate the percentage of overlap between the defined classes. The proposed algorithm is tested on the benchmark Tennessee Eastman process and it is presented later in Chapter 7.

Following the above, the novel contributions of this work can be summarized as follows:

1. Developed a supervised learning classification method for both static and dynamic DL models based on relevance and pruning of not significant inputs.
2. Developed an explainability based fault detection and diagnosis methodology based on DL models to identify the contributions of inputs to outputs of interest
3. Developed a hierarchical methodology for Fault detection and diagnosis of incipient faults
4. Developed an unsupervised fault detection and diagnosis methodology based on DNN models.
5. Developed a methodology to quantify the observability of outputs based on available data using DL models.

Chapter 2

Background

2.1 Introduction to Artificial Neural Networks

Artificial Neural Networks (ANNs) have been developed as a way to describe how complex information can be learned and processed by the biological nervous system. The way the human nervous system works has not been fully reproduced so far, neural networks are derived as a narrow - minded abstraction that takes advantage of the knowledge of the basic functionality and neuron organization. Human brain learn through experience similarly neural networks use examples (data samples) to generalize rules by performing multiple linear and nonlinear functions and adjusting the tuning weights of these functions to meet a specific fit criterion quality ([Goodfellow et al. \[2016\]](#)).

2.2 Multi-Layer Perceptron Neural Network (MLP-NN)

A MLP is a class of feed-forward network (also known as Artificial Neural Network (ANN)). Since single perceptrons (neurons) models are unable to explain complex non-linear functions [Goodfellow et al. \[2016\]](#) several stacked neurons and layers are needed to approximate non-linear complex functions. Each layer of neurons are stacked one after another and is

connected to each neuron in the following layer. Figure 2.1 shows a schematic for a basic Multi-Layer Perceptron of three layers. The network consists of three layers: input layer, hidden layer and an output layer. The input layer is fed with the data available in the problem. The hidden layers are used by the network to learn the important features that are contained in the input data. The output layer provides a final response that might be a class label in problems of classification or continuous values in general forecast/prediction problems. It is important to note that there are no limitations on the number of layers nor the number of neurons in a network. However, in feed-forward networks, there are no interconnections among neurons within the same layer and thus the data is transmitted only in the forward direction.

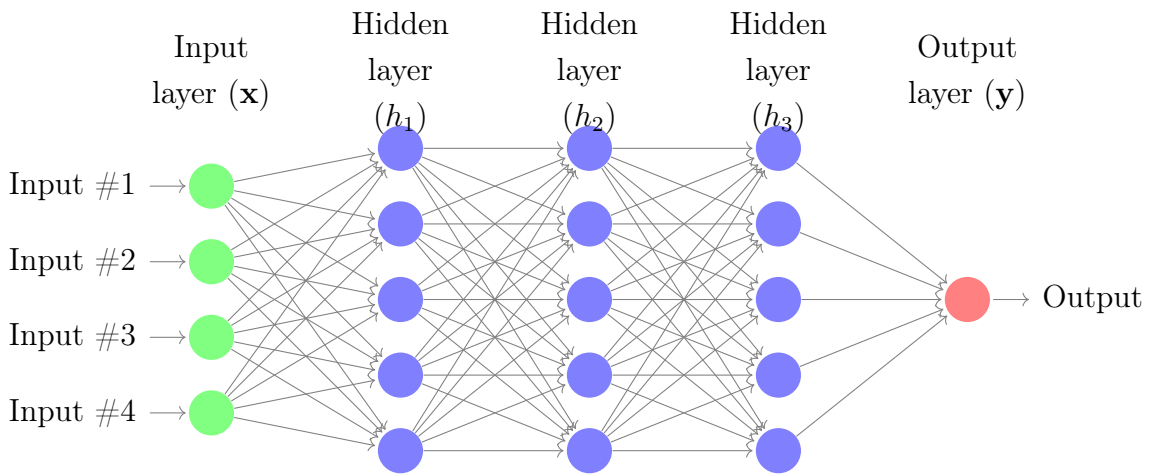


Figure 2.1: Multi-Layer Perceptron Neural Network

To summarize the figure, each neuron h_j^1 in the first hidden layer (L_1) is connected to each input neuron in the input layer $\mathbf{x} = x_1, x_2, \dots, x_n$, each of which have specific weights associated to each input-hidden layer connection $w_n = w_{1j}, w_{2j}, \dots, w_{nj}$ and a bias term θ_{ij} . The weighted sum of the inputs is then calculated using the following formula:

$$z_j = \sum_{i=1}^n x_i w_{ij} + \theta_{ij} \quad (2.1)$$

The value of z_j is then passed through an activation function $f(z_j)$, that is the key decision making unit, which depending on the task to be accomplished can take different forms. Consider for instance the following ‘log-sigmoid’ activation function:

$$f(z_j) = \frac{1}{1 + e^{-z_j}} \quad (2.2)$$

then the output of h_j^1 is computed as

$$h_j^1 = \frac{1}{1 + e^{-(\sum_{i=0}^n x_i w_{ij} + \theta_{ij})}} \quad (2.3)$$

The output from each node in the hidden layer is computed and the resulting values are used as inputs for the following layer. The outputs from the last hidden layer is considered as inputs for the output layer. For classification softmax layer is used as the output layer as follows:

$$y_i = softmax(o_i) = \frac{exp(o_i)}{\sum_j exp(o_j)} \quad (2.4)$$

2.3 Activation Functions

Table 2.1 shows the corresponding equation and graphical representation of the output that are obtained for each one of the most popular activation functions used in neural network models. The first type of activation function is the linear function, which is easy to compute but unable to learn complex nonlinear behavior. The binary step is an activation function that is usually used to compute the output of a classification network and the result of the function is either 0 or 1. In contrast, the three following rows, in table 1, represent the most popular activation functions used to describe non-linearities to the network. The sigmoid function computes an S-shaped output, which will return an output between 0 and 1 depending on the value of the weighted sum of the inputs. The tanh is also an S-shaped function, though instead of the output being between 0 and 1 it ranges from -1 to 1. The output range that the tanh function computes is often more desirable than the sigmoid due to the fact that it is zero-centered. The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the

tanh graph. Lastly, the rectified linear unit (ReLU) is currently the most used activation function in current neural network applications (Nair and Hinton [2010]). There are two main advantages of using ReLU as activation for units in deep neural networks as follows: i- the non-linearity of the function will allow the network to create sparse representations which results in transferring only important information through the network and ii- the form of the activation function dictates a linear dependency of the gradient with respect to the error that does not decay rapidly to zero as in the case of sigmoids thus avoiding the vanishing gradient problem (Glorot and Bengio [2010]).

It is important to mention that any of the activation units presented in Table 1 may be used in any of the layers of the network, either hidden or output. In addition, to all the previous layers, in many classification problems it is often desired to make the output vector a probabilistic distribution over n different classes to account for the presence of noise and stochastic disturbances. Such probabilistic classification can be generally addressed by applying softmax activation units in the output layers. The formula of the softmax function is obtained by normalizing z_i , where z_i is defined as

$$z_i = \sum x_i w_i + \theta_i \tag{2.5}$$

which then is applied to the softmax activation function as

$$y_i = f(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{2.6}$$

A strong prediction will result in only one category having a value close to 1, while a weak prediction will have the probability distributed among several categories.

2.4 Autoencoder Neural Networks (AE-NNs)

A traditional AE-NN is a neural network model composed of two parts: encoder and decoder, as shown in Figure 2.2. An AE is trained in an unsupervised fashion to extract

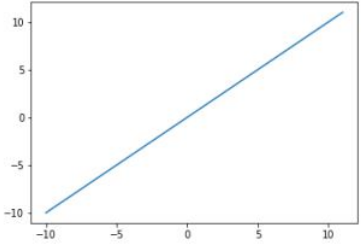
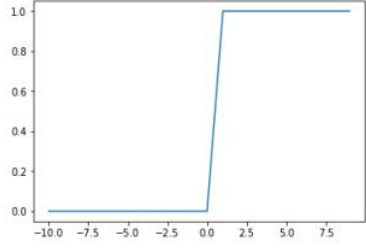
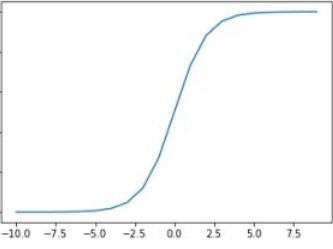
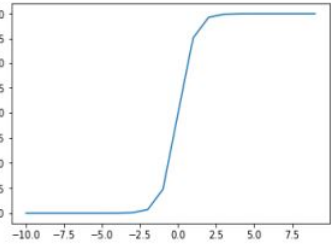
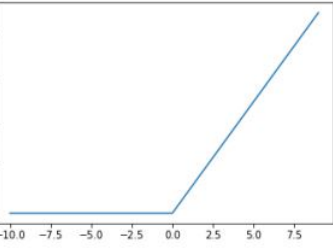
Name	Equation	Lab
Linear	$f(z) = az + b$	
Binary Step	$f(z) = \begin{cases} 0 & \text{for } z < 0 \\ 1 & \text{for } z \geq 0 \end{cases}$	
Sigmoid	$f(z) = \frac{1}{1+e^{-z}}$	
Tanh	$f(z) = \tanh(z) = \frac{2}{1+e^{-2z}} - 1$	
Rectified Linear Unit (ReLU)	$f(z) = \max(0, z) = \begin{cases} 0 & \text{for } z < 0 \\ z & \text{for } z \geq 0 \end{cases}$	

Table 2.1: Commonly used activation functions and plots

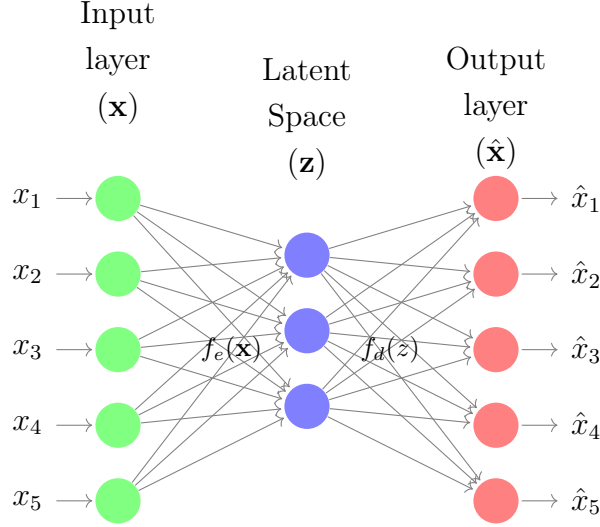


Figure 2.2: Traditional single layer Autoencoder Neural Network (AE-NN)

underlying patterns in the data and to facilitate dimensionality reduction. The encoder is trained so as to compress the input data onto a reduced latent space defined within a hidden layer and the decoder uncompresses back the hidden layer outputs into the reconstructed inputs. Let us consider the inputs to an AE-NN $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3 \ \dots \ \mathbf{x}_N]^T \in \mathbb{R}^{N \times d_x}$, 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 can be represented as follows:

$$\mathbf{z}_i = f_e(\mathbf{W}_e \mathbf{x}_i + \mathbf{b}_e) \quad (2.7)$$

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

$$\hat{\mathbf{x}}_i = f_d(\mathbf{W}_d \mathbf{z}_i + \mathbf{b}_d) \quad (2.8)$$

where f_d is a chosen activation function for the decoder, $\mathbf{W}_d \in \mathbb{R}^{d_x \times d_z}$ and $\mathbf{b}_d \in \mathbb{R}^{d_x}$ is a decoder weight 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 as an example here. The AE-NN is trained based on the following minimization problem:

$$l_{AE}(\mathbf{x}, \mathbf{W}_d \mathbf{W}_e \mathbf{x}) = \frac{1}{2N} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 = \frac{1}{2N} \sum_{s=1}^N (\mathbf{x}_s - \hat{\mathbf{x}}_s)^2 \quad (2.9)$$

where N is the number of samples.

2.5 Long-Short Term Memory (LSTM) Units

The LSTM unit is composed of three gated units and a memory cell [Hochreiter and Schmidhuber \[1997\]](#). Figure 2.3 shows a single LSTM unit that includes four major gates: the forget gate (\mathbf{f}_t), the input gate (\mathbf{i}_t), the output gate (\mathbf{o}_t) and the update gate (\mathbf{g}_t). The key component of the LSTM unit is the memory cell ($\mathbf{c}_t \in \mathbb{R}^{d_h \times 1}$) that is responsible for storing critical long term dependencies learned over time. The input gate (\mathbf{i}_t) is responsible for evaluating which part, if any, of the past historical data should be kept. Thus, the objective of the input gate is to allow the network to keep only relevant information from the previous time steps and discard the rest for a sample i .

Subsequently, the information that is worth keeping is determined by the memory cell (\mathbf{c}_t). The process of identifying information and storing in the memory cell consists of two parts: new information that is recorded and information that is discarded. The information that should be discarded from previous cell state \mathbf{c}_{t-1}^i is determined by the forget gate (\mathbf{f}_t), which is responsible for forgetting previously stored cell state values that have lost their relevance. Then new relevant information is added and existing cell-state values are updated by first selecting which values to update using the input gate \mathbf{i}_t^i and the output from the input gate is then multiplied by the new information generated by the update gate \mathbf{g}_t^i . Ultimately, the output \mathbf{h}_t is computed at every time step from the information contained in the memory cell and it is further gated by an output gate according to its relative importance or relevance. The mathematical equations describing these gating operations are as follows:

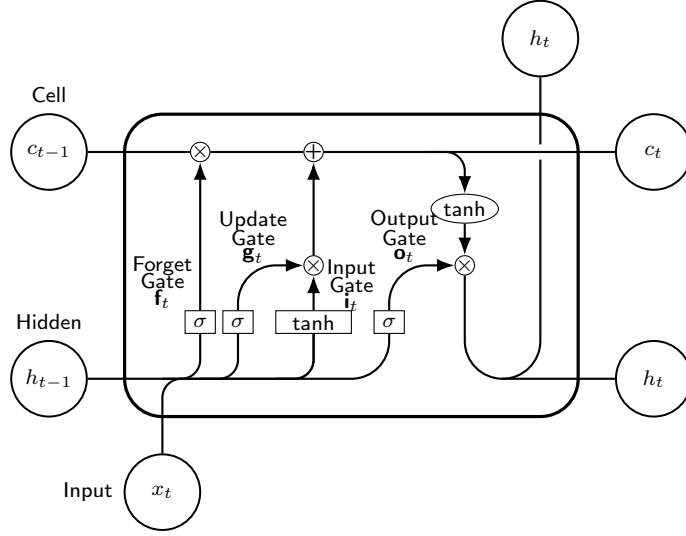


Figure 2.3: Schematic of a LSTM memory cell

$$\begin{aligned} \mathbf{i}_t^i &= \sigma(\mathbf{W}_i \mathbf{x}_t^i + \mathbf{R}_i \mathbf{h}_{t-1}^i + \mathbf{b}_i) \\ \mathbf{g}_t^i &= \tanh(\mathbf{W}_g \mathbf{x}_t^i + \mathbf{R}_g \mathbf{h}_{t-1}^i + \mathbf{b}_g) \end{aligned} \quad (2.10)$$

$$\mathbf{c}_t^i = \mathbf{f}_t^i \odot \mathbf{c}_{t-1}^i + \mathbf{i}_t^i \odot \mathbf{g}_t^i \quad (2.11)$$

where $\sigma()$ and $\tanh()$ are the element-wise sigmoid and hyperbolic tangent functions respectively.

$$\begin{aligned} \mathbf{o}_t^i &= \sigma(\mathbf{W}_o \mathbf{x}_t^i + \mathbf{R}_o \mathbf{h}_{t-1}^i + \mathbf{b}_o) \\ \mathbf{h}_t^i &= \mathbf{o}_t^i \odot \tanh(\mathbf{c}_t^i) \end{aligned} \quad (2.12)$$

where $\mathbf{R} = [\mathbf{R}_f \ \mathbf{R}_i \ \mathbf{R}_g \ \mathbf{R}_o]^T \in \mathbb{R}^{4d_h \times d_h}$ are known as recurrent weights, $\mathbf{W} = [\mathbf{W}_f \ \mathbf{W}_i \ \mathbf{W}_g \ \mathbf{W}_o]^T \in \mathbb{R}^{4d_h \times d_x}$ are all the input weights, $\mathbf{b} = [\mathbf{b}_f \ \mathbf{b}_i \ \mathbf{b}_g \ \mathbf{b}_o]^T \in \mathbb{R}^{d_h \times 1}$ are the bias parameters.

2.6 Generalization, Regularization and Dropout

One of the major challenges for neural network models is to provide them with the ability to predict accurate results for inputs that have not been used for model calibration. This capacity of models to predict with data never seen before during model calibration is referred to as generalization. To achieve good model generalization ability, the data is generally divided into three sets, a training set, a validation set and a test set. During the training procedure, the neural network is presented with the training data only and the model parameters are estimated so as to minimize a loss function. Validation data-set is used to choose the optimal hyper-parameters for a particular structure of the neural network. After minimizing the loss function, the generalization error is evaluated by using the trained neural network on test dataset. In order to evaluate the performance of the model, the gap between the testing and training error is particularly assessed in order to identify whether the model might be under-fitting or over-fitting the information. The under-fitting problem arises when the model is not able to reduce the training error to an acceptable standard. On the other hand, over-fitting arises when the model over fits the noisy data and thus becomes very sensitive to noise and consequently unable to generalize accurately.

When training a model, the common practice is to simultaneously monitor the training error and validation error until either one of them stops decreasing. If the errors achieved are not satisfactory the training procedure will continue with the goal of improving the model's feature extraction capabilities or by taking actions to prevent over-fitting. Several techniques have been developed to prevent over-fitting in neural networks, including regularization, early stopping and dropout.

Regularization is a broad concept that has been applied in many statistical models in a number of ways. However, in deep neural networks regularization refers to penalizing a particular norm of a parameter/s within the objective function of the optimization problem, e.g. the sum of square errors between predictions and data, with the goal of eliminating some of the parameters that are required to fit the data. Therefore, in order to calculate

the total gradient of the objective function, the gradients of both the error function and the regularization functions have to be calculated. Mathematically, in the objective function the regularization term is added as follows:

$$J(\theta) = Error + \alpha R(\theta) \tag{2.13}$$

where α is a hyper parameter that weights the norm penalty $R(\theta)$. Typically, in deep neural networks two types of norm are used for regularization of the parameters i.e. L_1 and L_2 norms.

The main difference between these two norms is that L_1 generally leads to a sparse representation where many parameters are left unchanged in the training procedure, whereas the L_2 norm tends to distribute the changes required for minimization of the objective more equally among the parameters. The early stopping algorithm is a simple way for preventing over-fitting when training deep neural networks. In this method of regularization, both the training error and the validation error are simultaneously monitored to ensure a certain balance between them at the solution. In theory, at the start of training, both errors will start decreasing. Then, if after certain time the validation error will start increasing but the validation error does not improve, the network is not trained any further ([Prechelt \[1998\]](#)).

Dropout was developed as a way to inexpensively prevent over-fitting in deep neural networks with large number of parameters ([Srivastava et al. \[2014\]](#)). The theory behind dropout is based on randomly dropping units in the neural network or preventing units from adapt during the training phase with the expectation that the model will generalize better in the absence of information. Technically, during each iteration or batch of iterations during the training procedure the value of each individual neuron is either kept based on a calculated probability p (hyper parameter) or dropped out by reducing its value to zero and cutting all the incoming and outgoing information to the neuron. It is important to denote that the process of dropping a neuron is only done during the training phase but when the model is tested for generalization the hyper parameter p value associated to the neuron that was dropped is changed back to 1 as the neuron is used. Some of the most

important advantages for dropout is that it is a computationally inexpensive operation , it is able to work well with any type of model (feedforward, recurrent, etc.) and it is compatible with other regularizers.

Chapter 3

Deep Learning for Classification of Profit-based Operating Regions in Industrial Processes

Overview¹

A classification approach is proposed for finding ranges of process inputs that result in corresponding ranges of a process profit function using Deep Learning. Two Deep Learning Tools are used to formulate models for use in classification, based on either supervised learning or unsupervised learning approaches. The supervised learning models are based on Long Short Term Memory Networks (LSTM) and Multi-Layer Perceptron (MLP) Networks while the unsupervised learning model consists of an Autoencoder Neural Network (AE-NN) connected to a Support Vector Machine classifier. An algorithm referred to as Sequential Layer-wise Relevance Propagation for Pruning (SLRPFP) is proposed and ap-

¹Adapted from Agarwal, Piyush, et al. "Deep learning for classification of profit-based operating regions in industrial processes." *Industrial & Engineering Chemistry Research* 59.6 (2019): 2378-2395

Piyush Agarwal and Hector Budman. "Classification of profit-based operating regions for the tennessee eastman process using deep learning methods." *IFAC-PapersOnLine*, 52(1):556-561, 2019"

plied to the aforementioned models for selecting relevant inputs and for pruning the Neural Networks (NNs) and its inputs such that the test accuracy at every step of the proposed sequential algorithm is maintained or even improved. It is also shown that the selected inputs from the proposed algorithm (SLRPFP) provides important process insights on the productivity i.e. profit-based objective function. The approaches are illustrated for the Tennessee Eastman Process (TEP) and for an industrial vaccine manufacturing process (industrial process). The efficacy of the proposed supervised and unsupervised deep learning approaches over linear model-based classification methods that are based on linear Dynamic Principal Component Analysis (DPCA) combined with Multi-class Support Vector Machines (MSVM) classification are shown by comparing the performance for both TEP and a vaccine manufacturing process.

3.1 Introduction

The work proposes supervised and unsupervised learning approaches to identify regions of process inputs that result in corresponding regions, i.e ranges of values, of process profit. This type of classification problem can help in identifying input variables that significantly affect defined profit function as well as for identifying ranges of inputs' values that result in high/low profit. Linear multivariate statistical models such as Principal Component Analysis (PCA) and Partial Least Squares (PLS) have been previously used for classification problems such as fault diagnosis. [Shams et al. \[2011b\]](#), [Kresta et al. \[1991\]](#), [Wise and Gallagher \[1996\]](#), [MacGregor et al. \[1994\]](#), [Li et al. \[2011\]](#). However, these methodologies may not be sufficiently accurate to model highly non-linear dynamic behavior that is common in chemical processes. The non-linear Kernel PLS based classification method has been proposed for similar problems but the results are sensitive to the selection of parameters that define the non-linear kernel functions [Rosipal et al. \[2003\]](#). Moreover, Support Vector Machines (SVMs) classification models have been used before for non-linear classification problems but their computational complexity increases with data samples. Also, it is difficult to provide physical interpretation of the SVM results since SVM does not compress the data into a lower dimensional space which generally facilitates interpretations. This work uses novel deep learning tools that have the ability to compress data into a lower

dimensional space [Hinton and Salakhutdinov \[2006\]](#) while accurately describing non-linear behaviour.

Deep learning techniques involve the use of multi-layered neural network (NN) models that are calibrated with process data. Although NNs were proposed several decades ago [Minsky and Papert \[1972\]](#) the interest in them was previously limited following the recognition that they required large datasets for training, learning of parameters is slow and the optimization required for model calibration often converged to a local minima of the loss-objective function. In particular, it was recognized that these convergence problems were due to the use of gradient based algorithms with random initialization of parameters [Glorot and Bengio \[2010\]](#). The resurgence of interest in NNs stems from a significant increase in computational power along with several theoretical developments that address some of the aforementioned limitations in learning of NNs. These developments involve new NNs architectures, such as Autoencoder NNs (AE-NN) [Poultney et al. \[2007\]](#), Long short term memory NNs (LSTM-NN) [Hochreiter and Schmidhuber \[1997\]](#), Convolutional NNs (CNNs) [Chellapilla et al. \[2006\]](#), General Adversarial Networks (GANs) [Goodfellow et al. \[2016\]](#) and novel “greedy” learning approaches [Bengio et al. \[2007\]](#) that are used to learn these deep NNs efficiently. In these greedy approaches each layer of the multi-layered (deep) NNs is pre-trained in an unsupervised fashion with the goal to provide good initial guesses for further tuning from the first layer (input) to the last layer (outputs) in a supervised manner.

Although deep learning modeling techniques have been applied extensively in the area of speech recognition and vision problems they have not been investigated, to the best knowledge of the authors, for the type of classification problems i.e. classification of profit-based operating regions as considered in this work. The current work investigates the ability of deep learning techniques to two case studies: the Tennessee Eastman problem (TEP) that has been widely used for comparison of control and fault detection approaches and an industrial study involving a fermentation process used in vaccine manufacturing. [Lv et al. \[2016\]](#) and [Heo and Lee \[2018\]](#) recently applied deep learning NNs for fault diagnosis in the Tennessee Eastman Problem and reported significant improvements in detection and diagnosis compared to linear Dynamic Principal Component Analysis (DPCA), Modified

Partial Least Squares (MPLS) and independent component analysis (ICA) based models. However, classification problems as presented in this work were not addressed in these earlier works.

A key difference between the two case studies is that while the TEP is a continuous system operated around nearly constant operating points, the fermentation process is a batch operation involving finite sets of data and large changes in operating conditions thus necessitating a modified approach as compared to the continuous case. In general data analysis for batch processes pose additional challenges such as unsynchronized and unequal batch lengths of process datasets. Also, seasonal changes such as changes in environmental temperature, pressure and feed composition or production motivated termination of batches by operators in a plant often result in non-uniform process operation for different batches. The uneven length duration causes additional numerical challenges for empirical modeling. In addition, critical operational changes and certain offline measurements e.g. the time for change from batch to fed-batch mode and the acquisition times of certain measurements may occur at different times for different batches. Many different methodologies have been previously used to address the issue of unequal and unsynchronized input variable profile such as Indicator Variable (IV) Method, Correlation Optimization Warping (COW) and Dynamic Time Warping (DTW). In indicator variable method, a variable is chosen to represent the maturity of the evolving batch and the batch trajectories are aligned using interpolation. Another method to deal with batches of differing duration is called Dynamic time warping (DTW) which make use of a distance measure to compresses and expands the similar patterns such that similar features are aligned. In this work DTW is used as a pre-processing step to align the batches for industrial vaccine manufacturing batch process. Few studies for aligning batch polymerization and fermentation datasets have been reported in [Ündey et al. \[2002, 2003\]](#).

A desirable property of a machine learning algorithm is to have the ability for providing process insight. In view of the large amount of data that are regularly collected from industrial processes and considering that this data is often highly correlated, the compression of data into a lower dimensional space is very useful for gaining process understanding,

for process visualization and for reducing the sensitivity of regressed models to noise and model structure error. Linear multivariate statistical methods such as PCA and PLS have been used to model correlated data but are not sufficiently accurate to represent highly non-linear correlations among inputs and between inputs to outputs. Non-linear PLS has been used but it is susceptible to the choice of the non-linear kernel functions. Alternatively SVM has been used for non-linear problems but it is computational demanding for large data sets and it cannot provide process understanding or visualization since it does not perform data compression. On the other hand Deep Learning NN architectures have the ability to deal with correlated data and can provide process insight since they perform data compression similar to other multivariate statistical methods [Hinton and Salakhutdinov \[2006\]](#).

Dynamic correlation is a key feature of time-series data. Long Short Term Memory NN (LSTM-NN) is a recurrent neural network (RNN) that utilizes a different units known as “memory cells” which maintains information for long periods of time and are capable of describing dynamically correlated data. LSTMs have proved to be effective to overcome the vanishing gradient problem that occurs during training of models with the use of different memory gates (input, output and forget gates). However, LSTM models and generally all other NN architectures have a large number of parameters which makes them computationally expensive and prone to data over-fitting. “Pruning” techniques have been proposed to eliminate irrelevant nodes, i.e. nodes that do not contribute significantly for minimizing the loss function. For example, [Han et al. Han et al. \[2015, 2016\]](#) trained the NN with additional L_1/L_2 penalty functions to eliminate neurons [Collins and Kohli \[2014\]](#) thus reducing the computational complexity. Most reported pruning methods are based on eliminating nodes while minimizing a non-linear loss function thus potentially converging to local minima. Instead, we propose in this study a novel method for pruning input variables of the NN by an algorithm to be referred to as Sequential Layer-wise Relevance Propagation for Pruning (SLRPFP) [Agarwal and Budman \[2019\]](#), preliminary results of which have been published earlier and have been extended further in this work, that is based on evaluation of the averaged-relative relevance of input variables. Beyond its ability of pruning input variables and the network it will also be shown that this proposed

methodology provides information on the inputs that have large and significant impact on profit and can be used to identify regions of input conditions that results in high or low profit for a defined profit-based objective function.

Following the above, three deep learning based classification models are proposed in this work : i) a supervised learning method that uses either an LSTM network or an Multi-Layer Perceptron (MLP) network with input-pruning by proposed algorithm (SLRPFP) and ii) an unsupervised classification model that uses an Autoencoder neural network (AE-NN) for feature extraction and SVM for classification. These deep learning NN classification models are then compared to a linear DPCA model combined with an SVM classifier. This work presents the following novel contributions: i) the use of the deep learning algorithms for profit based classification on an industrial vaccine manufacturing process (batch process), ii) a pruning approach (SLRPFP algorithm) combined with a threshold selection algorithm and its impact on classification accuracy, iii) a study to compare the performance of supervised and unsupervised deep learning methods with a linear unsupervised classification approach for both simulated and an industrial dataset case studies and iv) derivation of process insights regarding which inputs have large effect on profit function based on the proposed relevance-based pruning algorithm (SLRPFP) both for continuous (TEP) and an industrial batch process.

The rest of this chapter is organized as follows: [3.2](#) provides a brief overview on different NN architectures such as LSTM-NN, AE-NN, MLP-NN along with Layer-wise Relevance Propagation (LRP) for computation of relevances of input variables. The proposed algorithm for pruning is presented in [3.3](#). [3.4.1](#) and [3.4.2](#) briefly describes the Tennessee Eastman Benchmark Process (TEP) and the fermentation process for vaccine manufacturing, discusses the supervised and unsupervised deep learning methods for the classification of different ranges of productivity costs and shows the results of the comparison of the two deep learning approaches and the linear DPCA based method. Finally, [3.5](#) summarizes the work and provides the concluding remarks.

3.2 Preliminaries

This section briefly reviews LSTM classification NN and a method to compute relevances for input variables using NNs.

3.2.1 Long Short-Term Memory Neural Networks (LSTM-NN)

LSTM-NN are a special deep learning architecture of recurrent neural networks (RNNs) capable of learning long-term correlations. A schematic of a vanilla LSTM cell is shown in Figure 3.1. The functionality is somewhat similar to auto-regressive dynamic models that are widely used for modelling multivariate time-series with the addition of different memory gates to mitigate the problem of vanishing/exploding gradients that occurs when training RNNs by gradient descent algorithms Hochreiter and Schmidhuber [1997], Chung et al. [2014]. The four memory gates that allows the LSTM NN to learn long term dependencies by selectively memorizing the relevant information and forgetting the redundant features to the cell state for a sample i as $\mathbf{c}_t^i \in \mathbb{R}^{d_h \times 1}$ corresponding to the output/class-label are input \mathbf{i}_t^i , forget \mathbf{f}_t^i , update \mathbf{g}_t^i and output \mathbf{o}_t^i gates. Consider the input at time t as \mathbf{x}_t^i , $t = 1, 2, \dots, T \in \mathbb{R}^{d_h \times d_x}$. The LSTM cell determines the amount of past information to be discarded from the previous cell state \mathbf{c}_{t-1}^i . Element-wise sigmoid activation function in \mathbf{f}_t^i (refer forget gate) is a number between 0 and 1 for each of the corresponding elements of \mathbf{c}_{t-1}^i as:

$$\mathbf{f}_t^i = \sigma(\mathbf{W}_f \mathbf{x}_t^i + \mathbf{R}_f \mathbf{h}_{t-1}^i + \mathbf{b}_f) \quad (3.1)$$

Subsequently, it is required to add new relevant information and update the existing cell-state values. This is implemented by first selecting which values to update using the input gate \mathbf{i}_t^i and the output from the input gate is then multiplied by the new information generated by the update gate \mathbf{g}_t^i as follows:

$$\begin{aligned} \mathbf{i}_t^i &= \sigma(\mathbf{W}_i \mathbf{x}_t^i + \mathbf{R}_i \mathbf{h}_{t-1}^i + \mathbf{b}_i) \\ \mathbf{g}_t^i &= \tanh(\mathbf{W}_g \mathbf{x}_t^i + \mathbf{R}_g \mathbf{h}_{t-1}^i + \mathbf{b}_g) \end{aligned} \quad (3.2)$$

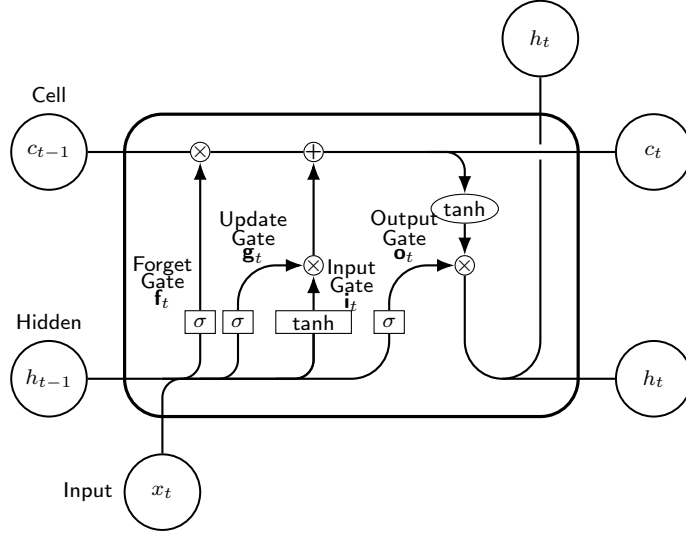


Figure 3.1: Schematic of a LSTM memory cell

The update of the cell-states are carried out by incorporating the new information and by forgetting the redundant features according to:

$$\mathbf{c}_t^i = \mathbf{f}_t^i \odot \mathbf{c}_{t-1}^i + \mathbf{i}_t^i \odot \mathbf{g}_t^i \quad (3.3)$$

where $\sigma()$ and $\tanh()$ are the element-wise sigmoid and hyperbolic tangent functions respectively. $\mathbf{W}_f, \mathbf{W}_i$ & \mathbf{W}_g are the input weight matrices, $\mathbf{R}_f, \mathbf{R}_i$ & \mathbf{R}_g , are the recurrent weights and \odot is a Hadamard product. Finally, the output value $\mathbf{h}_t^i \in \mathbb{R}^{d_h \times 1}$ of the LSTM cell is based on the updated cell-state values as shown below:

$$\begin{aligned} \mathbf{o}_t^i &= \sigma(\mathbf{W}_o \mathbf{x}_t^i + \mathbf{R}_o \mathbf{h}_{t-1}^i + \mathbf{b}_o) \\ \mathbf{h}_t^i &= \mathbf{o}_t^i \odot \tanh(\mathbf{c}_t^i) \end{aligned} \quad (3.4)$$

where $\mathbf{R} = [\mathbf{R}_f \ \mathbf{R}_i \ \mathbf{R}_g \ \mathbf{R}_o]^T \in \mathbb{R}^{4d_h \times d_h}$ are known as recurrent weights, $\mathbf{W} = [\mathbf{W}_f \ \mathbf{W}_i \ \mathbf{W}_g \ \mathbf{W}_o]^T \in \mathbb{R}^{4d_h \times d_x}$ are all the input weights, $\mathbf{b} = [\mathbf{b}_f \ \mathbf{b}_i \ \mathbf{b}_g \ \mathbf{b}_o]^T \in \mathbb{R}^{d_h \times 1}$ are the bias parameters. A sequence to label classification using LSTM units is implemented using the outputs at the last sequence element ($\mathbf{h}_T^i \in \mathbb{R}^{d_h \times 1}$). In this work, the LSTM outputs are used as inputs to a fully-connected layer as shown in Figure 3.2. The fully connected layer has the same number of output nodes as the number of defined classes. The last layer

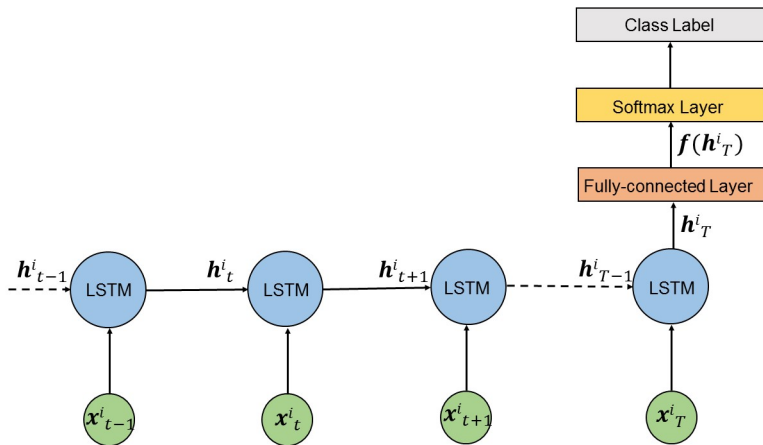


Figure 3.2: Sequence to label classification using LSTM-NN

consists of a softmax layer which translates the outputs $f(\mathbf{h}_T^i)$ of the fully-connected layer that satisfies $\text{softmax}(f(\mathbf{h}_T^i)) \in [0, 1]$ and $\sum_{k=1}^p (\text{softmax}(f(\mathbf{h}_T^i))) = 1$, where p is the number of total class labels. The class k is decided based on the largest $\text{softmax}(f(\mathbf{h}_T^i))$ value.

3.2.2 Layer-wise Relevance Propagation (LRP)

Layer-wise Relevance Propagation (LRP) was introduced by [Bach et al., \(2015\)](#) [Bach et al. \[2015\]](#) to assess the relevance of each input variable or features with respect to outputs. It is based on a layer-wise relevance conservation principle where each relevance $f(\mathbf{x}^i)$, $i = 1, 2, \dots, n$, where n is the number of samples, is calculated by propagating the output scores towards the input layer of the network. Previously, it has been used in the area of health-care for attributing a relevance value to each pixel of an image to explain the relevance in a image classification task using deep NNs [Yang et al. \[2018\]](#). LRP has also been implemented to explain the predictions of a NN in the area of sentiment analysis using LSTMs [Arras et al. \[2017\]](#) but it has not been studied for pruning input nodes as in the current study. To compute the relevance of input variables in LSTM-NN, the time-series for each input variable are organized in data packets while the class corresponding to each packet is available. Subsequently the score value of the corresponding class for a specific packet

$f(\mathbf{x}^i)$, computed during the training of the NN model, is propagated through the network towards the input. Depending on the nature of the connection between layers and neurons, a layer-by-layer relevance score is computed for each intermediate lower-layer neuron. The relevances computed for LSTM-NN classification model are averaged over time since the model is working on a time horizon, i.e. recursive in nature, so there is no meaning to relevances at each interval and is presented later in this chapter in 3.4.1 and 3.4.2. While relevances for MLP-NN model as shown in 3.4.2 are computed for each variable at each time-step. Different LRP rules have been proposed for attributing relevance for the input variables. In this work we use ϵ rule Bach et al. [2015] for computing relevances as is given by:

$$R_{i \leftarrow j} = \frac{w_{ij}x_i}{z_j + \epsilon \cdot \text{sign}(z_j)} R_j \quad (3.5)$$

where ϵ is used as a stabilizer to prevent numerical instability when z_j is close to zero and w_{ij} are the weights connecting lower layer neuron i and upper-layer neuron j .

3.3 Proposed Methodology: Sequential Layer-Wise Relevance Propagation for Pruning (SLRPFP)

This section proposes a novel method for pruning input variables and thereby reducing the size of NNs thus reducing the number of parameters to be estimated. The proposed algorithm is based on identifying relevant inputs for a particular classification task thus eliminating the non-contributing input variables. Algorithm 1 illustrates the steps for the implementation of Sequential Layer-wise Relevance for Pruning (SLRPFP) for identifying the important input variables and Algorithm 2 describes the methodology to determine the threshold for eliminating the irrelevant variables corresponding to the profit-based objective function.

The following section presents case-studies to demonstrate the effectiveness of the proposed procedure to prune the input variables and derive important process insights based on defined profit-based objective function.

Algorithm 1 Sequential Layer-wise Relevance Propagation for Pruning (SLRPFP)

- 1: Calibrate the NN on the training dataset and compute the classification accuracy on the test data.
 - 2: Evaluate average relative relevance of input variables using all correctly classified samples using LRP.
 - 3: A threshold value related obtained as per the procedure described in Algorithm 2 is used as the maximum average relevance below which the input variables are considered irrelevant and are discarded.
 - 4: Repeat step 1 i.e. re-train the NN with the remaining variables. Select the optimal number of hidden layer neurons by using validation data and early stopping technique so as to maintain the same or higher test accuracy.
 - 5: Stop the procedure when the relevances of remaining input variables are above the computed threshold value.
-

3.4 Results and Discussions

Two case studies are presented: the Tennessee Eastman Process (TEP) and an industrial vaccine manufacturing process (Sanofi Pasteur: antigen manufacturing process). While the TEP study uses simulated data the vaccine manufacturing process is based on actual industrial data. The key difference between the studies is that the TEP is continuous whereas the antigen manufacturing process is operated in batch mode. It is shown that the proposed algorithm is useful in acquiring information on inputs that have large and significant effect on profit. The relative advantages of deep learning models over models based on Dynamic PCA (DPCA) and Batch-Dynamic PCA (BDPCA) (Vaccine Manufacturing Process) are also presented.

3.4.1 Case Study 1: Tennessee Eastman Process (Simulated Case Study)

The TEP involves different unit operations including a vapor-liquid separator, a reactor, stripper a recycle compressor and a condenser. Four gaseous reactants (A, B, C and D)

Algorithm 2 Determining Threshold for Sequential Layer-wise Relevance Propagation

- 1: Evaluate Relevances $\mathbf{R}_c(\mathbf{x}; \mathbf{f}_c)_j = [R_{c_1}(x_1; f_c), R_{c_2}(x_2; f_c), \dots, R_{c_n}(x_n; f_c)]_j \in \mathbb{R}^n$, $i = 1, 2, \dots, n$ for each input feature x_i are calculated for the j^{th} sample with respect to a classification task.
- 2: Average the relevance scores over all the correctly classified samples. Therefore, the final input relevances with respect to the overall classification task c can be calculated as follows:

$$\mathbf{R}_c = \frac{1}{N_c} \sum_{j=1}^{N_c} |R_c(\mathbf{x}; \mathbf{f}_c)_j| \quad (3.6)$$

where N_c is the number of correctly classified samples in the training dataset. Furthermore, the least relevant input features are pruned based on the average relevance scores for all input variables calculated with Equation 3.6. In practise a threshold of $\lambda \times \max(\mathbf{R}_c)$ is chosen to prune the irrelevant variables where λ is an hyper-parameter that is determined by using the validation dataset (heuristically λ is chosen as 0.01 as the starting value).

- 3: Relevance of input variables below the threshold are removed from the dataset and the network is re-trained until the same or higher validation accuracy is achieved. If the desired validation accuracy is not achieved, the threshold is decreased.
-

forms two liquid products streams (G and H) and a by-product (F). A schematic of the Tennessee Eastman process is illustrated in Figure 3.3. [Downs and Vogel \[1993\]](#) reported the original simulator for this process that has been widely used as a benchmark for control and monitoring studies.

Although several TEP simulators are available, in this work the one developed by [Larsson et al. \[2001\]](#) was used. We have slightly modified the original control settings in order to simulate the process for different ranges of profit since the goal in the current study is to classify the inputs according to their resulting profit. The simulator involves 52 input variables of which 3 manipulated variables (Compressor Recycle Valve (XMV(5)), Stripper

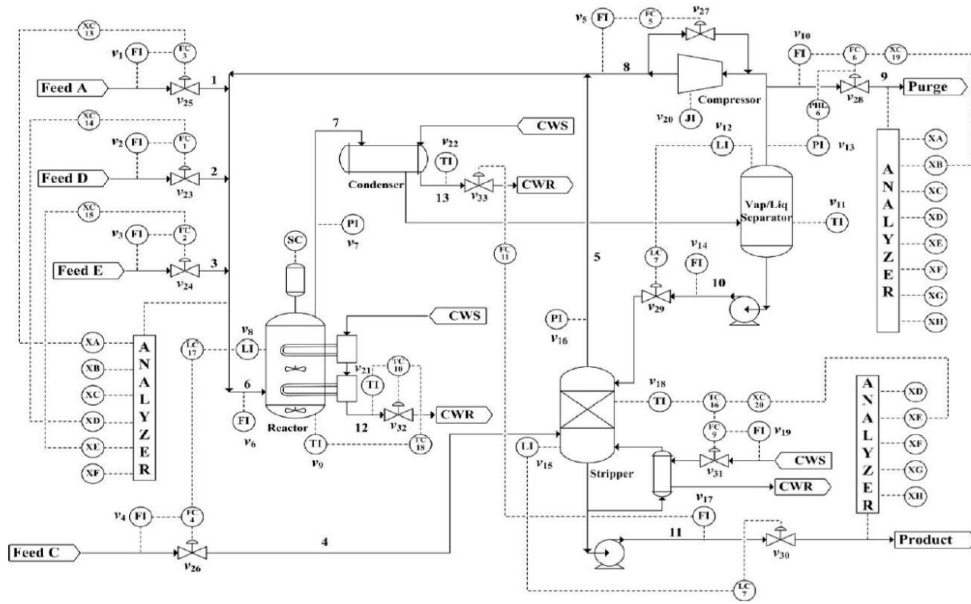


Figure 3.3: Tennessee Eastman plant process
(Downs & Vogel, 1993)

Steam Valve (XMV(9)) and Agitator Speed XMV(12)) were discarded initially (number of input variables = 50). The process profit for this case study is determined by the operating costs of the plant i.e. cost of productivity (COP). Downs and Vogel presented an equation for COP (operating costs) of the following form [Ricker \[1995\]](#):

$$\begin{aligned}
 \text{COP}_t = & (0.0536 \times \text{XMEAS}(20)_t) + (0.0318 \times \text{XMEAS}(19)_t) + \dots \\
 & (0.4479 \times \text{XMEAS}(10)_t) [(2.209 \times \text{XMEAS}(29)_t) + (6.177 \times \text{XMEAS}(31)_t) + \dots \\
 & (22.06 \times \text{XMEAS}(32)_t) + (14.56 \times \text{XMEAS}(33)_t) + (17.89 \times \text{XMEAS}(34)_t) + \dots \\
 & (30.44 \times \text{XMEAS}(35)_t) + (22.94 \times \text{XMEAS}(36)_t)] + (4.541 \times \text{XMEAS}(46)_t) [\dots \\
 & (0.2206 \times \text{XMEAS}(37)_t) + (0.1456 \times \text{XMEAS}(38)_t) + (0.1789 \times \text{XMEAS}(39)_t)]
 \end{aligned} \tag{3.7}$$

where $\text{COP}(t)$ is in \$/hr. To simulate different ranges of cost values, different process faults were introduced that result in corresponding changes in cost. Afterwards 8 datasets were generated, 1 normal operation and 7 each involving one known fault (IDV(1)-IDV(7), refer

COP (\$/hr)	Low (High Profit)	Intermediate	High (Low Profit)
Case 1	> 89.68	89.68 – 142.6	< 142.6
Case 2	> 108	108 – 130	< 130

Table 3.1: Profit-based defined classes for COP

Table S1). The respective datasets were produced for a total simulation time of 800 hours, i.e. a 100-hour duration for each dataset. Each fault was activated at the start of the corresponding 100-hour time period and data samples were collected at a sampling rate of 100 samples/hour (total number of samples 8×10^5 per dataset). Each of these datasets resulted in various ranges of COP values and were considered as the different classes to be targeted (refer Table 3.1).

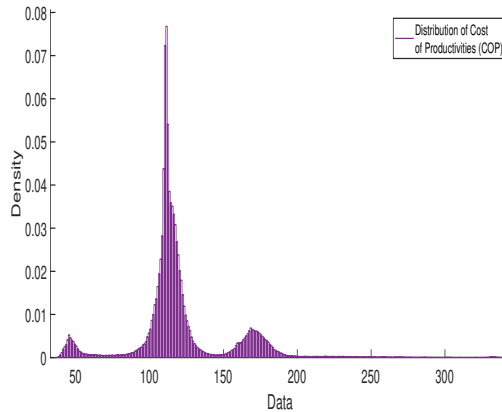


Figure 3.4: Distribution of COP values

Since the definition of the classes may be often subjective according to the economic goals set for the process, we examine two different cases that are defined in Table 3.1. The major difference between Case 1 and Case 2 is that they differ in the percentage of overlap between classes. The overlap between classes is calculated from the training data based on simulated frequency of occurrences of COP values. According to the histogram of productivity cost shown in Figure 3.4, case 1 corresponds to very low overlap while case 2 shows significant overlap between classes. Each dataset is divided into datasets for training

(75%) and for testing (25%). The dimensions of the training dataset is $50 \times 100 \times 4808$: the number of variables are 50, 100 time-steps of past and current values are considered for each variable, and 4808 data packets, each corresponding to an average operating cost value. The dimensions of the test data is $50 \times 100 \times 1536$. The calibration dataset is mean-centered and normalized by the standard deviation for each variable. The test dataset is mean centered and scaled according to the same means and variances that were used for training.

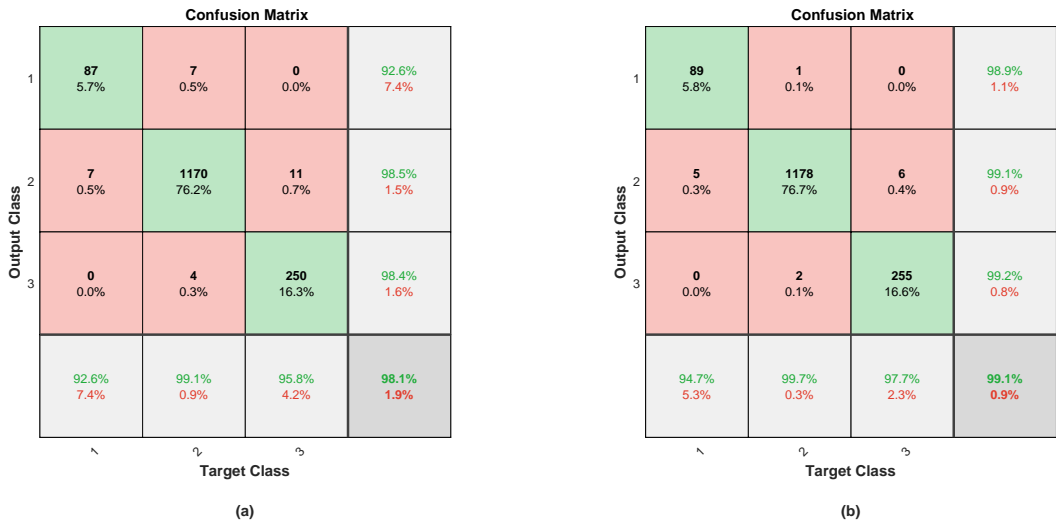


Figure 3.5: Confusion Matrices for Step 1 and Step 9 of the proposed method SLRPPF. (a) Confusion Matrix for Step 1. (b) Confusion Matrix for Step 9.

Two deep learning modeling approaches that are used for classification in this case study are: a supervised approach involving $LSTM-NN + SLRPPF$ and an unsupervised approach based on the combination $AE-NN + SVM$. It is to be noted that only the input variables remaining after pruning are used as inputs in the unsupervised approach. Both of these are compared to a widely used multivariate statistical method that combines $DPCA$ with an SVM classifier. The confusion matrix is used to summarize the percentages of correct and incorrect classifications for each class and an average of these values for a particular classification model. The aforementioned approaches are simulated and implemented using

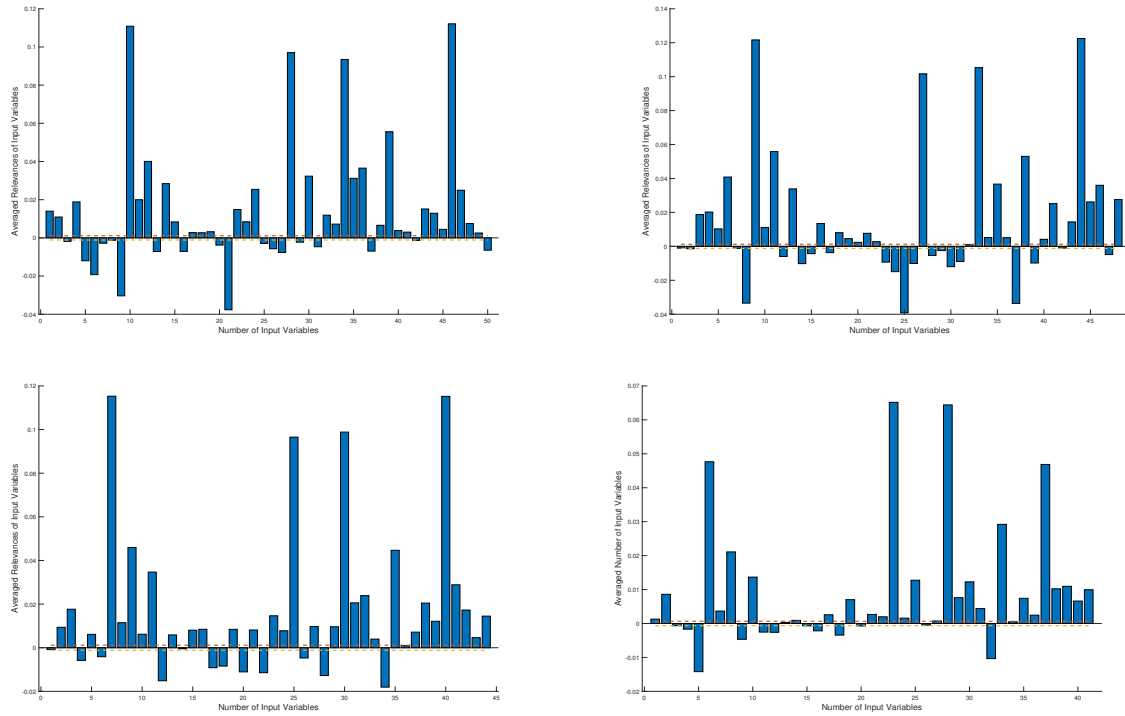


Figure 3.6: Average relevance of input variables for different iterations (SLRPF). (Step 1-4)

MATLAB[®].

LSTM + SLRPF Model (Supervised Classification)

The LSTM classification model for TEP consists of different layers namely: an input layer, an LSTM layer, dropout layer, a fully connected layer and a softmax layer. The hidden layer consisted of 80 units was tuned using a validation dataset (20% for the test-set) to minimize the loss error. The model performed well with a test accuracy of 98.11% for Case 1 and 89.7% for Case 2 on the test data. Confusion Matrix for Case 1 is shown in Figure 3.5(a). Subsequently, the proposed pruning algorithm presented in the previous section was implemented to reduce the size of the network and to identify the most relevant inputs for the different output classes. It should be noted that for LSTM-NN classification model,

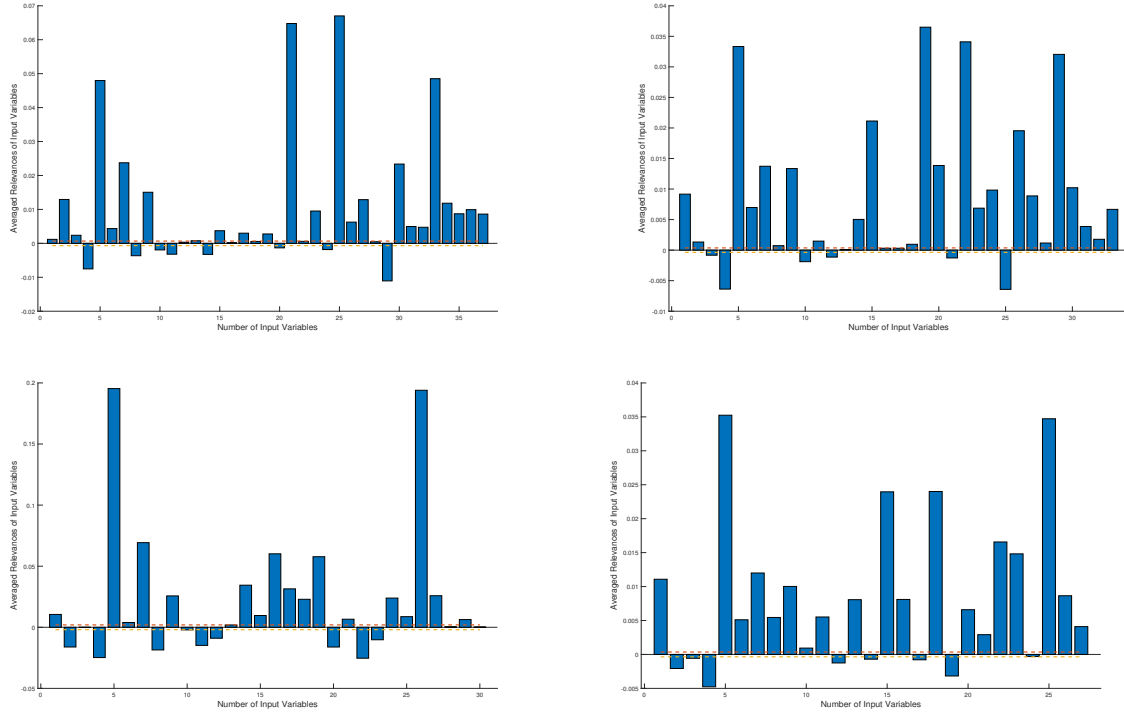


Figure 3.7: Average relevance of input variables for different iterations (SLRPF). (Step 5-8)

the relevances are averaged over time since the model is working on a time horizon, i.e. recursive in nature, so there is no meaning to relevances at each interval. The sequential elimination of non-contributing input variables are shown in Figure 3.6, 3.7 and 3.8. The number of relevant inputs was reduced from $n = 50$ to $n = 26$ for Case 1 and $n = 24$ for Case 2. Input variables with high averaged relevance scores are interpreted to have significant effect on profit-based objective function i.e. given by 3.7. Figure 3.8 displays all the input variables that have large impact on the defined profit-function. Note that with every single iteration of the proposed sequential algorithm, the corresponding testing accuracy increases for both Case1 and Case2 (see Table 3.2). Figure 3.5 illustrates the confusion matrices for Step-1 and Step-9 (the final step) for Case 1. This figure demonstrates the enhancement in test accuracy by reducing the number of irrelevant input variables with respect to the

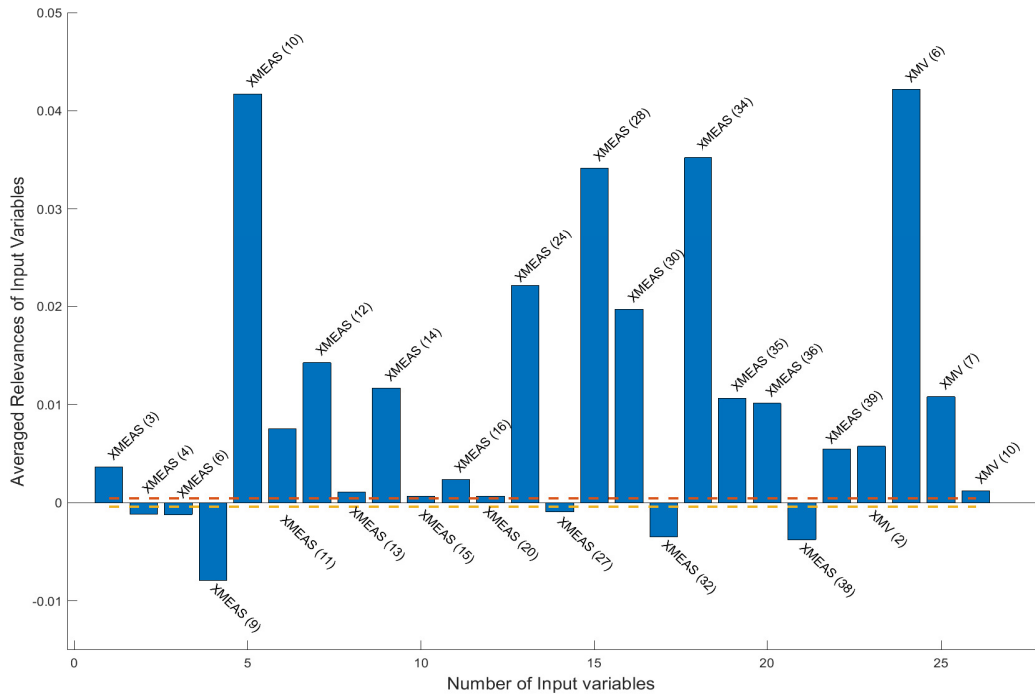


Figure 3.8: Average relevance of input variables for final iteration (SLRPFP). (Step 9)

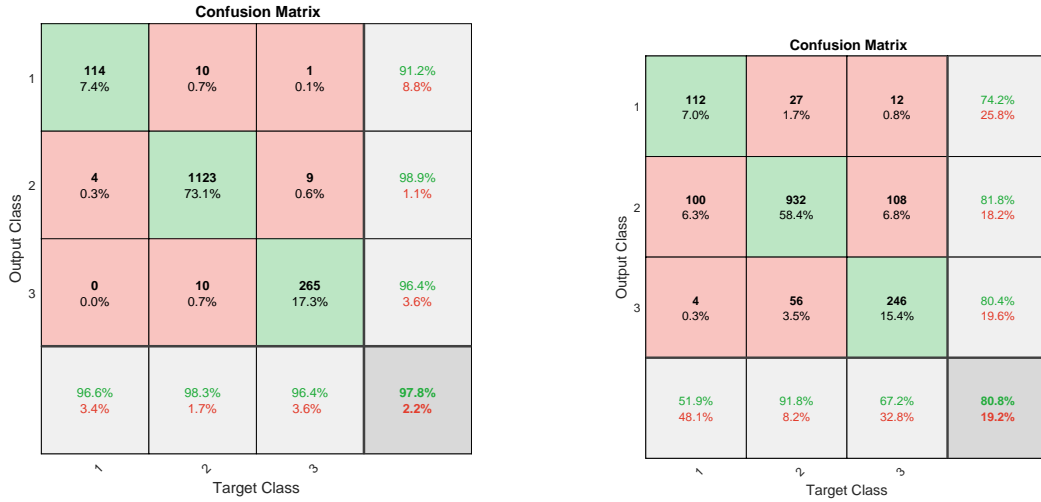
profit-based objective function. This improvement is due to a reduction in over-fitting of noisy data during training of the simplified network as compared to the original larger network used before pruning. It is also observed that there is a significant decrease in the total number of parameters of NN models from 37875 to 10840 for Case 1 and from 60500 to 11402 for Case 2. (see Table 3.2). Estimation of less number of parameters is also beneficial as it eases online implementation of deep-learning models. A significant reduction in the number of input variables for classification of profit-based operating regions were obtained following the application of the SLRPFP algorithm (see Algorithm 1 and 2).

Beyond the simplification of the network, a significant added benefit of the pruning algorithm is that it provides physical understanding about the process profitability based solely on process data. For example, it is observed that variables XMEAS(20), XMEAS(10),

Table 3.2: Implementation of SLRPFPP for pruning input variables in TEP using LSTM model

Steps	Number of input variables		Test Accuracy		Number of parameters	
	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2
Step 1	50	50	98.11%	89.7%	37875	60500
Step 2	48	36	98.44%	90.82%	33390	17450
Step 3	44	33	98.57%	90.95%	28665	19635
Step 4	41	28	98.63%	91.21%	24540	18535
Step 5	37	25	98.83%	93.03%	17650	12825
Step 6	33	24	98.89%	93.71%	16850	11402
Step 7	30	-	98.96%	-	11480	-
Step 8	27	-	99.02%	-	11000	-
Step 9	26	-	99.09%	-	10840	-

XMEAS(32), XMEAS(34), XMEAS(35), XMEAS(36), XMEAS(38) and XMEAS(39) that were identified as significant by the current relevance analysis are also present in the analytical expression describing the operating costs in the process (Equation 3.7). Since the datasets were generated by introducing known disturbances, the effect of the latter on input variables must be understood to explain the effects of input variables on the COP (cost of productivity). Changes in IDV(1) i.e. changes in the A/C ratio and IDV(6) i.e. changes in A will affect XMEAS(4) that is the addition of A and C. Thus, changes in both IDV(1) and IDV(6) will affect the ratio between components A and C. This ratio should be maintained constant if productivity is to be kept constant. For example, to maintain the A/C ratio constant in the presence of a reduction in IDV(6) (feed A), the sum of A and C (XMEAS(4)) must increase to provide more A and the purge rate (XMEAS(10)) must be increased to get rid of excess C. Similarly IDV(7) affects the feed rate of C which can be compensated by manipulating XMEAS(4) to maintain the ratio A/C constant. The composition of B is affected by IDV(2) which affects COP through the purge costs by means of XMEAS(24) and XMEAS(30). IDV(3) affects the feed temperature of compo-



(a) Autoencoder + SVM

(b) DPCA_{l=20} + SVM

Figure 3.9: Confusion Matrices for unsupervised approaches (Case 1)

ment D which is controlled by manipulating the reactor coolant flow (XMV(10)). IDV(5) disturbs the temperature of the condenser cooling water and is maintained at set-point by manipulating the separator cooling water flow (XMV(7)). The two variables stripper level (XMEAS(15)) and separator level (XMEAS(12)) are not affecting the steady-state values of COP Larsson et al. [2001] but may have significant dynamic effects during a 100-hour period.

Autoencoder + SVM (Unsupervised Classification)

To compare the performance of the supervised learning approach i.e. the LSTM model in 3.4.1 with an unsupervised learning approach, an AE was trained with the inputs that were found relevant following the pruning of the LSTM-NN. A single layer Autoencoder NN consists of 1000 neurons was calibrated for 1500 epochs. Reconstruction errors of 4.13% and 5.12% were obtained using ?? for the calibration and test dataset respectively. An MSVM classification model was built using the latent representation of the input variables from the

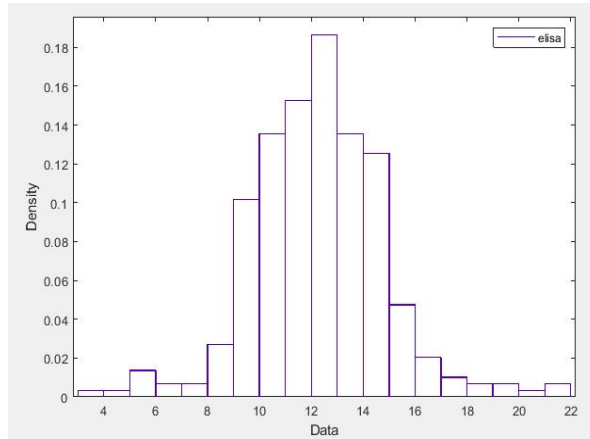


Figure 3.10: Distribution of PRN (ELISA) productivity

trained AE. The test accuracy evaluated for this classification model was 97.8% (shown in the right corner of Figure 3.9a) for Case 1 and 92.72% for Case 2. This demonstrates that the trained AE-NN could be also used for classification but the supervised deep learning classification perform marginally better.

DPCA + SVM (Unsupervised Classification)

Aforementioned models are compared with a linear unsupervised classification approach, i.e. based on DPCA combined with an SVM classifier, that is widely used in the process industries. The number of lags $d = 20$ was selected using the method described in [Ku et al. \[1995\]](#). The selected principal components explains 85% of variance of the input data. Then an MSVM was applied to the scores resulting from the DPCA for both the training and testing datasets. This resulted in 80.8% of classification accuracy of the testing dataset for Case 1 (shown in Figure 3.9b) and 68.8% for Case 2.

Hence the deep learning models perform significantly better than the classification algorithm that combines linear DPCA with MSVM thus corroborating the need for the non-linear deep learning classification models.

3.4.2 Case Study 2: Industrial Vaccine Manufacturing Process

Bordetella pertussis antigens contribute to the formulation of several pediatric vaccines, e.g. whooping cough, which are used across the world. Large variability in the manufacturing process in the process operated by Sanofi Pasteur may cause shortages in the worldwide supply from time-to-time. Vaccine antigens are not generally primary metabolism products. Hence, the production of these proteins is not necessarily correlated to microorganism growth and productivity may be low even if a significant amount of bacterial growth takes place. Therefore, it is essential to monitor the productivity to detect unfavorable process conditions. While operational improvements have been made, there are still difficulties in cell culture control and monitoring. For decomposing the three-way batch data, extensions of PCA and PLS techniques have been extended to Multiway PCA (MPCA) and Multiway PLS (MPLS) (MacGregor et al. [1994], Nomikos and MacGregor [1995]). On the other hand these algorithms are based on linear models and it is argued that non-linear modelling techniques may provide better inference of productivity in batch processes due to their ability to deal with inherent non-linear correlations in the historical process datasets. The current study assesses the use of deep learning models for classification of productivity classes with respect to regions of input values. The proposed methodology of pruning input variables (SLRPFP) provides understanding of the sources of variability during the fermentation process and the potential of uncovering mechanisms which would facilitate implementing pro-active process control techniques are presented in subsequent sections.

Sanofi Pasteur (Antigen Manufacturing Process)

The manufacturing process of the whooping cough vaccine involves two major phases: upstream and downstream. The upstream phase involves two parallel trains of three bioreactors each as shown in Figure 3.11 where fermentation of the cells is conducted. The downstream phase involves a series of purification steps where the pure antigens are synthesized from the product of the fermentation processes. Two parallel 20-L fermenters (initial fermentation stage) containing mainly Component Pertussis broth and growth factors are inoculated and grown for 22 to 25 hours. Subsequently, the cultures are transferred to two parallel 200-L fermenters (intermediate fermentation stage) for 22 to 25 hours, and

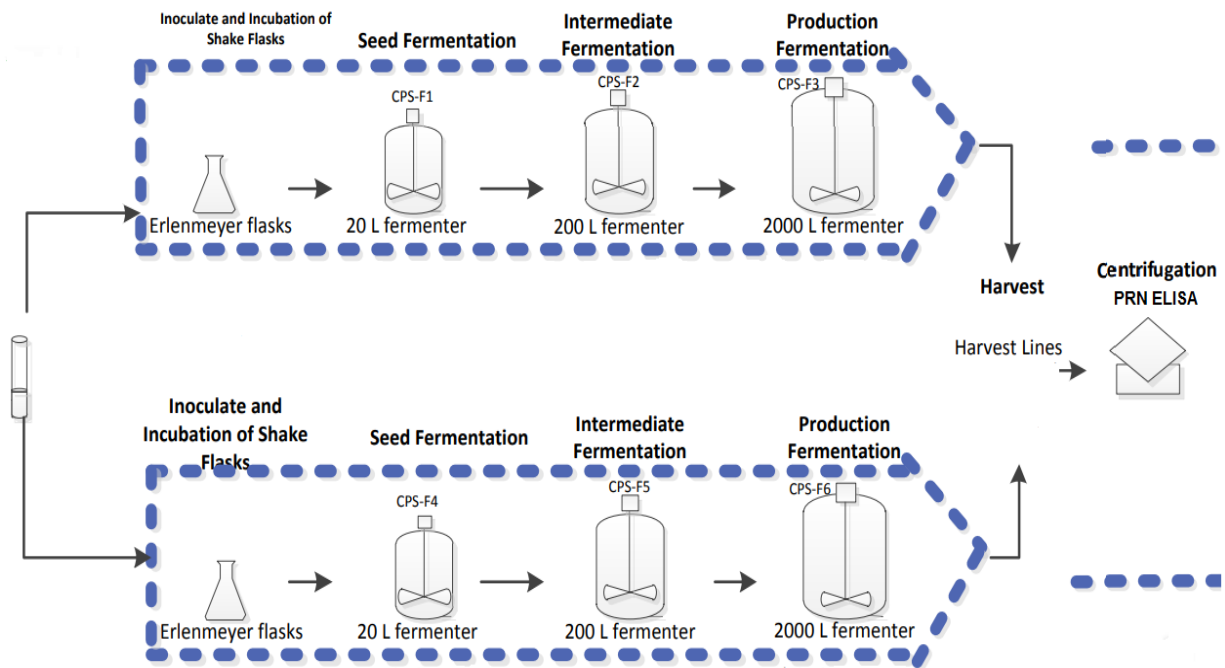


Figure 3.11: Schematic of vaccine manufacturing process at Sanofi Pasteur, Toronto, Canada

then to two parallel 2000-L fermenters (production fermentation stage) that are operated for up to 56 hours. All fermenters work under the same operating conditions in terms of temperature (36 °C), dissolved oxygen (35%) and pH (7.2) levels, which are continuously monitored and controlled in closed-loop. In the final production fermenters (2000-L), supplement feed media is added when the initial supply of the key nutrient (glutamate) is exhausted and at that point the operation changes from batch to fed-batch. The depletion of nutrients is detected by a significant sudden decrease in oxygen consumption which is accompanied by a corresponding spike in dissolved oxygen. These fermentation steps are followed by harvesting, which involves the centrifugation of the fermentation culture. The cell paste, coming from the centrifugation, is collected into a tank for further purification, and the supernatant to be referred as centrate is filtered to ensure the complete removal of the live cells.

This case study classifies the resulting productivity based on information of the two 2000-L fermenters to be referred to as F3 and F6. There are 11 process variables measured for both F3 and F6 fermenter. Data collected from 295 batches (over a period of 4-years) were considered for this study. About 240 batches were selected as training dataset and the rest (55 batches) were selected as testing dataset at random. The goal of the classification problem is to find input conditions that will result in either low or high levels of the productivity of Pertactin (PRN) that is a key antigen composing the acellular whooping cough vaccine. The level of the antigen is determined at the end of the fermentations by ELISA (enzyme-linked immunosorbent assay). The ELISA measurement is performed only on a sample collected from the combined stream of F3 and F6 after the centrifugation step. The two classes for the productivity of antigen that are targeted for classification are defined by the manufacturer, refer to Table 3.3. The distribution of productivity is shown in Figure 3.10. This classification task is extremely important for elucidating which process inputs are most relevant for low or high productivity of antigen and for gaining understanding about the possible causes of low productivity.

Table 3.3: Classes for productivity of Pertactin

Productivity of Pertactin (<i>g/batch</i>)	Low	High
Case	≤ 11.5	> 11.5

Although the vaccine contains several antigens, the focus in this work is on the production of the Pertactin (PRN) antigen since it is produced in very small quantities relative to the amount required in the vaccine and thus it is a potential bottleneck for the process.

Data-preparation

295 batches (ELISA batches) were selected from batch data obtained from approximately last 4 years. Some missing data occur for several batches. Missing data for process variables were estimated using a sparse-optimisation based algorithm [Agarwal and Tangirala \[2017b\]](#), that estimates missing data-points using the Fourier basis functions as the sparse basis. This algorithm has been applied previously for re-construction of causal networks [Agarwal](#)

and Tangirala [2017a]. As mentioned in the introduction, synchronization among batches is a key challenge for correct comparison of batch data collected from different batches. The batches were synchronised using the Dynamic Time Warping concept so as to have an even-length batch duration for all selected batches. Since NN models require large number of samples for the training process, additional low-resolution batches are generated from the 295 batches by re-sampling the profiles of process variables at slower sampling rates. The concept of re-sampling to generate more data for training has been recently reported by Tulsyan et al. [2019] for a Statistical Process Monitoring (SPM) application. In view that the sampling time for each process variable in the original dataset is 2 minutes, for generating the additional data the process variables were sampled every 15 samples (~ 30 minutes) in order to generate 15 low-resolution batches from a single batch data. This re-sampling operation resulted in 4425 samples in total, out of which 3600 samples are used for training and the other 825 samples as the test data-set. The training and testing dataset are normalised using min-max algorithm (Equation 3.8) such that entire range of values are mapped between 0 to 1.

$$\mathbf{x}_{norm} = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})} \quad (3.8)$$

MLP + SLRPFP (Supervised Classification)

It should be noticed that this problem is substantially different from the continuous TEP problem discussed below since here the profit related variable is only measured once after the completion of the batch/fed-batch operation. Thus, for the current study it is more relevant to extract information from the entire time profiles of the input variables rather than exploiting local dynamic correlations as done for the TEP study for which the costs are measured at each time interval. Thus, it was hypothesized that a fully connected MLP NN that connects the information among inputs and between inputs to outputs occurring at all times during the operation may provide better inference of the end process profitability (productivity). Hence, an MLP neural network was developed consisting of an input layer, two hidden layers that use *tanh* as the non-linear activation function and a softmax based

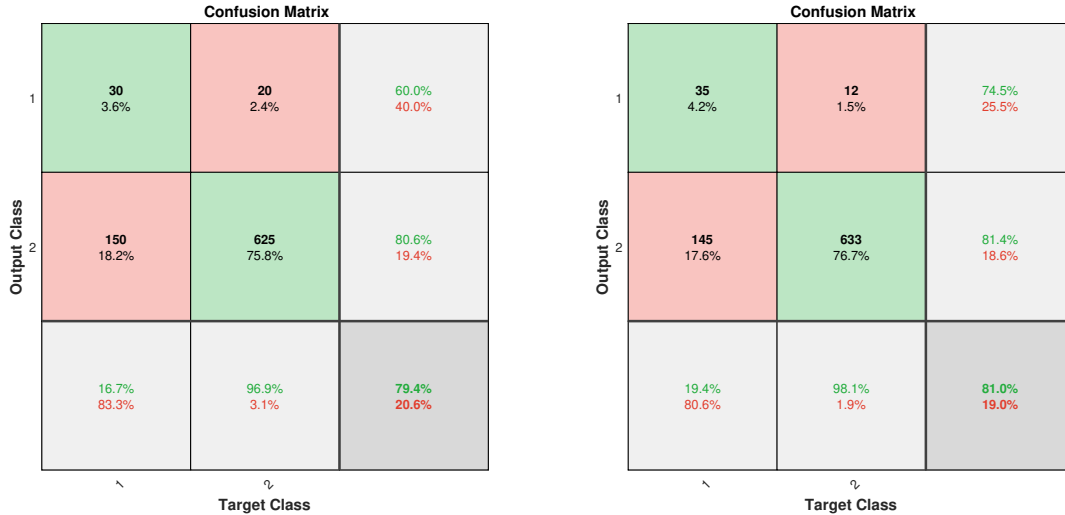


Figure 3.12: MLP + SLRPFP: Test dataset accuracy before (left) and after SLRPFP (right) for industrial process

classification layer. This network was initially trained with the normalized training dataset. The number of nodes, mini-batch sizes for training and epochs were selected to minimize the cross-validation error that was based on 20% of the testing set. The input layer consists of 1760 input nodes, the first hidden layer consists of 100 neurons and second hidden layer consists of 2 neurons with a mini-batch size of 5 samples for 10000 epochs. This model resulted in an accuracy of 79.4% on the testing dataset as shown in the right corner of the confusion matrix in Figure 3.12 (left). Subsequently, the SLRPFP pruning algorithm was applied to the MLP model to compute a relevance value for each input variable and at each time interval of the batch. Also, this increased the test accuracy from 79.4% to 81% (shown in Figure 3.12 (right)). In contrast with the continuous TEP problem for which an average relevance over time was calculate for the current batch study since the inputs to the model change significantly during the batch and the property being classified (Pertactin level) is obtained only at the end of the batch it is more informative to calculate the relevance of each input at each time interval (see Algorithm 1). Since the entire time profiles of the input variables are fed simultaneously to the network, pruning of the network is done

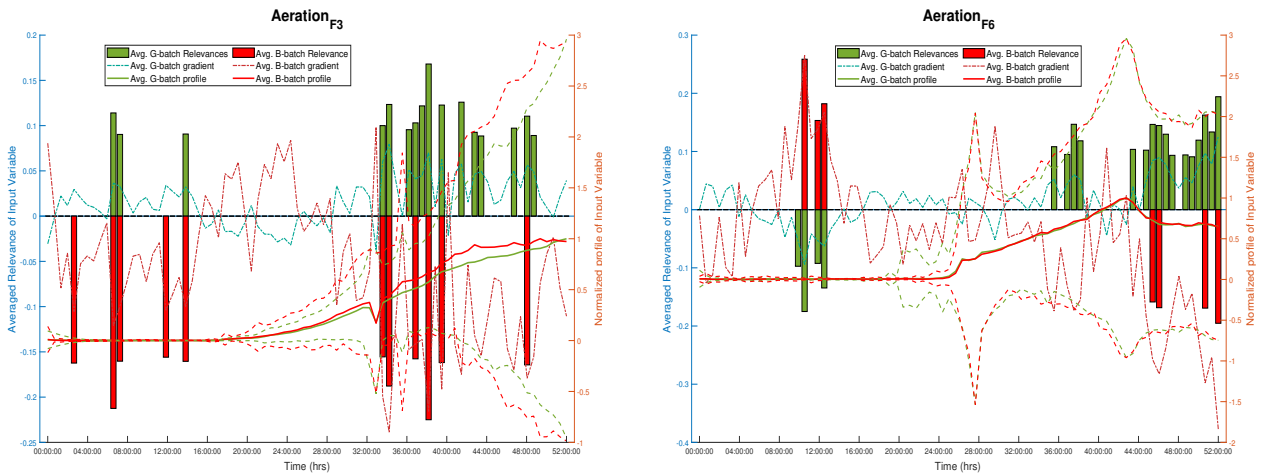


Figure 3.13: Averaged Time-based Relevance of Aeration profile for both High and Low PRN productivity batches

with respect to each input variable value at each time interval during the batch. Final relevances of input variables (after all pruning steps) for a class k are computed using the scores of train, validation and test samples that are correctly classified for a particular class using LRP for each input at each time interval. Through the analysis of relevances (see Algorithm Algorithm 1 and 2), a considerable number of input variables were found to be irrelevant with respect to the classification thus reducing the number of relevant variables from 1760 to 443.

Figure 3.12 demonstrates the initial and final iteration of SLRPFP until the relevance of all input variables are above a set threshold value (see Algorithm 2). Similar to Case Study 1 the pruning of the network results in higher test accuracy (improvement of 1.6%).

Beyond the capability of identifying relevant inputs and pruning the network, the interpretation of the relevances can provide, as shown for Case Study 1, important physical understanding about the effect of inputs process productivity. In order to facilitate such understanding, the relevance scores are calculated with respect to each of the two classes, i.e. low and high range of PRN levels. Relevance scores of each input-profile for both high (green bars) and low (red bars) PRN productivity classes are shown for few process variables (such as pH1, aeration and agitation) in Figures 3.13,3.14,3.15 and 3.16. The

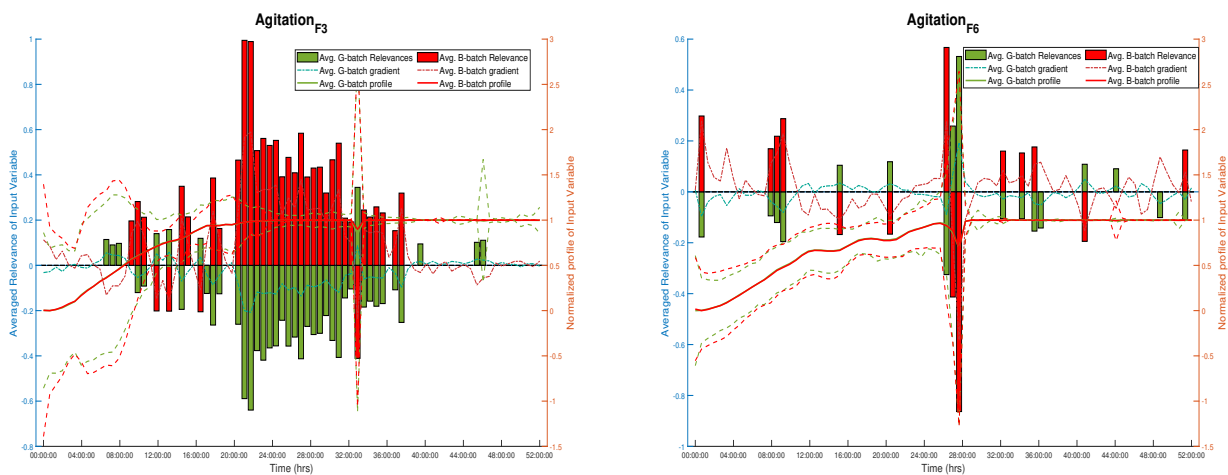


Figure 3.14: Averaged Time-based Relevance of Agitation profile for both High and Low PRN productivity batches

averaged-standardized variable profiles for both high and low PRN batches are plotted with bold green and red lines along with 2*standard-deviation. As observed from the variances in Figures 14-17 there is large variability among the different batches which in a separate work have been related to metabolic changes due to variability in media composition [Budman et al. \[2013\]](#). It should be remembered that the relevances indicate the relative contribution of each input variable to the productivity classes. When separate relevances are calculated for each one of the two classes, the relevance of each input reflects its relative positive or negative contribution to the probability to belong to the class for which the relevance is calculated. To verify the same, derivatives with respect to softmax probabilities were calculated and are shown with every relevance plots. It was found that the relative relevances for each class follows the same trend as of the gradients with respect to softmax probabilities. Accordingly, the physical/biological interpretations of the calculated relevances are as follows:

i- According to Figure 3.13 high aeration levels towards the end of the fermentation process both for fermenters F3 and F6 are significantly (positively) correlated with the probability to be within the high class of productivity, i.e. higher aeration increases the probability to be within the high productivity class. On the other hand Figure 3.13 also shows that

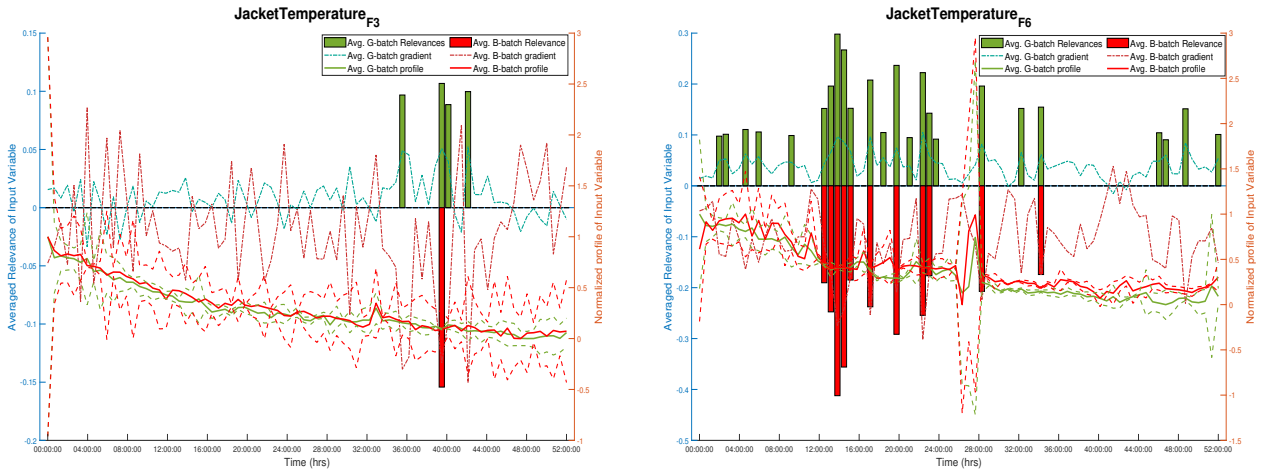


Figure 3.15: Averaged Time-based Relevance of Jacket Temperature profile for both High and Low PRN productivity batches

the aeration rates are not as significantly correlated to the probability to be within the low productivity class. Thus in bad batches variations in aeration are not clearly correlated to productivity

ii- According to Figure 3.14 the agitation rates in the fermenters are significantly (positively) correlated to the probability to be within the low productivity class between fermentation time of 20 hours to 32 hours. This is a particularly important observation which indicates that the growth in the low productivity batches is delayed as compared to the high productivity batches. The increase in agitation rate after 20 hours indicates that growth started to increase at that time in the low productivity batches since increased growth will require increased supply of oxygen through increased agitation (as indicated by the relevances' results) to maintain the dissolved oxygen target. Although, increased aeration could have also supply the additional required oxygen, a particular split oxygen control strategy is implemented in the process where the controller uses first the agitation to increase oxygen demand until the agitation reaches a saturation limit and only afterward it will start to use the aeration rate to satisfy the demand. The delayed increase in growth is clearly shown in Figure 3.18b where biomass values as measured by optical density are shown for a few low and high productivity batches. At this point optical density cannot

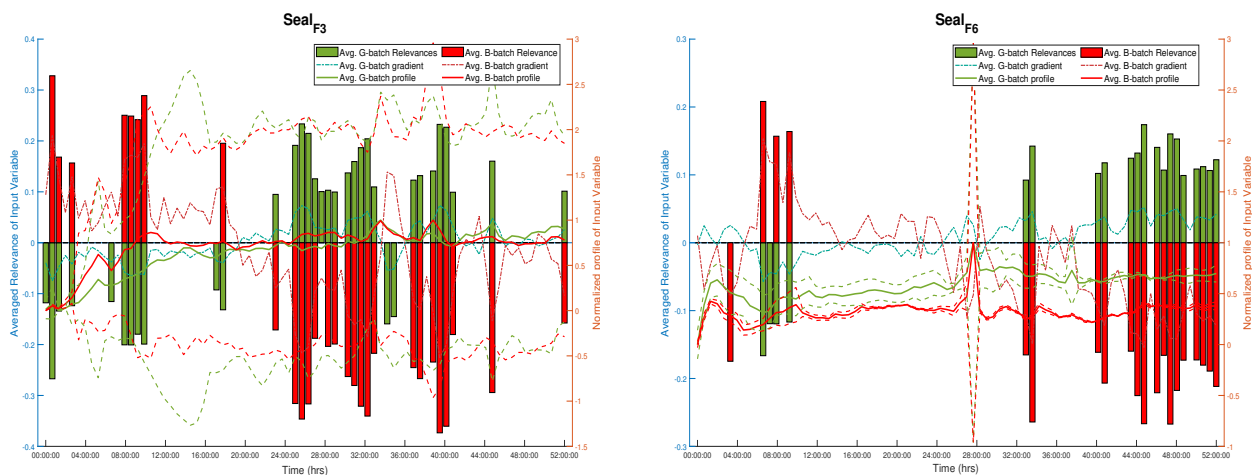


Figure 3.16: Averaged Time-based Relevance of Seal Temperature profile for both High and Low PRN productivity batches

be measured online and it is available for only few batches. Hence, the latter observation cannot be corroborated for all batches that were investigated. A possible explanation for the variability is the occurrence of different levels of oxidative stress due to changes in media composition as reported in a separate study [Zavatti et al. \[2016\]](#).

iii- It can be noted from Figure 3.15 that the jacket temperature in F3 fermenter does not contribute much to the classification model, on the other hand higher jacket temperature in F6 is favourable for the batch to be classified as the higher PRN batch. The above observation is confirmed through personal communication with Sanofi Pasteur staff which found that PRN specific production rates were higher at high temperature levels.

iv- In Figure 3.16 it can be observed that higher seal temperature at the supplementation and end of fermentation period increases the probability of a batch to be a High PRN batch. The seal temperature is monitored to ensure sterilization-in-place of the impeller. The seal ensures that nothing enters the motor that drives the impeller inside the fermenter. In general higher seal temperature is highly correlated to the temperature inside the fermenter which is correlated to higher productivity of antigen.

It should be noticed that in the current case study, in contrast with the simulator

based TEP study, not all inputs, e.g. disturbances, are known or measured. Thus, it is not possible as for the Case Study 1 to assess through the relevances all the input variables that can significantly affect productivity. For example, it is expected that growth media composition changes will surely affect the productivity but this cannot be identified with the relevances since these changes cannot be realistically measured due to the complex composition of the media. Another important observation from the relevances is regarding differences in relevances between the two parallel F3 and F6. Although these fermenters are supposed to behave identically, based on manufacturing experience and differences in growth curves shown in 3.18a, they respond differently. These differences may be possibly related to different performance of the impeller, operator related variations, etc. Thus we can not expect similar relevance of input variables in both of them.

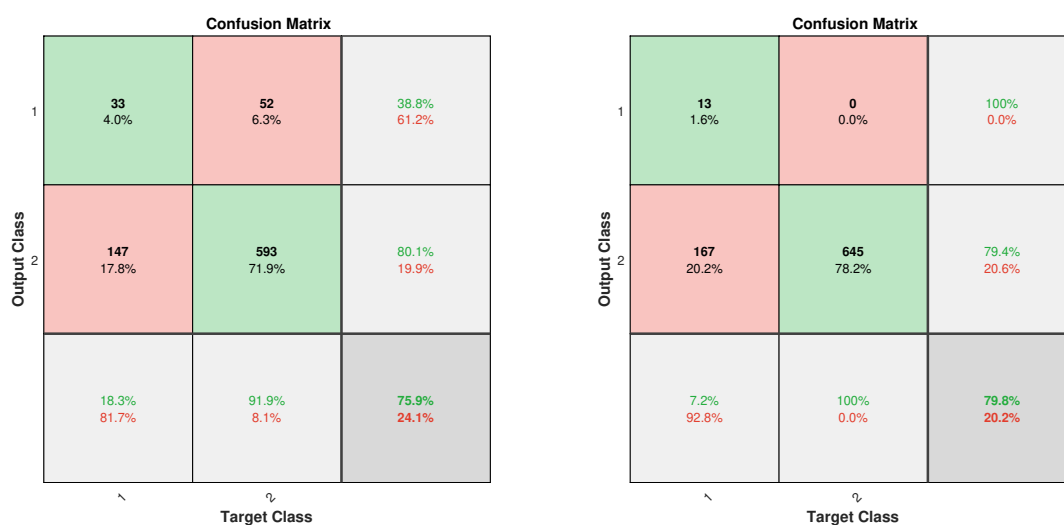


Figure 3.17: LSTM + SLRPFP: confusion matrix of test dataset before (left) and after (right) SLRPFP

For comparison purposes *LSTM*, *AE + SVM* and *BDPCA + SVM* classification models are investigated for this case study.

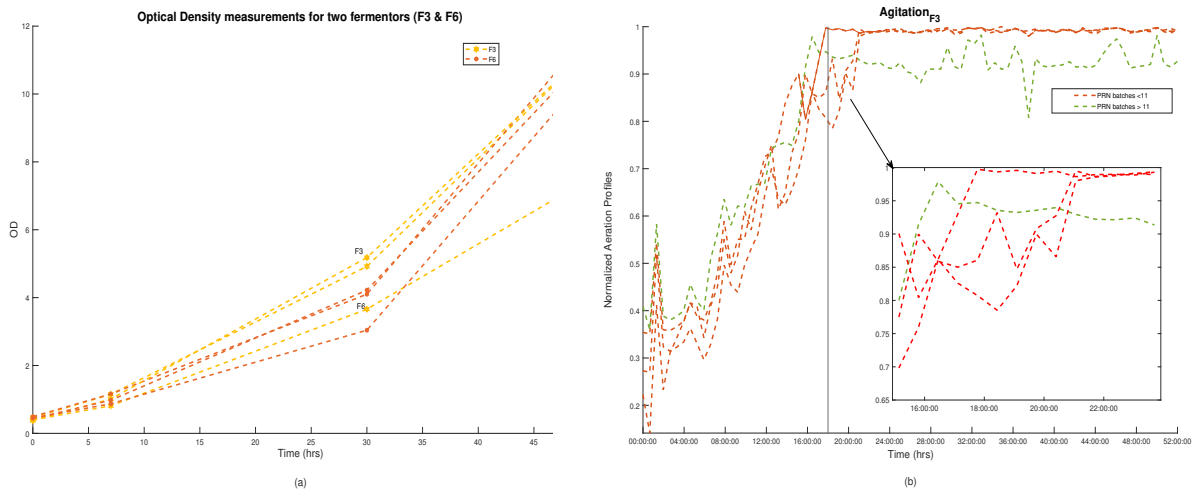
LSTM + SLRPFM Model (Supervised Classification)

The LSTM model for Sanofi Pasteur batch datasets consists of an input layer, followed by a dropout layer, a fully-connected layer and a softmax layer. The hidden layer consists of 8 neurons and the model is trained with a mini-batch size of 1000 samples for maximum of 6500 epochs. The testing accuracy achieved for this model was 75.9% on the same test-dataset as the one used in the MLP model in the previous section. The SLRPFM (see 1) algorithm was applied to prune the irrelevant variables (pH_{F3} and Jacket Temperature $_{F3}$). From the relevances for each variable shown in Figure 3.19 it is observed that the variable pH_{F3} and Jacket Temperature $_{F3}$ do not contribute to the overall LSTM classification model. Subsequently, it was found that following pruning of the two input variables the test accuracy increased from 75.9% to 79.8%. The confusion matrix of test-dataset for both before and after pruning are shown in Figure 3.17. The obtained testing accuracies are acceptable since the expected error in the measurement of antigen is of the order of at least $\pm 10\%$

It is further noticed that averaged relevances for each variable at each time-step computed from the MLP model in the previous sub-section and overall averaged relevance for each input variable from the LSTM model indicate the same variables i.e. pH_{F3} and Jacket temperature in F3 as non-relevant for classification. This corroborates the significance of each input variable with respect to the productivity and that different models lead to the same conclusions. It is also evident that the testing accuracies in Case Study 1 are lower than the testing accuracies in this case study. However, this is not surprising if one considers that in the TEP problem all the inputs and disturbances are available for model calibration whereas in Case Study 2 some of the disturbances are not measured, e.g. changes in media composition that cannot be exactly measured for each batch.

Autoencoder + SVM (Unsupervised Classification)

To compare the performance of supervised deep learning models i.e. the MLP and the LSTM model in Section 3.4.2 and 3.4.2 with an unsupervised learning approach, an AE



(a) OD for F3 & F6 fermenter

(b) Normalized agitation profiles

Figure 3.18: (a) Difference in the growth profile for two fermentors; (b) Increase in agitation profiles around 18 hours because of delayed increase of growth in low PRN batches

was trained with the pruned training dataset used for training MLP-NN and LSTM-NN. A single layer AE-NN consists of 1000 neurons was calibrated for 1500 epochs. A reconstruction error of 7% was evaluated using ?? for the calibration dataset. A MSVM classification model was built based on the outputs from the trained AE. The test accuracy was 78.18% which corroborates an earlier observation in Section 4.1.2 that the supervised modeling approach performs marginally better than the unsupervised one.

BDPCA + SVM

The deep learning models were compared with linear unsupervised *BDPCA + SVM* approach. BDPCA was chosen for comparison since it was reported to perform better than Multiway Principal Component Analysis (MPCA) for monitoring batch processes [Chen and Liu \[2002\]](#). The method for determining number of lags d is described in [Ku et al., \(1995\)](#) [Ku et al. \[1995\]](#) for each batch and an average time lag $d_{avg} = 15$ is selected. Estimation of the average loading vectors for each batch is carried out by evaluating an averaged

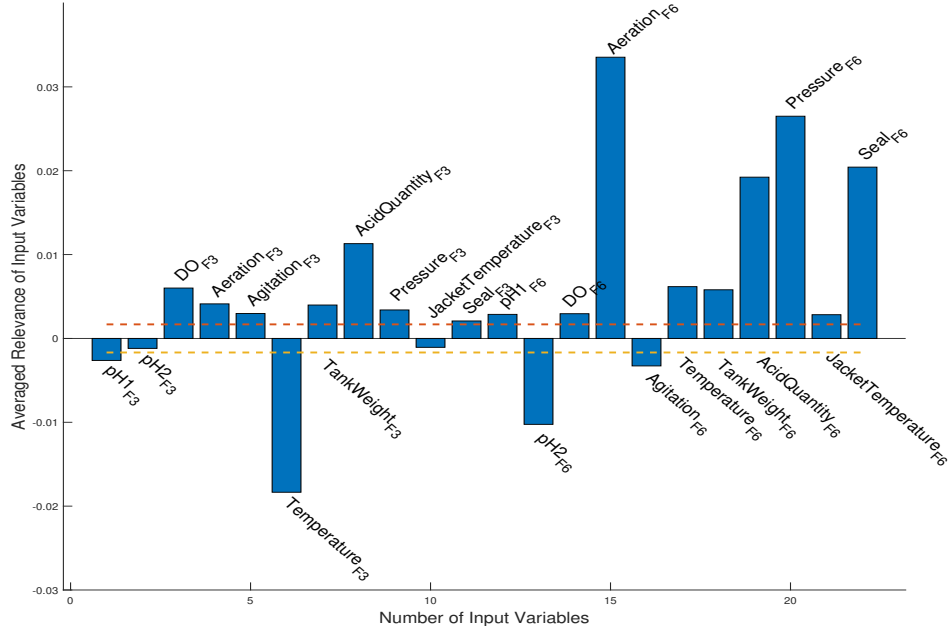
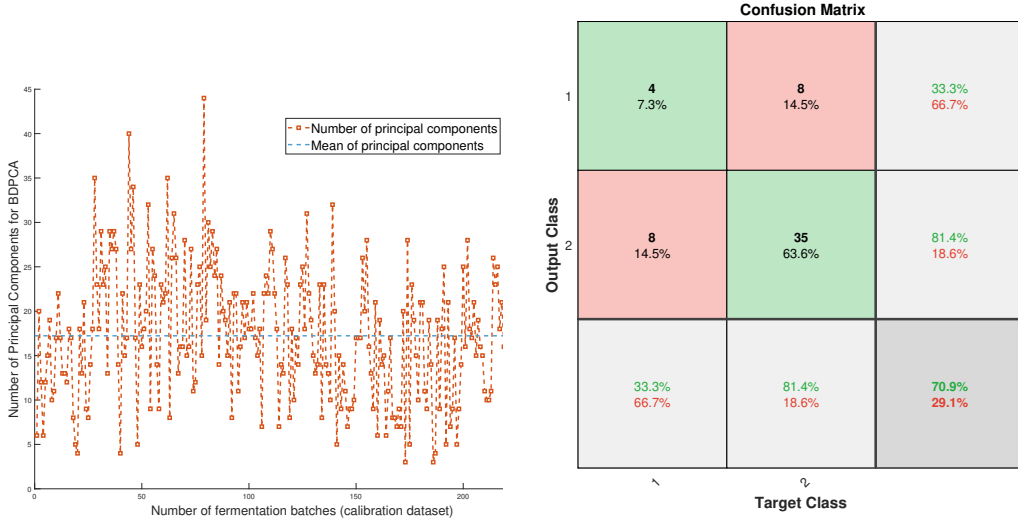


Figure 3.19: Averaged Relevance of all Input Variables from the LSTM model

dynamic co-variance matrix $\mathbb{S}_{\mathbb{X}_d \mathbb{X}_d}^{avg}$ as follows:

$$\mathbb{S}_{\mathbb{X}_d \mathbb{X}_d}^{avg} = \frac{(N - d_{avg} + 1) \sum_{i=1}^N \mathbb{S}_{\mathbb{X}_d \mathbb{X}_d}^i}{J(N - d_{avg})} \quad (3.9)$$

where $J = 22$ (11 process variables from each fermenter) are the number of process variables monitored during the fermentation process; $N = 240$ are the number of batch-datasets used for calibration and $\mathbb{S}_{\mathbb{X}_d \mathbb{X}_d}^i$ is the co-variance matrix for each batch. An average number of principal components $n_{princomp_{avg}} = 18$ are selected (shown in Figure 3.20a) such that it explains 90% of the variance on average for each batch. Subsequently, the test dataset is projected onto the selected average principal components and the obtained scores are used as the inputs to an MSVM classification model. The classification accuracy for the training and testing dataset were 66.7% and 70.91% (illustrated in Figure 3.20b) respectively which are significantly lower than the values obtained with the deep learning models, e.g. 90.3% for training and 79.8% for testing with the MLP model.



(a) Avg. principal components (BDPCA) (b) Confusion Matrix (BDPCA)

Figure 3.20: (a) Selection of avg. number of principal components in BDPCA; (b) Confusion Matrix for test dataset (BDPCA)

3.5 Conclusion

This chapter presents a novel neural network (NN) pruning algorithm referred to as Sequential Layer-wise Relevance Propagation for Pruning (SLRPFP) based on relevance of input variables. The proposed method first computes the relative relevance scores for all input variables followed by eliminating the input variables which are below a certain threshold value. It significantly reduces the number of parameters to be estimated and it improves the performance of the classification model by avoiding data over-fitting. An added benefit is that it reduces the computational load thus facilitating online implementation in continuous processes.

Also, the pruning methodology provides important physical insights on the system regarding the inputs that have positive and negative effect on the profit function and to detect significant changes in process phenomena, e.g. variability in cell growth for different

fermentations in the vaccine manufacturing process.

This work also demonstrated the superiority of deep learning approaches with both supervised (LSTM and MLP neural networks) and unsupervised (AE based model) classification models as compared to linear multivariate models. The efficacy of the proposed method is demonstrated for both a continuous (Tennessee Eastman process) and a batch fermentation process (an industrial antigen manufacturing process).

Chapter 4

Explainability: Relevance based Dynamic Deep Learning Algorithm for Fault Detection and Diagnosis in Chemical Processes

Overview¹

The focus of this work is on Statistical Process Control (SPC) of a manufacturing process based on available measurements. Two important applications of SPC in industrial settings are fault detection and diagnosis (FDD). In this work a deep learning (DL) based methodology is proposed for FDD. We investigate the application of an explainability concept to enhance the FDD accuracy of a deep neural network model trained with a data set of relatively small number of samples. The explainability is quantified by a novel relevance measure of input variables that is calculated from a Layerwise Relevance Propagation (LRP) algorithm. It is shown that the relevances can be used to discard redundant

¹Adapted from Agarwal, Piyush, et al. "Explainability: Relevance based Dynamic Deep Learning Algorithm for Fault Detection and Diagnosis in Chemical Processes" *Computers & Chemical Engineering*, Volume 154 (2021)

input feature vectors/ variables iteratively thus resulting in reduced over-parametrization and over-fitting of noisy data, increasing distinguishability between output classes and superior FDD test accuracy. The efficacy of the proposed method is demonstrated on the benchmark Tennessee Eastman Process.

4.1 Introduction

The fourth industrial revolution also known as ‘Industry 4.0’ and Big Data paradigm has enabled the manufacturing industries to boost its performance in terms of operation, profit and safety. The ability to store large amount of data have permitted the use of deep learning (DL) and optimization algorithms for the process industries. In order to meet high levels of product quality, efficiency and reliability, a process monitoring system is needed. The two important aspects of Statistical Process Monitoring (SPM) are fault detection and diagnosis (FDD). Normal operation of process plants can be detected by determining if the current state of the process is normal or abnormal where abnormal refers to a situation where a fault has occurred. This problem is referred to as “Fault Detection”. Following the detection of abnormality in the process, the next step is to diagnose the specific fault that has occurred. This step is referred to as “Fault Diagnosis/ Classification”. The presence of noise, correlation, non-linear process dynamics and high dimensionality of the process inputs greatly hinders the FDD mechanism in process plants. Previously, traditional multivariate statistical methods such as PCA, PLS and its variants [Wise et al. \[1990\]](#), [Kaistha and Upadhyaya \[2001\]](#), [Harmouche et al. \[2014\]](#), [Yuan et al. \[2018\]](#) have been extensively used for fault detection and prognosis. However, the inherent non-linearity in the process pose challenges while using these linear methods and non-linear techniques can provide better accuracy. To this end, recent DL methods have shown considerable improvement over traditional methods. DL fault detection and classification techniques have been widely researched for applications in several engineering fields [Agarwal and Budman \[2019\]](#), [Agarwal et al. \[2019\]](#). In chemical engineering, machine learning techniques have been applied for the detection and classification of faults in the Syschem plant, which contains 19 different faults [Hoskins et al. \[1991\]](#) and for the TEP problem. Beyond their application in the process industries Several studies on DL approaches have been conducted for the

prevention of mechanical failures. For example DL models have been used for detecting and diagnosing faults present in rotating machinery [Janssens et al. \[2016\]](#), [Jia et al. \[2016\]](#), motors [Sun et al. \[2016\]](#), wind turbines [Zhao et al. \[2018b\]](#), rolling element bearings [Gan et al. \[2016\]](#), [He and He \[2017\]](#) and gearboxes [Jing et al. \[2017\]](#), [Chen et al. \[2015\]](#). Many DL studies have been recently conducted on TEP using DL models.

Although DL based models have better generalization capabilities, they are poor in interpretation abilities because of their black box nature. Using these methods it is difficult to identify the root cause of faults i.e. input variables that are most correlated to the occurrence of the faults by significantly deviating from their normal trajectories following the occurrence of the fault. Following the development of complex novel neural network architectures, there is an increasing interest in investigating the problems associated with DL models. For example, to understand how a particular decisions are made, which input variable/feature is greatly influencing the decision made by the DL-NN models, etc. This understanding is expected to shed light on why biased results can be obtained, why a wrong class is predicted with a higher probability in classification problems etc. Explainable Artificial Intelligence (XAI) is an emerging field of study which aims at explaining predictions of Deep Neural Networks (DNNs). Several different methods have been proposed in order to explain the predictions by assigning a relevance or contribution to each input variable for a given sample. Methodologies that are used to assign scores to each input feature with respect to a particular task can be classified into two class of methods: perturbation based methods and backpropagation based methods. Perturbation based methods perturb the individual input feature vectors (one by one) and estimate the impact on the output [Zeiler and Fergus \[2014\]](#), [Zhou and Troyanskaya \[2015\]](#) . On the other hand backpropagation methods are based on backward propagation of the probabilities calculated by Softmax output neurons in case of classification problem through different layers of the NN back to the input layer. Most perturbation methods are computationally expensive and often underestimate the relevance of the input features. To this end, different backward propagation methods have been proposed in the XAI literature for explaining the predictions such as Layer-wise Relevance Propagation (LRP) [Bach et al. \[2015\]](#), LIME [Ribeiro et al. \[2016\]](#), SHAP values [Lundberg and Lee \[2017\]](#), DeepLIFT [Shrikumar et al. \[2017\]](#). LRP

as a explainability technique has been successfully used in many different areas such as healthcare, audio source localization, biomedical domain and recently also in process systems engineering [Montavon et al. \[2019\]](#), [Agarwal and Budman \[2019\]](#), [Agarwal et al. \[2019\]](#) and have been shown to perform better than both SHAP and LIME [Rios et al. \[2020\]](#). In this work, we use LRP for explainability of the network by evaluating the relevance of input variables. LRP was proposed by Bach et al., 2015 [Bach et al. \[2015\]](#) to explain the predictions of DNNs by back-propagating the classification scores from the output layer to the input layer. In particular, for a specific output class c , the goal of LRP is to determine the relevance $R_c(x_i; f_c)$ of the individual input variables/ feature vectors $\mathbf{R}_c(\mathbf{x}; \mathbf{f}_c) = [R_{c_1}(x_1; f_c), R_{c_2}(x_2; f_c), R_{c_3}(x_3; f_c), \dots, R_{c_i}(x_i; f_c), \dots, R_{c_n}(x_n; f_c)] \in \mathbb{R}^n, i = 1, 2, \dots, n$ of each input feature x_i to the output $f_c(x)$.

[Xie and Bai, 2015](#) proposed neural network based methodology as a solution for the diagnosis problem in the Tennessee Eastman simulation that combines the network model with a clustering approach. The classification results obtained by this method were satisfactory for most faults. Both Wang et al., 2018 [Wang et al. \[2018\]](#) and Spyridon et al., 2018 [Spyridon and Boutalis \[2018\]](#) proposed the use of Generative Adversarial Networks (GANs), as a fault detection scheme for the TEP. GANs are an unsupervised technique composed of a generator and a discriminator trained with the adversarial learning mechanism, where the generator replicates the normal process behavior and the discriminator decides if there is abnormal behavior present in the data. This unsupervised technique can detect changes in the normal behavior achieving good detection rates. Lv et al., 2016 [Lv et al. \[2016\]](#) proposed a stacked sparse autoencoder (SSAE) structure with a deep neural network to extract important features from the input to improve the diagnosis problem in the Tennessee Eastman simulation. The diagnosis results applying this DL technique showed improvements compared to other linear and non-linear methods. To account for dynamic correlations in the data, Long Short Term Memory (LSTM) units have been recently applied to the TEP for the diagnosis of faults [Zhao et al. \[2018a\]](#). A model with LSTM units was used to learn the dynamical behaviour from sequences and batch normalization was applied to enhance convergence. An alternative to capture dynamic correlations in the data is to apply a Deep Convolutional Neural Networks (DCNN) composed of convo-

lutional layers and pooling layers [Wu and Zhao \[2018\]](#).

The fault detection problem in the current work is formulated as a binary classification problem where the objective is to classify whether the current state of the process plant is normal or abnormal while the fault diagnosis problem is formulated as a multi-class classification problem to identify the type of fault. Then, the application of the concept of explainability of Deep Neural Networks (DNNs) is explored with its particular application in FDD problem. While the explainability concept has been studied for general DNN based models as mentioned above, it has not been investigated before in the context of FDD problems. In this work, the relevance of input variables for FDD are interpreted using LRP and the irrelevant input variables' for the supervised classification problem are discarded. It is shown that the resulting pruning of the input variables results in enhanced fault detection as well as fault diagnosis test accuracy. Lastly, we show that the use of a Dynamic Deep Supervised Autoencoder (DDSAE) NNs along with the pruning of the network for both fault detection and diagnosis further improves the overall classification ability as compared to other methods reported before.

To conduct a fair comparison of the proposed algorithm to previously reported methods, careful attention should be given to the data used as the basis for comparison. For example, there is a vast literature on FDD for the TEP problem that uses differing amounts of data. In this work, we have used a standard dataset as a basis for comparison which further challenges the training of DL models and accuracy of FDD with a DL model and that has been used for comparison in other studies. The proposed DL based detection method with Deep Supervised Autoencoder (DSAE) or Deep Dynamic Supervised Autoencoder (DDSAE) is compared to several techniques: linear Principal Component Analysis (PCA) [Zhang \[2009\]](#), [Yin et al. \[2012\]](#), [Lau et al. \[2013\]](#), [Shams et al. \[2010\]](#), Dynamic Principal Component Analysis (DPCA) [Chiang et al. \[2000\]](#), [Yin et al. \[2012\]](#), [Ku et al. \[1995\]](#), [Rato and Reis \[2013\]](#), [Odiowei and Cao \[2009\]](#), Independent Component Analysis (ICA) [Hsu et al. \[2010\]](#) and with two other recently reported methods that use DL models based on Sparse Stacked Autoencoder NNs (SAE-NN) [Lv et al. \[2016\]](#) and Convolutional NN (CNN) [Chadha and Schwung \[2017\]](#) for the same data set. For the Fault Diagnosis prob-

lem, the proposed method is compared with Support Vector Machines (SVM) [Kulkarni et al. \[2005\]](#), [Chiang et al. \[2004\]](#), [Mahadevan and Shah \[2009\]](#), Random Forest, Structure SVM, and sm-NLPCA (architecture used: Stacked Autoencoder). It will be shown that the proposed relevance based method with DSAE or DDSAE networks significantly increases the average fault detection and diagnosis accuracy over other methods.

The following chapter is organized as follows. The mathematical modelling tools including basic Autoencoder (AE), Deep Supervised Autoencoder (DSAE), its dynamic version Dynamic Deep Supervised Autoencoder (DDSAE) neural networks and Layerwise Relevance Propagation (LRP) for computing the relative importance of input variables for explaining the predictions of DNNs are introduced in Section 4.2. The developed methodology for both Fault Detection and Diagnosis (FDD) is presented in Section 4.3. The application of the proposed method to the case study of Tennessee Eastman Process and comparisons to other methods are presented in Section 4.4 followed by conclusions presented in Section 4.5.

4.2 Preliminaries

This section briefly reviews the fundamentals of a Supervised Deep Autoencoder Neural Networks (DSAE-NNs), Dynamic Deep Supervised Autoencoder (DDSAE-NNs) and Layerwise Relevance Propagation (LRP).

4.2.1 Deep Supervised Autoencoder Classification Neural Networks (DSAE-NNs)

The overall goal is to learn a function that predicts the class labels in one-hot encoded form $\mathbf{y}_i \in \mathbb{R}^m$ from inputs $\mathbf{x}_i \in \mathbb{R}^{d_x}$. The operation performed by the encoder for a single hidden layer between the input variables to the latent variables $\mathbf{z}_i \in \mathbb{R}^{d_z}$ can be mathematically described as follows:

$$\mathbf{z}_i = f_e(\mathbf{W}_e \mathbf{x}_i + \mathbf{b}_e) \tag{4.1}$$

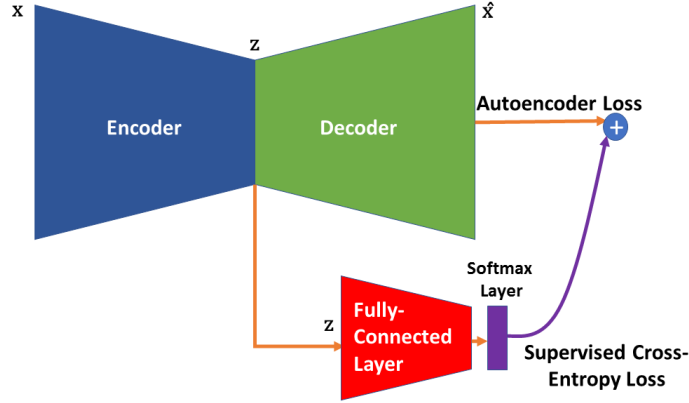


Figure 4.1: Schematic of a single layer Supervised Autoencoder Neural Network (SAE-NN)

The latent variables are used both to predict the class labels and to reconstruct back the inputs \mathbf{x} as follows:

$$\hat{\mathbf{x}}_i = f_d(\mathbf{W}_d \mathbf{z}_i + \mathbf{b}_d) \quad (4.2)$$

$$\hat{\mathbf{y}}_i = f_c(\mathbf{W}_c \mathbf{z}_i + \mathbf{b}_c) \quad (4.3)$$

where f_c is a non-linear activation function applied for the output layer. $\mathbf{W}_c \in \mathbb{R}^{m \times d_z}$ and $\mathbf{b}_c \in \mathbb{R}^m$ are output weight matrix and bias vector respectively. The training of an Deep Supervised Autoencoder Neural Network (DSAE-NN) model, schematically shown in Figure 4.1, is based on the minimization of a weighted sum of the reconstruction loss function and the supervised classification loss corresponding to the first and second terms in (Equation (4.4)) respectively. The reconstruction loss function in Equation (4.4) is ensuring that the estimated latent variables are able to capture the variance in the input data while the classification loss is ensuring that only those non-linear latent variables are extracted that are correlated with output classes. Mean squared error function is used as

a reconstruction loss and softmax cross-entropy as the classification loss.

$$\begin{aligned}
l_{DSAE} &= \lambda_1 \sum_{s=1}^N L_r^s(\mathbf{x}_s, \mathbf{W}_d \mathbf{W}_e \mathbf{x}_s) + \sum_{s=1}^N L_p^s(\mathbf{W}_c \mathbf{W}_e \mathbf{x}_s, \mathbf{y}_s) \\
&= \frac{\lambda_1}{N} \|\mathbf{x}_s - \hat{\mathbf{x}}_s\|_2^2 + \frac{1}{N} \sum_{s=1}^N \sum_{c=1}^m -y_{s,c} \log(p_{s,c}) \\
&= \frac{1}{N} \left[\lambda_1 \|\mathbf{x}_s - \hat{\mathbf{x}}_s\|_2^2 + \sum_{s=1}^N \sum_{c=1}^m -y_{s,c} \log(p_{s,c}) \right] \tag{4.4}
\end{aligned}$$

$$p_{s,c} = \frac{e^{(y_{s,c}^{\hat{}})}}{\sum_{c=1}^m e^{(y_{s,c}^{\hat{}})}} \tag{4.5}$$

where λ_1 is the weight for the reconstruction loss L_r , m is the number of classes, $y_{s,c}$ is a binary indicator (0 or 1) equal to 1 if the class label c is the correct one for observation s and 0 otherwise, $y_{s,c}^{\hat{}}$ is the non-normalized log probabilities and $p_{s,c}$ is the predicted probability for a sample s of class c . Moreover, to avoid over-fitting, a regularization term is added to the objective function in Equation 4.4. Hence, the objective function for Deep Supervised Autoencoder NNs used for Fault Detection (number of classes $m = 2$, normal or faulty) is as follows:

$$\min_{\mathbf{W}} l_{DSAE} = \min \frac{1}{N} \left[\lambda_1 \|\mathbf{x}_s - \hat{\mathbf{x}}_s\|_2^2 + \lambda_2 \sum_{s=1}^N \sum_{c=1}^m -y_{s,c} \log(p_{s,c}) + \lambda_3 \sum_L \sum_k \sum_j \mathbf{W}_{kj}^{2[L]} \right] \tag{4.6}$$

where $\mathbf{W}_{kj}^{[L]}$ are the weight matrices for each layer L in the network ($L = 1$ in this example) and the weights on the individual objective functions $\lambda_1, \lambda_2, \lambda_3$ are chosen using validation data.

4.2.2 Dynamic Deep Supervised Autoencoder Classification Neural Networks (DDSAE-NNs)

The static DSAE-NN presented above assumes that the sampled data are independent to each other, and hence, temporal correlations are ignored. To account for the correlations

in time between the data samples, a Dynamic Deep Supervised Autoencoders (DDSAE) model has been proposed by using a dynamic extension matrix. Accordingly, the original DSAE-NN model can be extended to take into account auto-correlations in time correlated data by augmenting each sample vector with the previous l observations and stacking the data matrix with the resulting vectors, each corresponding to different time intervals.

The dynamic augmentation of the input data matrix \mathbf{X}_D by stacking previous l observations to input data matrix \mathbf{X} is as follows:

$$\mathbf{X}_D = [\mathbf{x}_{l+1}^D \ \mathbf{x}_{l+2}^D \ \mathbf{x}_{l+3}^D \ \dots \ \mathbf{x}_N^D]^T \in \mathbb{R}^{(N-l) \times ((l+1)d_x)} \quad (4.7)$$

where $\mathbf{x}_n^D = [\mathbf{x}_n \ \mathbf{x}_{n-1} \ \mathbf{x}_{n-2} \ \dots \ \mathbf{x}_1]$, where \mathbf{x}_n is a \mathbb{R}^{d_x} dimensional vector of all the input feature vectors/ variables. The different time window length is chosen to build DDSAE models, in which the best classification performance is the final time window length. The following objective function (Equation 4.8) is minimized with training data $\mathbf{X}_D^{N-l+1} = \{\mathbf{x}\}_{i=1}^{N-l+1}$, $\mathbf{y}_D = \{y\}_{i=1}^{N-l+1}$, where $N - l + 1$ is the total number of samples:

$$\min_{\mathbf{W}} l_{DDSAE} = \min \frac{1}{(N-l+1)} \left[\lambda_1 \|\mathbf{x}_s^D - \hat{\mathbf{x}}_s^D\|_2^2 + \lambda_2 \sum_{s=1}^{N-l+1} \sum_{c=1}^m -y_{s,c} \log(p_{s,c}) + \lambda_3 \sum_L \sum_k \sum_j \mathbf{W}_{kj}^{2[L]} \right] \quad (4.8)$$

Note that the number of samples for the augmented dynamic matrix for the training data decreases as compared to the static DSAE case.

4.2.3 Layer-wise Relevance Propagation (LRP)

Layer-wise Relevance Propagation (LRP) was introduced by [Bach et al. \[2015\]](#) to assess the relevance of each input variable or features with respect to outputs using a trained NN. It is based on a layer-wise relevance conservation principle where each relevance $[R_c(x_i; f_c)]_j$, $i = 1, 2, \dots, n$, where n is the number of input variables/ feature vectors (x_i , $i = 1, 2, \dots, n$) for a j^{th} sample where $j = 1, 2, \dots, N$, where N is the total number

of samples in the training dataset, is calculated by propagating the output scores for a particular task c towards the input layer of the network. Previously, it has been used in the area of health-care for attributing a relevance value to each pixel of an image to explain the relevance in a image classification task using DNNs [Yang et al. \[2018\]](#), to explain the predictions of a NN in the area of sentiment analysis [Arras et al. \[2017\]](#), to identify the audio source in reverberant environments when multiple sources are active [Perotin et al. \[2019\]](#), and to identify EEG patterns that explain decisions in brain-computer interfaces [Sturm et al. \[2016\]](#). In process systems engineering LRP has been recently applied by the authors for the first time for FDD problems. The method was used for identifying relevant input variables and pruning irrelevant input variables (input nodes) with respect to a specific classification task for both Multi-layer Perceptron (MLP) NN and Long-Short Term Memory (LSTM) NN by [Agarwal and Budman \[2019\]](#), [Agarwal et al. \[2019\]](#).

To compute the relevance of each input variable $x_i, i = 1, 2, \dots, n$ for the DSAE-NNs and DDSAE-NNs models (used in this work), are trained for both fault detection and fault classification using the training dataset $\chi : \{\mathbf{X}^l, \mathbf{y}^l\}$ and $\chi_D : \{\mathbf{X}_D^l, \mathbf{y}_D^l\}$ (χ_D : dynamic version of χ) respectively. Subsequently, the score value f_c of the corresponding class for the j^{th} sample is back-propagated through the network towards the input. Depending on the nature of the connection between layers and neurons, a layer-by-layer relevance score is computed for each intermediate lower-layer neuron. Different LRP rules have been proposed for attributing relevance for the input variables. In this work, we use the ϵ epsilon rule [Bach et al. \[2015\]](#) for computing relevances that are given as follows:

$$R_{l \leftarrow u} = \sum_u \frac{x_l w_{lu}}{\sum_l x_l w_{lu} + \epsilon} R_u \quad (4.9)$$

where ϵ is used to prevent numerical instability when z_u is close to zero and w_{lu} are the weights connecting lower layer neurons l and upper-layer neurons u . As ϵ becomes larger, only the most salient explanation factors survive the absorption. This typically leads to explanations that are sparser in terms of input features and less noisy [Montavon et al. \[2019\]](#).

Let us consider a simple example for the propagation of relevances from the output

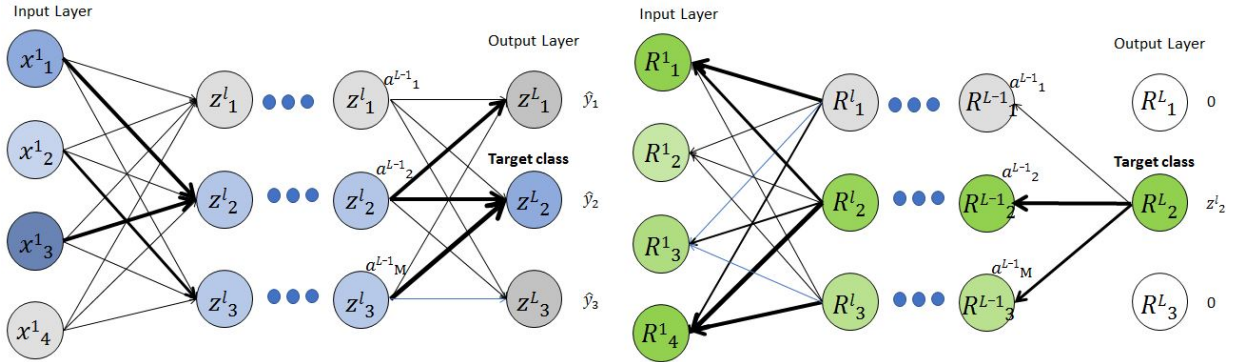


Figure 4.2: Left figure: Represents forward contribution of each node to the output layer; Right figure: Represents the relevance propagation from output layer to the input layer

layer to the input layer of a NN as shown in Figure 4.2. For each layer l in a network with L total layers, $1, \dots, m, \dots, M$ are the nodes in layer $l - 1$, and $1, \dots, n, \dots, N$ nodes are the nodes in the layer l . z_n^l is the pre-activation function value of the node, w_{mn}^{l-1} is the weight connecting nodes m and n and a_m^{l-1} is the output of a node post activation function for a node m in layer $l - 1$. Then, the relevance for a sample at node m in layer $l - 1$ is calculated as follows:

$$R_m^{l-1} = \sum_n \frac{a_n^{l-1} w_{nm}^{l-1}}{\sum_m a_m^{l-1} w_{mn}^{l-1}} R_n^l \quad (4.10)$$

Usually the contribution of relevance for a node m comes from all the nodes n of a given layer l . However, specifically the propagation of relevance from the the output softmax (last) layer to the layer before the last one, we consider the contribution coming from the target node only.

Relevances $\mathbf{R}_c(\mathbf{x}; \mathbf{f}_c)_j = [R_{c_1}(x_1; f_c), R_{c_2}(x_2; f_c), \dots, R_{c_n}(x_n; f_c)]_j \in \mathbb{R}^n$, $i = 1, 2, \dots, n$ for each input feature x_i are calculated for the j^{th} sample with respect to a classification task c in the training dataset \mathcal{X} or \mathcal{X}_D . Since the goal is to prune the irrelevant input features DNNs based on estimated relevance scores using LRP, it is important to average the relevance scores over all the correctly classified samples in the training dataset. Therefore,

the final average input relevances with respect to the overall classification task c can be calculated as follows:

$$\mathbf{R}_c = \frac{1}{N_c} \sum_{j=1}^{N_c} |R_c(\mathbf{x}; \mathbf{f}_c)_j| \quad (4.11)$$

where N_c is the number of correctly classified samples in the training dataset. Furthermore, the least relevant input features are pruned based on the average relevance scores for all input variables calculated with Equation 4.11. In practise a threshold of $\lambda \times \max(\mathbf{R}_c)$ is chosen to prune the irrelevant variables where λ is an hyper-parameter that is determined by using the validation dataset (heuristically λ is chosen as 0.01 as the starting value). Relevance of input variables below the threshold are removed from the dataset and the network is re-trained until the same or higher validation accuracy is achieved. It is to be noted that the DNN has to be re-trained with the set of remaining input variables after pruning and the testing accuracy increases with successive iterations as shown later in Section 4.4. For the dynamic augmented input matrix χ_D shown in Equation 4.7, the final input relevances (combining effects of individual lagged variables) with respect to the overall classification task c is:

$$\mathbf{R}_c = \frac{1}{N_c} \sum_{j=1}^{N_c} \sum_{i=1}^{l+1} |R_{c_i}(x_i; f_c)|_j \quad (4.12)$$

where l is the number of time lag window included in the dataset \mathbf{X}_D^l .

4.3 Proposed Fault Detection and Diagnosis Methodology based on DSAE-NNs and DDSAE-NNs

Both the Deep Supervised Autoencoder NN (DSAE-NN) and Dynamic Deep Supervised Autoencoder NN (DDSAE-NN) are used for FDD and are the basis for the explainable-pruning based methodology presented in the previous section. The proposed fault detection

algorithm is first used to extract deep features to detect if the process is operating in a normal or faulty region. Then, a fault diagnosis algorithm is applied in case the sample indicates faulty operation to identify the particular fault and possible root-cause of the occurring fault in the process using an DDSAE-NN. Since the latter is iteratively trained by using the LRP based pruning procedure that provides explainability of input variables the resulting DDSAE-NN model will be referred to as xDDSAE-NN.

4.3.1 Fault Detection Methodology

First, a DSAE-NN is trained using the training data $(\mathbf{X}^l, \mathbf{y}^l)$. The fault detection process is formulated as a binary classification problem. Often, this binary classification task for Fault Detection is susceptible to a ‘class imbalance problem’ because of the unequal distribution of classes in the training dataset. For example, the number of training samples for the normal operating region may be far less than the samples for abnormal operating region or vice-versa. To address this class imbalance problem an extra weight δ is introduced in the loss functions in Equation 4.13 and Equation 4.14 is as follows:

$$\min_{\mathbf{w}} l_{DSAE} = \min \frac{1}{N} \left[\lambda_1 \sum_{i=1}^N \|\mathbf{x}_s - \hat{\mathbf{x}}_s\|_2^2 - \lambda_2 \sum_{s=1}^N (\delta y_{s,1} \log(p_{s,1}) + y_{s,2} \log(p_{s,2})) + \lambda_3 \sum_L \sum_k \sum_j \mathbf{w}_{kj}^{2[L]} \right] \quad (4.13)$$

$$\min_{\mathbf{w}} l_{DDSAE} = \min \frac{1}{N} \left[\lambda_1 \sum_{i=1}^N \|\mathbf{x}_s^D - \hat{\mathbf{x}}_s^D\|_2^2 - \lambda_2 \sum_{s=1}^N (\delta y_{s,1} \log(p_{s,1}) + y_{s,2} \log(p_{s,2})) + \lambda_3 \sum_L \sum_k \sum_j \mathbf{w}_{kj}^{2[L]} \right] \quad (4.14)$$

For example, if there are more data samples of faulty operation than samples for normal operation higher weights would be assigned to the samples belonging to the normal operating region class. The value of δ dictates a trade-off between false positives and true negatives and is considered as an additional hyper-parameter to the model that is ultimately chosen using the validation data-set. Initially the DSAE-NN is trained on the training dataset $\mathbf{X}^l, \mathbf{y}^l$ using all the input-variables. The best performing model is chosen

using a validation dataset $\mathbf{X}^v, \mathbf{y}^v$. Then, the LRP is implemented to explain the predictions of the chosen DSAE-NN with a set of hyper-parameters by computing the relevance of each input variable. Relevances $\mathbf{R}_c(\mathbf{x}; \mathbf{f}_c)_j = [R_{c_1}(x_1; f_c), R_{c_2}(x_2; f_c), \dots, R_{c_n}(x_n; f_c)]_j \in \mathbb{R}^n$, $i = 1, 2, \dots, n$ for each input feature x_i are calculated for the j^{th} sample with respect to a classification task c in the training dataset \mathcal{X} or \mathcal{X}_D . Since the goal is to prune the irrelevant input features DNNs based on estimated relevance scores using LRP, it is important to average the relevance scores over all the correctly classified samples in the training dataset. Therefore, the final input relevances with respect to the overall classification task c can be calculated as follows:

$$\mathbf{R}_c = \frac{1}{N_c} \sum_{j=1}^{N_c} |R_c(\mathbf{x}; \mathbf{f}_c)_j| \quad (4.15)$$

where N_c is the number of correctly classified samples in the training dataset. Furthermore, the least relevant input features are pruned based on the average relevance scores for all input variables calculated with Equation 4.15. In practise a threshold of $\lambda \times \max(\mathbf{R}_c)$ is chosen to prune the irrelevant variables where λ is an hyper-parameter that is determined by using the validation dataset (heuristically λ is chosen as 0.01 as the starting value). Relevance of input variables below the threshold are removed from the dataset and the network is re-trained until the same or higher validation accuracy is achieved. It is to be noted that the DNN has to be re-trained with the set of remaining input variables after pruning and the testing accuracy increases with successive iterations as shown later in Section 4. For the dynamic augmented input matrix \mathcal{X}_D shown in Equation 4.7, the final input relevances (combining effects of individual lagged variables) with respect to the overall classification task c is:

$$\mathbf{R}_c = \frac{1}{N_c} \sum_{j=1}^{N_c} \sum_{i=1}^{l+1} |R_{c_i}(x_i; f_c)|_j \quad (4.16)$$

where l is the number of time lag window included in the dataset \mathbf{X}_D^l . Subsequently, the eXplainable DSAE (xDSAE) neural network is re-trained using the reduced training dataset $\{\mathbf{X}^l, \mathbf{y}^l\} \rightarrow \{\mathbf{X}_r^l, \mathbf{y}_r^l\}$ at each iteration. The premise for reducing the dimensionality of the input data by discarding less relevant inputs is that the information content can often be represented by a lower dimensional space, implying that only a few fundamental variables are sufficient to account for the variation in the data that are most informative about the identification of faults and normal regions. Once all the relevant input variables that are significant to the classification task are chosen, an eXplainable DDSAE-NN (xDDSAE-NN) is trained with the remaining inputs and the reduced training data matrix \mathbf{X}_r^l is augmented with the lagged variables of the remaining input variables $\mathbf{X}_r^{l,D}$. The process of discarding input feature vectors is iterative and xDDSAE-NN is iteratively retrained using the validation dataset. This approach has multiple advantages over other reported methods used for fault detection as follows:

1. Improvement in test classification accuracy.
2. Identification of an eXplainable empirical model
3. Synthesis of a smaller network with fewer parameters

4.3.2 Fault Diagnosis Methodology

After detecting that the process has deviated from the normal operation and a fault has occurred, it is desired to diagnose the type of fault. For fault diagnosis, a similar methodology to the one used for fault detection is applied for the classification of the type of fault. First, the static DSAE is used to extract deep features and predict the type of fault in a process plant. For this task one-hot encoded outputs are utilized as the labels for training the model. Initially, the DSAE-NN is trained on the training dataset $\mathbf{X}^l, \mathbf{y}^l$ using all the input-variables. The best performing model is chosen using a validation $\mathbf{X}^v, \mathbf{y}^v$. LRP is subsequently implemented to explain the predictions of the selected DSAE-NN by computing the relevance of each input variable. The irrelevant features are removed by comparing the relevances to a threshold. Then an xDSAE NN is trained using the reduced

training dataset $\mathbf{X}_r^l, \mathbf{y}_r^l$ by successive iterations of pruning of irrelevant inputs and model re-training until the relevance of all the remaining input variables are above the threshold. Since data collected from chemical processes have strong dynamic/temporal correlations, the input data matrix \mathbf{X} is augmented with observations at l previous time steps for each input feature dimension (refer Equation 4.7) and a DDSAE-NN is trained. The iterative procedure of discarding input variables from the reduced dynamic matrix $\mathbf{X}_r^{l,D}$ is implemented and pruning and re-training is applied as long as validation accuracy continue to increase after discarding features. The decision of adding lagged variables only to the remaining input variables of the final iteration of xDDSAE model is justified by the fact that the input variables that were eliminated do not have an instantaneous effect of x_k on fault detection. Then, since the pruned input variables at current time are auto-correlated in time to previous values ($\mathbf{x}_k \propto f(\mathbf{x}_{k-1}, \mathbf{x}_{k-2}, \dots, \mathbf{x}_{k-n})$), if current values are not correlated to the model outputs then their corresponding previous values (lagged variables) are also not correlated to these outputs.

4.3.3 Proposed Methodology for FDD

The proposed methodology for FDD is schematically described in Figure 4.3 and it is summarized by the following steps.

1. Pre-process the input data. $\mathbf{X}_{\text{raw}} \rightarrow \mathbf{X}^l$.
2. Build a DSAE-NN that maps the input vectors into the latent feature space by using a DSAE model structure. The goal is to extract discriminative features that capture the latent manifold in the input data that are most correlated with the output classes. The parameters are optimized using a combination of the reconstruction error, L_2 regularization error and binary softmax cross-entropy error of the input data (refer Equation 4.13).
3. Select the best performing model architecture (number of layers and nodes) along with the set of hyper-parameters that include learning rate, batch size, and all other

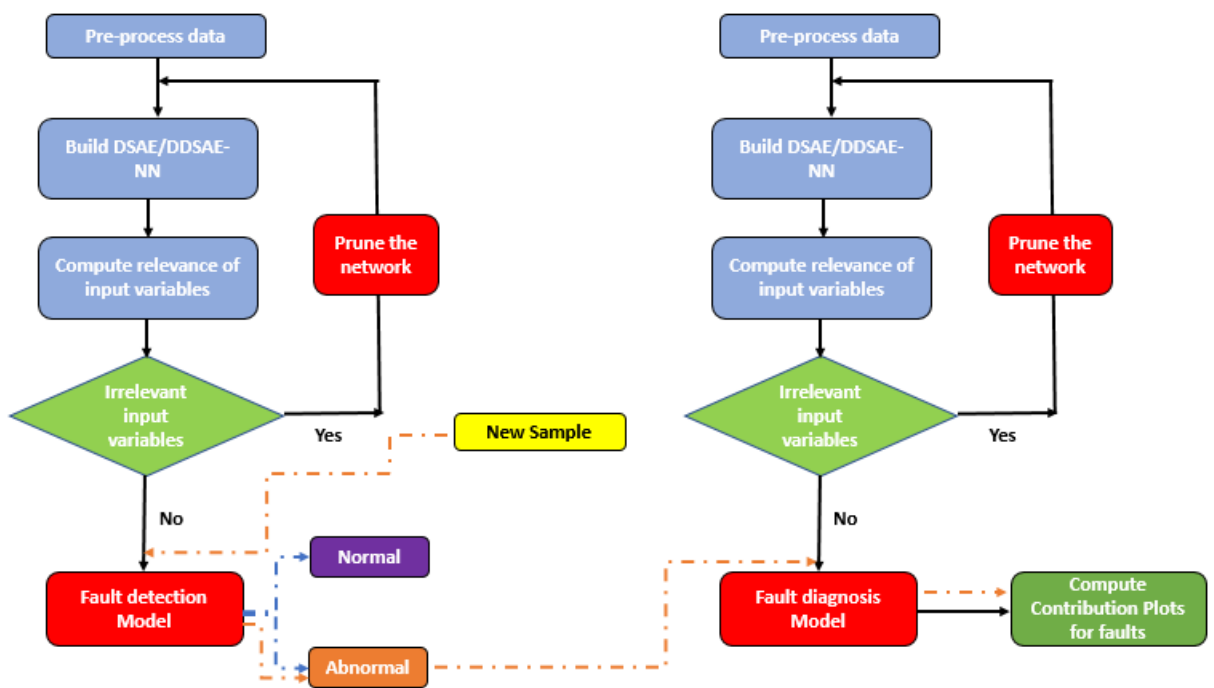


Figure 4.3: Flowchart for fault detection and diagnosis based on explainable DNN

weighting ($\lambda_1, \lambda_2, \lambda_3$ and δ) parameters using the validation dataset. $(\mathbf{X}^v, \mathbf{y}^v)$.

4. Evaluate the classification accuracy using the chosen trained DSAE model in Step 3 for testing dataset $(\mathbf{X}^t, \mathbf{y}^t)$. If the model in testing is satisfactory, the model will be used for further analysis; if unsatisfactory, return to Step 3 to redesign the DSAE-NN model.
5. Compute input relevances using the LRP method on the training and validation dataset and discard irrelevant input features from the dataset $\mathbf{X}^l, \mathbf{X}^v$ and \mathbf{X}^t .
6. Repeat steps 3,4 and 5 until relevances of all input variables are above the threshold and no improvement over the validation accuracy is achieved for xDSAE NN.
7. Build xDDSAE model using the reduced input dataset \mathbf{X}_r^l computed in Step 6 along with augmenting l lagged variables.
8. Select the best performing model architecture (number of layers and nodes) using the validation dataset $(\mathbf{X}_r^v, \mathbf{y}_r^v)$.
9. Repeat steps 3,4 and 5 until relevances of all input variables are above the threshold and no improvement of the validation accuracy is achieved.
10. For online process monitoring: When a new data vector \mathbf{X}_{new} becomes available, import it into the model after normalizing to determine whether the current state of the process is in normal or abnormal operating region and to determine the type of fault that is responsible for the deviation.

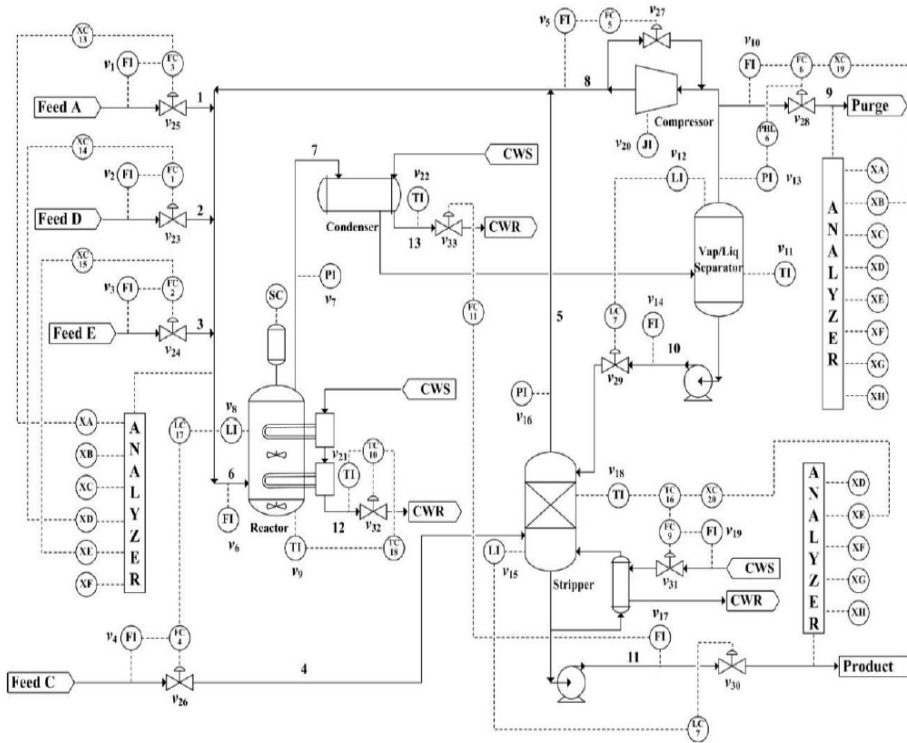


Figure 4.4: Schematic: Tennessee Eastman plant process (Downs and Vogel, 1993)

4.4 Case Study: Tennessee Eastman Process

In this section the proposed methodology is implemented for FDD and the performance is compared with different approaches in the literature on the benchmark Tennessee Eastman Process (TEP). The Tennessee Eastman plant has been used widely for testing several process monitoring and fault detection algorithms [Chiang et al. \[2000\]](#), [Lau et al. \[2013\]](#), [Rato and Reis \[2013\]](#), [Xie and Bai \[2015\]](#), [Ricker \[1996\]](#), [Bathelt et al. \[2015\]](#), [Kulkarni et al. \[2005\]](#), [Larsson et al. \[2001\]](#). The TEP involves different unit operations including a vapor-liquid separator, a reactor, stripper a recycle compressor and a condenser. Four gaseous reactants (A, B, C and D) forms two liquid products streams (G and H) and a by-product (F). A schematic of the Tennessee Eastman Process is illustrated in Figure 4.4. [Downs and Vogel \(1993\)](#)[Downs and Vogel \[1993\]](#) reported the original simulator for this process

Table 4.1: Measured and manipulated variables (from Downs and Vogel, 1993)

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

and has been widely used as a benchmark process for control and monitoring studies (simulator available at <http://depts.washington.edu/control/LARRY/TE/download.html>). The process simulator involves a total of 52 measured variables including 22 process (output) variables, 11 manipulated variables and 19 composition measurements. A complete list of output measurements and manipulated variables are presented in Table 4.1. Additional details about the process model can be found in the original paper Downs and Vogel [1993] and descriptions of the different control schemes that have been applied to the simulator can be found in Ricker [1996] and its revised version Bathelt et al. [2015]. Several data-driven statistical process monitoring approaches have been reported for the detection and diagnosis of disturbances in the Tennessee Eastman simulation. There are 20 different process disturbances (fault types) in the industrial simulator (shown in Table 4.2) though only 17 were used in this work to be consistent with other methods in the

Table 4.2: Process Faults for classification in TE Process

Fault	Description	Type
IDV(1)	A/C feed ratio, B composition constant (stream 4)	step
IDV(2)	B composition, A/C ratio constant (stream 4)	step
IDV(3)	D Feed Temperature	step
IDV(4)	Reactor cooling water inlet temperature	step
IDV(5)	Condenser cooling water inlet temperature (stream 2)	step
IDV(6)	A feed loss (stream 1)	step
IDV(7)	C header pressure loss reduced availability (stream 4)	step
IDV(8)	A, B, C feed composition (stream 4)	random variation
IDV(9)	D Feed Temperature	random variation
IDV(10)	C feed temperature (stream 4)	random variation
IDV(11)	Reactor cooling water inlet temperature	random variation
IDV(12)	Condenser cooling water inlet temperature	random variation
IDV(13)	Reaction kinetics	slow drift
IDV(14)	Reactor cooling water	valve sticking
IDV(15)	Condenser Cooling Water Valve	stiction
IDV(16)	Deviations of heat transfer within stripper	random variation
IDV(17)	Deviations of heat transfer within reactor	random variation
IDV(18)	Deviations of heat transfer within condenser	random variation
IDV(19)	Recycle valve of compressor, underflow stripper and steam valve stripper	stiction
IDV(20)	unknown	random variation

literature. Each of these methods has shown different levels of success in detecting and diagnosing the faults considered in the simulations. Several statistical studies have reported faults 3, 9 and 15 as unobservable or difficult to diagnose due to the close similarity in the responses of the noisy measurements used to detect these faults [Lau et al. \[2013\]](#), [Shams et al. \[2010\]](#), [Chiang et al. \[2000\]](#), [Du and Du \[2018\]](#) and therefore these 3 faults were not considered in the current study.

The training data consists of 500 samples of normal data and 480 samples for each fault. The testing dataset has 960 samples for both faulty and normal operation data. For the faulty testing dataset, the fault is introduced at 160 time-sample. A part of the training data $\{\mathbf{X}^l, \mathbf{y}^l\}$ is used as the validation dataset $\{\mathbf{X}^v, \mathbf{y}^v\}$ for tuning the hyper-parameters (learning rate, weights: $\lambda, \lambda_1, \lambda_2, \lambda_3$ and δ , number of epochs, layers and nodes in each layer) for both DSAE/ xDSAE and DDSAE/ xDDSAE DNNs for all iterations. These hyper-parameters, such as number of layers, number of neurons in each layer, classification weights, learning rate, time-horizon etc. are selected using validation data. It should

be noticed that the data were divided into 3 sets: training, validation and testing data. The weights of the network are obtained for a certain 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. The hyper-parameter search is implemented using keras-tuner in Python. Firstly, a grid of hyper-parameters is defined, for example number of encoder layers = [1,2,3,4,5,6,7], number of neurons units for each of these layers ranging from 2 to 400, learning rate = [1e⁻¹,2e⁻¹,3e⁻¹, 1e⁻²], value of weights in the objective function, etc. 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 test accuracy.

The network architectures and test accuracy for both fault detection and diagnosis are presented in Table 4.4 and 4.5 respectively. For example, for a particular entry in Table 4 of an architecture as 52-5-10-5-52 the notation is as follows. The first number represents the number of neurons in the input layer, subsequently the second layer consists of 5 neurons and the bottleneck layer of autoencoder consists of 10 neurons. Another dense layer network is attached to the bottleneck layer of 10 neurons with an output layer of 2 neurons for fault detection (refer to Figure 4.1). This dense network is used for classification. The decoder layers are also connected to the bottleneck layer with 5 neurons and finally the output layer has 52 neurons for the reconstruction of inputs. After applying the LRP procedure to the static fault detection model, it is found that only 24 out of 52 variables are the most important for obtaining the highest testing accuracy for detecting the correct state of the process plant. After every iteration of the input pruning-relevance (LRP) based procedure, it is shown in Table 4.4 that the removal of irrelevant input variables results in successive improvement of fault class separability. To account for the dynamic information after identifying the 24 most relevant process variables, the reduced input data matrix $\{\mathbf{X}_r^l\}$ is stacked with lagged time stamps and an DDSAE NN model is retrained. The best fault detection test accuracy of 96.43% is achieved by stacking two previous time-stamp process

values. The fault detection rates for all the faults are shown in Table 4.6. These results are compared in the same Table 4.6 with several methods as follows: PCA [Lv et al. \[2016\]](#), DPCA [Lv et al. \[2016\]](#), ICA [Hsu et al. \[2010\]](#), Convolutional NN (CNN) [Singh Chadha et al. \[2019\]](#), Deep Stacked Network (DSN) [Chadha and Schwung \[2017\]](#), Stacked Autoencoder (SAE) [Chadha and Schwung \[2017\]](#), Generative Adversarial Network (GAN) [Spyridon and Boutalis \[2018\]](#) and One-Class SVM (OCSVM) [Spyridon and Boutalis \[2018\]](#). It can be seen from Table 4.6 that the proposed method outperformed the linear multivariate methods and other DL based methods for most fault modes. For example, for PCA with 15 principal components, the average fault detection rates are 61.77% and 74.72% using T^2 and Q statistic respectively. Since the principal components extracted using PCA captures static correlations between variables, DPCA (Dynamic PCA) is used to account for temporal correlations (both auto-correlations and cross-correlations) in the data. Since DPCA is only an input data compression technique, it must be combined with a classification model for the purpose of fault detection. Accordingly, the output features from the DPCA model are fed into an SVM model that is used for final classification. The effect of increasing the number of lagged variables in the dataset is also investigated following the hypothesis that increasing the time horizon will enhance classification accuracy. It can be seen in Table 4 that the increasing number of lags and simultaneous pruning improves the classification accuracy. The average detection rate obtained was 72.35%. ICA ([Hsu et al. \[2010\]](#)) based monitoring scheme was found to perform better than both PCA and DPCA based methods with an averaged accuracy of approximately 90%. In addition to the comparison to linear methods, the proposed methodology was also compared with different DNN architectures such as CNN ([Chadha and Schwung \[2017\]](#)), DSN ([Chadha and Schwung \[2017\]](#)), SAE-NN (results reported in [Chadha and Schwung,2017](#)) and GAN ([Spyridon and Boutalis \[2018\]](#)), OCSVM (results reported in [Spyridon and Boutalis,2018](#)) reported previously. It can be seen that the proposed method also outperforms these DNN based methodologies. Also, the false alarm rate (FAR) i.e. normal samples miss-classified as faulty is 1.46% which is the lowest as compared to all the other methods. Another metric that is predominantly used in the area of fault detection is detection delay i.e. number of samples required for fault detection for the first time. Table 4.3 represents the detection delay for different faults. It is observed that the detection delay for faults 18 and 20 is significantly high.

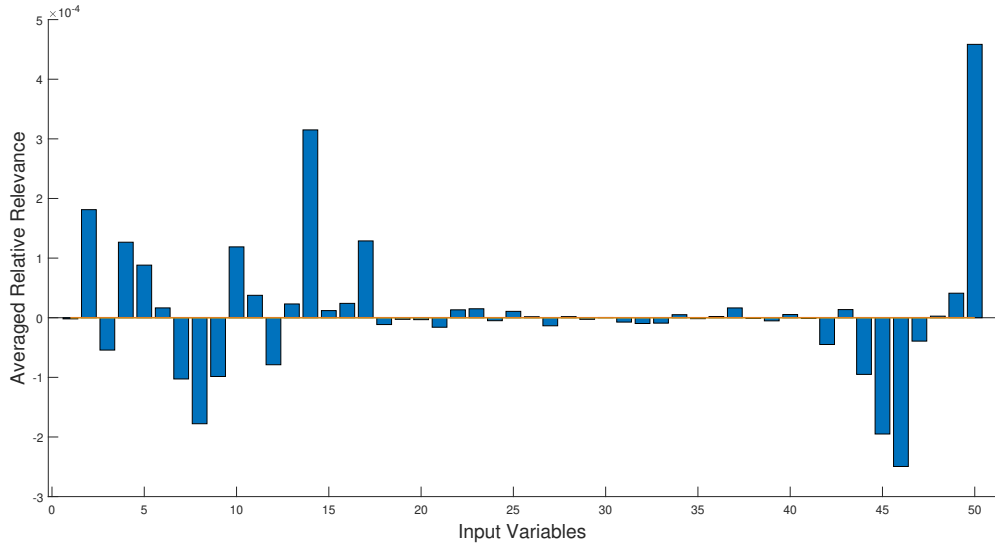


Figure 4.5: Final Iteration: Averaged Relative Relevance Plot for Fault Detection (DDSAE Model with 2 lagged input variables as \mathbf{X}^l)

Some methods are specifically designed for faster detection rate such as [Gajjar et al. Gajjar et al. \[2020\]](#) but have lower FDR.

Table 4.3: Detection Delay for different faults

Faults	1	2	4	5	6	7	8	10	11	12	13	14	16	17	18	19	20
Detection Delay (samples)	0	10	0	0	0	0	8	17	3	0	11	0	5	16	76	0	68

For fault classification with the static DSAE-NN models it is observed that only 235 out of 363 input variables are the most relevant features for obtaining the highest testing accuracy in identifying the type of fault. Every iteration of the proposed methodology conducted for pruning of irrelevant input features increase the classification accuracy as

Table 4.4: Network Architecture and iterations for fault detection methodology

Iteration	Network Type DSAE/ DDSAE	Architecture	Averaged Test Classification Accuracy (FDR)
1	DSAE	52 – 5 – 10* – 5 – 52	91.55%
2	xDSAE	30 – 5 – 10* – 5 – 30	93%
3	xDSAE	24 – 7 – 6* – 7 – 30	93.23%
1	DDSAE (lag1)	48 – 4 – 6* – 4 – 48	93.96%
2	xDDSAE (lag1)	46 – 5 – 7* – 5 – 46	95.52%
3	xDDSAE (lag1)	41 – 5 – 10* – 5 – 41	95.63%
4	xDDSAE (lag1)	40 – 5 – 10* – 2 – 40	95.85%
1	DDSAE (lag2)	72 – 4 – 10* – 4 – 72	93.5%
2	xDDSAE (lag2)	70 – 2 – 10* – 2 – 70	93.53%
3	xDDSAE (lag2)	54 – 4 – 10* – 4 – 54	95.44%
4	xDDSAE (lag2)	50 – 6 – 12* – 6 – 50	96.43%

* A dense layer is present where the number of input nodes are shown with an asterisk and the number of output nodes are 2 (equal to the number of classes).

shown in Table 4.5. After identifying the 33 most relevant process variables with the static DSAE-NN model, the reduced input data matrix $\{\mathbf{X}_r^l\}$ is stacked with lagged time stamps $\{\mathbf{X}_r^l\} \rightarrow \{\mathbf{X}_r^{lD}\}$ and the network is retrained. The best test classification accuracy of 88.41% is achieved by stacking ten previous time-stamp process values. The confusion matrices for the first iteration and the final iteration are shown in Figure 4.6 and 4.7 respectively. It can be seen that there is a significant improvement in the average test classification due to the implementation of proposed methodology. For example: there is an increase of 38% in degree of separability in IDV(8) and 20% increase in IDV(13). The averaged test accuracy for fault classification problem is compared with various non-linear classification algorithms such as Sparse representation, SVM, Random Forest, Structure SVM, AE based classification (sm-NLPCA) method. As shown in Figure 4.14 the proposed methodology outperforms other methods by a significant margin. The averaged relative

Table 4.5: Network Architecture and iterations for fault diagnosis methodology

Iteration	Network Type D _{SAE} / D _{DDSAE}	Architecture	Averaged Test Classification Accuracy (FDR)
1	D _{SAE}	52 – 25 – 20 – 20* – 20 – 25 – 52	81.90%
2	x _D _{SAE}	45 – 10 – 10 – 20* – 10 – 10 – 45	82.60%
3	x _D _{SAE}	33 – 21 – 20 – 20* – 20 – 21 – 33	83.15%
1	D _{DDSAE} (5 lags)	198 – 14 – 10 – 30* – 10 – 14 – 198	83.41%
2	x _D _{DDSAE} (5 lags)	159 – 24 – 10 – 30* – 10 – 24 – 159	85.14%
3	x _D _{DDSAE} (5 lags)	155 – 24 – 8 – 30* – 8 – 24 – 155	85.87%
4	x _D _{DDSAE} (5 lags)	140 – 30 – 20 – 17* – 20 – 30 – 140	86.91%
1	D _{DDSAE} (10 lags)	363 – 14 – 20 – 30* – 20 – 14 – 363	83.08%
2	x _D _{DDSAE} (10 lags)	317 – 18 – 15 – 30* – 15 – 18 – 317	85.04%
3	x _D _{DDSAE} (10 lags)	293 – 24 – 18 – 30* – 18 – 24 – 293	85.51%
4	x _D _{DDSAE} (10 lags)	259 – 28 – 18 – 30* – 18 – 28 – 259	87.07%
5	x _D _{DDSAE} (10 lags)	244 – 34 – 20 – 30* – 20 – 34 – 244	87.86%
6	x _D _{DDSAE} (10 lags)	235 – 38 – 21 – 30* – 21 – 38 – 235	88.41%

* A dense layer is present where the number of input nodes are shown with an asterisk and the number of output nodes are 17 (equal to the number of classes).

importance of each relevant input feature \mathbf{R}_c (estimated using Equation 4.16) towards the classification task c (fault classification) is shown in Figure 4.8. An important by-product of the proposed pruning methodology is that it can explain which input variables significantly deviate from their normal trajectories while the fault is occurring or to identify the root cause of the process fault. Towards that task, averaged input relevances' values for the correctly classified samples for a specific process fault are computed using LRP. For example for Fault 1 (a step change in A/C Feed ratio) the average relative relevance plot for IDV(1) is shown in Figure 4.9. Then, using this plot which are the variables that will significantly deviate from their trajectories during normal operation following the occurrence of the fault. Figure 4.10 shows the evolution of the IDV(1) relevant input variables

Confusion Matrix

1	790 5.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	42 0.3%	6 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	94.2%	5.8%																		
2	0 0.0%	784 5.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	55 0.4%	8 0.1%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	92.3%	7.7%																		
3	0 0.0%	0 0.0%	769 5.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	84 0.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	6 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	89.5%	10.5%																		
4	0 0.0%	0 0.0%	0 0.0%	799 5.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	64 0.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	92.6%	7.4%																		
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	800 5.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100%	0.0%																		
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	798 5.9%	1 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	99.8%	0.2%																		
7	5 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	287 2.1%	29 0.2%	2 0.0%	21 0.2%	142 1.0%	0 0.0%	11 0.1%	1 0.0%	1 0.0%	1 0.0%	5 0.0%	5 0.0%	5 0.0%	56.8%	43.2%																		
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	12 0.1%	555 4.1%	7 0.1%	30 0.2%	14 0.1%	0 0.0%	33 0.2%	0 0.0%	4 0.0%	4 0.0%	12 0.1%	12 0.0%	12 0.0%	82.7%	17.3%																		
9	0 0.0%	1 0.0%	31 0.2%	0 0.0%	0 0.0%	0 0.0%	6 0.0%	5 0.0%	586 4.3%	8 0.1%	10 0.1%	3 0.0%	12 0.1%	23 0.2%	8 0.1%	9 0.1%	13 0.1%	13 0.1%	13 0.1%	82.0%	18.0%																		
10	1 0.0%	2 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	167 1.2%	24 0.2%	4 0.0%	601 4.4%	250 1.8%	1 0.0%	6 0.0%	5 0.0%	25 0.2%	6 0.0%	2 0.0%	2 0.0%	2 0.0%	54.9%	45.1%																		
11	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	36 0.3%	0 0.0%	7 0.1%	15 0.1%	174 1.3%	0 0.0%	1 0.0%	0 0.0%	3 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	73.1%	26.9%																		
12	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	5 0.0%	6 0.0%	6 0.0%	4 0.0%	772 5.7%	0 0.0%	26 0.2%	3 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	93.8%	6.2%																		
13	3 0.0%	5 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	44 0.3%	77 0.6%	30 0.2%	20 0.1%	79 0.6%	0 0.0%	650 4.8%	23 0.2%	27 0.2%	14 0.1%	56 0.4%	56 0.4%	56 0.4%	63.2%	36.8%																		
14	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	22 0.2%	10 0.1%	13 0.1%	3 0.0%	30 0.2%	24 0.2%	4 0.0%	709 5.2%	1 0.0%	1 0.0%	10 0.1%	10 0.1%	10 0.1%	85.7%	14.3%																		
15	0 0.0%	4 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	15 0.1%	16 0.1%	12 0.1%	11 0.0%	0 0.0%	0 0.0%	18 0.1%	1 0.0%	686 5.0%	7 0.1%	31 0.2%	31 0.2%	31 0.2%	85.6%	14.4%																		
16	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	5 0.0%	36 0.3%	30 0.2%	4 0.0%	15 0.1%	0 0.0%	22 0.2%	7 0.1%	14 0.1%	738 5.4%	30 0.2%	30 0.2%	30 0.2%	81.9%	18.1%																		
17	0 0.0%	4 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	103 0.8%	34 0.3%	19 0.1%	15 0.1%	80 0.6%	0 0.0%	43 0.3%	5 0.0%	22 0.2%	19 0.1%	641 4.7%	641 4.7%	641 4.7%	65.0%	35.0%																		
	98.8%	98.0%	96.1%	99.9%	100%	99.8%	35.9%	69.4%	73.3%	75.1%	21.8%	96.5%	81.3%	88.6%	85.8%	92.3%	80.1%	81.9%	81.9%	1.2%	2.0%	3.9%	0.1%	0.0%	0.2%	64.1%	30.6%	26.7%	24.9%	78.3%	3.5%	18.8%	11.4%	14.2%	7.8%	19.9%	18.1%		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17																						

Figure 4.6: Confusion Matrix for Fault Classification (First Iteration: DSAE Model with 52 input variables)

as a function of time. This figure corroborates that all the identified variables according to the average relevance analysis do significantly deviate from their nominal operation values during the occurrence of fault IDV(1). Similarly, an averaged relative relevance plot for IDV(2) is shown in Figure 4.11 and the input variables identified as significant to detect this fault are then plotted in Figure 4.12 as functions of time corroborating that the variables identified as significant in the plot 4.11 are significantly deviating following the fault from their trajectories during normal operation. Similar diagnostics can be run for all the other faults for the TEP problem.

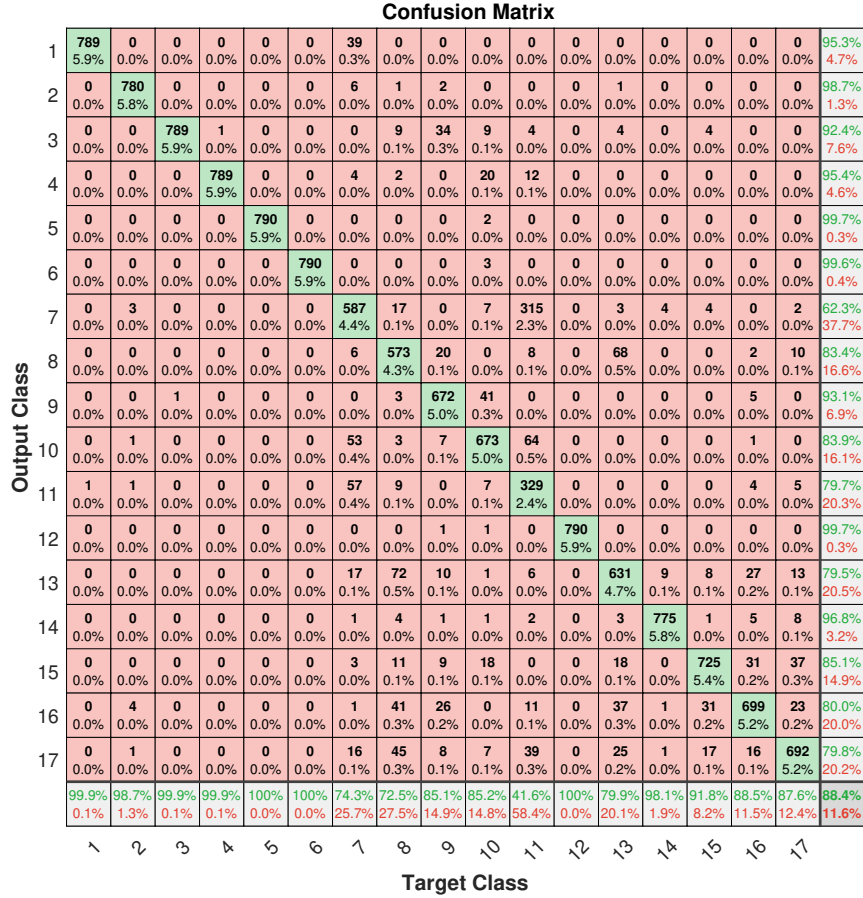


Figure 4.7: Confusion Matrix for Fault Classification (Final Iteration: DDSAE Model with 10 lagged input variables)

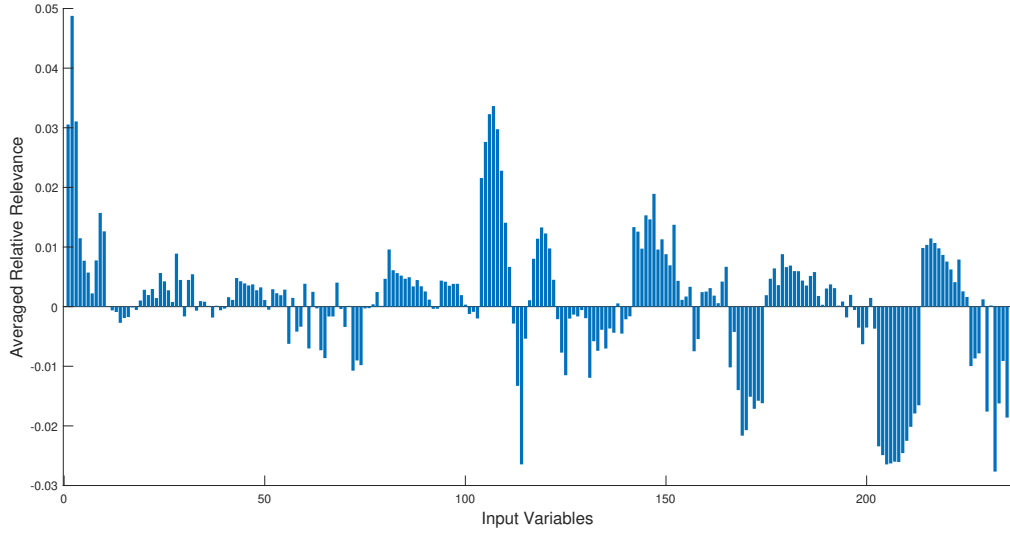


Figure 4.8: Final Iteration: Averaged Relative Relevance Plot for Fault Diagnosis (DDSAE Model with 10 lagged input variables as \mathbf{X}^l)

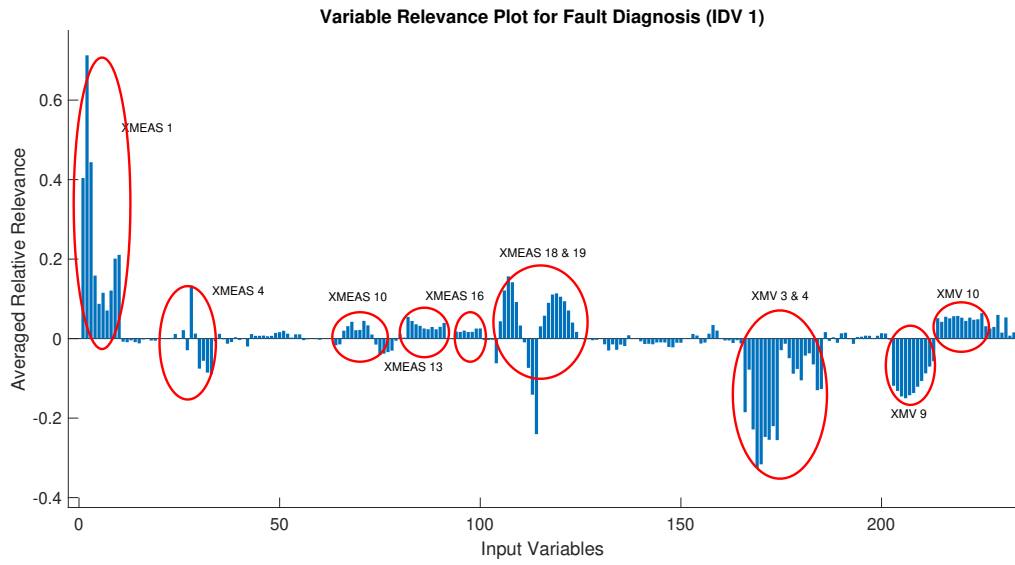


Figure 4.9: Input variable relevance plot for Fault Diagnosis (IDV 1)

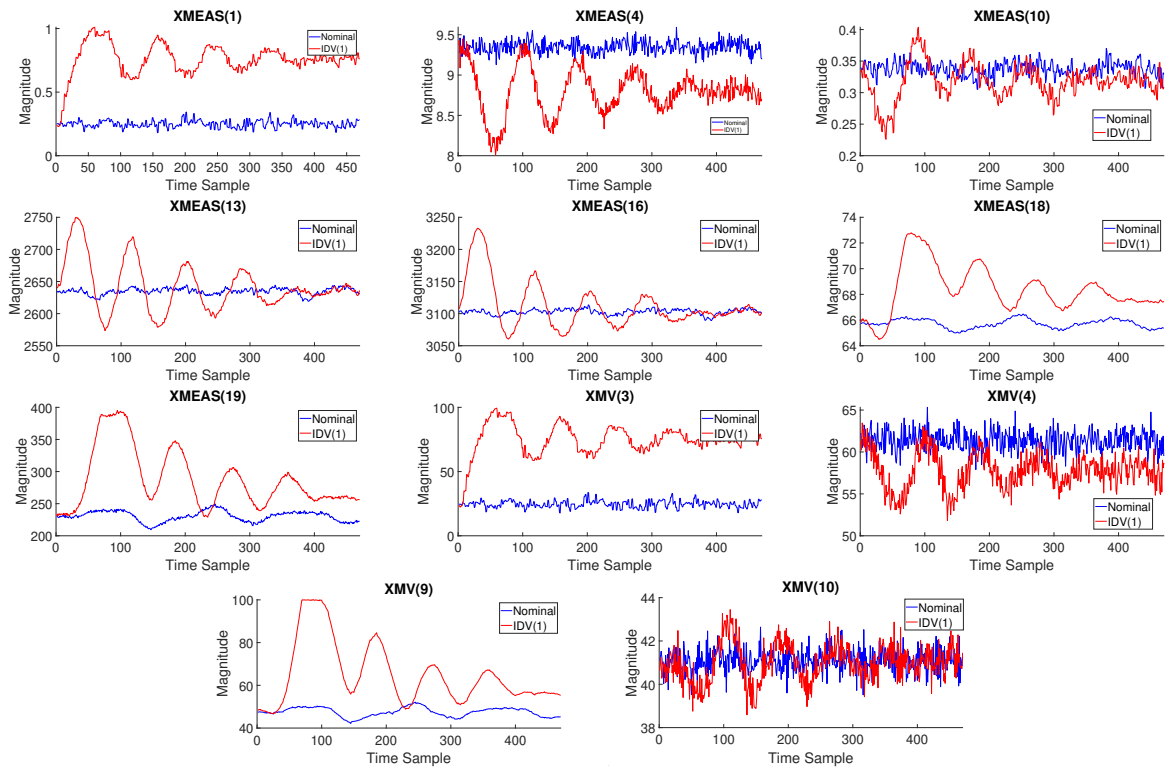


Figure 4.10: Relevant variables contributing to IDV(1) with nominal and abnormal profiles

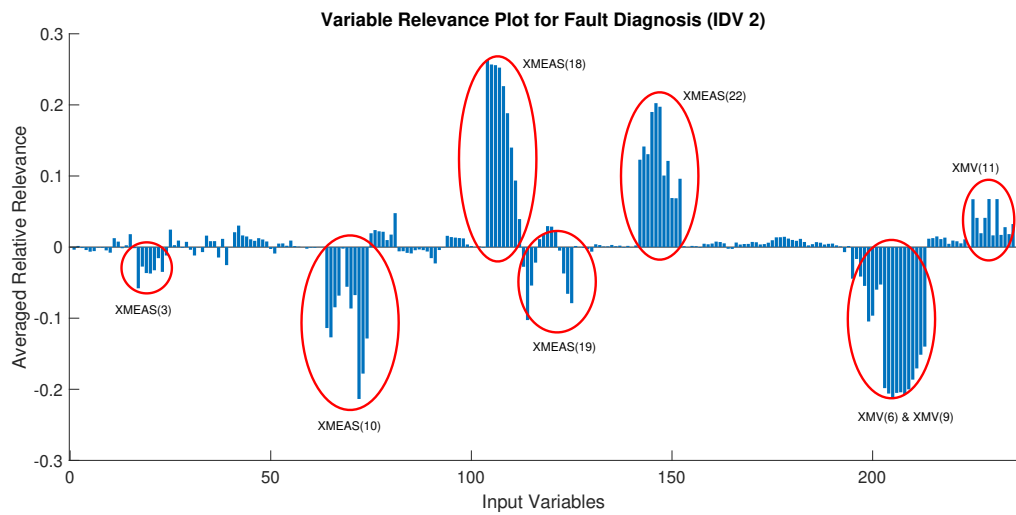


Figure 4.11: Input variable relevance plot for Fault Diagnosis (IDV 2)

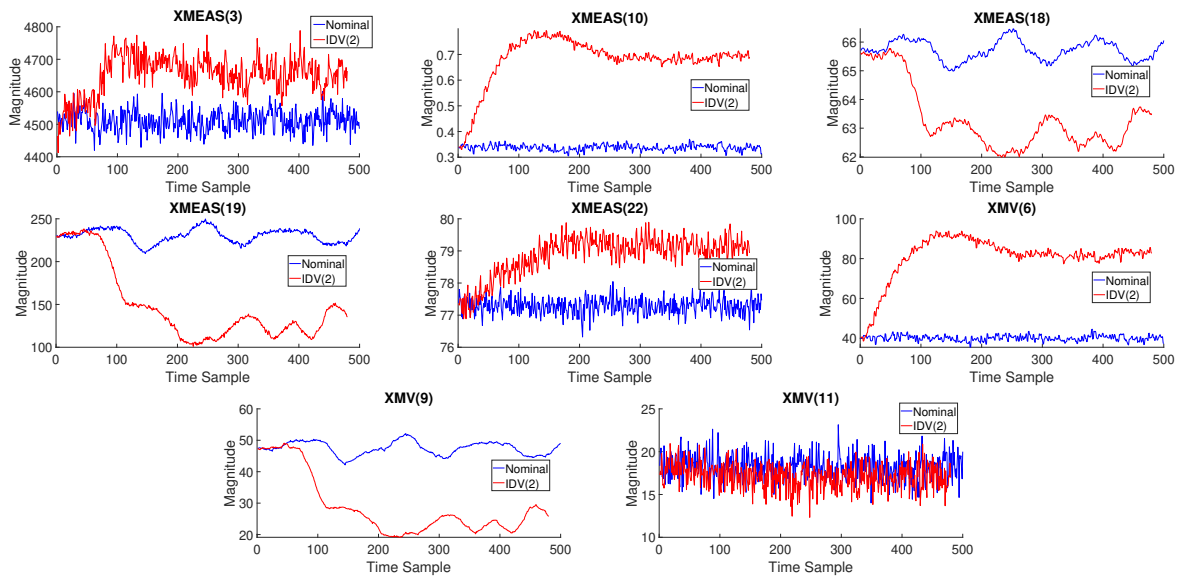


Figure 4.12: Relevant variables contributing to IDV(2) with nominal and abnormal profiles

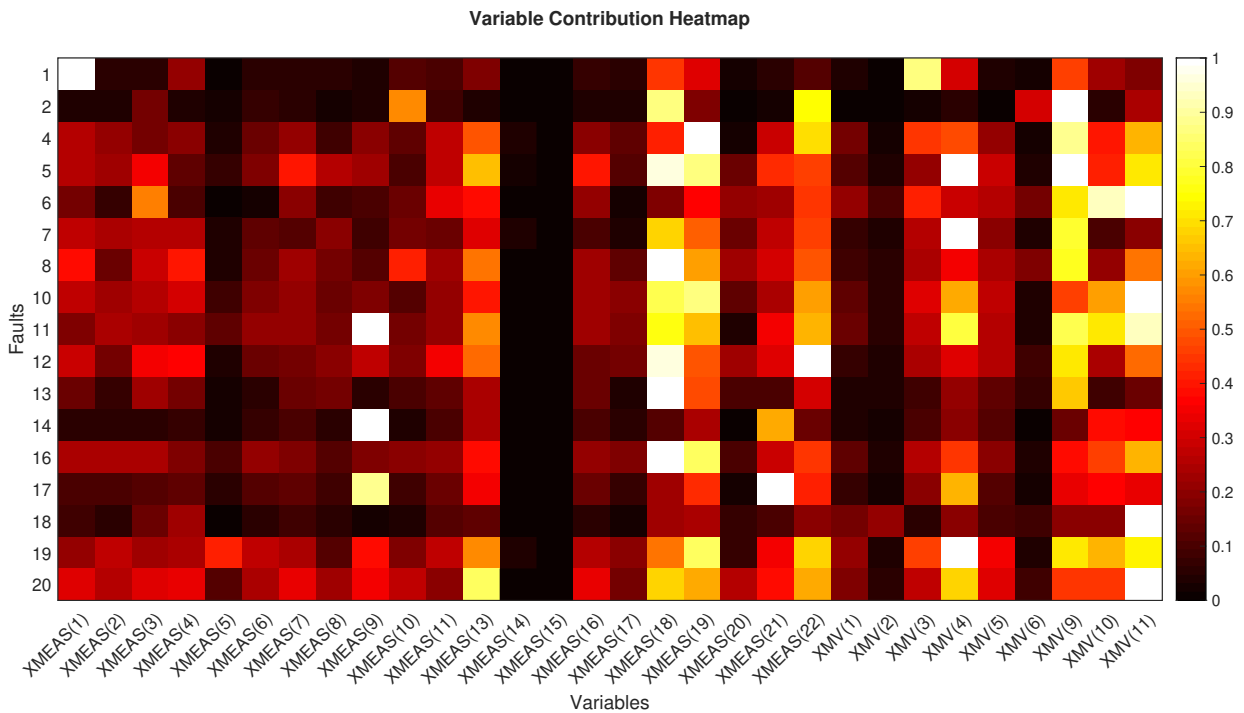


Figure 4.13: Variable Contribution Heatmap corresponding to all faults

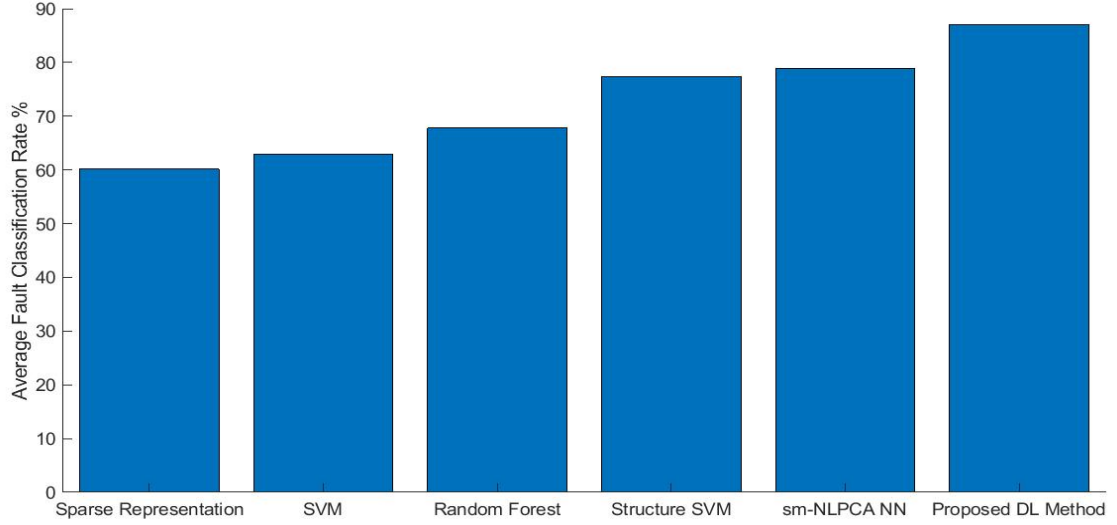


Figure 4.14: Comparison of Fault Classification rate with different methods

Finally, for easier visualization, using these average relative relevances' values for each fault a variable contribution heatmap is generated and shown in Figure 4.13 can be calculated using normalized version of relevances from Equation 4.16. In this heatmap the significance of different variables are color coded to identify the possible root-cause of different variables for each particular fault. For example for fault 1, variables 1 and 3 are shown as significant in the heatmap and it can be verified from Figure 4.10 that these same variables deviate the most during faulty operation from their trajectories during normal operation.

4.5 Conclusion

In this work, an explainability based fault detection and classification methodology is proposed using both a deep supervised autoencoder (DSAE) and dynamic supervised autoencoder (DDSAE) for the extraction of features. A Layerwise Relevance Propagation

Table 4.6: Comparison of Fault Detection Rate with different methods with non-incipient faults only

Fault	PCA (15 comp.)		DPCA (22 comp.)		ICA (9 comp.)		DL (2017)	DL (2017)	DL (2018)	DL (2018)	DL (2019)	Proposed DL
	T^2	SPE	T^2	I^2	AO	SAE-NN	DSN	GAN	OCSVM	CNN	DDSAE	
1	99.2%	99.8%	99%	100%	100%	77.6%	90.8%	99.62%	99.5%	91.39%	99.9%	
2	98%	98.6%	98%	98%	98%	85%	89.6%	98.5%	98.5%	87.96%	98.2%	
4	4.4%	96.2%	26%	61%	84%	56.6%	47.6%	56.25%	50.37%	99.73%	99.7%	
5	22.5%	25.4%	36%	100%	100%	76%	31.6%	32.37%	30.5%	90.35%	100%	
6	98.9%	100%	100%	100%	100%	82.8%	91.6%	100%	100%	91.5%	99.7%	
7	91.5%	100%	100%	99%	100%	80.6%	91%	99.99%	99.62%	91.55%	99.9%	
8	96.6%	97.6%	98%	97%	97%	83%	90.2%	97.87%	97.37%	82.95%	94%	
10	33.4%	34.1%	55%	78%	82%	75.3%	63.2%	50.87%	53.25%	70.05%	89.1%	
11	20.6%	64.4%	48%	52%	70	75.9%	54.2%	58%	54.75%	60.16%	87.7%	
12	97.1%	97.5%	99%	99%	100%	83.3%	87.8%	98.75%	98.63%	85.56%	99.4%	
13	94%	95.5%	94%	94%	95%	83.3%	85.5%	95%	94.87%	46.92%	95.6%	
14	84.2%	100%	100%	100%	100%	77.8%	89%	100%	100%	88.88%	100%	
16	16.6%	24.5%	49%	71%	78%	78.3%	74.8%	34.37%	36.37%	66.84%	94%	
17	74.1%	89.2%	82%	89%	94%	78%	83.3%	91.12%	87.25%	77.11%	97.7%	
18	88.7%	89.9%	90%	90%	90%	83.3%	82.4%	90.37%	90.12%	82.74%	90.9%	
19	0.4%	12.7%	3%	69%	80%	67.7%	52.4%	11.8%	3.75%	70.87%	89.9%	
20	29.9%	45%	53%	87%	91%	77.1%	44.1%	58.37%	52.75%	72.88%	89.6%	
Average	61.77%	74.72%	72.35%	87.29%	91.70%	77.7%	76.84%	74.04%	62.78%	85.47%	96.43%	

(LRP) algorithm is used as the main tool for explaining the classification predictions for the deep neural networks. The explainability measure serves two major objectives: i) Pruning of irrelevant input variables and further improvement in the test classification accuracy and ii) Identification of possible root cause of different faults occurring in the process. The fault detection and classification performance of the proposed DSAE/xDSAE and DDSAE/xDSSAE DNN models together is tested on the TE benchmark process. The proposed methodology outperforms both multivariate linear methods and other DL based methods reported in the literature on the same standard data.

Although this study make use of the powerful feature extraction capability of deep learning neural network models and XAI (eXplainable AI) their use in industrial processes must face practical challenges such as availability of data for training and a longer develop-

ment time for off-line model calibration because of the sequence of pruning and re-training steps.

Chapter 5

Hierarchical Deep LSTM for Fault Detection and Diagnosis for a Chemical Process

Overview¹

A Hierarchical algorithm based on a Deep Neural Network (DNN) is proposed for the detection and classification of faults in industrial plants. The proposed algorithm has the ability to classify incipient faults that are difficult to detect and diagnose with other methods. In the proposed hierarchical structure faults are grouped into subsets according to their similarity thus facilitating detection within each subset. External pseudo-random binary signals (PRBS) are injected into the system to enhance identification of incipient faults. The proposed approach is tested on the Tennessee Eastman Process resulting in significant improvements in classification as compared to both multivariate linear model-based strategies and non-hierarchical nonlinear model-based strategies.

¹Adapted from Agarwal, Piyush, et al. "Hierarchical Deep LSTM for Fault Detection and Diagnosis for a Chemical Process". Submitted to ISA Transactions. (Under Review)

5.1 Introduction

The faults in a chemical plant often propagate along the process, significantly impacting the profit of chemical plants. Hence it is imperative to detect them soon upon their occurrence. The operation of industrial plants employs sensors and control loops to mitigate the economic losses resulting from these faults. However, in the presence of process faults and manipulated variable constraints, these control schemes are not sufficiently resilient to avoid abnormal operation [Chiang et al. \[2000\]](#). Thus, process faults must be diagnosed and addressed by implementing a suitable corrective measure.

A typical process monitoring system consists of two parts: fault detection and diagnosis methodology. The objective of a fault detection system is to make a binary decision whether the current state of the process is in normal or faulty operation region. Once an abnormal operation is detected, the fault diagnosis system is used to infer the type of fault or identify the root cause of the process fault. In the current study, we perform both detection and classification with a single algorithm by considering the normal operation condition as an additional fault class to be identified in the classification step.

Process monitoring schemes rely on estimated process models using historical data to infer faults. Based on the type of model, the methodologies are divided into two main approaches: mechanistic model-based (e.g. using first principles models) and data-driven model-based approaches [Chiang et al. \[2000\]](#). Data-driven models for FDD, such as the one used in the current study, are based on a comparison between different sensor measurements under normal operation versus faulty operation [Yin et al. \[2014a\]](#). Within the class of data-driven approaches, several reported algorithms are based on multivariate statistical methods such as Principal Component Analysis (PCA) [Zhang \[2009\]](#), [Yin et al. \[2012\]](#), [Lau et al. \[2013\]](#), [Shams et al. \[2010\]](#) or its dynamic version such as Dynamic Principal Component Analysis (DPCA) [Chiang et al. \[2000\]](#), [Yin et al. \[2012\]](#), [Ku et al. \[1995\]](#), [Rato and Reis \[2013\]](#), [Odiwei and Cao \[2009\]](#). These methods assume process behaviour is linear. However, most chemical processes are inherently non-linear in nature. Thus, non-linear modeling techniques such as Deep Neural Networks (DNNs) are employed in the current

work. In the last decade, a new generation of Deep Neural Networks (DNNs) algorithms has emerged that capitalizes both the significant increase in computational power and novel algorithmic developments that facilitate the training and calibration of these networks. The use of these algorithms for fault detection in the process industry has recently received increased attention. However, despite the improvements in detection accuracy obtained with these techniques, some faults are still difficult to detect and diagnose (incipient faults). The current study focuses on the detection and diagnosis of such difficult to detect faults while maintaining good detection accuracy for the other faults. The difficult to observe/detect faults will be referred to as incipient faults.

Lack of observability often arises due to the low signal to noise ratio in the measurements used for fault detection and diagnosis (FDD) and feedback control [Isermann \[2005\]](#) [Shams et al. \[2011a\]](#). Specifically, the controller forces the controlled variables to remain close to their set-points at all times. Further, with the addition of noise, the effects of faults are masked. Also, the lack of distinguishability between different process faults is related to the fact that various process faults have a similar effect on the dynamic responses of the measured variables.

FDD algorithms that rely on data collected from the process operation are referred to as passive, while active FDD approaches have also been proposed to improve detection [Mhaskar et al. \[2006\]](#). Active FDD involves injecting persistently exciting input signals into the system and using the resulting input-output data for incipient fault detection and diagnosis. [Heirung and Mesbah \[2019\]](#), [Cusidó et al. \[2011\]](#), [Busch and Peddle \[2014\]](#). The disadvantage of active FDD is that it introduces an external disturbance to the process which may temporarily impact the operation and thus its use should be limited. To the knowledge of the authors, the combination of active and passive FDD approaches into one algorithm for detecting a mix of non-incipient and incipient faults have not been studied.

Following the above, the focus of the current work is on developing deep learning techniques for the detection of faults with an emphasis on the detection of incipient faults. However, faults and their effects on process variables are strongly coupled with each other.

Thus, improving detection of incipient faults should be achieved without degrading the detection of the regular faults. Towards this goal, a novel hierarchical classification strategy based on DNN models is proposed that involves identifying separate models for different subsets of faults with different degree of difficulty to detect. A combination of both passive and active FDD approaches are used. The DNN models used for the passive FDD component are of Recursive Neural Network (RNN) type to exploit the dynamic information in the data. It is also demonstrated that the detection accuracy of most faults can be enhanced by increasing the time horizon of the LSTM based model. While the passive approach is used in the higher level of the hierarchy, the active approach involving the injection of external signals is only used in the last level of the hierarchy for detecting incipient faults that cannot be diagnosed otherwise. It is shown that the passive FDD approach is effective for identifying most faults but the active approach is required for detecting incipient faults.

All studies in this work are conducted with a standard set of simulated data from the Tennessee Eastman Process (TEP) for a fair comparison with several algorithms reported for this system [Spyridon and Boutalis \[2018\]](#), [Lv et al. \[2016\]](#), [Hsu et al. \[2010\]](#), [Singh Chadha et al. \[2019\]](#), [Chadha and Schwung \[2017,?\]](#). Since its introduction, the TEP has served as a benchmark problem for testing control and fault detection algorithms and it is thus ideal for comparing existing approaches to our proposed algorithm. It should be emphasized that due to the difficulty in detecting a set of incipient faults for TEP (faults 3, 9 and 15) many studies on FDD for this system were carried out by ignoring these faults altogether [Lv et al. \[2016\]](#), [Hsu et al. \[2010\]](#). For those studies of FDD for the TEP process that consider all the faults together, the regular faults were detected with different level of success but the detection of incipient faults was very inaccurate ?. Additional reported methods applied to TEP are further reviewed in the Results section. The comparison of our approach to several reported methods shows that our approach provides comparable or superior FDD accuracy for regular faults but clear superiority for incipient faults.

The main contributions of the current study are:

1. Study of the effect of data horizon in the LSTM based deep learning model on the

fault classification ability.

2. Development of a hierarchical structure combining passive FDD with active FDD to enhance the detection and classification accuracy for incipient faults.
3. A comparison of the proposed method to several other methods that shows comparable or superior FDD accuracy for both regular and incipient faults.

This chapter is organized as follows. Fundamentals used in the work are presented in Section 5.2. Explanation on the hierarchical structure of the proposed methodology is presented in Section 5.3. Section 5.4 presents the proposed methodology. Section 5.5 describes the case study. The results and comparisons with previously reported approaches are presented in Section 5.6 followed by conclusions in Section 5.7.

5.2 Preliminaries

5.2.1 Deep LSTM Supervised Autoencoder Neural Network (LSTM-SAE NN)

The training of a Deep Supervised Autoencoder Neural Network (DSAE-NN) model is based on the minimization of a weighted sum of the reconstruction loss function and the supervised classification loss corresponding to the first and second terms in (Equation (5.1)) respectively. Addition of unsupervised loss function i.e. reconstruction loss function improves the generalization of supervised autoencoder model ?. Further, it serves as the regularization term which constraints the problem in terms of latent variables, thus reducing over-fitting. while the minimization of the classification loss function i.e multi-class cross-entropy loss function ensures the non-linear latent variables extracted are the predictors of the output label. The mean squared error function is used as a reconstruction loss and softmax cross-entropy as the classification loss. The overall goal is to learn a function that predicts the class labels in one-hot encoded form $\mathbf{y}_i \in \mathbb{R}^m$ from inputs $\mathbf{x}_i \in \mathbb{R}^{d_x \times 1}$.

For training DSAE-NN, the following loss function is minimized:

$$l_{DSAE} = \frac{\lambda_1}{N} \|\mathbf{x}_s - \hat{\mathbf{x}}_s\|_2^2 + \frac{1}{N} \sum_{s=1}^N \sum_{c=1}^m -y_{s,c} \log(p_{s,c}) \quad (5.1)$$

In this work, we use LSTM units instead of dense layers for both the encoder and decoder as shown in Figure 5.1. The goal is to reconstruct and classify input sequences at time t simultaneously. The encoder transforms the input time sequences using the Equations 2.10, 2.11 and 2.12 to learn important features and encode these features $\mathbf{z} \in \mathbb{R}^{d_h \times 1}$. The decoder function reconstruct the input using the extracted feature vectors. The operation performed by the encoder for a single LSTM layer between the input variables to the latent variables $\mathbf{z}_t^i \in \mathbb{R}^{d_h \times 1}$ can be mathematically described as follows:

$$\mathbf{z}_t^i = \zeta_e(\mathbf{x}_t^i) \quad (5.2)$$

The latent variables \mathbf{z}_t^i are used both to predict the class labels and to reconstruct back the inputs \mathbf{x} as follows:

$$\hat{\mathbf{x}}_t^i = \zeta_d(z_t^i) \quad (5.3)$$

$$\hat{\mathbf{y}}_t^i = f_c(\mathbf{W}_c \mathbf{z}_t^i + \mathbf{b}_c) \quad (5.4)$$

where ζ_e and ζ_d is the LSTM encoder and decoder function respectively. f_c is a non-linear activation function (softmax layer) for the output layer. $\mathbf{W}_c \in \mathbb{R}^{m \times d_z}$ and $\mathbf{b}_c \in \mathbb{R}^m$ are output weight matrix and bias vector respectively.

$$p_{s,c} = \frac{e^{(\hat{y}_{s,c})}}{\sum_{c=1}^m e^{(\hat{y}_{s,c})}} \quad (5.5)$$

where λ_1 is the weight multiplying the reconstruction loss L_r in the cost to be minimized, m is the number of classes, $y_{s,c}$ is a binary indicator (0 or 1) equal to 1 if the class label c is

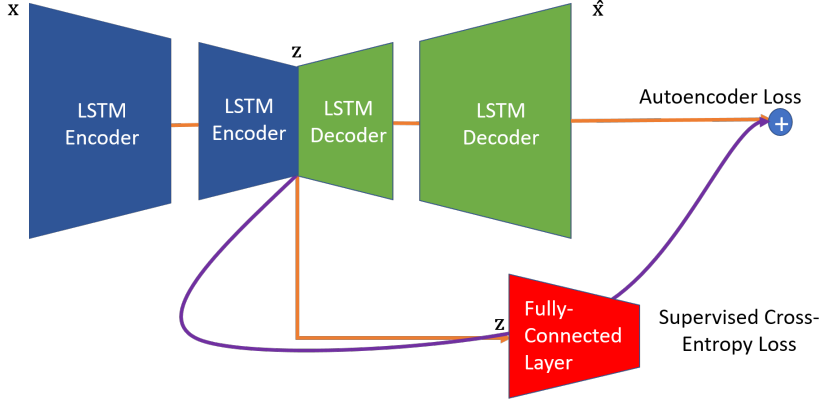


Figure 5.1: Schematic of a Deep LSTM Supervised Autoencoder Neural Network (DLSTM-SAE NN)

the correct one for observation s and 0 otherwise, $y_{s,c}$ is the non-normalized log probabilities and $p_{s,c}$ is the predicted probability for a sample s of class c . Moreover, to avoid over-fitting, a regularization term is added to the objective function in Equation 5.1. Accordingly, the objective function for Deep Supervised LSTM NNs used for FDD is as follows:

$$\min_{\mathbf{W}} l_{DSAE} = \min \frac{1}{N} \left[\lambda_1 \|\mathbf{x}_s - \hat{\mathbf{x}}_s\|_2^2 + \lambda_2 \sum_{s=1}^N \sum_{c=1}^m -y_{s,c} \log(p_{s,c}) + \lambda_3 \sum_L \sum_k \sum_j \mathbf{W}_{kj}^{[L]2} \right] \quad (5.6)$$

where $\mathbf{W}_{kj}^{[L]}$ are the weight matrices for each layer L in the network and the weights on the individual objective functions $\lambda_1, \lambda_2, \lambda_3$ are chosen using validation data.

5.2.2 Model Structure and Specifications

The DLSTM-SAE model used in the current study was developed with training and testing data sets generated from the Tennessee Eastman Process (TEP) simulation. The data are extracted from simulations of the system conducted at either the normal state or when each of the 20 different faults is occurring in the process. It is assumed that at each sampling interval 52 different variables are measured and organized into a vector. Each

such vector of measurements is acquired every 3 minutes. It should be noticed that during testing of the methods proposed in this study the normal state is considered as a different separate class and hence a total of 21 different classes, i.e. 20 faulty plus one normal operations, are considered for classification. The standard dataset can be downloaded from <http://depts.washington.edu/control/LARRY/TE/download.html>. The simulator is ran for 72 hours (training: 24 hours; testing: 48 hours) for each fault generating 1440 samples for each fault class and normal class. The data is then divided between calibration and validation data sets where the first 480 samples are used as training data and the rest are used for testing for each class. This results in a total of 10,080 training samples and 19,200 testing samples. A small fraction of training dataset is used as validation dataset for selecting the optimal hyper-parameters. It is important to note that the number of training, validation and testing samples vary depending on the time horizon used in DLSTM-SAE model. The results reported in the following section are based on the classification accuracy of test dataset, i.e. on data that was not used for model calibration. The experiments in this paper have been implemented on an Intel Core i7-7700HQ PC (2.80GHz, 16GB RAM) and NVIDIA GeForce GTX 1060 (6GB) 64Bit Windows 10 operating system in Python (®) environment. The models are developed using Keras Chollet et al. [2015] (an open deep learning library) on TensorFlow platform. Abadi et al. [2016]. All hyper-parameters such as number of LSTM encoder layers, LSTM units in each layer, weights and learning rate are optimized using Keras-tuner.

5.3 Hierarchical Structure

The key goal of the work is to improve the detection and diagnosis of incipient faults but without sacrificing the detection accuracy for the regular (non-incipient) faults. Thus, we need to increase the sensitivity of the nonlinear FDD algorithm with respect to the incipient faults but without losing sensitivity with respect to the non-incipient faults. The sensitivity of nonlinear models such as deep neural networks is highly dependent on the variability of the data used for calibration. Accordingly, a key data pre-processing step towards model calibration involves data standardization, i.e. mean centering and normal-

ization. It is hypothesized that by building separate models for different groups of faults it is possible to increase the sensitivity of different models and distinguish-ability between faults because of the different re-normalization conducted within each group.

Following the above, a hierarchical structure is proposed as shown in Figure 5.2. This structure includes the following sequential steps for training of the model with a training data set:

1. The training data is mean centered and normalized
2. The faults are classified into two groups: group 1- easily distinguishable faults and group 2- difficult to distinguish faults which include the incipient faults along with normal operation data class.
3. A Deep LSTM-SAE model denoted as M1 is designed for identifying the faults of group 1 or identifying all faults in group 2 as a single fault.
4. The data for group 2 identified in the previous step is mean centered and re-normalized.
5. A neural network model is designed specifically for group 2 denoted as M2.
6. For faults that are not accurately identified by M2, a PRBS is designed and injected into locations in the system that are informative about these faults.

Based on the trained hierarchical structure, online detection and diagnosis for any new sample proceeds as follows:

1. The data corresponding to the sample is mean centered and normalized as in step 1 of the training procedure.
2. The sample is classified as either in group 1 of easy to observe faults or group 2 of difficult to identify faults.
3. If sample is in group 1 is classified accordingly by model M1. If it is in group 2 it is re-normalized according to the re-normalization in step 4 of the training procedure.

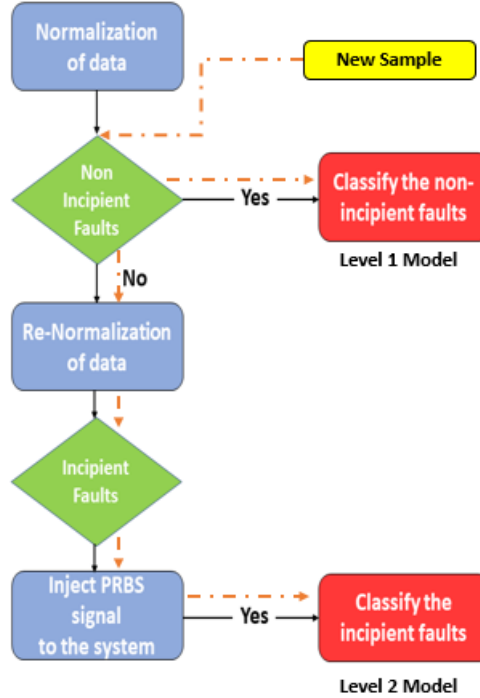


Figure 5.2: Hierarchical structure used for fault detection and diagnosis

4. If the sample is within group 2 is identified by model M2 in step 5 of the training procedure.
5. If the sample is not identified accurately by the model for group 2, PRBS signals are injected as specified in step 6 of the training procedure and the corresponding faults are diagnosed from the resulting data.

It should be noticed that in this algorithm the normal operation is treated as an additional fault-class denoted as Class 1. Then, if the incipient faults are characterized by responses that are very similar to the normal state, a model that is trained to predict all the faults together will be shown to be unable to accurately discern between these responses. Also if the incipient faults are grouped along with the normal state as per step 1 of the training procedure they may also be miss-classified as other faults. Hence, the

overall classification accuracy for the incipient faults must be assessed after the execution of the entire hierarchical procedure.

For model M1 the normalized data is fed to a first level model where the softmax layer of LSTM-SAE NN uses 18 units instead of the 21 units (incipient faults and normal state grouped as one) as used in the non-hierarchical type model. The structure of model M2 is similar to model M1 but the difference is that the softmax layer involves only 4 units each for one of the incipient faults (3,9,15) and for the normal state (fault 0). The PRBS is injected only when the incipient fault cannot be properly identified with either models M1 or M2. Additional details about the PRBS signal design are given in the following section.

5.3.1 Design: Pseudo-random Binary Signal (PRBS)

Although the hierarchical structure proposed in the previous section enhances the diagnosability of few faults, detection of incipient faults is still challenging due to lack of excitation to detect these faults in the presence of noise. This problem is particularly acute in the TEP since the data-set contains variables that are used in closed-loop control thus exhibiting small variation with respect to their set-point values making it difficult to estimate the occurrence of faults from such variables. To increase diagnosability of incipient faults the use of active fault detection, as reviewed in the Introduction, is proposed for the TEP process. The lack of diagnosability/distinguishability of the incipient faults can be viewed as a problem of inaccurate identification of a model relating variability in measured values to faults. To improve the identification accuracy it is required to use inputs that sufficiently excite the system dynamics in the presence of noise [Ljung \[1999\]](#) which will result in larger changes in the measured quantities and larger sensitivity to fault changes. Thus, it is required to introduce additional excitation to the one available in regular operation of the system. Accordingly, external forcing signals are injected at particular points of the control loops, e.g. an excitation signal to the set-points of the loops that involve variables related to the difficult to detect faults. The addition of such excitation signals in combination with a separate deep neural network model (second level) in the hierarchical structure described in the previous section is investigated in the current study for detecting and

diagnosing incipient faults that cannot be accurately identified with the regular operating data collected from the process.

To avoid a large negative impact of the external signals on the profitability of the plant the input signals should meet certain constraints as follows:

1. Reduce input move sizes (to reduce wear and tear on actuators).
2. Reduce input and output amplitudes, power, or variance.
3. Short experimental time to prevent losses

In a practical implementation, the added excitation signal should result in variations in the measured quantities that will be large in magnitude relative to the noise. Towards this goal it is necessary to include information of frequencies lower than the crossover frequency of the closed loop transfer function [Rivera and Gaikwad \[1995\]](#). PRBS signals are used as excitation signals in this study since they have a finite length that can be synthesized repeatedly with simple generators while presenting favorable spectra. The spectrum at low frequencies are flat and constant while at high frequencies the spectra drop off. Thus, the PRBS can be designed to have a specific bandwidth, which can be utilized for exciting the processes within the required range of frequencies [Garcia-Gabin and Lundh](#). The analytical expression for the power spectrum of a PRBS is given by:

$$s(\omega) = \frac{A^2(R+1)t_{cl}}{R} \left[\frac{\sin \omega t_{cl}/2}{\omega t_{cl}} \right]^2 \quad (5.7)$$

where ω is the frequency, t_{cl} is the clock period (minimum time between a change in levels) which is a multiple of the sampling time (T_s) and A is the amplitude of the signal. The sequence repeats itself after $T = R \times t_{cl}$ units of time, where $R = 2n - 1$ and n is the number of shift registers used to generate the sequence. Thus, for designing the PRBS signal it is necessary to estimate the amplitude and the frequency range.

$$\frac{2\pi}{T} \leq \omega \leq \frac{2.8}{t_{cl}} \quad (5.8)$$

Rivera and Gaikwad, 1995, Rivera and Gaikwad [1995] Lee and Rivera, 2005 and Garcia-Gabin and Lundh Garcia-Gabin and Lundh provided practical guidelines for estimating the range of frequency needed for process closed-loop identification using time domain information. The primary frequency band of interest for excitation is determined by the dominant time constants of the system.

$$\omega_{low} = \frac{1}{S_f t^{ol}} \quad (5.9)$$

where $t^{ol} = 4\tau^{ol} + t_d^{ol}$

$$\omega_{high} = \frac{4S_f}{t^{cl}} \quad (5.10)$$

$$\omega_{high} \leq \omega_N \quad (5.11)$$

where S_f is a safety factor used to augment the bandwidth of the excitation signal, t^{ol} is the open loop settling time and t^{cl} is the settling time of closed loop process without considering the time delays. t_d^{ol} is the time delay of the open loop process. Also, the upper value of the frequency must be lower than the Nyquist frequency ω_N to avoid aliasing. Although the magnitude of the signal has not been optimized in the current work, it could be further optimized by taking a profit function of the plant into consideration for minimal losses and using the validation data used for the FDD model.

5.4 Results and discussion

In this section, the industrial benchmark TEP is used to validate and demonstrate the effectiveness of the proposed method. We investigated the multi-class classification performance using a total of 20 fault modes presented in Table 4.2 which involve all of the compositions, manipulated and measurement variables in the TE process (Table 4.1. For an individual class IDV(i), the performance was typically evaluated by a confusion matrix which consists of true positives (TP_i), false positives (FP_i), true negatives (TN_i) and false negatives (FN_i). The notation used in the confusion matrix is as follows:

Output Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
1	6373 36.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	371 2.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	452 2.6%	0 0.0%	88.6% 11.4%
2	0 0.0%	649 3.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	60 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	91.5% 8.5%
3	0 0.0%	0 0.0%	649 3.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	649 3.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	40 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	94.2% 5.8%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	649 3.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	649 3.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	649 3.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	566 3.2%	0 0.0%	0 0.0%	0 0.0%	57 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	90.9% 9.1%
9	2 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	223 1.3%	0 0.0%	0 0.0%	145 0.8%	0 0.0%	69 0.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	50.8% 49.2%
10	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	609 3.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
11	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	23 0.1%	2 0.0%	0 0.0%	0 0.0%	588 3.4%	135 0.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	78.6% 21.4%
12	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	265 1.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
13	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	649 3.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
14	12 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	53 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	580 3.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	89.9% 10.1%
15	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	649 3.7%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
16	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	61 0.3%	47 0.3%	0 0.0%	0 0.0%	0 0.0%	649 3.7%	0 0.0%	0 0.0%	85.7% 14.3%
17	22 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	197 1.1%	0 0.0%	90.0% 10.0%
18	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	649 3.7%	100% 0.0%
	99.4% 0.6%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	87.2% 12.8%	84.4% 15.6%	93.8% 6.2%	90.6% 9.4%	40.8% 59.2%	100% 0.0%	89.4% 10.6%	100% 0.0%	100% 0.0%	30.4% 69.6%	100% 0.0%	91.1% 8.9%
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	

Figure 5.3: Confusion Matrix for the first level model of the hierarchical structure (i.e. classification of non-incipient faults and considering incipient faults as a normal class)

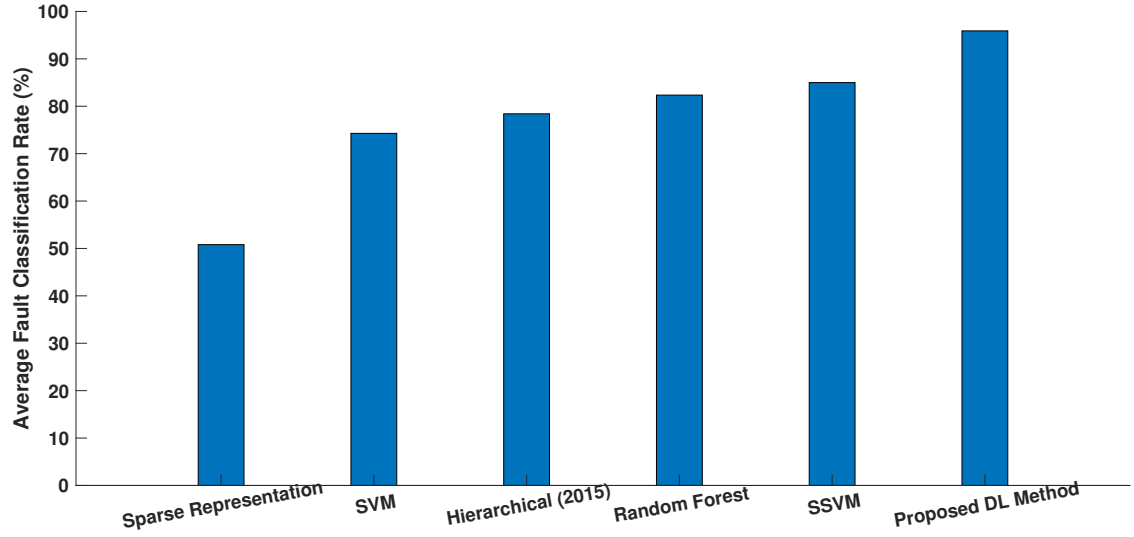


Figure 5.4: Comparison of averaged fault classification rates (non-incipient faults only)

	Counts of predicted label i	Counts of predicted label other than i
Counts of real label i	TP_i	TN_i
Counts of real label other than i	FP_i	FN_i

Table 5.1: Confusion Matrix for each fault (IDV(i))

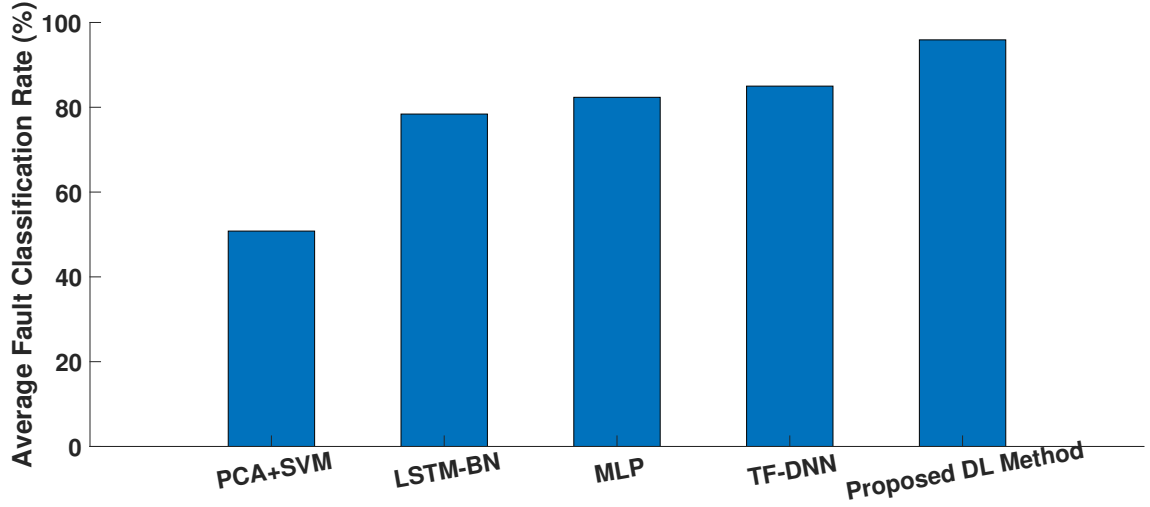


Figure 5.5: Comparison of averaged fault classification rates (all faults)

Two main important metrics for quantifying the performance of the proposed process monitoring methodology are as follows:

- Fault Detection Rate (FDR):

$$\begin{aligned}
 FDR &= \frac{\text{number of fault data that have been detected as fault}}{\text{total number of faulty samples}} \\
 &= \frac{TP_i}{TP_i + FP_i}
 \end{aligned} \tag{5.12}$$

FDR represents the probability that the abnormal conditions are correctly detected which is an important criterion to compare between different methods in terms of their detection efficiency. Evidently, a very high FDR is desirable.

- False Alarm Rate (FAR):

$$\begin{aligned}
FAR &= \frac{\text{number of normal data that have been detected as fault}}{\text{total number of normal samples}} \\
&= \frac{FP_i}{TP_i + TN_i}
\end{aligned} \tag{5.13}$$

where the class corresponding to normal operation is considered as the positive class. FAR represents the probability that the normal operation is wrongly identified as abnormal and thus a very low FAR is desired.

The fault detection results obtained with the hierarchical LSTM SAE NN model are compared with both linear multivariate statistical methods and deep learning methods reported in previous studies. For a fair comparison between the methods, for studies where only non-incipient faults were considered the results were compared to fault detection results obtained from the first level of the hierarchical structure model whereas for studies where all the faults were considered, the comparisons were done for results obtained from second level of the hierarchical structure model. The fault detection rate (FDR) for all the faults is compared for the proposed method, PCA [Lv et al. \[2016\]](#), DPCA [Lv et al. \[2016\]](#), ICA [Hsu et al. \[2010\]](#), Convolutional NN (CNN) [Singh Chadha et al. \[2019\]](#), Deep Stacked Network (DSN) [Chadha and Schwung \[2017\]](#), Stacked Autoencoder (SAE) [Chadha and Schwung \[2017\]](#), Generative Adversarial Network (GAN) [Spyridon and Boutalis \[2018\]](#) and One-Class SVM (OCSVM) [Spyridon and Boutalis \[2018\]](#). The fault detection rates for all non-incipient faults and incipient faults are shown in [Table 5.2](#) and [5.3](#) respectively for different methodologies along with the results from the proposed method. It can be seen from [Table 5.2](#) that the proposed method outperformed the linear multivariate methods and other DL based methods for most fault modes. For example, for PCA with 15 principal components, the average fault detection rates are 61.77% and 74.72% using T^2 and Q statistic respectively. Since the principal components extracted using PCA captures static correlations between variables, DPCA is used to account for temporal correlations (both auto-correlations and cross-correlations) in the data. The effect of increasing the number of time samples in the Tennessee Eastman simulation is also investigated following the hypothesis that increasing the time horizon will enhance classification accuracy. In the case of DPCA, the number of lags used in the observation matrix is a key parameter. Since DPCA

is only a data compression technique it must be combined with a classification model for the purpose of fault detection. Accordingly, the output features from the DPCA model are fed into an SVM model that is used for final classification. Different time horizons were tried for training the DPCA model. Based on validation results the best DPCA model was obtained with 22 lags. The average detection rate obtained was 72.35%. ICA [Hsu et al. \[2010\]](#) based monitoring scheme perform better than both PCA and DPCA based methods with an averaged accuracy of approximately 90%. It should be noted that all these methods (PCA, DPCA and ICA) perform poorly for detecting incipient faults. In addition to the comparison to linear methods the proposed methodology was also compared with different DNN architectures such as CNN [Chadha and Schwung \[2017\]](#), DSN [Chadha and Schwung \[2017\]](#), SAE-NN (results reported in [Chadha and Schwung,2017](#)) and GAN [Spyridon and Boutalis \[2018\]](#), OCSVM (results reported in [Spyridon and Boutalis,2018](#)) reported previously. It can be seen that the proposed method also outperforms these DNN based methodologies. The relative advantage of our method versus these other DNN architectures (Table 4) is mostly due to the inclusion of the incipient faults within the normal class. This reduces the confusion between the normal samples with other non-incipient faults. However, the additional advantage of the proposed method over the other DNN architectures is realized when the hierarchical structure is used in combination with the PRBS signals as further discussed below. It should be noted that all these comparisons were based on an identical data set. Similarly, fault detection rate for all faults are compared with different DL based models in Table 5.3 including SAE-NN, DSN, GAN, OCSVM, CNN, Optimized LSTM [Zhao et al. \[2018a\]](#) and LSTM along with attention mechanism ?. It can be seen that the proposed methodology improves the averaged test classification accuracy for all faults significantly.

To improve diagnosis of the non-incipient faults the proposed hierarchical structure was applied where the first level model of the hierarchical structure classifies non-incipient faults and the second level model classifies incipient faults. For the first level model, there are 7382 training samples and 17,442 testing samples in total with a time horizon of 150 time-steps. The model consists of 182 encoder LSTM units, followed by 116 LSTM units for processing of the output of the encoding layer. Thereafter, the output of the second

LSTM layer is passed through a dense layer for classification. Hyper-parameters such as number of layers, number of LSTM units in each layer, classification weights, learning rate, time-horizon etc. are selected using validation data that are a subset of the training dataset. The hyper-parameter search is implemented using the keras-tuner. To this purpose, a grid of hyper-parameters is defined, for example number of encoder layers = [1,2,3], number of LSTM units for each of these layers ranging from 2 to 200 with an interval of 2 = [10:2:200], learning rate = [$1e^{-1}$, $2e^{-1}$, $3e^{-1}$, $1e^{-2}$], value of weights in the objective function, etc. 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. The best run with highest validation accuracy and the combination of hyper-parameters for the run are used to evaluate test accuracy. A study was also conducted to select the optimal time horizon for the LSTM based model with the hierarchical structure. It can be seen from Figure 5.7 that the classification averages can be enhanced by extending the length of the time horizon of past data fed to the LSTM based model. 150 time steps were chosen as the optimal time-horizon. The Confusion matrix for the level 1 model is presented in Figure 5.3.

The next important design parameter for the second level hierarchical model is the location in the process at which the external excitation signal should be introduced to maximize information about the occurring incipient fault. In this work, this choice is based on the flow-sheet and by identifying which variables are mostly correlated to the incipient faults under consideration. Specifically, the excitation signals were added to process set-points in control loops that are most correlated to the incipient faults. When the selection of the variable to be excited by a PRBS is not obvious from the process flow-sheet, a more systematic approach is to use sensitivity analysis, e.g. sensitivity of changes in the variable connected to the fault to all process variables. Since it may be detrimental to perturb the set-point continuously by the PRBS signal the latter can be introduced intermittently into the process. In the current work an excitation signal of length 40 time-steps was intermittently introduced every 4 hours into the process by assuming that such event will not impact significantly the profitability of the process (for test data). Changes in the

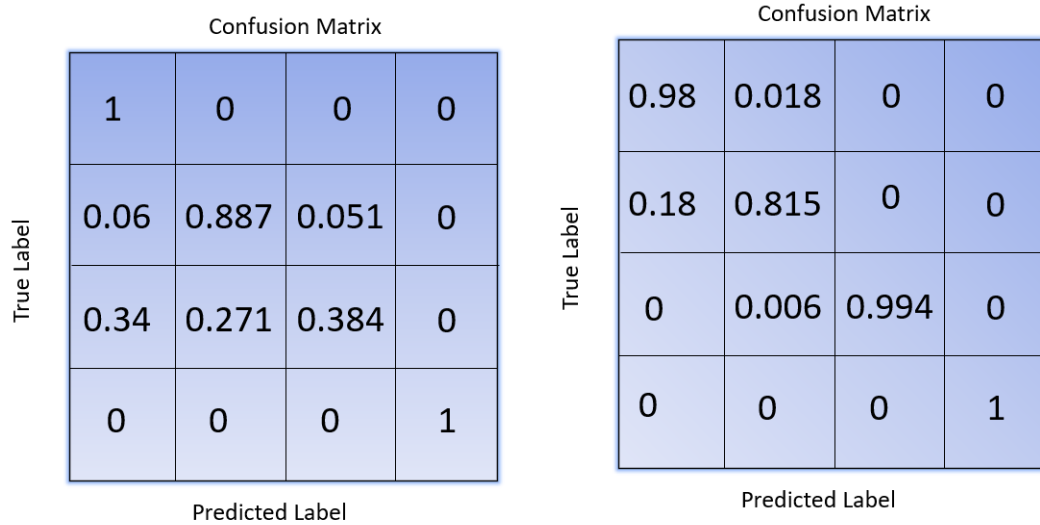


Figure 5.6: Confusion Matrix on test data for the second level model of the hierarchical structure: a) After adding designed PRBS signal w.r.t. fault 15 b) After adding designed PRBS signal w.r.t. fault 9 and fault 15

separator temperature set-point will force changes in the condenser temperature. Since the fault to be identified is stiction in the valve that affects the condenser temperature, the imposed PRBS in the separator set-point indirectly helps in identifying fault 15. For fault 9 i.e. random variation in D feed temperature (refer Table 4.2) the PRBS excitation ($\omega \in [\omega_{cl}, \omega_n]$ where $\omega_{cl} = 0.0087$ rad/s and $\omega_n = 1.74$ rad/s) signal is introduced to the D feed ratio in order to create a suitable excitation. After developing this PRBS signal, we added both signals to the process at different times during the simulation. For fault 15, the PRBS signal is designed with a frequency range of $\omega \in [\omega_{cl}, \omega_n]$ where $\omega_{cl} = 0.005$ rad/s and $\omega_n = 1.74$ rad/s

A systematic ablation study is conducted in Table 5.4 in order to demonstrate the gradual improvements in the results by showing fault detection rates of incipient faults, normal operation and non-incipient faults for 4 cases: i-without the hierarchical structure with one DL model, ii- with hierarchical structure and iii- with hierarchical structure and with addition of one PRBS signal related to fault 15 and iv- with hierarchical structure and with addition of two PRBS signals related to fault 15 and fault 9. Other than a slight

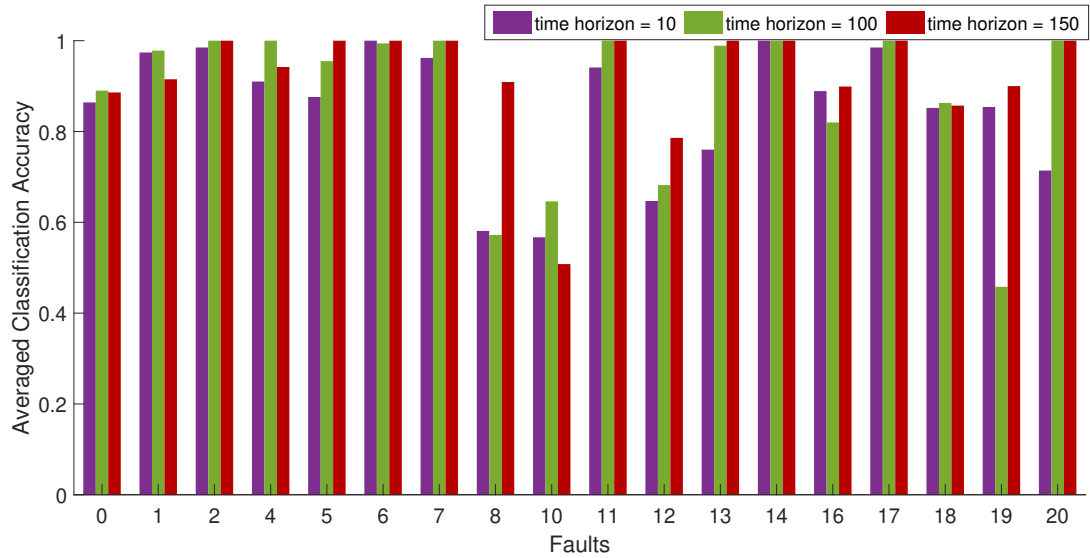


Figure 5.7: Selection of optimal time horizon for Hierarchical LSTM-SAE Level 1 model

decrease in the detection of the Normal operation with the hierarchical structure and the addition of the two PRBS signals, the improvements in all other faults and in the average test accuracy are evident.

For the second level model, there are 1,796 training samples and 4,196 testing samples in total with a time horizon of 150 time-steps. The model consists of 284 encoder LSTM units in the first hidden layer, second layer consists of 100 LSTM units, followed by 278 LSTM units for processing of the output of the encoding layer. Thereafter, the output of the third LSTM layer is passed through a dense layer for classification. Hyper-parameters such as number of layers, number of LSTM units in each layer, classification weights, learning rate, time-horizon, weights in the loss function etc. are selected using the validation data which is part of the training dataset. The hyper-parameter search is implemented again using the keras-tuner. For the second level model, the samples corresponding to fault 0 (normal) and incipient faults are considered. Figure 5.6 (a) shows the confusion matrix after introducing the PRBS signal that was designed for identifying fault 15 and Figure 5.6 (b) shows the confusion matrix after introducing both PRBS signals that were designed

for identifying fault 15 and fault 9. The total FAR calculated using Equation 17 was 2.41%.

The averaged fault classification rate for all non-incipient faults and for all faults (including incipient faults) are shown in Figure 5.4 and 5.5 respectively. Figure 5.4 shows a bar-chart comparison of the proposed method with several non-linear methods such as Sparse representation Wu et al. [2012], SVM Yin et al. [2014b], Hierarchical model based method Xie and Bai [2015], Random Forest, Structural SVM. It can be seen that the Hierarchical Deep RNN based method outperforms other methods with a significant margin. It should be noted that comparisons made in Figure 5.4 do not consider incipient faults. In Figure 5.5, the averaged test accuracy of all faults (both incipient and non-incipient faults) are compared with other DL based methods Luo et al. [2020]. It can be seen that the second level hierarchical model combined with the introduction of the designed PRBS signals significantly improves the classification of the incipient faults and thus the averaged test accuracy for fault diagnosis increases significantly.

5.5 Conclusions

This work studied the application of a deep learning model within a hierarchical structure as a way to increase the detection and classification of faults in the Tennessee Eastman Process (TEP). The TEP simulation contains 20 different faults that were used during this study to make the classification problem. As previously reported by other researchers, a subset of these faults - referred to in this study as incipient - are particularly difficult to diagnose due to low signal to noise ratio and similarities in the resulting dynamic responses corresponding to different faults.

A comparison between deep learning techniques to a multivariate linear technique for fault detection such as PCA, DPCA, ICA and other deep learning methods is also presented. It is observed that Hierarchical LSTM based model is superior to traditional linear and other deep learning based methods for fault classification due to their ability to capture nonlinear dynamic behaviour. It was also shown that the classification aver-

Table 5.2: Comparison of Fault Detection Rate with different methods with non-incipient faults only

Fault	PCA (15 comp.)		DPCA (22 comp.)		ICA (9 comp.)		DL (2017)	DL (2017)	DL (2018)	DL (2018)	DL (2019)	Prop- osed DL
	T^2	SPE	T^2	I^2	AO	SAE-NN	DSN	GAN	OCSVM	CNN	HDRNN (LSTM-SAE)	
1	99.2%	99.8%	99%	100%	100%	77.6%	90.8%	99.62%	99.5%	91.39%	100%	
2	98%	98.6%	98%	98%	98%	85%	89.6%	98.5%	98.5%	87.96%	100%	
4	4.4%	96.2%	26%	61%	84%	56.6%	47.6%	56.25%	50.37%	99.73%	100%	
5	22.5%	25.4%	36%	100%	100%	76%	31.6%	32.37%	30.5%	90.35%	100%	
6	98.9%	100%	100%	100%	100%	82.8%	91.6%	100%	100%	91.5%	100%	
7	91.5%	100%	100%	99%	100%	80.6%	91%	99.99%	99.62%	91.55%	100%	
8	96.6%	97.6%	98%	97%	97%	83%	90.2%	97.87%	97.37%	82.95%	100%	
10	33.4%	34.1%	55%	78%	82%	75.3%	63.2%	50.87%	53.25%	70.05%	42.8%	
11	20.6%	64.4%	48%	52%	70	75.9%	54.2%	58%	54.75%	60.16%	100%	
12	97.1%	97.5%	99%	99%	100%	83.3%	87.8%	98.75%	98.63%	85.56%	100%	
13	94%	95.5%	94%	94%	95%	83.3%	85.5%	95%	94.87%	46.92%	100%	
14	84.2%	100%	100%	100%	100%	77.8%	89%	100%	100%	88.88%	100%	
16	16.6%	24.5%	49%	71%	78%	78.3%	74.8%	34.37%	36.37%	66.84%	100%	
17	74.1%	89.2%	82%	89%	94%	78%	83.3%	91.12%	87.25%	77.11%	100%	
18	88.7%	89.9%	90%	90%	90%	83.3%	82.4%	90.37%	90.12%	82.74%	100%	
19	0.4%	12.7%	3%	69%	80%	67.7%	52.4%	11.8%	3.75%	70.87%	40.4%	
20	29.9%	45%	53%	87%	91%	77.1%	44.1%	58.37%	52.75%	72.88%	100%	
Average	61.77%	74.72%	72.35%	87.29%	91.70%	77.7%	76.84%	74.04%	62.78%	85.47%	93.13%	

Table 5.3: Comparison of Fault Detection rate with different methods (with all faults)

Fault	DL	DL	DL	DL	DL	DL	DL	Prop-
	(2017)	(2017)	(2018)	(2018)	(2019)	(2018)	(2021)	osed DL
	SAE-NN	DSN	GAN	OCSVM	CNN	Optimized LSTM	LSTM (attention)	LSTM-SAE
1	77.6%	90.8%	99.62%	99.5%	91.39%	68%	100%	100%
2	85%	89.6%	98.5%	98.5%	87.96%	78%	89%	100%
3	79.4%	14.4%	10.375%	7.62%	50.59%	45%	94%	81.58%
4	56.6%	47.6%	56.25%	50.37%	99.73%	75%	99%	100%
5	76%	31.6%	32.37%	30.5%	90.35%	45%	94%	100%
6	82.8%	91.6%	100%	100%	91.5%	75%	100%	100%
7	80.6%	91%	99.99%	99.62%	91.55%	89%	100%	100%
8	83%	90.2%	97.87%	97.37%	82.95%	100%	99%	100%
9	50.6%	16.3%	8.625%	7.125%	49.53%	89%	81%	99.38%
10	75.3%	63.2%	50.87%	53.25%	70.05%	71%	99%	42.84%
11	75.9%	54.2%	58%	54.75%	60.16%	67%	88%	100%
12	83.3%	87.8%	98.75%	98.63%	85.56%	77%	99%	100%
13	83.3%	85.5%	95%	94.87%	46.92%	83%	89%	100%
14	77.8%	89%	100%	100%	88.88%	56%	99%	100%
15	55.5%	26.7%	12.5%	14%	43.54%	89%	22%	100%
16	78.3%	74.8%	34.37%	36.37%	66.84%	99%	31%	100%
17	78%	83.3%	91.12%	87.25%	77.11%	0%	97%	100%
18	83.3%	82.4%	90.37%	90.12%	82.74%	89%	95%	100%
19	67.7%	52.4%	11.8%	3.75%	70.87%	20%	97%	40.4%
20	77.1%	44.1%	58.37%	52.75%	72.88%	88%	85%	100%
Average	75.355%	65.32%	64.51%	62.78%	79.84%	70.15%	87.85%	93.23%

Table 5.4: Ablation study for the proposed method

Faults	Non-Hierarchical DL NN	Hierarchical DL NN (no PRBS)	Hierarchical DL NN+ PRBS addition for fault 15	Hierarchical + PRBS addition for fault 15 and fault 9
Fault 3	36%	42%	88.7%	81.5%
Fault 9	32%	18%	38.4%	99.3%
Fault 15	12%	30%	99.4%	100%
Normal Operation	18%	25%	100%	98.1%
Average of all other Faults	85%	87%	93.1%	93.1%
Averaged Test Accuracy	73.4%	75.90%	90.9%	93.4%

ages can be enhanced by extending the length of the time horizon of past data fed to the RNN based model. However, most of these improvements in classification occurred for the non-incipient faults. Therefore, an active fault detection approach was pursued where a hierarchical model structure combined with external PRBS signals was proposed that proved to be particularly effective for classifying incipient faults. Future studies will address the trade-off between the impact of the injected PRBS signals on quality and productivity versus the benefit from early detection of incipient faults.

Chapter 6

A Novel Unsupervised Approach for Batch Process Monitoring using Deep Learning

Overview¹

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 work 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 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.

¹Adapted from Agarwal, Piyush, et al. "A Novel Unsupervised Approach for Batch Process Monitoring using Deep Learning". Submitted to Computers and Chemical Engineering. (Under Review)

6.1 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. \[2019, 2021\]](#) 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 (MPCA) [Wise et al. \[1990\]](#), independent component analysis (ICA) [Kano et al. \[2003\]](#), or supervised, e.g. partial least squares (MPLS) [MacGregor et al. \[1994\]](#), qualitative trend analysis (QTA) [Maurya et al. \[2005\]](#), and Fisher discriminant analysis [Chiang et al. \[2000\]](#). While MPLS 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 MPLS for FD in batch processes use the fermentation time 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 MPCA or MPLS 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 from input space that are most informative about the output by compressing a high dimensional input space into a lower dimensional latent variable space that can be used for FD. However, these algorithms are based on linear decomposition techniques and fail to accurately extract non-linear information when applied to nonlinear processes. To mitigate these drawbacks, a deep learning 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. \[2019b\]](#), [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 features 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. \[2020\]](#). 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 Analysis (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 following chapter 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 6.2. Section 6.3 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 6.4 followed by conclusions in Section 6.5.

6.2 Preliminaries

This section briefly reviews the fundamentals of MPCA and MPLS 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.

6.2.1 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:

$$\mathbf{X} = \sum_{k=1}^K \mathbf{t}_k \mathbf{p}_k^T + \mathbf{E} = \mathbf{TP}^T + \mathbf{E} \quad (6.1)$$

$$s.t. \quad \mathbf{P}^T \mathbf{P} = \mathbf{I} \quad (6.2)$$

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 :

$$T_{new}^2 = \mathbf{t}_{new} \frac{(\mathbf{TT}^T)^{-1}}{n-1} \mathbf{t}_{new}^T \quad (6.3)$$

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

$$\mathbf{t}_{new} = \mathbf{x}_{new} \mathbf{P} (\mathbf{P}^T \mathbf{P})^{-1} \quad (6.4)$$

An abnormal observation, i.e. an observation denoting a faulty condition, can be detected if T_{new}^2 exceeds the control limit T_α^2 [Lee et al. \[2004b\]](#) which can be calculated as follows:

$$T_\alpha^2 = \frac{K(n-1)}{n-K} F_{K, n-K, \alpha} \quad (6.5)$$

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:

$$\begin{aligned} Q_{new} &= \mathbf{e}_{new} \mathbf{e}_{new}^T \\ \mathbf{e}_{new} &= \mathbf{x}_{new} - \mathbf{t}_{new} \mathbf{P}^T \end{aligned} \quad (6.6)$$

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

$$\begin{aligned}
Q_\alpha &= \theta_1 \left[\frac{z_\alpha (2\theta_2 h_0^2)^{0.5}}{\theta_1} + 1 + \frac{\theta_2 h_0 (h_0 - 1)}{\theta_1^2} \right]^{\frac{1}{h_0}} \\
\mathbf{V} &= \frac{\mathbf{E}\mathbf{E}^T}{n-1} \\
\theta_i &= \text{trace}(\mathbf{V}^i); i = 1, 2, 3 \\
h_0 &= 1 - \frac{2\theta_1\theta_3}{3\theta_2^2}
\end{aligned} \tag{6.7}$$

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_α^2 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_α^2 and Q_α are evaluated using Eq 6.5 and 6.7. ii- Each newly acquired sample is mean-centered and normalized based on the training set mean and variance. T_{new}^2 and Q_{new} of the new sample are evaluated using Eq 6.3 and 6.6 and abnormality is determined if any one of these metrics exceeds its control limits.

6.2.2 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:

$$\begin{aligned}
 \mathbf{X} &= \sum_{k=1}^K \mathbf{t}_k \mathbf{p}_k^T + \mathbf{E} = \mathbf{T} \mathbf{P}^T + \mathbf{E} \\
 \mathbf{Y} &= \sum_{k=1}^K \mathbf{u}_k \mathbf{q}_k^T + \mathbf{F} = \mathbf{U} \mathbf{Q}^T + \mathbf{F} \\
 \text{s.t. } \max & \quad \mathbb{E} \left\{ \left(\mathbf{T} - \bar{\mathbf{T}} \right) \left(\mathbf{U} - \bar{\mathbf{U}} \right) \right\} \\
 \implies \max & \quad \text{corr}(\mathbf{T}, \mathbf{U}) \times \sqrt{\mathbf{T}^T \mathbf{T}} \times \sqrt{\mathbf{U}^T \mathbf{U}} \tag{6.8}
 \end{aligned}$$

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. 6.3-6.7.

6.3 Proposed Methodology

6.3.1 Multiway Partial Least Squares Autoencoder (MPLS-AE)

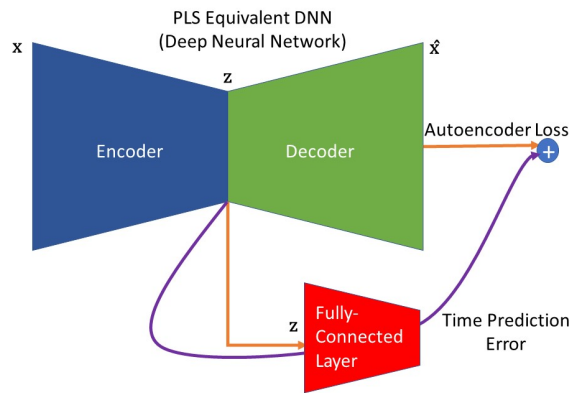


Figure 6.1: Schematic of a Multiway Partial Least Squares Autoencoder (MPLS-AE)

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 Equation 6.8) 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 Equation 6.8. 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 Equation 6.9. The weight λ in equation below is equivalent to maximizing correlation between inputs and outputs. 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 6.1 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:

$$J^{\text{MPLS-AE}} = \frac{1}{n} \left(\sum_{s=1}^n (t_s - \hat{t}_s)^2 + \lambda \sum_{s=1}^n (\mathbf{x}_s - \hat{\mathbf{x}}_s)^2 \right) \quad (6.9)$$

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:

$$H_{k,b}^2 = \sum_{d_z=1}^R \frac{(z_{k,b,d_z}^{deepest} - z_{k,d_z}^{mean})^2}{z_{k,d_z}^{variance}} \quad (6.10)$$

where $H_{k,b}^2$ is H^2 of the k^{th} sample in the b^{th} batch, $z_{k,b,d_z}^{deepest}$ is the d_z^{th} dimension of the latent variables of the k^{th} sample in b^{th} batch, z_{k,d_z}^{mean} and $h_{k,d_z}^{variance}$ 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:

$$SPE_{k,b} = \sum_{d_z=1}^R (\mathbf{x}_{k,d_z}^b - \hat{\mathbf{x}}_{k,d_z}^b)^2 \quad (6.11)$$

where $SPE_{k,b}$ is the SPE of the k^{th} sample in the b^{th} batch, \mathbf{x}_{k,d_z}^b and $\hat{\mathbf{x}}_{k,d_z}^b$ 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 \[1962\]](#) procedure is used to obtain control limits H_α^2 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_α^2 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 $H_{k,\alpha}^2$ and $SPE_{k,\alpha}$. To calculate these limits, we use Gaussian kernels to estimate the probability distribution function of univariate data (\mathbf{H}_k^2 and \mathbf{SPE}_k) for each time interval k as follows:

$$F(H_k^2) = \sum_{i=1}^b G\left(\frac{H_k^2 - H_i^2}{b_w}\right) \quad (6.12)$$

$$G(x_i; \sigma) \propto \exp\left(-\frac{x_i^2}{2\sigma^2}\right) \quad (6.13)$$

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 \[1986\]](#). The $H_{k,\alpha}^2$ for a pre-specified significance level α is then calculated using the equation below:

$$1 - \alpha = \int_{\min \mathbf{H}_{k,train}^2}^{H_{k,\alpha}^2} F(H_k^2) dH_k^2 \quad (6.14)$$

Similar procedure (Eqs. [6.12-6.14](#)) can be followed to evaluate the residual control limits $SPE_{k,\alpha}$.

6.3.2 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 Equation 6.15) in MPCA or the MSE of output prediction (refer Equation 6.16) in MPLS.

$$J^{MPCA} = \frac{1}{n} \sum^b \sum^k (x_k - \hat{x}_k)^2 \quad (6.15)$$

$$J^{MPLS} = \frac{1}{n} \sum^b \sum^k (\mathbf{t}_k - \hat{\mathbf{t}}_k)^2 \quad (6.16)$$

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 as follows:

It should be emphasized that since all the samples used for training of an unsupervised learning model correspond to normal operation, $T_{k,train}^2$ of each sample at time ‘ k ’ should be less than the associated control limit i.e. $T_{k,\alpha}^2$ (Dynamic control limits). An objective function that is most relevant for fault detection should measure the number of missed 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:

$$J^{MPCA-MPLS,FDR} = \frac{1}{n} \sum^b \sum^k ([T_{k,train}^2 - T_{k,\alpha}^2(t)] > 0) \parallel ([Q_{k,train} - Q_{k,\alpha}] > 0) \quad (6.17)$$

where $T_{k,train}^2$ and $T_{k,\alpha}^2$ are a function of scores and nPCs K . These hyper-parameters are selected using Eq 6.17 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:

$$J^{MPLS-AE,FDR} = \frac{1}{n} \sum^b \sum^k ([H_{k,train}^2 - H_{k,\alpha}^2] > 0) \parallel ([SPE_{k,train} - SPE_{k,\alpha}] > 0) \quad (6.18)$$

The above loss function (Equation 6.18) evaluates the number of miss-detected samples with the MPLS-AE model. Hence the loss function is directly related to the (\overline{FDR}) which is a function of $H_{k,train}^2$ and $Q_{k,train}$ and consequently these are a function of latent variables z (refer Eqs. 6.10 and 6.11). The proposed methodology for fault detection is schematically described in Figure 6.2.

6.3.3 Average Fault Detection Rate (\overline{FDR})

Thereafter the average fault detection rate (\overline{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:

$$M_1 : \begin{cases} \overline{FDR}_{normal} &= \frac{1}{n_{normal}} \sum_{k=1}^{n_{normal}} \sum^b ([T_{k,b}^2 - T_{k,\alpha}^2] \leq 0) \&\& ([Q_{k,b} - Q_{k,\alpha}] \leq 0) \\ \overline{FDR}_{abnormal} &= \frac{1}{n_{abnormal}} \sum_{k=1}^{n_{abnormal}} \sum^b ([T_{k,b}^2 - T_{k,\alpha}^2] > 0) \parallel ([Q_{k,b} - Q_{k,\alpha}] > 0) \end{cases} \quad (6.19)$$

$$M_2 : \begin{cases} \overline{FDR}_{normal} &= \frac{1}{n_{normal}} \sum_{k=1}^{n_{normal}} \sum^b ([H_{k,b}^2 - H_{k,\alpha}^2] \leq 0) \&\& ([SPE_{k,b} - SPE_{k,\alpha}] \leq 0) \\ \overline{FDR}_{abnormal} &= \frac{1}{n_{abnormal}} \sum_{k=1}^{n_{abnormal}} \sum^b ([H_{k,b}^2 - H_{k,\alpha}^2] > 0) \parallel ([SPE_{k,b} - SPE_{k,\alpha}] > 0) \end{cases} \quad (6.20)$$

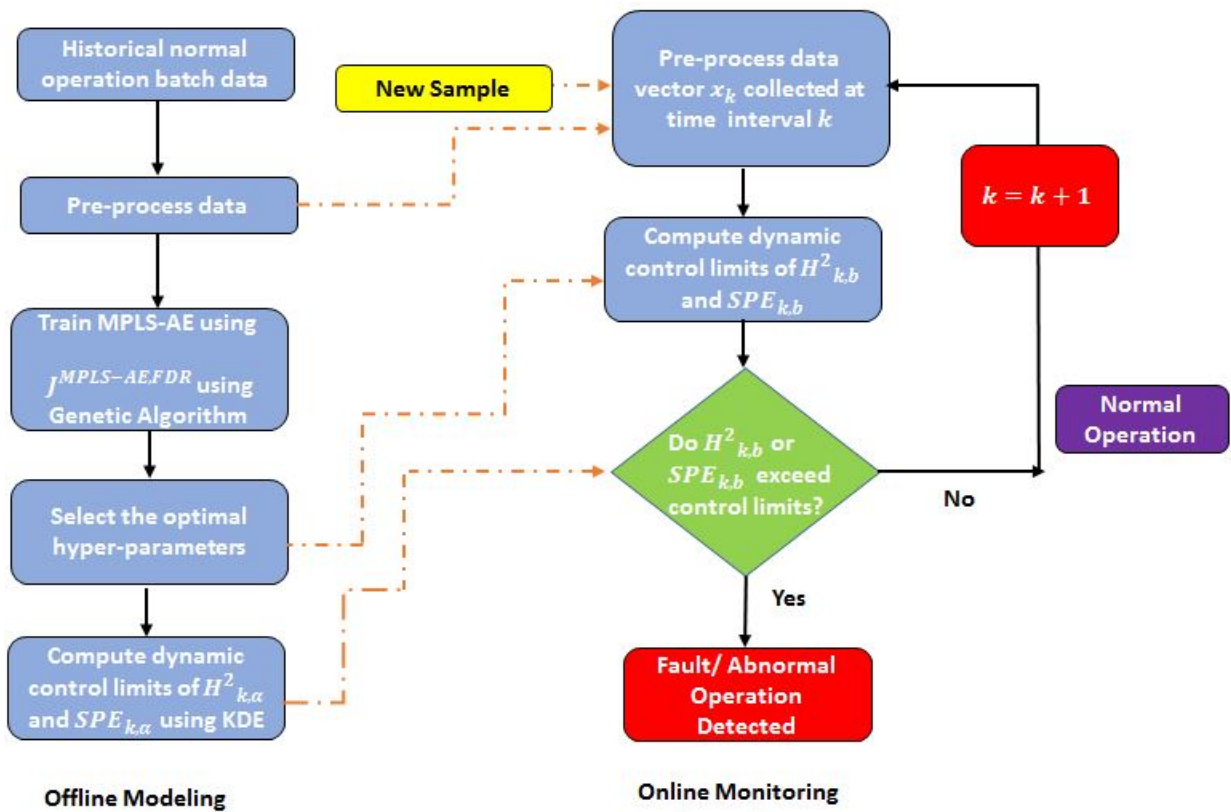


Figure 6.2: Flowchart for fault detection based on novel objective function

$$\overline{FDR} = \frac{n_{normal}\overline{FDR}_{normal} + n_{abnormal}\overline{FDR}_{abnormal}}{n_{normal} + n_{abnormal}} \quad (6.21)$$

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 $H_{k,\alpha}^2$ and $SPE_{k,\alpha}$ are the control bounds $H_{k,\alpha}^2$ and $SPE_{k,\alpha}$ of the k^{th} sample, respectively. A normal operation sample is detected correctly if both $T_{k,b}^2$ or $H_{k,b}^2$ 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 $H_{k,b}^2$ or $SPE_{k,b}$ or both exceed their control limits in case of MPLS-AE and $T_{k,b}^2$ 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 Equation 6.21).

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.

6.3.4 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 6.3. 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):

$$\begin{aligned}
\frac{dA_0}{dt} &= r_b - r_{diff} - \frac{F_{in}A_0}{V} \\
\frac{dA_1}{dt} &= r_e - r_b + r_{diff} - r_{deg} - \frac{F_{in}A_1}{V} \\
\frac{dA_2}{dt} &= r_{deg} - r_a - \frac{F_{in}A_2}{V} \\
\frac{dA_3}{dt} &= r_a - \frac{F_{in}A_3}{V}
\end{aligned} \tag{6.22}$$

Total biomass ($X_{biomass}$):

$$X_{biomass} = A_0 + A_1 + A_2 + A_3 \tag{6.23}$$

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):

$$\frac{dP}{dt} = r_P - r_h - \frac{F_{in}P}{V} \tag{6.24}$$

Substrate consumption (S):

$$\frac{dS}{dt} = -Y_{S/X}r_e - Y_{S/X}r_b - m_s r_m - Y_{S/P}r_P + \frac{F_S C_S}{V} + \frac{F_{oil} C_{oil}}{V} - \frac{F_{in}S}{V} \tag{6.25}$$

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 6.1. The dynamic profiles of these faults (except for fault number 6 which is a combination of all the other faults) are shown in Figure 6.4. 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 ($\mathbf{X}_1 \in \mathbb{R}^{1150 \times 25 \times 80}$) is unfolded to a matrix as described in Section 2.1 and 2.2 with dimensions $\mathbf{X} \in \mathbb{R}^{92000 \times 25}$. 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.

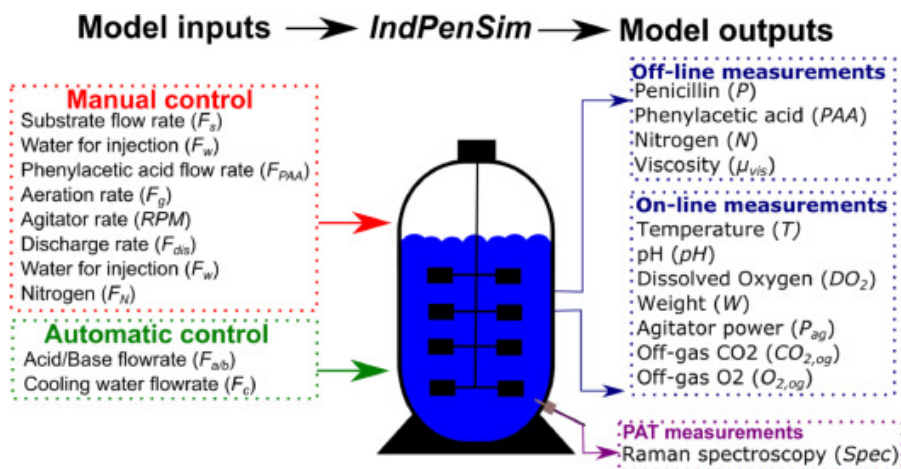


Figure 6.3: Summary of all model inputs and outputs recorded by IndPenSim

6.4 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

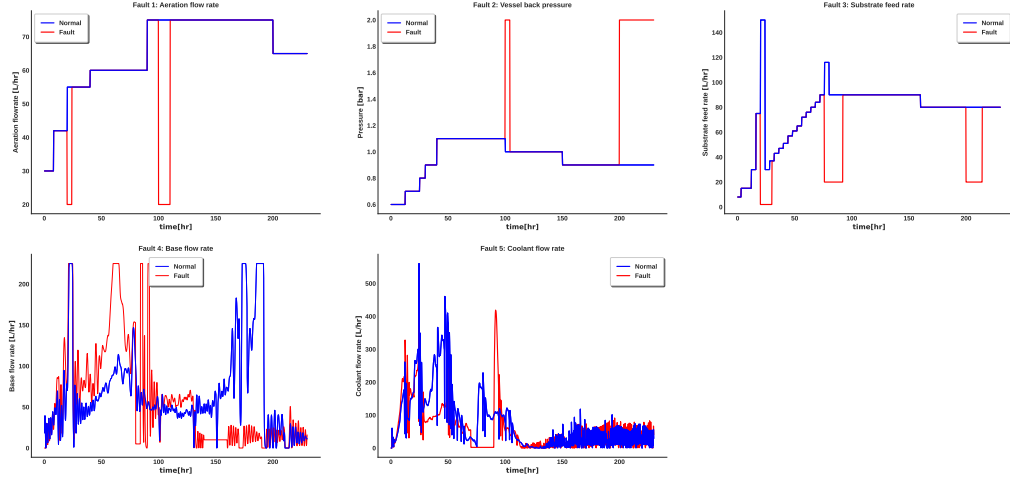


Figure 6.4: Profiles of measured variables in normal and faulty conditions.

Table 6.1: Description of different types of 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		

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 6.3.2).

6.4.1 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_α^2 and Q_α

Following the methodology presented in Section 6.2, the nPCs K of MPCA or MPLS are determined by evaluating the traditional objective function reported in other studies (refer Equation 6.15 and 6.16) 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 6.2.). Figs. (6.5)-(6.6) 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. 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.

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 control limits $T_{k,\alpha}^2$ 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 6.4.1. It was

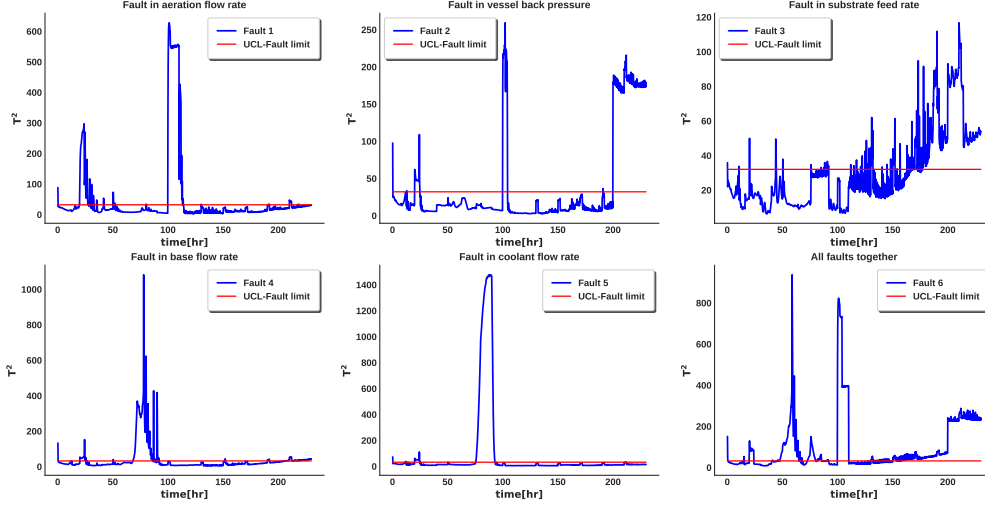


Figure 6.5: Plots of T_k^2 and static control limits T_α^2 for six different faulty batches of the test dataset based on J^{MPCA} as the objective function ($\alpha = 0.01$).

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 6.2). 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.

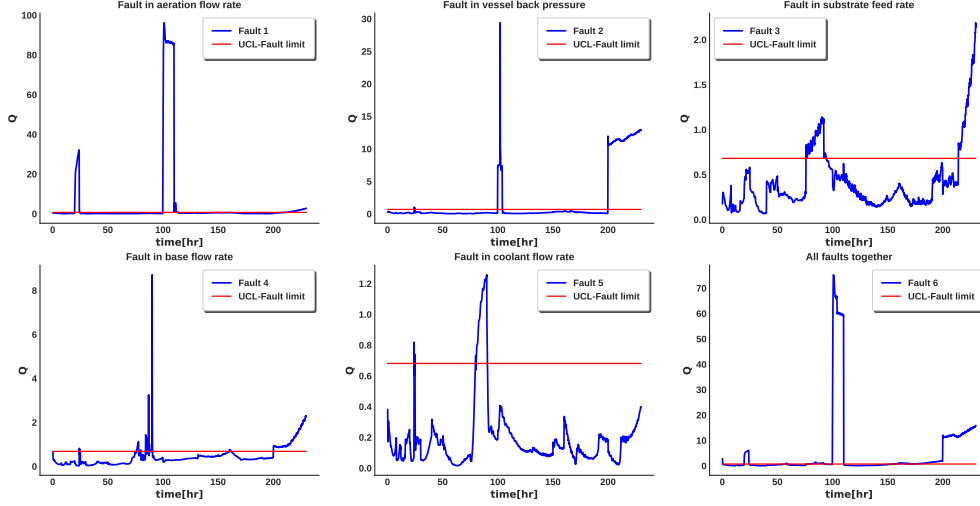


Figure 6.6: Plots of Q_k and static control limits Q_α for six different faulty batches of the test dataset based on J^{MPCA} as the objective function ($\alpha = 0.01$).

6.4.2 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. (6.3)-(6.7) 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 $J^{MPCA-MPLS,FDR}$ on validation dataset. It is observed (presented in Table 6.2) 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 $\overline{\text{FDR}}$ accu-

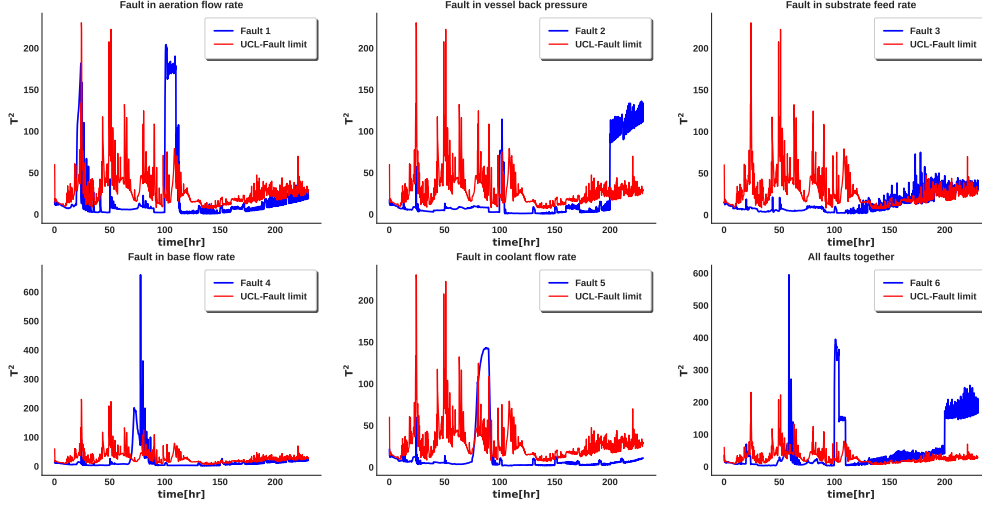


Figure 6.7: Plot of T_k^2 and dynamic control limits $T_{k,\alpha}^2$ of six different faulty batches of the test dataset by using J^{MPLS} as the objective function ($\alpha = 0.01$).

racy is highly related to the nPCs and the over-fitting of noise. 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 slightly better for faults 1 and 4. Conversely, Model No. 5 has better performance in all other faults and has lower false alarm rate (by about 10%). Moreover, the results are consistent in terms of the higher fault detection rates for MPLS models as compared to MPCA FD models.

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

Based on the novel objective function $J^{MPCA-MPLS,FDR}$ with dynamic control limits $T_{k,\alpha}^2$ 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

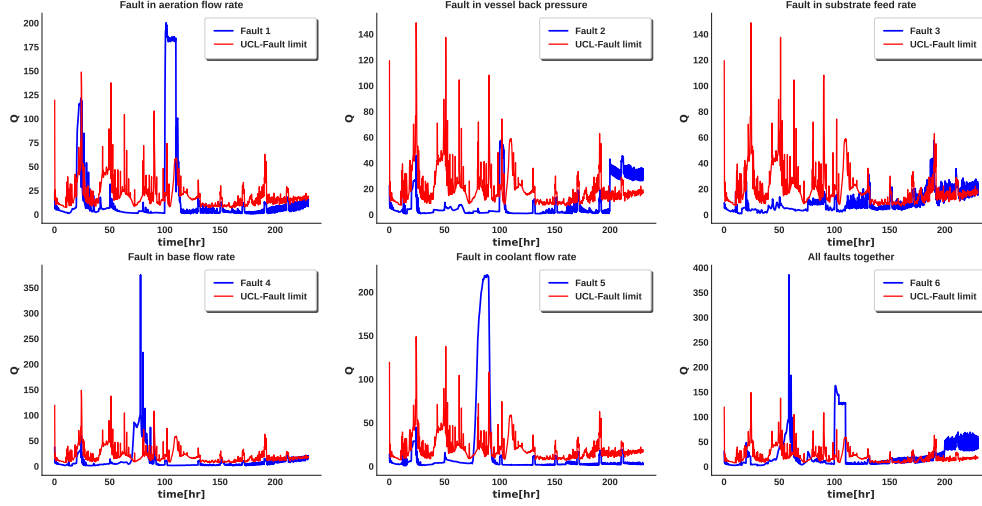


Figure 6.8: Plots of Q_k and dynamic control limits $Q_{k,\alpha}$ for six different faulty batches of the test dataset based on J^{MPLS} as the objective function ($\alpha = 0.01$).

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. (6.9)-(6.10).

6.4.3 MPLS-AE with $J^{MPLS-AE}$

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 predic-

Table 6.2: Comparison of ($\overline{\text{FDR}}$) test accuracy for different models

No.	Objective function	Model type	control limits	nPCs	$\overline{\text{FDR}}$ %(train/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

tion 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], value of weights in the objective function, 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 layer of the MPLS-AE network is shown in Figure 6.11. This figure indicates that the loss function does not significantly decreased beyond 15 latent variables (optimal number of

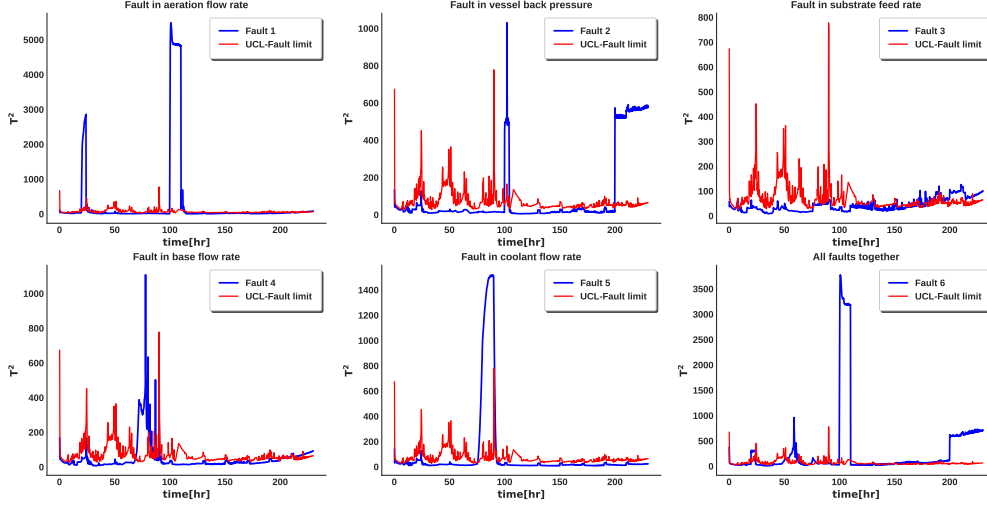


Figure 6.9: Plots of T_k^2 and dynamic control limits $T_{k,\alpha}^2$ for six different faulty batches of the test dataset based on $J^{MPCA-MPLS,FDR}$ as the objective function for the MPLS model ($\alpha = 0.01$).

units). Similar to the procedure described in section 6.4.1, 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.

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.

6.4.4 MPLS-AE with $J^{MPLS-AE,FDR}$

Finally, we evaluate the performance of MPLS-AE trained with the proposed loss function to maximize the $\overline{\text{FDR}}$ given by Equation (6.18) in combination with dynamic control limits.

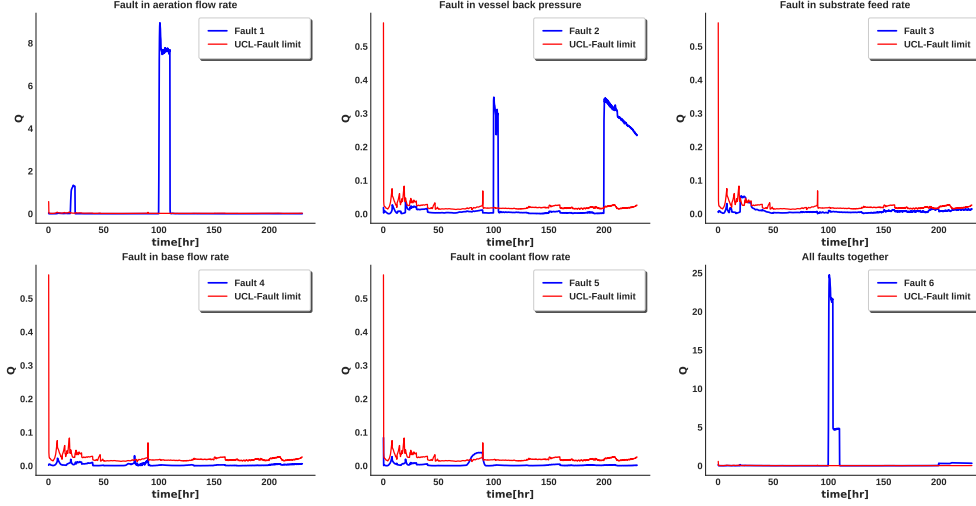


Figure 6.10: Plots of Q_k and dynamic control limits $Q_{k,\alpha}$ for six different faulty batches of the test dataset based on $J^{MPCA-MPLS,FDR}$ as the objective function for the MPLS model ($\alpha = 0.01$).

In order to train MPLS-AE with the proposed objective function, it is essential to evaluate the $H_{k,\alpha}^2$ and $SPE_{k,\alpha}$ by estimating the distribution of \mathbf{H}_k^2 and \mathbf{SPE}_k through KDE. Additionally, we need to evaluate the cumulative distribution function for a pre-specified significance level α (refer Equation 6.14). 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 popula-

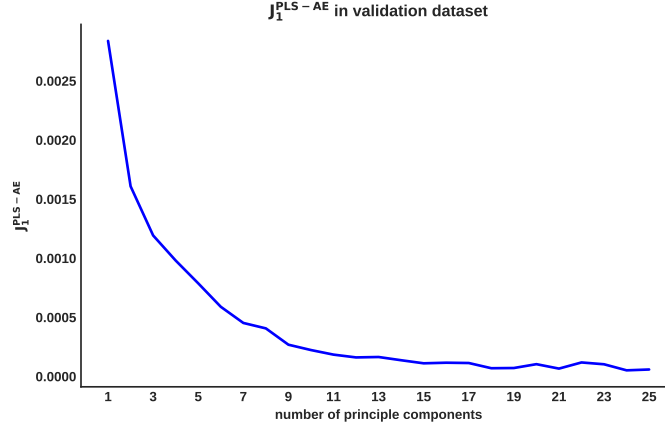


Figure 6.11: Loss of the validation dataset with MPLS-AE model using $J^{MPLS-AE}$ as the objective function to train the network.

tion (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. (6.12)-(6.13) 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 2) \overline{FDR} accuracy is obtained with the MPLS-AE model that use $J^{MPLS-AE,FDR}$ thus corroborating 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 6.2, the MPLS-AE trained with the novel objective function $J^{MPLS-AE,FDR}$ results in a higher \overline{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.

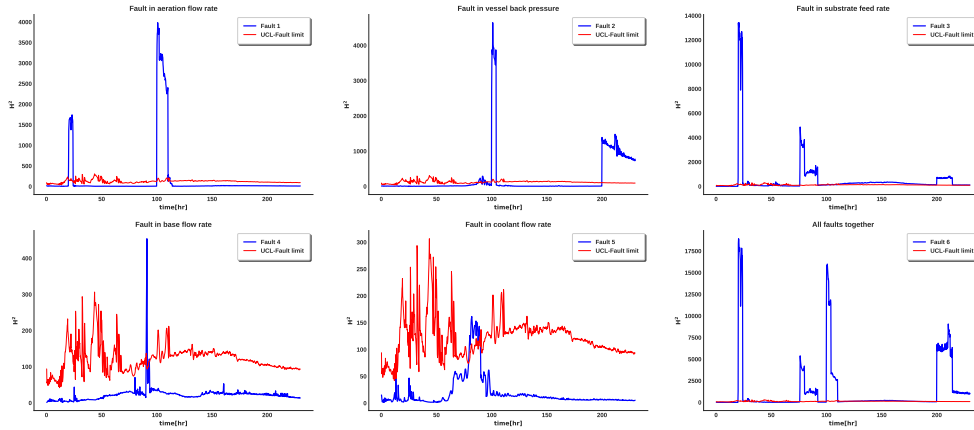


Figure 6.12: Plots of H_k^2 and dynamic control limits $H_{k,\alpha}^2$ for six different faulty batches of the test dataset based on $J^{MPLS-AE,FDR}$ as the objective function to train MPLS-AE ($\alpha = 0.01$).

6.5 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

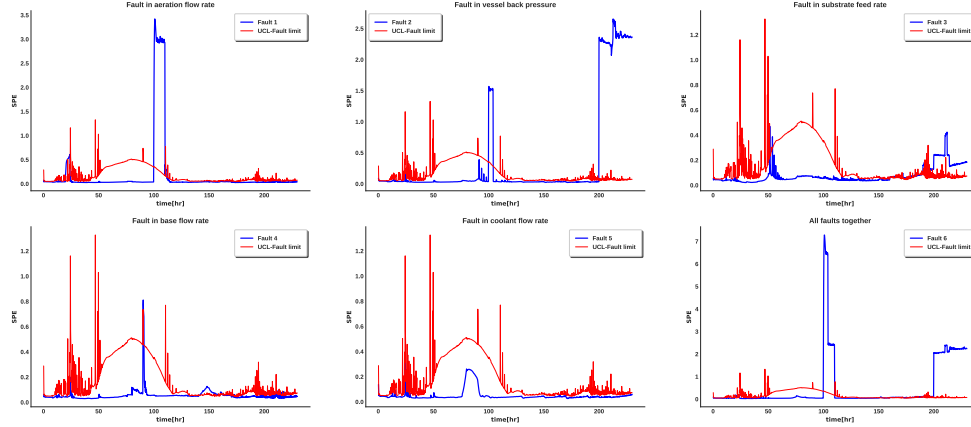


Figure 6.13: Plots of SPE_k^2 and dynamic control limits $SPE_{k,\alpha}^2$ for six different faulty batches of the test dataset based on $J^{MPLS-AE,FDR}$ as the objective function to train MPLS-AE ($\alpha = 0.01$).

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 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 7

Assessing Observability using Supervised Autoencoders with Application to Industrial Processes

Overview¹

This chapter presents a novel approach to calculate classification observability using a supervised autoencoder (SAE) neural network (NN) for classification. This metric is based on a minimal distance between every two classes in the latent space defined by the hidden layers of the auto-encoder. Quantification of classification observability is required to address whether the available sensors in a process are sufficient to observe certain outputs (phenomenon) and whether additional measurements are to be included in the dataset to improve classification accuracy. Using this metric an approach is proposed for enhancing the observability of the output classes from input data by selectively choosing the input features or variables to the NN which are relevant to the classification task and by discarding the variables that contribute to overlap between regions of the latent space corresponding

¹Adapted from Agarwal, Piyush, et al. "Assessing Observability using Supervised Autoencoders with Application to Tennessee Eastman Process"

to different output classes. Large overlap between regions will result in high confusion between classes and high miss-classification rates. To this end, we use a recently proposed Layer-wise Relevance Propagation (LRP) to identify input variables that are most relevant for the classification task and contribute to the overlapping regions corresponding to output classes. The efficacy of the proposed method is illustrated through two case-studies: the Tennessee Eastman Benchmark Process and Sanofi Vaccine Manufacturing Process.

7.1 Introduction

Identification of inputs that are highly informative and well correlated to an output of interest, e.g. productivity of a process, is crucial for efficient chemical process monitoring, improved system knowledge and operational robustness with respect to unknown disturbances. An input design space is defined as "the multidimensional combination and interaction of input variables and process parameters" [Laky et al. \[2019\]](#), that assures quality of product within specified operational constraints. Classification of the input variables' space into distinct regions that result in correspondingly distinct output classes is often challenging due to the proximity between input values that correspond to different classes combined with the presence of measurement noise. We focus here on the classification of regions that differ in terms of economic profit as a function of process inputs. The classification is based on a Supervised Autoencoder Neural Networks (SAE-NNs) model. SAE-NNs are Autoencoders (AE) that predicts both reconstructed inputs as well as outputs. Previously, SAE-NNs or its variants have been used for image classification and other regression tasks in a semi-supervised setting i.e. making use of both labelled and unlabelled data ([Epstein and Meir \[2019\]](#), [Seeger \[2001\]](#)). To accomplish this task the following objective function is minimized with respect to the weights of the SAE-NN:

$$l_{SAE} = \sum_{s=1}^N L_r^s(\mathbf{y}_s, \hat{\mathbf{y}}_s) + \lambda_1 \sum_{i=1}^N L_p^s(\mathbf{x}_s, \hat{\mathbf{x}}_s) \quad (7.1)$$

The addition of the input reconstruction loss L_r^s (first term in Equation (7.1)) to the supervised learning related term L_p^s for a sample s (second term in Equation (7.1)) in the

objective function has been a subject of study as to why better input reconstruction helps in better classification [Rigollet \[2007\]](#). This point is explicitly addressed in the current work. The focus of this work is to provide a robust lower bound on classification observability (C_{obs}) of output classes based on inputs fed to the SAE-NNs models. The ability of classifying regions of the input space that result in corresponding classes of a process output, e.g. process productivity/profit, depends on the degree of observability of the output from the measured process inputs. Quantifying observability of the classification task can help answering several important industrial questions such as: are the available sensors sufficient to provide acceptable classification accuracy? which sensors are more informative for the classification task? It should be noticed that observability cannot be assessed by standard state observability methods since a state model is assumed to be unavailable. Hence, the novelty of the current proposed method is in assessing observability directly from input-output data that to the knowledge of the author has not been thoroughly researched in the literature.

The development of an SAE-NN model to be used for classification involves several elements: i- feeding the inputs to the encoder, ii- feeding the outputs from the encoder to a fully connected layer and iii- feeding the outputs from the fully connected layer a to classification layer consisting of softmax functions. iv- simultaneous training of the autoencoder and classifier. Due to the data projection (compression) operation achieved by the encoder, the outputs from the encoder are referred to as latent variables. The difficulty in observing the output classes from input data is due to the proximity/overlap among sets of input data, model structure error and noise. Moreover, the support of the encoder functions corresponding to different output classes define regions in the latent variable space (output space of the encoder) that may strongly overlap with each other. This overlap may cause miss-classification of new samples, i.e. samples that were not used for model training. In this study we perform numerical evaluation of the overlap between regions in the latent space that correspond to different classes and the observability of the classes is quantified from the degree of overlap. The overlap is estimated for any two input data points \mathbf{x}_i and $\mathbf{x}_j \in \mathbb{R}^{d_x}$ based on a distance d_{ij} between their projections in the latent variable space $\mathbf{z} \in \mathbb{R}^{d_z}$ where points \mathbf{x}_i and \mathbf{x}_j corresponds to different classes ($d_x > d_z$). If these distances are large enough as compared to certain threshold d_{ij} (robust

observability distance measure) related to the noise in the input measurements, the classes are considered to be observable while if the distance is smaller than the threshold the system is considered unobservable. This observability criterion is mathematically given as follows:

$$d = \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 = \begin{cases} \text{observable,} & \text{if } d > d_{ij}. \\ \text{non-observable,} & \text{otherwise.} \end{cases} \quad (7.2)$$

where points \mathbf{z}_i and \mathbf{z}_j are projections of \mathbf{x}_i and \mathbf{x}_j in the latent space that corresponds to different output classes.

Quantifying the observability in the latent variable space as opposed to the input space capitalizes on the lower dimensions of the former as compared to the latter. Thus, drastically simplifying the calculation. Also, the latent space is more informative about productivity since the loss function used to train the model includes the prediction of productivity. Beyond its use for assessing classification observability, the degree of classification observability C_{obs} can be further enhanced by discarding (pruning) inputs that are not informative for classification (Agarwal and Budman [2019], Agarwal et al. [2019]) and contribute to overlap between regions corresponding to different output classes. The discarded inputs do not contribute to the classification task and instead they decrease the classification accuracy since they increase confusion among classes. A metric known as ‘Pruning Index’ (*PI*) is proposed to identify and prune input variables that contribute to this overlap or are irrelevant to the classification problem. Eliminating sensors that do not contribute significantly to classification may help to reduce cost and to reduce misclassification resulting from potentially faulty sensors/ irrelevant sensors.

Following the above, the three main contributions of this work: i) Assessment of the use of the reconstruction error for training the classification model; ii) Derivation of a robust observability distance measure (RODM) to evaluate the degree of classification observability C_{obs} ; iii) Identification of input variables contributing to the overlap. iv) Enhancement of the observability and classification accuracy by discarding inputs based on the proposed robust observability bound. The proposed contributions are illustrated through two case-studies: the Tennessee Eastman Benchmark Process and vaccine manufacturing process at

Sanofi Pasteur, Toronto.

The chapter is organized as follows: Section 7.2 provides a brief review on Autoencoder NNs. Section 7.3 provides the problem description for both TEP and vaccine manufacturing process case-study. The two algorithms used for quantifying a robust lower bound on classification observability are presented in Section 7.4. Results and discussions on the case-studies are shown in Section 7.5 for TEP and Sanofi Process followed by concluding remarks in Section 7.6.

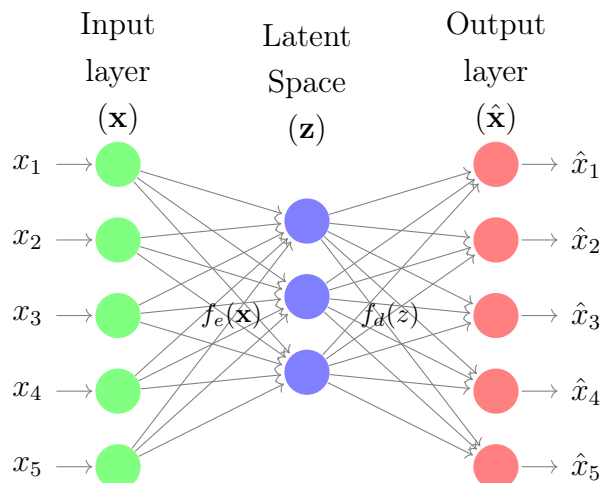


Figure 7.1: Traditional single layer Autoencoder Neural Network (AE-NN)

7.2 Preliminaries

This section briefly reviews the fundamentals of a Supervised Autoencoder Neural Networks (SAE-NNs) models.

7.2.1 Supervised Autoencoder Classification Neural Networks (SAE-NNs)

The Supervised Autoencoder Neural Network (SAE-NN) model, shown in Figure ??, is trained based on the minimization of a combination of the reconstruction loss function and the supervised classification loss corresponding to the first and second terms in (Equation (7.6)) respectively. The reconstruction loss function in Equation (7.1 and 7.6) is ensuring that the calculated latent variables are able to reconstruct the input data with good accuracy. The goal is to learn a function that predicts the class labels in one-hot encoded form $\mathbf{y} \in \mathbb{R}^m$ from inputs $\mathbf{x} \in \mathbb{R}^{d_x}$. The encoder operation for a single hidden layer between the input variables to the latent variables $\mathbf{z} \in \mathbb{R}^{d_z}$ is represented as follows:

$$\mathbf{z} = f_e(\mathbf{W}_e \mathbf{x} + \mathbf{b}_e) \quad (7.3)$$

The latent variables are used both to predict the class labels and reconstruct inputs \mathbf{x} as follows:

$$\hat{\mathbf{x}} = f_d(\mathbf{W}_d \mathbf{z} + \mathbf{b}_d) \quad (7.4)$$

$$\hat{\mathbf{y}} = f_c(\mathbf{W}_c \mathbf{z} + \mathbf{b}_c) \quad (7.5)$$

where f_c is a non-linear activation function for the output layer. $\mathbf{W}_c \in \mathbb{R}^{m \times d_z}$ and $\mathbf{b}_c \in \mathbb{R}^m$ are output weight matrix and bias vector respectively. For training the SAE, the following loss function is minimized:

$$\begin{aligned} l_{SAE} &= \lambda_1 \sum_{s=1}^N L_r^s(\mathbf{x}_s, \mathbf{W}_d \mathbf{W}_e \mathbf{x}_s) + \sum_{s=1}^N L_p^s(\mathbf{W}_c \mathbf{W}_e \mathbf{x}_s, \mathbf{y}_s) \\ &= \frac{\lambda_1}{N} \sum_{s=1}^N \|\mathbf{x}_s - \hat{\mathbf{x}}_s\|_2^2 + \frac{1}{N} \sum_{s=1}^N \sum_{c=1}^m -y_{s,c} \log(p_{s,c}) \\ &= \frac{1}{N} \left[\lambda_1 \sum_{i=1}^N \|\mathbf{x}_s - \hat{\mathbf{x}}_s\|_2^2 + \sum_{s=1}^N \sum_{c=1}^m -y_{s,c} \log(p_{s,c}) \right] \end{aligned} \quad (7.6)$$

$$p_{s,c} = \frac{e^{(y_{s,c})}}{\sum_{c=1}^m e^{(y_{s,c})}} \quad (7.7)$$

where λ_1 is the weight for the reconstruction loss L_r , m is the number of classes, $y_{s,c}$ is a binary indicator (0 or 1), 1 if class label c is the correct classification for observation s , $\hat{y}_{s,c}$ is the non-normalized log probabilities and $p_{s,c}$ is the predicted probability for a sample of class c .

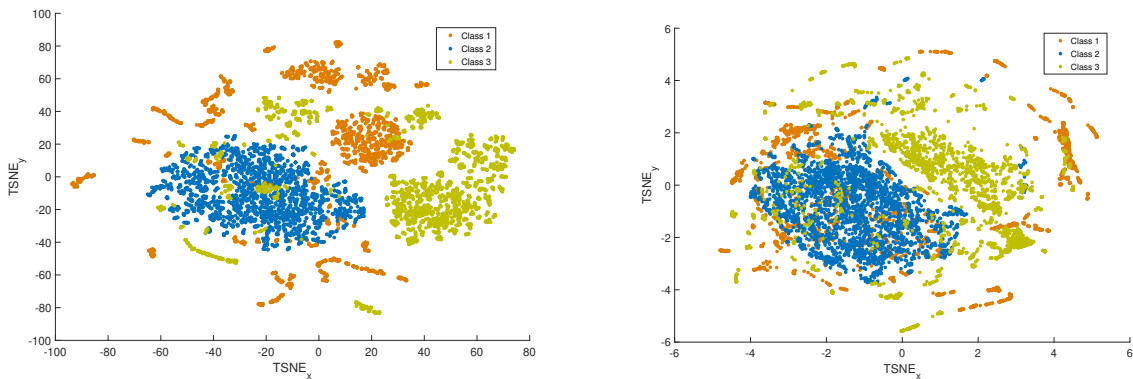


Figure 7.2: Left: Projection of input space in 2 dimensions using TSNE for non-overlapping case (Case 1), Right: Projection of input space in 2 dimensions using TSNE for overlapping case (Case 2)

7.3 Description of Case Studies

This study was motivated by the intention of Sanofi to measure Off-gas compositions in the Pertussis vaccine manufacturing process. This exercise was performed with the aim to investigate whether the off-gas data (eg. Argon, Carbon, Oxygen, Nitrogen, Carbon Emission Rate (CER) and Oxygen Uptake Rate(OUR)) contribute information in correlating the quality variables such PRN ELISA and Kjeldahl measurements to the classification of high productivity versus low productivity batches. The data consist of fermentation profiles for 295 batches and 52 batches of both fermentation and off-gas profiles. Owing to this fact, we first develop the algorithm for evaluating classification observability and test it on TEP simulator and then the same methodology was applied to the vaccine manufacturing process.

Tennessee Eastman problem: The TEP involves several unit operations including a vapor-liquid separator, a reactor, stripper, a recycle compressor and a condenser. Four gaseous reactants (A, B, C and D) form two liquid products streams (G and H) and a by-product (F). Although several TEP simulators are available, in this work the one developed by Larsson et al. [2001] was used and a schematic of the process is shown in Figure 7.3. The original controller settings were modified and different disturbances, i.e. referred to as faults in the TEP simulator, were introduced in order to generate different ranges of values of process profit since the goal in the current study is to classify the inputs according to their resulting process profit. The simulator involves 53 input variables of which 3 manipulated variables (Compressor Recycle Valve (XMV(5)), Stripper Steam Valve (XMV(9)) and Agitator Speed XMV(12)) were discarded initially (number of input variables = 50). Since the process profit for this case study is determined solely by the operating costs of the plant, this profit will be referred to as cost of productivity (COP).

COP (\$/hr)	High Profit	Intermediate	Low Profit
Case 1	> 89.6	89.6 – 142.6	< 142.6
Case 2	> 108	108 – 130	< 130

Table 7.1: Profit-based defined classes for COP

Also, since the boundaries between classes corresponding to different ranges of COP values can be chosen arbitrarily, we examine two different cases that are defined in Table 7.1. These cases differ in the overlap between classes. This overlap is calculated from the training data based on simulated frequency of occurrences of COP values as shown in Figure 7.4. As shown in this figure, Case 1 corresponds to very low overlap while Case 2 results in significant overlap between classes. The overlap is illustrated by TSNE (t-distributed Stochastic Neighbor Embedding, Maaten and Hinton [2008]) projections of the input design space for the high overlap in Figure 7.2.

A total of 8 datasets were generated, 1 normal operation and 7 each involving one known fault (IDV(1)-IDV(7)) for a total simulation time of 800 hours, i.e. a 100-hour duration for each dataset. Each fault was activated at the start of the corresponding 800-hour time

period and data samples were collected at a sampling rate of 100 samples/hour (total number of samples 8×10^4 per dataset). Out of which 3×10^4 samples were considered as training dataset and 1.5×10^4 samples as validation dataset and testing dataset. Each of these datasets resulted in various ranges of COP values, i.e. different classes (refer Figure 7.4 and Table 1) to be identified by the SAE-NN model.

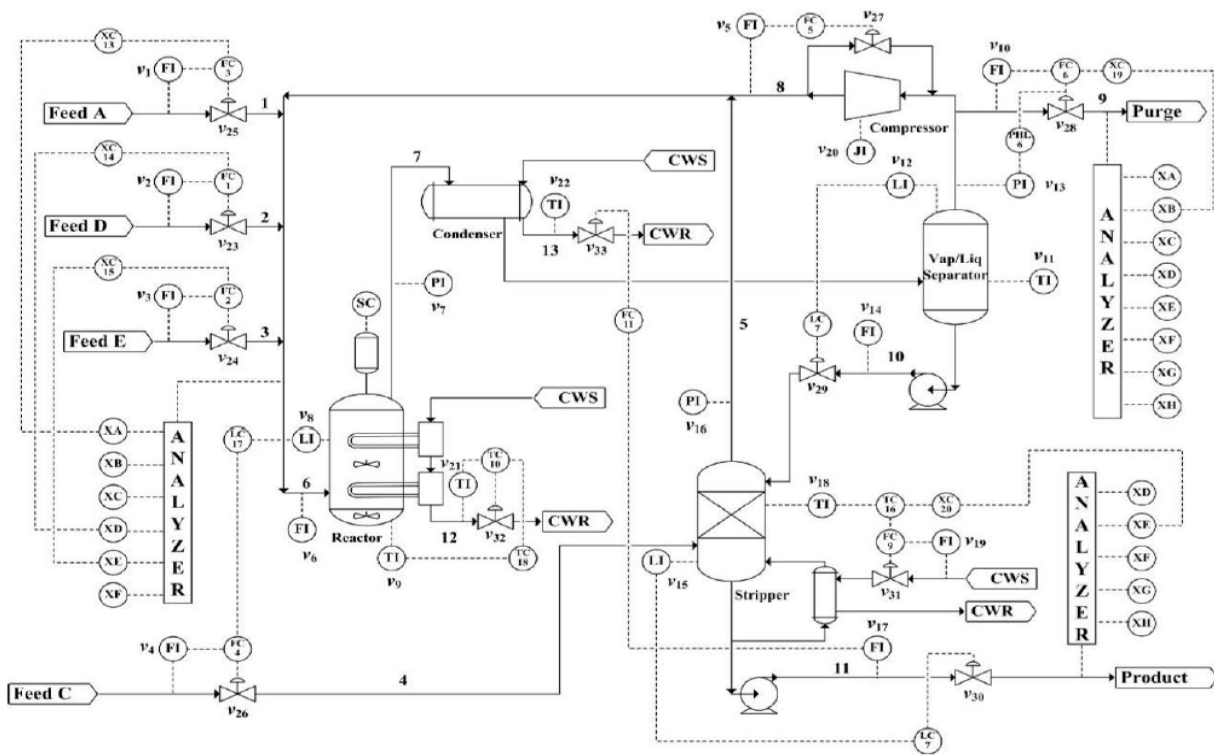


Figure 7.3: Schematic of Tennessee Eastman plant process (Downs & Vogel, 1993)

7.4 Proposed Methodology

The goal is to find an observability measure which is robust to measurement noise in the input data. The proposed algorithm is based on calculating a distance measure in the latent space using SAE with respect to noisy inputs using a boot-strapping approach. The two algorithms are as follows:

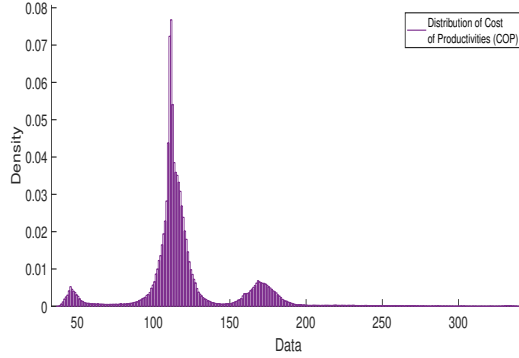


Figure 7.4: Distribution of Cost Of Productivities (COP)

1. Algorithm 1 (Robust Observability Distance Measure (RODM)) for the computation of the RODM ($d_{ij}, i \neq j$).
2. Algorithm 2 (Evaluation of degree of observability C_{obs}) for the computation of a degree of observability C_{obs} which is defined as the percentage of overlap between points within a neighbourhood of distance ($d_{ij}, i \neq j$) from each point, where i and j are points in different class labels.

To compute the RODM d_{ij} , first a SAE-NN model is trained on the training data collected from the process. Subsequently a bootstrapping approach is used where the inputs are perturbed around different nominal values for R number of realizations of white-noise and the variances in the latent variables resulting from these input perturbations are evaluated (refer Equations (7.8) and (7.9)). Finally, a distance measure is defined as the maximum of l_2 norm of the variance of perturbations in latent variables due to noisy inputs (refer Equation and (7.10)) across R realizations (see Algorithm 3).

To calculate the degree of classification observability C_{obs} (see Algorithm 4), we first evaluate pairwise Euclidean distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ where N is the number of samples in the validation dataset. Afterwards, inter-class samples that are closer than RODM (d_{ij}) are selected (referred as Total percentage overlap (%TOv) in Algorithm 4). Thereafter, the points which are correctly classified are discarded and the remaining samples are used

Algorithm 3 Robust Observability Distance Measure (RODM)

- 1: Train an SAE classification NN $g(\mathbf{W}_c \mathbf{W}_e \mathbf{x})$ using an optimal weighting of the reconstruction and classification loss-functions, where this weighting is found by using a validation dataset.
- 2: Perturb the input variables, \mathbf{x}_l ($l = 1, 2, \dots, d_x$) (mean $\mu_{\mathbf{x}_l} = 0$; variance $\sigma_{\mathbf{x}_l}^2$), with input perturbations $\Delta \mathbf{x}_l$. Where $\Delta \mathbf{x}_l$ ($l = 1, 2, \dots, d_x$) are independent normally distributed (i.i.d) random variables that has mean $\mu_{\Delta \mathbf{x}_l} = 0$ and variance $\sigma_{\Delta \mathbf{x}_l}^2$ for R uncorrelated realizations such that Signal-to-noise ratio (SNR) $\frac{\sigma_{x_l}^2}{\sigma_{\Delta x_l}^2} = 10$ is maintained.
- 3: Compute the latent feature vectors \mathbf{z}_k ($k = 1, 2, \dots, d_z$) for R realizations of $\mathbf{x}_l + \Delta \mathbf{x}_l$ using the trained SAE model.
- 4: Estimate the variances of the latent variables resulting from the introduced perturbations to the inputs (noise) for R realizations in the latent space as follows:

$$V(\Delta \hat{\mathbf{z}}_k) = \mathbb{E} \left[\left(\Delta \mathbf{z}_k - \mathbb{E}(\Delta \mathbf{z}_k) \right) \left(\Delta \mathbf{z}_k - \mathbb{E}(\Delta \mathbf{z}_k) \right)^T \right] \quad (7.8)$$

$$\Delta \mathbf{z}_k = \left(g(\mathbf{W}_e(\mathbf{x}_i + \Delta \mathbf{x}_i)) - g(\mathbf{W}_e \mathbf{x}_i) \right) \quad (7.9)$$

where $k = 1, 2, \dots, d_z$ & \mathbb{E} is an expectation operator.

- 5: Robust Observability Distance Measure (RODM) is computed as the maximum of l_2 norm of the estimated variance of $\Delta \mathbf{z}$ for R realizations as:

$$d_{ij} = \max \left\{ \sqrt{V(\hat{\Delta \mathbf{z}}_1) + V(\hat{\Delta \mathbf{z}}_2) + \dots + V(\hat{\Delta \mathbf{z}}_k)} \right\}_R \quad (7.10)$$

where $i \neq j$.

Algorithm 4 Evaluation of degree of observability C_{obs}

```

1: Evaluate pairwise Euclidean distance matrix  $\mathbf{D} \in \mathbb{R}^{N \times N}$ 
2: Determine the indices ( $ind_c$ , where  $c = 1, 2, \dots, m$ ) of samples corresponding to  $m$  different classes.
3:  $ind = \{ind_1, ind_2, \dots, ind_m\}$ ;  $t = 0$ 
4: for  $i$  in number of classes ( $m$ ) do
5:   for  $j$  in number of classes ( $m$ ) do
6:     if  $i \neq j$  then
7:        $t = t + 1$ 
8:       for  $q$  in  $ind_i$ .length (samples of class  $i$ ) do
9:         for  $r$  in  $ind_j$ .length, where  $j_1 = ind_j$  do
10:           $to_{ij} = \text{list}() \ \& \ cov_{n(u)l(u)} = \text{list}()$ 
11:          if  $\mathbf{D}(ind_i(q), ind_j(r)) < d_{ij}$  then
12:             $to_{ij}.add = ind_j(r)$ 
13:          end if
14:        end for
15:      end for
16:    end if
17:    return  $n(t) = i$ 
18:    return  $l(t) = j$ 
19:  end for
20: end for
21: for  $i$  in  $m(m-1)$  i.e. #overlapping regions ( $i \neq j$ ) do
22:    $to_{ij} = \text{unique}(to_{ij})$ 
23: end for
24:  $to_{v} = \{to_{12}, to_{13}, \dots, to_{1m}, to_{21}, to_{23}, \dots, to_{m(m-1)}\}$ 
25: Total Percentage Overlap (% TOv) is determined by:

```

$$\%TOv = \frac{\sum_{i \neq j} \text{unique}(to_{ij}).\text{length}}{ind.\text{length}} \quad (7.11)$$

```

26: Total Classification Percentage Overlap (% TCOv) is determined by calculating SAE-NN output probabilities  $p_{i,c}$ , where  $c = (1, 2, \dots, m)$  into account.
27: for  $u$  in  $m(m-1)$  i.e. #overlapping regions ( $i \neq j$ ) do
28:   for  $v$  in the length of  $to_{n(u)l(u)}$  do
29:     if  $p_{v,l(u)} > p_{v,l \setminus l(u)}$  then
30:        $cov_{n(u)l(u)}.add = to_{ij}(u, v)$ 
31:     end if
32:   end for
33: end for
34:  $cov = \{cov_{12}, cov_{13}, \dots, cov_{1m}, cov_{21}, cov_{23}, \dots, cov_{m(m-1)}\}$ 
35: Total Classification Percentage Overlap (% TCOv) is determined by:

```

$$\%TCOv = \frac{\sum_{n(u) \neq l(u)} \text{unique}(cov).\text{length}}{ind.\text{length}} \quad (7.12)$$

$$C_{obs} = 100\% - (\%TCOv + (100\% - \text{Training \%Accuracy}))$$

$$C_{obs} = \%Training \text{ Accuracy} - \%TCOv \quad (7.13)$$

to evaluate the Total Classification Percentage Overlap (%TCOV) as:

$$\%TCOV = \frac{\sum_{n(u) \neq l(u)} \text{unique}(\text{cov}).\text{length}}{\text{ind.length}} \quad (7.14)$$

Finally the degree of observability C_{obs} is calculated as

$$C_{obs} = \%Training \text{ Accuracy} - \%TCOV \quad (7.15)$$

The obtained C_{obs} is evaluated for the worst-case using RODM and represents the lower bound on the degree of observability i.e. new samples are expected to exhibit equal or larger classification accuracy.

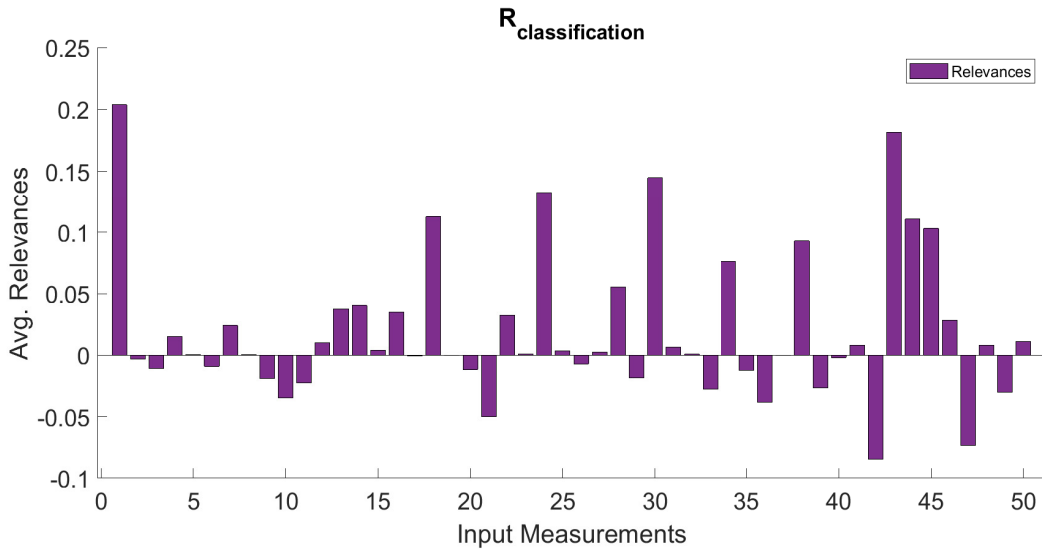


Figure 7.5: Average Relevance corresponding to correctly classified samples for low overlapping case (Case 1)

7.5 Results and Discussion

The following section presents the results of the application of the algorithms of the previous section to the TEP case-study. The advantage of adding the reconstruction loss function L_r in Equation (7.6) is also assessed.

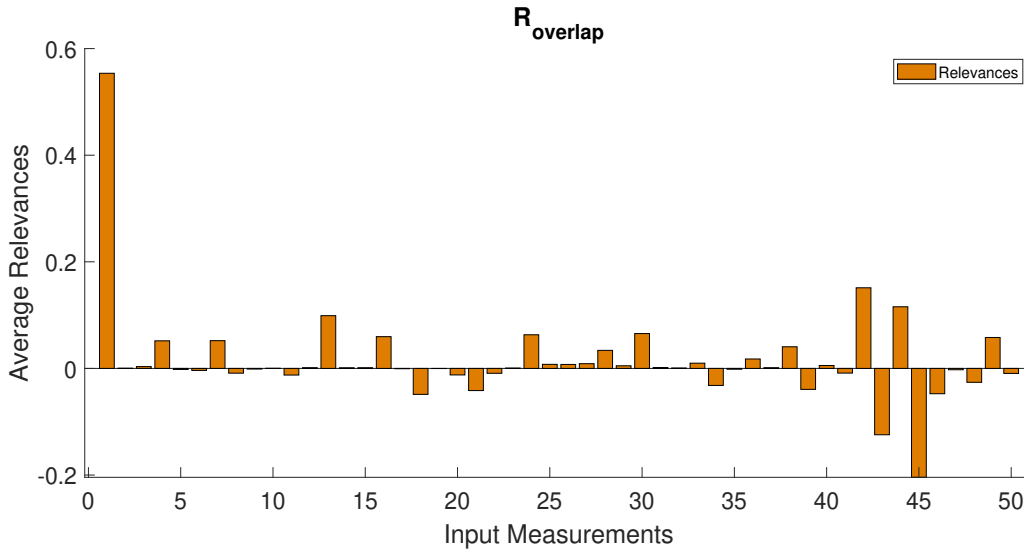


Figure 7.6: Average Relevance corresponding to overlapping samples for Case 1

7.5.1 Effect of Reconstruction Error Loss function on classification accuracy

The first objective of the case study is to investigate whether the addition of the reconstruction loss L_r term to the supervised loss function L_p (see Equation (7.6)) for training the classification AE-NN model helps to improve classification accuracy. An SAE-NN with a single layer was trained with different weights λ_1 with a validation dataset. The hyperparameters including the weight multiplying the reconstruction loss L_r term in Equation (7.6) and the dimension of the latent space \mathbf{z} were chosen based on the highest classification accuracy achieved on the validation set for both cases 1 and 2. It can be observed in Table 7.3 that for different dimensions of the latent space \mathbf{z} , the validation classification accuracy and test classification accuracy with reconstruction loss function L_r was always higher than the NN architecture without the reconstruction loss function L_r .

Output Class	1	2	3	
1	4654 31.0%	21 0.1%	303 2.0%	93.5% 6.5%
2	216 1.4%	4979 33.2%	0 0.0%	95.8% 4.2%
3	130 0.9%	0 0.0%	4697 31.3%	97.3% 2.7%
	93.1% 6.9%	99.6% 0.4%	93.9% 6.1%	95.5% 4.5%
	1	2	3	
	Target Class			

Figure 7.7: Confusion Matrix for Case 1 (Validation Data-set)

7.5.2 Degree of Classification Observability for the TEP problem

The degree of classification observability C_{obs} is calculated according to Algorithms 1 and 2, presented in Section 4. First the RODM is calculated for both the cases i.e. Case 1 and Case 2 using $R = 1000$ realizations of input perturbations. It can be observed that RODM is larger i.e. $d_{ij} = 1.14$ for Case 1 as compared to Case 2 i.e. $d_{ij} = 0.5168$ which indicates that Case 1 has higher degree of observability than Case 2. The confusion matrix for Case 1 and the classification overlap (COv) matrix evaluated using both Algorithm 3 and 4 are shown in Figures 7.7 and 7.8 respectively. The computation of COv matrix explains the root cause for the miss-classification of samples. The numbers shown in coloured boxes of COv matrix represents the number of samples that are miss-classified because of the proximity between each two different regions. The numbers shown below (in the brackets) shows the total number of samples miss-classified. The degree of classification observability C_{obs} for Case 1 is:

$$C_{obs} = 97.74\% - 2.46\% = 95.28\%$$

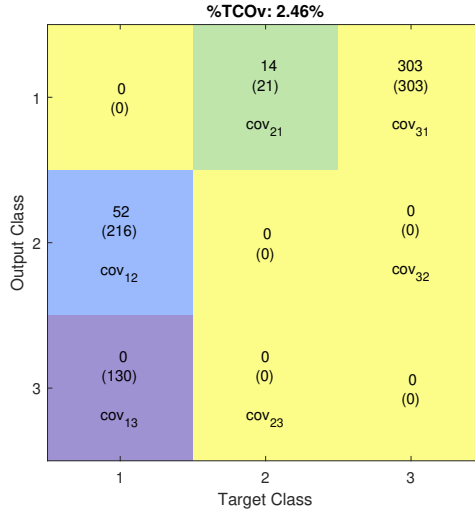


Figure 7.8: Classification Overlap Matrix (COv) for Case 1 (Validation Data-set)

It can be seen that the C_{obs} (lower-bound) is smaller than the validation data-set accuracy 95.5% (shown in the right corner of Figure 7.7) and 97.5% for the test data-set accuracy (see Table 7.2). The results for both Case 1 and Case 2 are summarized in Table 7.2. The results corroborate that Case 1 is easily separable than Case 2 i.e. the degree of classification observability for Case 1 is much higher than Case 2.

7.5.3 Enhancing Classification Observability

A key application of the degree of observability estimate provided above is for identifying the input variables that are informative about the classification problem and the variables that are not informative since they result in overlap between regions of the latent space that correspond to different classes. Thus, based on the degree of classification observability C_{obs} , a method is proposed to enhance the observability by pruning (discarding) input variables and the corresponding neural network interconnections that are not relevant for the classification task using Layer wise Relevance Propagation (LRP). Firstly, the average of relevances of each input variable is assessed using the LRP for both correctly classified samples ($\bar{R}_{classification}$) and overlapped samples ($\bar{R}_{overlap}$). LRP algorithm is based on a

Table 7.2: Degree of classification observability (C_{obs}) for Case 1 and Case 2

	COV ₁₂	COV ₂₁	COV ₁₃	COV ₃₁	COV ₂₃	COV ₃₂	$d_{ij}(i \neq j)$	% TCOV	C_{obs}
Case 1	52	14	0	303	0	0	1.14	2.46%	95.28%
Case 2	4565	8	48	1931	0	1070	0.5168	43.84%	53.78%
Enhanced Classification Observability									
Case 1	54	1	10	240	0	0	1.5818	2.02%	96.16%
Case 2	4542	7	74	774	0	969	0.8654	42.40%	54.47%

Table 7.3: Classification Accuracy for both cases ($\mathbf{z} \in \mathbb{R}^{d_z}$, $d_z = 7$)

	L_r Weights	Validation Accuracy	Training Accuracy	Test Accuracy
Case 1	0.5	95.53%	97.74%	97.5%
Case 1	0	94.81%	97.12%	-
Case 2	0.5	55.94%	97.62%	55.8%
Case 2	0	55.59%	92.05%	-

backward propagation of scores obtained at the output layer of each class label to the input variables through the interconnections of the neural network (Bach et al. [2015], Agarwal and Budman [2019], Agarwal et al. [2020]). Figure 7.5 and 7.6 shows the average of relevance of input variables for all correctly classified and overlapping samples respectively. Secondly, to assess the importance of whether a variable can be discarded or not, we define a ratio referred to as ‘Pruning Index’ (PI) for an input variable l ($l = 1, 2, \dots, d_x$) as follows:

$$PI_l = \left| \frac{\bar{R}_{overlap}^l}{\bar{R}_{classification}^l} \right| \times \left(\frac{N_{overlap}}{N_{classification}} \right) \quad (7.16)$$

where $N_{overlap}$ are the number of samples that result in overlap in the latent space based on the RODM defined above and $N_{classification}$ are the number of samples correctly classified. If the ratio is greater than 1 for an input variable, the variable is dropped while if the ratio PI is smaller than 1 then the variable is deemed important for classification.

$$PI_l = \begin{cases} \text{pruned,} & \text{if } PI_l \geq 1. \\ \text{not pruned,} & \text{if } PI_l < 1. \end{cases} \quad (7.17)$$

There is a trade-off between the contribution of an input variable to the classification task and to overlap. PI can be interpreted as the ratio between total relevance of an input variable to the class overlap to the total relevance of an input variable to classification. Finally, an SAE-NN is re-trained with the remaining variables using the training data-set and validated on a validation data-set. For the non-overlapping case (Case 1) 48 variables (two variables were discarded) were used while for the high overlap case (Case 2) 22 input variables were discarded to improve the degree of observability. After discarding the irrelevant inputs the C_{obs} was re-calculated using Algorithm 3 and 4. The results are shown in Table 7.2. It can be seen that the classification observability increased for both cases. Also, similar to the results shown in Section 7.5.1, the test accuracy and validation accuracy were higher for both cases with the presence of the reconstruction loss in the objective function. The key advantages in the elimination of input variables for classification, especially in Case 2 (22 variables were eliminated), is that it potentially reduces costs and the risk for miss-classification from potentially faulty measurements while slightly enhancing the observability of the classes.

7.5.4 Degree of Classification Observability for the Vaccine Manufacturing Process at Sanofi Pasteur, Toronto

Classification observability C_{obs} is evaluated for 4 years (2014-2018) of fermentation data and for 52 new batches measuring both fermentation data and off-gas data combined with 4 years of data with respect to PRN antigen.

Case 1: Fermentation data for 295 batches

Case 2: Fermentation data for 295 batches + Fermentation data for 52 new batches + Off-gas data for 52 new batches.

C_{obs} for Case 1 is 74.48% and for Case 2 is 62%.

The results led to the conclusion that there is no additional information in 52 new batches (off-gas data) with respect to PRN yield. The following points have to be considered when analyzing these results:

1. Miss-alignment of off-gas data variables:

There is no a reference trajectory for off-gas data, the off-gas data in 52 new batches can not be aligned to the same length as of fermentation profiles in 295 batches.

2. Miss-match between fermentation variables for Case 1 and Case 2:

Case 1: Acid Quantity, Pressure, Jacket Temperature, Seal Temperature

Case 2: Antifoam, Supply Quantity

Variables in Case 1 were not available in Case 2 and variables in Case 2 were not available in Case 1.

3. High weighting of 295 samples:

The model is expected to be biased towards 295 samples for Case 2 as compared to the 52 new samples that are added since the overall objective is to have high classification accuracy.

A simple mechanistic balance of oxygen was formulated to understand the lack of information of the off-gas data about the process productivity.

DO: Dissolved oxygen

$$\frac{dO_2^l}{dt} = K_{la}(O_2^* - O_2^l) - \text{Oxygen consumed by cell or produced}$$

O_2^l = DO: Conc. of oxygen in liquid phase

O_2^* : Saturated oxygen concentration K_{la} : Mass-transfer rate

O_2^{met} : Metabolic conc. of oxygen

$$\frac{dO_2^l}{dt} = K_{la}(O_2^* - O_2^l) - O_2^{met} \quad (7.18)$$

$$O_2^{met} = O_2^{\text{consumed by cells}} - O_2^{\text{produced by cells}}$$

O_2^g : Conc. of oxygen in gas phase

$$\frac{dO_2^g}{dt} = \underbrace{O_2^{in}}_{\text{Aeration}} - \underbrace{O_2^{out}}_{\text{off-gas}} - K_{la}(O_2^* - O_2^l) \quad (7.19)$$

Since there is a control of $DO = O_2^l$

$$\frac{dO_2^l}{dt} = 0 = K_{la}(O_2^* - O_2^l) - O_2^{met}$$

$$\implies K_{la}(O_2^* - O_2^l) = O_2^{met} \quad (7.20)$$

And assuming $\frac{dO_2^g}{dt} = 0$

$$O_2^{in} - O_2^{out} = K_{la}(O_2^* - O_2^l) \quad \text{from (2)}$$

or

$$O_2^{out} = O_2^{in} - K_{la}(O_2^* - O_2^l) \quad (7.21)$$

Equation 7.24 implies that O_2^{out} can be inferred from

$$O_2^{out} = O_2^{in} - K_{la}(O_2^* - O_2^l)$$

Since $K_{la} = f(\text{RPM, Aeration, etc})$

then $O_2^{out} = f(\text{aeration, RPM, etc})$

Based on earlier studies it has been hypothesized that oxidative stress may be a main source of disturbances and changes in productivity. If the disturbance is in the oxidative stress (O_2^{produced} from catalase action)

$$O_2^{\text{net}} = O_2^{\text{consumed by cells}} - O_2^{\text{produced by cells}}$$

From Equation 7.23, we can estimate the net O_2^{net} but we can not observe individually $O_2^{\text{consumed by cells}}$ and $O_2^{\text{produced by cells}}$. So we cannot observe possible disturbances that depends upon the individual contributions of O_2^{consumed} and O_2^{produced} . On the other hand, based on Equation 7.24, the off-gas data could be used to detect dissolved oxygen or temperature sensor errors in O_2^l (faulty probe problems or out of calibration sensor) and/or mixing problems. Notice K_{la} is a function of RPM (stirring rate) and aeration rate, O_2^* is the saturation concentration (generally function of temperature). Thus, to observe disturbances related to reactive oxidative stresses (ROS) we need to look at other variables such as NADPH or ROS by fluorescence, cytometry or other analytical devices. Another small scale study was also performed for predicting off-gas data trajectories using the fermentation data profiles. A sequence to sequence LSTM model was trained. The results from the small scale study demonstrated that the off-gas data can be predicted with high accuracy using the fermentation profiles.

7.6 Conclusion

This work presents a novel method to compute a robust observability distance measure (RODM) and evaluate degree of classification observability C_{obs} for a classification problem based on noisy input data. The proposed method first computes a distance metric such that two clusters of points belonging to different classes should be at least distance d_{ij} ($i \neq j$) apart in the worst case-scenario where i and j are points corresponding to different labels in representation space for a good classification. The merit of the method is that it can be used to assess the observability of output classes from available input data that is corrupted by noise. Furthermore, it is shown that the observability and classification

accuracy can be enhanced by discarding variables that are not relevant for the classification task and contribute to overlap between different regions corresponding to output classes. The proposed methodology is demonstrated on two case studies i.e. on TEP and vaccine manufacturing process. Though the results on the Sanofi Process with respect to off-gas data were not positive, the method correctly predicted that off-gas data is not informative about the main suspected disturbance to the process.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This thesis proposes the use of Deep Learning algorithms in chemical/ bio-pharmaceutical industries for deriving process insights, for designing of supervised and unsupervised statistical process monitoring methodologies and for evaluating observability of normal/abnormal operation from available measurements.

8.1.1 Classification of profit-based operating conditions

The findings from this part of the research are as follows:

1. Deep learning models are inherently over-parameterized and thus their use for identifying the sources of process variability is challenging. In addition, the use of many parameters increases the potential of over-fitting the data and increasing the sensitivity to noise. To address these issues this research work presented a novel neural network (NN) pruning algorithm referred to as Sequential Layer-wise Relevance Propagation for Pruning (SLRPFP) based on relevance of input variables. The proposed method first computes the relative relevance scores for all input variables followed by eliminating the input variables which are below a certain threshold value.

2. Pruning of irrelevant variables significantly reduced the number of parameters required to calibrate the models. For TEP the number of parameters were reduced from 37,875 to 10,840 for Case 1 and from 60,500 to 11,402 for Case 2. For the vaccine manufacturing process at Sanofi Pasteur, the number of input variables reduced from 1760 to 443. Reduction in both parameters and input variables resulted in 4% and 1.6% improvement in test classification accuracy for MLP and LSTM classification model respectively on Sanofi Process. Similarly, there was an improvement of 1.02% and 4% for Case 1 and Case 2 respectively for TEP.
3. The pruning methodology provided important physical insights on the system regarding the inputs that have positive and negative effect on the profit function and to detect significant changes in process phenomena. For example, it was found that high aeration levels towards the end of the fermentation process are significantly correlated to the high productivity. Further, it was also identified that the initiation of growth in the low productivity batches were delayed by several hours as compared to the high productivity batches. Both temperature inside the fermenter and the jacket temperature has a positive correlation with the productivity of the Pertactin antigen. However, the exact source of the variability in the productivity of the pertactin antigen was not identified since the variables used in the classification model are mere responses to the actual unmeasured disturbances. For example, it is believed that oxidative stress is a possible source of variability in the process since it has a significant effect on growth. However, the level of oxidative stress could not be inferred from the measured inputs. Hence, it is required to add more measurements such as NADPH level that is indicative of oxidative stress.

8.1.2 Statistical Process Control and Monitoring

In this part of the research, an explainability based fault detection and classification methodology is proposed using both a deep supervised autoencoder (DSAE) and dynamic supervised autoencoder (DDSAE) for the extraction of features. This problem is different from the first part of the research where an output (productivity) was utilized to differentiate the low productivity region from the higher productivity region whereas the fault

detection problem deals with differentiating abnormal evolution of input variables from normal operation and diagnosing the fault for the supervised learning models. The explainability measure serves two major objectives: i) Pruning of irrelevant input variables and further improvement in the fault detection accuracy and ii) Identification of possible root cause of different faults occurring in the process via contribution plots. The proposed methodology outperforms both multivariate linear methods and other DL based methods with a significant margin of 11% for fault detection and 10% for fault diagnosis as reported in the literature on the same standard data. Although this study make use of the powerful feature extraction capability of deep learning neural network models and XAI (eXplainable AI) concepts for deriving contribution plots, implementation to the industrial Sanofi process was not possible since the faults have not been clearly identified for this process. Also, it was not possible to observe the incipient faults using the explainability based fault detection and diagnosis methodology.

Thus, another method was devised to detect incipient faults effectively using a hierarchical structure as a way to increase the detection and classification of faults in the Tennessee Eastman Process (TEP). The hierarchical structure merges the detection and fault classification problem into one. An active fault detection approach was pursued where a hierarchical model structure combined with external PRBS signals was proposed that proved to be particularly effective for classifying incipient faults. It was also observed that LSTM-Hierarchical based model is superior than the traditional linear methods and other deep learning based methods for fault classification due to the ability of the LSTM based algorithm to capture process dynamics. It was also shown that the classification averages can be enhanced by extending the length of the time horizon of past data fed to the RNN based model. This methodology demonstrates the ability of detecting the incipient faults well on the Tennessee Eastman process simulator. However, actual implementation of external excitation for detecting incipient faults will result in additional variability. Hence, a more comprehensive study is required to select the amplitude of the excitation signal and also the time duration to cause minimal process disturbance.

Since both the methods described above utilize labeled data, they were not directly

applicable to the Sanofi vaccine manufacturing process since the faults for the latter process have not been clearly identified as yet. Thus, an unsupervised process monitoring algorithm was also developed. In this part of the research, 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. We demonstrated that the use of the proposed objective function provides a significant improvement in fault detection accuracy for both linear and nonlinear models but the improvement is larger with the nonlinear MPLS-AE method. A case study of industrial penicillin batch dataset suggests that the use of dynamic control limits along with the novel objective function result in significant improvements in detection for all methods. Implementation of this methodology to a real process such as the Sanofi vaccine process may be challenging. One of the major obstacles is the selection of normal operating profiles in the input variables as the productivity of the process that can serve to select these normal profiles is measured only once at the end of the fermentation.

8.1.3 Evaluating observability

This part of the research was motivated by the need to evaluate the implementation of new sensors in a process. In general, the adoption of new sensors in the pharmaceutical industry requires extensive validation and increases the maintenance costs. Thus, the goal was to develop an observability measure that will assess the contribution of a new sensor to the accuracy of a deep learning based model. A novel method was presented to compute a robust observability distance measure (RODM) and evaluate degree of classification observability based on noisy input data. The proposed method first computes a distance metric such that two clusters of points belonging to different classes should be at least distance d_{ij} ($i \neq j$) apart in the worst case-scenario where i and j are points corresponding to different labels in representation space for a good classification. The merit of the method is that it can be used to assess the observability of output classes from available input data that is corrupted by noise. Furthermore, it is shown that the observability and

classification accuracy can be enhanced by discarding variables that are not relevant for the classification task and contribute to overlap between different regions corresponding to output classes. It is argued that the proposed method to evaluate observability can be used in the future for selecting sensors to increase the observability of the classes. The proposed methodology was applied to the Sanofi Vaccine Process to evaluate the contribution of off-gas data analysis in order to differentiate the low productivity batches from the higher productivity batches. It was found that the additional data do not contribute towards the classification task but could be used to detect changes in calibration of the dissolved oxygen sensor.

8.2 Future Work

Following the conclusions of this research, this section outlines future work both in terms of development of new methodologies and their implementation in the case studies (Tennessee Eastman Process and Sanofi Pasteur Vaccine Manufacturing Process).

1. Since the sources of variability has not been identified with the developed explainable DNN model, it is important to perform more experiments with respect to different composition of media and study its effect to the productivity. By relating changes in raw materials to productivity it will be possible to diagnose these changes from the measurements following a supervised learning approach.
2. In view of the small sample size currently available at Sanofi Pasteur there is a motivation to generate synthetic data by using GANs (Generative Adversial Networks). MLP and LSTM regression network models should be re-trained to evaluate the effect of synthetic data created by GANs on test accuracy.
3. Develop framework for utilizing the data from different sources (NIR, flow cytometry, spectro-fluorescence etc.) to correlate with productivity values. For the purpose of utilizing and reconciling information from different sources, it is envisioned to use CNNs (Convolutional Neural Networks) for spectro-fluorescence data along with RNNs (Recurrent Neural Networks) for time-series data. From other parallel research

in our group we know that spectro-fluorescence can be used to measure NADPH levels that are correlated to oxidative stress, a suspected major source of variability in the process.

4. While the current research deals with the upstream process only, it is important to monitor also the downstream (purification) process to seek for additional sources in variability of the final antigen productivity (after purification). Sanofi regularly collects downstream data of antigen levels by Elisa and Kjeldahl methods, i.e. tests that are used to quantify productivity values. These data should be used separately or in combination with the upstream data to develop NNs based models.
5. While the current studies involving the Sanofi process have only considered the last two of the parallel trains of fermenters, it would be of interest to develop predictive models that include data from previous fermenters train. This may increase the predictability of antigen productivity from input data and may help to further explain the sources of variability.
6. The developed unsupervised process monitoring methodology should be applied to the Sanofi's Vaccine manufacturing process after careful selection of a region of normal operation. Since it is very difficult to measure antigens' levels at different times, the selection of normal operating profiles could be based on frequent measurements of biomass that are easier to obtain.
7. This thesis deals with empirical modeling using deep learning models. A hybrid modelling approach that combines mechanistic models and NNs may further improve the developed algorithms. For example, mechanistic models that contain a clear relation between changes in raw materials to growth and productivity may permit better diagnosis of the sources of variability in each fermentation.

References

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
- Piyush Agarwal and Hector Budman. Classification of profit-based operating regions for the tennessee eastman process using deep learning methods. *IFAC-PapersOnLine*, 52(1):556–561, 2019.
- Piyush Agarwal and Arun K Tangirala. Reconstruction of missing data in multivariate processes with applications to causality analysis. *International Journal of Advances in Engineering Sciences and Applied Mathematics*, 9(4):196–213, 2017a.
- Piyush Agarwal and Arun K Tangirala. Reconstruction of causal graphs for multivariate processes in the presence of missing data. In *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 0389–0394. IEEE, 2017b.
- 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*, 2019.
- 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, 2020. doi: 10.1021/acs.iecr.9b04737. URL <https://doi.org/10.1021/acs.iecr.9b04737>.

- 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. doi: <https://doi.org/10.1016/j.compchemeng.2021.107467>. URL <https://www.sciencedirect.com/science/article/pii/S0098135421002453>.
- 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.
- Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Explaining recurrent neural network predictions in sentiment analysis. *arXiv preprint arXiv:1706.07206*, 2017.
- Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.
- Andreas Bathelt, N Lawrence Ricker, and Mohieddine Jelali. Revision of the tennessee eastman process model. *IFAC-PapersOnLine*, 48(8):309–314, 2015.
- Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.
- Léon Bottou et al. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nimes*, 91(8):12, 1991.
- Hector Budman, Nilesh Patel, Melih Tamer, and Walid Al-Gherwi. A dynamic metabolic flux balance based model of fed-batch fermentation of bordetella pertussis. *Biotechnology progress*, 29(2):520–531, 2013.
- Regardt Busch and Iain K Peddle. Active fault detection for open loop stable lti siso systems. *International Journal of Control, Automation and Systems*, 12(2):324–332, 2014.

- 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.
- Gavneet Singh Chadha and Andreas Schwung. Comparison of deep neural network architectures for fault detection in tennessee eastman process. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2017.
- 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.
- Kumar Chellapilla, Sidd Puri, and Patrice Simard. High performance convolutional neural networks for document processing. 2006.
- 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.
- 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, 2020.
- ZhiQiang Chen, Chuan Li, and René-Vinicio Sanchez. Gearbox fault identification and classification with convolutional neural networks. *Shock and Vibration*, 2015, 2015.
- Feifan Cheng, Q Peter He, and Jinsong Zhao. A novel process monitoring approach based on variational recurrent autoencoder. *Computers & Chemical Engineering*, 129:106515, 2019.
- 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.

- 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.
- 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.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- 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.
- Sang Wook Choi, Julian Morris, and In-Beum Lee. Dynamic model-based batch process monitoring. *Chemical Engineering Science*, 63(3):622–636, 2008.
- François Chollet et al. Keras documentation. *Keras. io*, 2015.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- Maxwell D Collins and Pushmeet Kohli. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*, 2014.
- Jordi Cusidó, Luis Romeral, Juan Antonio Ortega, Antoni Garcia, and Jordi Riba. Signal injection as a fault detection technique. *Sensors*, 11(3):3356–3380, 2011.
- James J Downs and Ernest F Vogel. A plant-wide industrial process control problem. *Computers & chemical engineering*, 17(3):245–255, 1993.

- Yuncheng Du and Dongping Du. Fault detection using empirical mode decomposition based pca and cusum with application to the tennessee eastman process. *IFAC-PapersOnLine*, 51(18):488–493, 2018.
- John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.
- Baruch Epstein and Ron Meir. Generalization bounds for unsupervised and semi-supervised learning with autoencoders. *arXiv preprint arXiv:1902.01449*, 2019.
- Shriram Gajjar, Murat Kulaheci, and Ahmet Palazoglu. Least squares sparse principal component analysis and parallel coordinates for real-time process monitoring. *Industrial & Engineering Chemistry Research*, 59(35):15656–15670, 2020.
- Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.
- Meng Gan, Cong Wang, et al. Construction of hierarchical diagnosis network based on deep learning and its application in the fault pattern recognition of rolling element bearings. *Mechanical Systems and Signal Processing*, 72:92–104, 2016.
- 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.
- Winston Garcia-Gabin and Michael Lundh. Input prbs design for identification of multi-variable systems.
- Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

- Stephen Goldrick, Andrei Ștefan, David Lovett, Gary Montague, and Barry Lennox. The development of an industrial-scale fed-batch fermentation simulation. *Journal of biotechnology*, 193:70–82, 2015.
- 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.
- 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.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pages 243–254. IEEE, 2016.
- Jinane Harmouche, Claude Delpha, and Demba Diallo. Incipient fault detection and diagnosis based on kullback–leibler divergence using principal component analysis: Part i. *Signal Processing*, 94:278–287, 2014.
- Miao He and David He. Deep learning based approach for bearing fault diagnosis. *IEEE Transactions on Industry Applications*, 53(3):3057–3065, 2017.
- 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.

- Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.
- Tor Aksel N Heirung and Ali Mesbah. Input design for active fault diagnosis. *Annual Reviews in Control*, 47:35–50, 2019.
- Seongmin Heo and Jay H Lee. Fault detection and classification using artificial neural networks. *IFAC-PapersOnLine*, 51(18):470–475, 2018.
- Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.
- Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- John H Holland. Adaptation in natural and artificial systems, university of michigan press. *Ann arbor, MI*, 1(97):5, 1975.
- JC Hoskins, KM Kaliyur, and David M Himmelblau. Fault diagnosis in complex chemical plants using artificial neural networks. *AIChE Journal*, 37(1):137–141, 1991.
- Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.
- Chun-Chin Hsu, Mu-Chen Chen, and Long-Sheng Chen. A novel process monitoring approach with dynamic independent component analysis. *Control Engineering Practice*, 18(3):242–253, 2010.
- Rolf Isermann. *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media, 2005.

- J Edward Jackson and Govind S Mudholkar. Control procedures for residuals associated with principal component analysis. *Technometrics*, 21(3):341–349, 1979.
- Olivier Janssens, Viktor Slavkovikj, Bram Vervisch, Kurt Stockman, Mia Loccufer, Steven Verstockt, Rik Van de Walle, and Sofie Van Hoecke. Convolutional neural network based fault detection for rotating machinery. *Journal of Sound and Vibration*, 377:331–345, 2016.
- Feng Jia, Yaguo Lei, Jing Lin, Xin Zhou, and Na Lu. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing*, 72:303–315, 2016.
- 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.
- Luyang Jing, Ming Zhao, Pin Li, and Xiaoqiang Xu. A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox. *Measurement*, 111:1–10, 2017.
- N Kaistha and BR Upadhyaya. Incipient fault detection and isolation in a pwr plant using principal component analysis. In *Proceedings of the 2001 American Control Conference. (Cat. No. 01CH37148)*, volume 3, pages 2119–2120. IEEE, 2001.
- Manabu Kano, Shouhei Tanaka, Shinji Hasebe, Iori Hashimoto, and Hiromu Ohno. Monitoring independent components for fault detection. *AIChE Journal*, 49(4):969–976, 2003.
- Athanassios Kassidas, John F MacGregor, and Paul A Taylor. Synchronization of batch trajectories using dynamic time warping. *AIChE Journal*, 44(4):864–875, 1998.
- 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.

- 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.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- 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.
- Theodora Kourti and John F MacGregor. Multivariate spc methods for process and product monitoring. *Journal of quality technology*, 28(4):409–428, 1996.
- James V Kresta, John F Macgregor, and Thomas E Marlin. Multivariate statistical monitoring of process operating performance. *The Canadian journal of chemical engineering*, 69(1):35–47, 1991.
- Wenfu Ku, Robert H Storer, and Christos Georgakis. Disturbance detection and isolation by dynamic principal component analysis. *Chemometrics and intelligent laboratory systems*, 30(1):179–196, 1995.
- Abhijit Kulkarni, Vaidyanathan K Jayaraman, and Bhaskar D Kulkarni. Knowledge incorporated support vector machines to detect faults in tennessee eastman process. *Computers & chemical engineering*, 29(10):2128–2133, 2005.
- Saïd Ladjal, Alasdair Newson, and Chi-Hieu Pham. A pca-like autoencoder. *arXiv preprint arXiv:1904.01277*, 2019.
- Daniel Laky, Shu Xu, Jose S Rodriguez, Shankar Vaidyaraman, Salvador García Muñoz, and Carl Laird. An optimization-based framework to define the probabilistic design space of pharmaceutical processes with model uncertainty. *Processes*, 7(2):96, 2019.
- Truls Larsson, Kristin Hestetun, Espen Hovland, and Sigurd Skogestad. Self-optimizing control of a large-scale plant: The tennessee eastman process. *Industrial & engineering chemistry research*, 40(22):4889–4901, 2001.

- 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.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Hyun Jin Lee and Daniel E Rivera. *An integrated methodology for plant-friendly input signal design and control-relevant estimation of highly interactive processes*. American Institute of Chemical Engineers, 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, 2004a.
- Jong-Min Lee, ChangKyoo Yoo, and In-Beum Lee. Statistical process monitoring with independent component analysis. *Journal of process control*, 14(5):467–485, 2004b.
- 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.
- Gang Li, S Joe Qin, and Donghua Zhou. Geometric properties of partial least squares for process monitoring. *Automatica*, 46(1):204–210, 2010.
- Gang Li, Carlos F. Alcalá, S. Joe Qin, and Donghua Zhou. Generalized reconstruction-based contributions for output-relevant fault diagnosis with application to the tennessee eastman process. *IEEE Transactions on Control Systems Technology*, 19:1114–1127, 2011.
- Lucas Liebenwein, Cenk Baykal, Brandon Carter, David Gifford, and Daniela Rus. Lost in pruning: The effects of pruning neural networks beyond test accuracy. *Proceedings of Machine Learning and Systems*, 3, 2021.
- Lennart Ljung. System identification. *Wiley encyclopedia of electrical and electronics engineering*, pages 1–19, 1999.

- Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.
- Lin Luo, Lei Xie, and Hongye Su. Deep learning with tensor factorization layers for sequential fault diagnosis and industrial process monitoring. *IEEE Access*, 8:105494–105506, 2020.
- Feiya Lv, Chenglin Wen, Zejing Bao, and Meiqin Liu. Fault diagnosis based on deep learning. In *American Control Conference (ACC), 2016*, pages 6851–6856. IEEE, 2016.
- Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- 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.
- Sankar Mahadevan and Sirish L Shah. Fault detection and diagnosis in process data using one-class support vector machines. *Journal of process control*, 19(10):1627–1639, 2009.
- 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.
- Prashant Mhaskar, Adiwinata Gani, Nael H El-Farra, Charles McFall, Panagiotis D Christofides, and James F Davis. Integrated fault-detection and fault-tolerant control of process systems. *AIChE Journal*, 52(6):2129–2148, 2006.
- Marvin Minsky and Seymour A Papert. Artificial intelligence progress report. 1972.
- Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 193–209, 2019.
- Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

- Viet Ha Nguyen and Jean-Claude Golinval. Fault detection based on kernel principal component analysis. *Engineering Structures*, 32(11):3683–3691, 2010.
- Niels-Peter Vest Nielsen, Jens Michael Carstensen, and Jørn Smedsgaard. Aligning of single and multiple wavelength chromatographic profiles for chemometric data analysis using correlation optimised warping. *Journal of chromatography A*, 805(1-2):17–35, 1998.
- Paul Nomikos and John F MacGregor. Multivariate spc charts for monitoring batch processes. *Technometrics*, 37(1):41–59, 1995.
- Pabara-Ebiere Patricia Odiwei and Yi Cao. Nonlinear dynamic process monitoring using canonical variate analysis and kernel density estimations. *IEEE Transactions on Industrial Informatics*, 6(1):36–45, 2009.
- 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.
- 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.
- Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962. doi: 10.1214/aoms/1177704472.
- Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.
- Lauréline Perotin, Romain Serizel, Emmanuel Vincent, and Alexandre Guérin. Crnn-based multiple doa estimation using acoustic intensity features for ambisonics recordings. *IEEE Journal of Selected Topics in Signal Processing*, 13(1):22–33, 2019.
- Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

- Christopher Poultney, Sumit Chopra, and Yann L Cun. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2007.
- Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.
- Tiago J Rato and Marco S Reis. Fault detection in the tennessee eastman benchmark process using dynamic principal components analysis based on decorrelated residuals (dpca-dr). *Chemometrics and Intelligent Laboratory Systems*, 125:101–108, 2013.
- 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.
- Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.
- Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.
- N Lawrence Ricker. Decentralized control of the tennessee eastman challenge process. *Journal of Process Control*, 6(4):205–221, 1996.
- NL Ricker. Optimal steady-state operation of the tennessee eastman challenge process. *Computers & chemical engineering*, 19(9):949–959, 1995.
- Philippe Rigollet. Generalization error bounds in semi-supervised classification under the cluster assumption. *Journal of Machine Learning Research*, 8(Jul):1369–1392, 2007.
- Andre Rios, Vaibhav Gala, Susan Mckeever, et al. Explaining deep learning models for structured data using layer-wise relevance propagation. *arXiv preprint arXiv:2011.13429*, 2020.
- Daniel E Rivera and Sujit V Gaikwad. Systematic techniques for determining modelling requirements for siso and mimo feedback control. *Journal of Process Control*, 5(4): 213–224, 1995.

- 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.
- Roman Rosipal, Leonard J Trejo, and Bryan Matthews. Kernel pls-svc for linear and nonlinear classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 640–647, 2003.
- David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- Matthias Seeger. Learning with labeled and unlabeled data (technical report). *Edinburgh University*, 2001.
- Hadi Shahnazari. Fault diagnosis of nonlinear systems using recurrent neural networks. *Chemical Engineering Research and Design*, 153:233–245, 2020.
- Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International conference on machine learning*, pages 71–79. PMLR, 2013.
- M Bin Shams, H Budman, and T Duever. Finding a trade-off between observability and economics in the fault detection of chemical processes. *Computers & chemical engineering*, 35(2):319–328, 2011a.
- MA Bin Shams, HM Budman, and TA Duever. Fault detection, identification and diagnosis using cusum based pca. *Chemical Engineering Science*, 66(20):4488–4498, 2011b.
- 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.
- Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.

- BW Silverman. Density estimation for statistics and data analysis, chapman and hall, london, 1986. *Crossref*, 1986.
- G. Singh Chadha, M. Krishnamoorthy, and A. 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, 2019. doi: 10.1109/IECON.2019.8926924.
- Plakias Spyridon and Yiannis S Boutalis. Generative adversarial networks for unsupervised fault detection. In *2018 European Control Conference (ECC)*, pages 691–696. IEEE, 2018.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- Irene Sturm, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Interpretable deep neural networks for single-trial eeg classification. *Journal of neuroscience methods*, 274:141–145, 2016.
- Wenjun Sun, Siyu Shao, Rui Zhao, Ruqiang Yan, Xingwu Zhang, and Xuefeng Chen. A sparse auto-encoder-based deep neural network approach for induction motor faults classification. *Measurement*, 89:171–178, 2016.
- Gerald Tesauro. Practical issues in temporal difference learning. *Machine learning*, 8(3): 257–277, 1992.
- 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.
- Aditya Tulsyan, Christopher Garvin, and Cenk Undey. Industrial batch process monitoring with limited data. *Journal of Process Control*, 77:114–133, 2019.

- Cenk Ündey, Bruce A Williams, and Ali Çınar. Monitoring of batch pharmaceutical fermentations: Data synchronization, landmark alignment, and real-time monitoring. *IFAC Proceedings Volumes*, 35(1):271–276, 2002.
- 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.
- Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.
- Venkat Venkatasubramanian and King Chan. A neural network methodology for process fault diagnosis. *AIChE Journal*, 35(12):1993–2002, 1989.
- Huan-gang Wang, Xin Li, and Tao Zhang. Generative adversarial network based novelty detection using minimized reconstruction error. *Frontiers of Information Technology & Electronic Engineering*, 19(1):116–125, 2018.
- 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.
- 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.
- Barry M Wise, NL Ricker, DF Veltkamp, and Bruce 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.
- Svante Wold, Paul Geladi, Kim Esbensen, and Jerker Öhman. Multi-way principal components-and pls-analysis. *Journal of chemometrics*, 1(1):41–56, 1987.
- Hao Wu and Jinsong Zhao. Deep convolutional neural network model based chemical process fault diagnosis. *Computers & chemical engineering*, 115:185–197, 2018.

- Lijun Wu, Xiaogang Chen, Yi Peng, Qixiang Ye, and Jianbin Jiao. Fault detection and diagnosis based on sparse representation classification (src). In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 926–931. IEEE, 2012.
- D. Xie and L. Bai. A hierarchical deep neural network for fault diagnosis on tennessee-eastman process. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 745–748, 2015. doi: 10.1109/ICMLA.2015.208.
- Danfeng Xie and Li Bai. A hierarchical deep neural network for fault diagnosis on tennessee-eastman process. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 745–748. IEEE, 2015.
- Weiwei 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.
- Yinchong Yang, Volker Tresp, Marius Wunderle, and Peter A Fasching. Explaining therapy predictions with layer-wise relevance propagation in neural networks. In *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 152–162. IEEE, 2018.
- Daniel S Yeung and Xuequan Sun. Using function approximation to analyze the sensitivity of mlp with antisymmetric squashing activation function. *IEEE Transactions on Neural Networks*, 13(1):34–44, 2002.
- Chunyang 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.
- Shen Yin, Steven X Ding, Adel Haghani, Haiyang Hao, and Ping Zhang. A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark tennessee eastman process. *Journal of process control*, 22(9):1567–1581, 2012.
- Shen Yin, Steven X Ding, Xiaochen Xie, and Hao Luo. A review on basic data-driven approaches for industrial process monitoring. *IEEE Transactions on Industrial Electronics*, 61(11):6418–6428, 2014a.

- Shen Yin, Xin Gao, Hamid Reza Karimi, and Xiangping Zhu. Study on support vector machine-based fault detection in tennessee eastman process. In *Abstract and Applied Analysis*, volume 2014. Hindawi, 2014b.
- Jie Yu and S Joe Qin. Multimode process monitoring with bayesian inference-based finite gaussian mixture models. *AIChE Journal*, 54(7):1811–1829, 2008.
- 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.
- Xiaofeng Yuan, Jiao Zhou, Yalin Wang, and Chunhua Yang. Multi-similarity measurement driven ensemble just-in-time learning for soft sensing of industrial processes. *Journal of Chemometrics*, 32(9):e3040, 2018.
- Vanessa Zavatti, Hector Budman, Raymond Legge, and Melih Tamer. Monitoring of an antigen manufacturing process. *Bioprocess and biosystems engineering*, 39(6):855–869, 2016.
- Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
- 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, 2019a.
- Xu Zhang, Yuanyuan Zou, Shaoyuan Li, and Shenghu Xu. A weighted auto regressive lstm based approach for chemical processes modeling. *Neurocomputing*, 367:64–74, 2019b.
- Yingwei Zhang. Enhanced statistical analysis of nonlinear processes using kpca, kica and svm. *Chemical Engineering Science*, 64(5):801–811, 2009.
- Haitao Zhao, Shaoyuan Sun, and Bo Jin. Sequential fault diagnosis based on lstm neural network. *IEEE Access*, 6:12929–12939, 2018a.

Hongshan Zhao, Huihai Liu, Wenjing Hu, and Xihui Yan. Anomaly detection and fault analysis of wind turbine components based on deep learning network. *Renewable energy*, 127:825–834, 2018b.

Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature methods*, 12(10):931–934, 2015.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

Qiuyu Zhu and Ruixin Zhang. A classification supervised auto-encoder based on predefined evenly-distributed class centroids. *arXiv preprint arXiv:1902.00220*, 2019.