PROCESS MONITORING AND FAULT DIAGNOSIS

A project report submitted in partial fulfillment of requirement for the degree of

BE

In
Chemical Science and Engineering

By

Name: Aashish Acharya	Roll no: 01
Name: Ritesh Dahal	Roll no: 07
Name: Aakarshan Khanal	Roll no: 12
Name: Niraj Neupane	Roll no: 18
Name: Aayush Parajuli	Roll no: 21
Name: Prajwol Poudel	Roll no: 22

DEPARTMENT OF CHEMICAL SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING

KATHMANDU UNIVERSITY

MAY 2022

BONAFIDE CERTIFICATE

This is to certify that the project titled **PROCESS MONITORING AND FAULT DIAGNOSIS** is a bonafide record of the work done by

Name: Aashish Acharya	Roll No: 1
Name: Ritesh Dahal	Roll No: 7
Name: Aakarshan Khanal	Roll No: 12
Name: Niraj Neupane	Roll No: 18
Name: Aayush Parajuli	Roll No: 21
Name: Prajwol Poudel	Roll No: 22

In partial fulfillment of the requirements for the award of the degree of **Bachelor in Engineering in Chemical Science and Engineering** of the **Kathmandu University, Dhulikhel** during the year 2022.

Mr. Harish Chandra Bhandari

Project Supervisor

Department of Mathematics

Dr. Bibek Uprety

Program Coordinator

Department of Chemical Science and Engineering

Project Viva-voice held on 6th March 2022

Internal Examiner

External Examiner

ABSTRACT

Process Monitoring is defined as an activity that ensures a process is steady, predictable and constantly operating at the normal level with less variation. A fault refers to any abnormal operation that is of significant concern, regardless of whether the abnormality was due to faulty equipment or is an extreme disturbance. Process Monitoring consists of three parts: Fault detection, fault diagnosis and process recovery. Process Monitoring methods are generally classified into three categories: data driven, analytical and knowledge based. Data driven methods are widely applied in industries because of their simplicity and efficiency compared to other methods. Fault diagnosis of chemical process data becomes one of the important practices in modern industries and chemical process companies. Conventional fault diagnosis and classification methods firstly extract important features from raw data and its diagnosis is done by a certain classifier, but they lack the adaptive processing of dynamic information in raw data. This project is a fault diagnosis method based on a long short-term memory (LSTM) neural network. Long short-term memory (LSTM) is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs. In LSTM firstly, raw process data is used to train the LSTM neural network until the cost function of LSTM converges below a predefined small positive value and dynamic information of raw process data is learned by LSTM. Testing data are used to obtain the diagnosis results of the trained LSTM neural network. The application of LSTM to fault identification and analysis is evaluated in the Tennessee Eastman benchmark process (TEP). Comparison of results shows that LSTM can better separate different faults and provide fault diagnosis performance.

Keywords: abnormality, diagnosis, classifier, recurrent, identification

ACKNOWLDEGEMENT

We are extremely thankful to Dr. Bibek Upreti, program coordinator of the Department of Chemical Science and Engineering who provided us the possibility to complete this project.

A special gratitude we give to our project Coordinator and Supervisor, Mr Harish Chandra Bhandari, Department of Mathematics for his guidance and constant supervision as well as for providing necessary information regarding the project & also for their guidance and support in completing this project.

A special thanks to all the people who have been directly or indirectly linked with this project.

Table of Contents

List of Tables

List of Figures

List of Symbols, Abbreviations and Nomenclature

FDD	Fault detection and diagnosis
NOC	Normal Operating Condition
TEP	Tennessee Eastman Process
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
LSTM	Long-Short term memory
ROC curve	Receiver operating characteristic curve
AUC	Area under the ROC curve

CHAPTER 1: INTRODUCTION

1.1 Background

With the progress in the industries, industries processes are getting smarter. In particular, many modernized industrial processes are equipped with the high end sensors to gather process-related data for discovering faults existing or arising in process as well as monitoring the process status. With the change of industry with full-automation of equipment and process, more caution supervision that includes process control and suitable corrective actions is required to guarantee the process efficiency. With the advent of industry, current industrial processes are transforming into smart ones. In particular, many modernized industrial processes are equipped with several well-elaborated sensors to gather process-related data for discovering faults existing or arising in processes as well as monitoring the process status. For this change of industrial environments with the full-automation of equipment and process, more cautious supervision that includes process control and suitable corrective actions is required to guarantee the process efficiency. It is an important task to maintain a desirable performance in industrial processes which commonly hold several kinds of faults. Among various process supervision techniques, the fault detection and diagnosis (FDD) is a significantly critical control method for accomplishing this task because most industries hope to improve their process performance through a higher level of FDD capability. The basic functions of FDD can be summed up into two parts, namely (1) monitoring the behavior of processes (variables) and (2) revealing the fault presence, its characteristics, and root causes of faults. Thus, in order to maintain high process yield and throughput in industrial processes, it is necessary to adopt fast, accurate, and effective detection and diagnosis tools for process or equipment faults that may degrade the performance of the entire system.

Process monitoring is defined as an activity that ensures a process is steady, predictable, and constantly operating at the set level of performance with the help of normal variation. In simple terms, it is described as a method for improving and controlling a process with the help of statistical analysis.

Process monitoring is an important tool that is implemented in several industries like power generation plants, chemical processing, food and beverage industry, paper manufacturing, and oil refining. This production process is used to achieve a safe, economical, and consistent production level that is not possible just with manual control by humans.

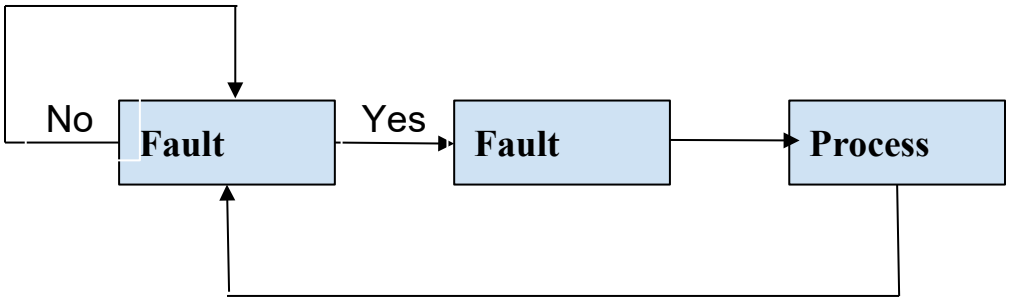


Figure 1: block diagram of process monitoring and diagnosis

A fault refers to any abnormal operation that is of significant concern, regardless of whether the abnormality was due to faulty equipment or is an extreme disturbance, e.g., from switching raw material from different companies for a process that is sensitive to variation in contaminant profiles, or flooding in a distillation column caused by a sudden thunderstorm raining on the external surface of the column. Process monitoring consists of three parts:

- Fault detection is the determination of whether a fault has occurred.
- fault diagnosis identifies the observation variables and determines which fault has occurred.
- Process recovery removes the effect the fault

1.2 Fault detection

Process monitoring methods are generally classified into three categories: data driven, analytical, and knowledge-based.

1.2.1 Data-driven methods

Data-driven methods are widely applied in industries because of their simplicity and efficiency compared to the other methods. Process monitoring systems based on data-driven techniques handle large quantities of data by transforming the high-dimensional data into a lower dimensional space that captures the most important information. Data-driven analytics techniques do not require such expert knowledge about the process, rather leverage on a large amount of data which is collected nowadays in production systems. Furthermore, with the advent of Big Data technologies for storing and analyzing large amounts of digital data, the computation obstacles are also circumvented. The data-driven techniques for fault detection focus on analyzing historical data from the system to

understand the underlying relationship between the different process parameters and based on this understanding, predict current or future occurrence of a fault.

1.2.2 Analytical based methods

There are different model-based fault detection methods. Fault detection using input and output measurements of the system is the basic one among them. In some cases, we measure only the output signal, in such a case model based methods such as spectral analysis and band-pass filters are used for fault detection. Among these methods, the most frequently used techniques are parameter estimation and observer based methods. Fault detection using these methods is done by comparing the system‘s measured variables with the information obtained from the system‘s mathematical model.

1.2.3 Knowledge-Based Methods

Fault detection using knowledge based methods is a heuristic process. System characteristic values like amplitude, variance, state variables, model parameters and vibrations are used to extract features during normal and faulty conditions by using heuristic and analytical knowledge. After extracting the features under both the conditions (faulty and normal), they are then compared and methods of change detection are applied. Artificial neural networks, fuzzy logic and neuro-fuzzy based can be regarded as knowledge-based methods.

1.3 PCA based control charts

One of the most studied techniques for detecting faults has been the use of multivariate control charts based on PCA(Principal Component Analysis)[4,8]. In this type of control charts the process monitoring scheme is built by firstly reducing the initial high dimensional data into a lower dimensional representation using dimensional reduction techniques known as PCA. Which reduces the dimension to the first r dimension with the maximum variance. The T^2 statistic which measures the Mahalanobis distance of the principal scores to the historical mean and the Squared Prediction Error(SPE) which measures the Euclidean norm of the residuals are used to monitor the variations in the two subspaces, respectively. The control limits are set accordingly for both of these measurements.

This method creates two different control charts for the process commonly known as T^2 control chart and SPE control chart. Below is the code for creating these control charts for the TEP dataset.

https://github.com/phantom-balance/TEP/blob/master/TEP_PCA.ipynb

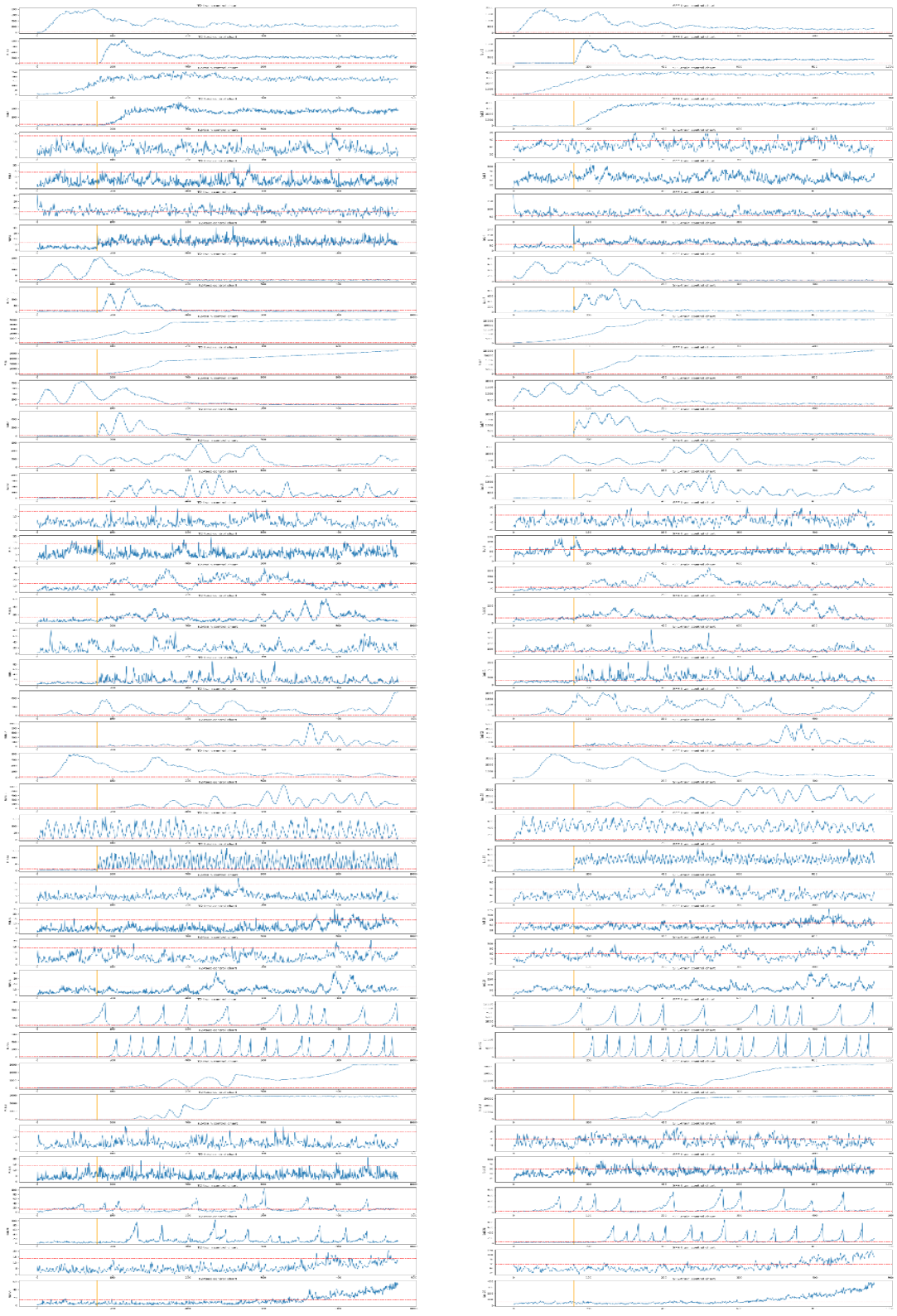


Figure 2: PCA based control chart

Recently, however, a trend in the deep learning community has emerged towards an end-to-end manner, which combines feature extraction and classifier design into one neural network. The motivation behind this idea is that the neural network automatically learns both the features of raw data and the classifier which better suit the fault diagnosis task and hence lead to improved performance. In order to deal with the dynamic information of time series, Recurrent neural network (RNN) architectures such as long short-term memory (LSTM) neural network and variants have exhibited state-of-the-art performance on a wide range of complicated sequential problems including signal processing, speech classification and video captioning. LSTM can adaptively learn the dynamic information of time sequences by non-linear gating units regulating the information into and out of the memory cells of LSTM.

1.3 Motivation

A small fault in an industrial system can propagate into a large and disastrous accident, for example, Bhopal gas tragedy in 1984, the chernobyl disaster in 1986 and many more. These catastrophic incidents provide a significant incentive for designing and implementing process monitoring systems. Apart from increasing process safety, process monitoring systems can also improve process efficiency, flexibility, reliability, and product quality. As in Nepal many industries are growing, implementation of process monitoring systems helps the industries to study what is process monitoring and diagnosis.

1.4 Objectives

- To study process monitoring techniques.
- Detection of fault present in the Tennessee Eastman process datasets of industrial process
- To study about the importance of data collection and analysis of the industrial process
- To early prediction about the efficiency of the industrial process .
- To study about the faults which may occur and how those faults can be diagnosed in an early stage using different machine learning and deep learning techniques.

1.5 Scope

The main purpose of the project is to study about the different types of processes, faults occurring during the operating conditions, making the industries run with less faults and with the maximum product yields. In Process monitoring there are many ways so, here we discussed briefly about all of them and with special attention to the data driven model.

1.6 Limitations

As everything has its pros and cons, with process monitoring there comes the limitations which are of different types, the limitation of process monitoring varies with the fault types, process type and for what type of industrial process we are doing process monitoring. We have three approaches for process monitoring: data-driven, analytical, and knowledge-based. The main drawback of data-driven measures is that their proficiency is highly dependent on the quantity and quality of the process data. For the small process whose dynamical models are relatively easy to model and understand, analytical, and knowledge are best suited. For the large industries with complex dynamics which are significantly harder to model, data-driven methods are more suitable. For handling Multivariate data, a Multivariate statistical method is introduced which is Principal Component Analysis (PCA). And other newly proposed approaches for handling multivariate data are based on machine learning and deep learning methods which can handle large and complex datasets.

CHAPTER 2: METHODOLOGY

2.1 Recurrent Neural Network (RNN)

RNNs are an efficient tool used to deal with complex and nonlinear dependencies in multivariate times series data. Generally speaking, RNNs are able to review information at each time point and choose the pertinent information to appropriately generate the outputs. RNNs can be trained to retain information in the long term through discovering features and modeling sequential (time) dependencies from the training dataset. The efficiency of RNNs has been proven over the last decade through several applications involving sequential or temporal data. This success can be attributed to their ability to extract complex nonlinearity between data points in the training dataset and project them onto a new feature space. RNNs have been widely exploited in speech recognition, natural language processing, and machine translation.

The so-called simple RNNs, or vanilla RNNs, with a single hidden layer comprise a memory **h** that permits summarizing the past in order to predict the future. Generally speaking, RNNs predict the output O_t by using the input vector, x_t , and the memory state, h_t The memory state h of VRNNs is updated with the recurrence formula

$$h_t = \sigma\,(\mathrm{V}\, h_{t-1} + \mathrm{U}\, x_t), \tag{2.1}$$

where σ is the activation function, and matrices V and U are trained to properly update the history vector h. Then, the predicted output O_t can be obtained by

$$O_t = \mathrm{W}\, h_t\ , \tag{2.2}$$

where the weight matrix W is trained to use the history h and predict the next output. The weights matrices, V, U, and W, represent the internal parameters of the RNN. The dynamic behavior and prediction of the RNN changes by updating these internal parameters, which can be computed by backpropagation. RNNs uncover and learn temporal dependencies in time series data by using the former output as inputs in addition to the actual input and recent past inputs to generate new outputs, as shown in Fig. 2.1. The architecture of the RNN model was designed using nonlinear stacked units, where links between units form a directed cycle (Fig. 2.1).

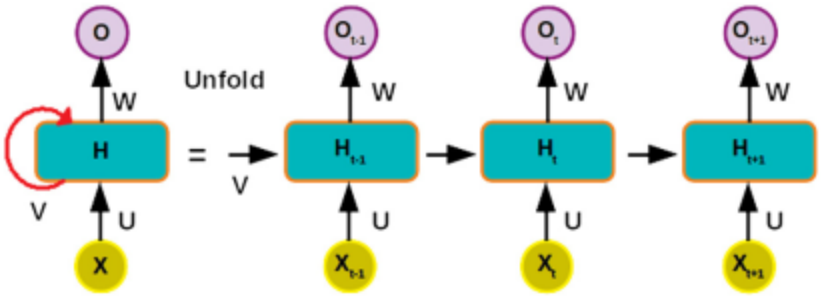


Figure 3: Basic illustrative representation of RNN

Note that RNNs are very deep in time when unrolled. The architecture of RNNs improves feature extraction and discovery of long-term dependencies from sequential time-series data. RNN is designed to model time varying or sequential patterns of input, where the input can have fixed or variable size, while the size of the RNN output is fixed. Of course, RNNs are characterized by feedback. RNN topology is represented as closed loop connections, equipped with a memory that captures and stores the information processed so far. The ability of multiple mapping schemes is not supported in the traditional neural network based on the feed-forward mechanism; this kind of network architecture supports only a fixed input and output size. Another desirable property of RNNs is their capability to handle input variables with various sizes, which makes them very useful for operating over sequences of vectors. In other words, RNNs are able to map input sequences to produce output sequences, where the length or size of the inputs depends on data nature and structure (Fig. 2.2). Below are a few examples to clarify this more concretely.

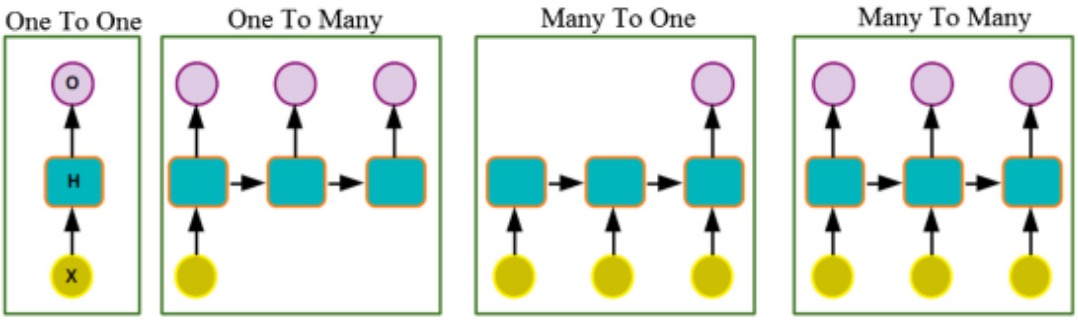


Figure 4: Different configuration of RNN model

Single to multiple:
In the case of image captioning, the network takes the input pixels of an image and generates a sequence of words.

Multiple to single:
Such a situation appears in sentiment analysis, where the input is a sequence of words, while for the output the network generates a single Boolean classification of true or false (i.e., positive or negative sentiments). Another example is in voice classification which is based on a sequence of voice records, and the network identifies the speaker.

Multiple to Multiple (a):
This happens in the case of language translation, for instance, from English to Arabic.

Multiple to Multiple (b):
This case is of video captioning and video classification.

RNNs have been widely exploited to capture relevant dependencies in time series. However, in training deep neural networks, RNNs can face two common problems of vanishing and exploding gradients, where the error gradients can increase to explosion or decrease to vanishing (close to zero). Indeed, the error gradients are employed for updating weights. The problem becomes more severe when increasing the depth of the network. Several techniques have been developed to alleviate the problem of exploding gradients: changing the network design to make it shallower, using a gradient clipping approach that aims to threshold the error gradients, and using weight regularization to reduce the severity through L_1 or L_2 penalties on the recurrent weights.

2.2 Long Short-term Memory (LSTM)

Machine learning has been researched extensively over the past three decades. Conventional neural networks are one such intensively used machine learning approach. As mentioned above, the main characteristics underlying these networks are the presence of full connections between adjacent layers and the absence of connections between the nodes within the same layer. Thus, this type of network is suited to handle sequential data and describe temporal dependencies in the data because it considers only the current measurement and without memorizing past measurements.
As discussed above, to overcome this problem, the internal memory of an RNN has been used to handle sequential data by considering the actual and previously received measurements. In other words, the hidden units in an RNN receive feedback from the previous state to the current state. Because the depth of the RNN is the time span,

information can be lost through time and the error can propagate back. Accordingly, the accuracy of the RNN can be degraded when the time span becomes longer due to the vanishing gradient and exploding gradient problems. To address this issue, long short-term memory (LSTM), which is an extended version of RNN, was first introduced by Hochreiter and Schmidhuber in 1997. LSTM models have been largely utilized in many applications, including handwriting recognition, language modeling and translation, acoustic modeling of speech, speech synthesis, protein structure prediction, and analysis of audio and video data. Moreover, in the modern LSTM architecture, there are peephole connections between internal cells and the gates in the same cell for learning the accurate timing of the outputs. This section first provides an overview of LSTMs and then discusses how they can be used for modeling and process monitoring. We also describe gated recurrent units (GRU), another improved version of RNNs.

The LSTMs are designed as an extension of simple RNNs to solve the vanishing gradient problem by explicitly incorporating a memory unit into the network. They are based on memories and gates making them suitable for learning long-term dependencies. Fig. 2.3 displays a schematic representation of an LSTM.

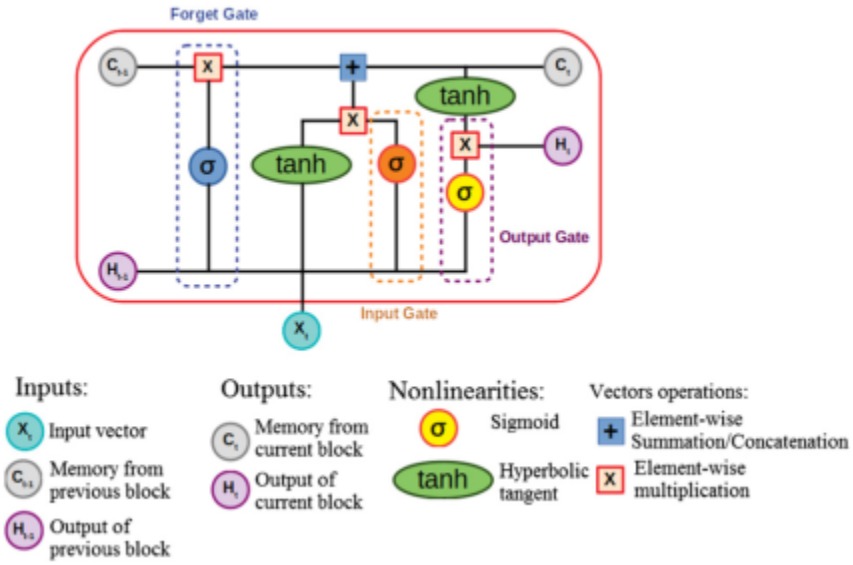


Figure 5: Basic illustrative representation of LSTM

The principal component of LSTM resides in its cell state, which is the horizontal chain shown in the top of the graph (Fig. 2.4). In LSTM, information can be removed or added to the cell by using structures called gates.

We designed a common LSTM model as a concatenation of several cell units instead of conventional neural network layers. We now briefly investigate the properties of the LSTM unit. As shown in Fig. 2.3, an LSTM cell comprises three inputs: X_t is the input observation at the current time point, h_{t-1} represents the output generated from the preceding LSTM cell, and C_{t-1}denotes the memory of the preceding cell. Also, each LSTM cell contains two outputs,h_t and c_t, which are the output of the actual network and the memory of the current unit, respectively.It is composed of three kinds of gates, namely the input gate, the forget gate, and the output gate (Fig. 2.3).Each gate comprises a sigmoid neural net layer and a pointwise multiplication operation. LSTMs can be seen when unfolded as a chain-like structure. In other words, it is a loop repeating module with a different structure (Fig. 2.5). Depending on the input and the internal feedback (memory state), the output will be generated in a special manner based on gates and four embedded layers. The layers here have activation units based on sigmoid and tanh.

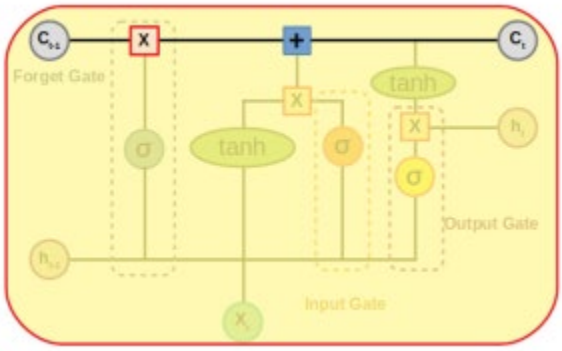


Figure 6: Cell state in LSTM model

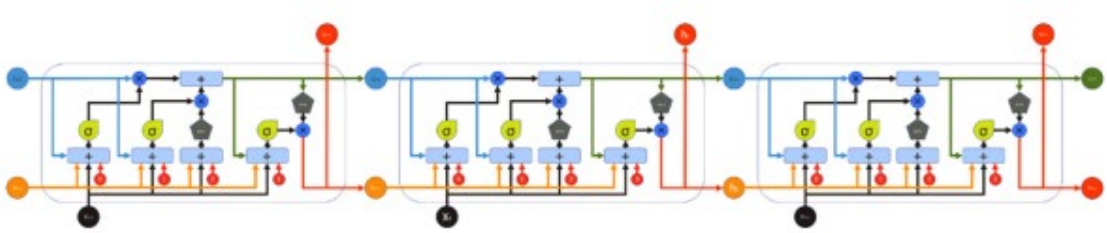


Figure 7: Schematic representation of an LSTM when unrolled in time.

This LSTM unit provides a decision based on the actual input, precedent output and memory, and then produces a new output and upgrades its memory.

Input gate:
The key role of the input gate is to control the flow of input activation into the memory cell. Here, the sigmoid layer controls the flow by generating an output between 0 and 1 into the memory cell.

Output gate:
The aim of the output gate is to control the output flow generated from cell activation to be injected into the network. This gate has the sigmoid layer that controls how much memory should be fed into the next LSTM unit.

Forget gate:
The sigmoid layer output removes information from the cell state that is no longer required (when the output is 0, otherwise it will keep it). In other words, the forget gate overcomes the weakness of LSTM models by preventing them from processing continuous input streams. By scaling the content of block memory, it forgets the cell's memory content in an adaptive way.

2.2.1 LSTM implementation steps

The input of each LSTM cycle is $(t-1)^{th}$ memory state C_{t-1} and hidden layer units h_{t-1}(output), but for the first cycle we start with zero or randomized values. The main steps when implementing LSTM are described below.
The first step of LSTM consists of deciding which information is to be forgotten or retained in the particular instant. This step is particularly useful when the past information is no longer useful for the actual cycle which is mainly related to the current input.
In other words, the aim of this step is to reveal the information that is not needed and can be omitted from the cell state. This is achieved by the sigmoid function. It looks at the past state (h_{t-1})and the actual input x_t and calculates the function accordingly (Fig. 2.6).

$$f_t = \sigma(x_t U^f + h_{t-1} W^f), \tag{2.3}$$

where W^f is weight and h_{t-1} is the output from the previous time.

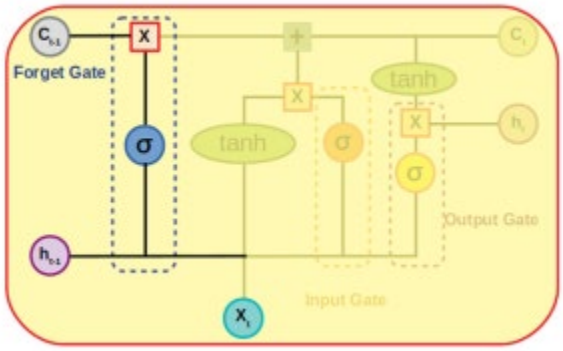


Figure 8: First step in LSTM, to decide what information is to be forgotten or retained.

The second step in the LSTM consists of updating the content of the memory cell. This step selects the new information that will be stored in the cell state (Fig. 2.7). The second layer (input gate) contains two parts, namely the sigmoid function and the tanh. The sigmoid layer chooses which values are to be updated. In the case of 1, the value of the input gate is unchanged, and in the case of 0 it is dropped. Next, a tanh layer creates a vector of new candidate values that can be added to the state.

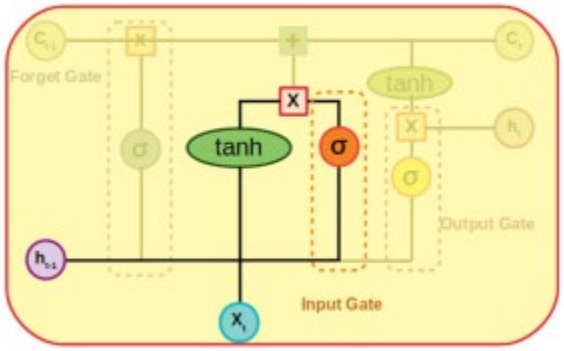


Figure 9: Second step in LSTM: updating the content of the memory cell.

It provides weight to the selected values for their level of importance (−1 to 1). After that, the two are combined to update the state. The Ĉ is combined with C_{t-1} to update both the memory state C_t . The H_t (output) is computed according to the output gate sigmoid and the tanh of C_t :

$$i_t = \sigma(x_t U^i + h_{t-1} W^i) , \quad (2.4)$$

$$\tilde{C}_t = \tanh(x_t U^g + h_{t-1} W^g), \quad (2.5)$$

C̃t is the candidate memory cell, W^i, W^g are weight parameters.

In this step, the old cell state, C_{t-1} , is actualized with the new cell state,C_t , and the past cell state,C_{t-1} , is actualized with the new cell state, C_t (Fig. 2.8). To do so, the old state is multiplied by f_t to forget the irrelevant information, and the candidate, $\tilde{C}_t$, is added. This represents the new candidate values scaled by how much we choose to update every state value:

$$C_t = f_t . C_{t-1} + i_t \cdot \tilde{C}_t . \quad (2.6)$$

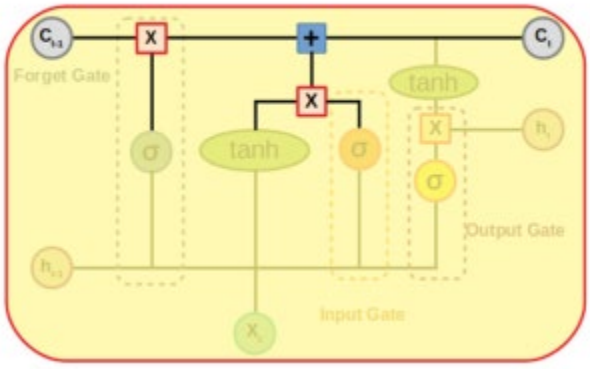


Figure 10: Step 3 in LSTM modeling.

Finally, the output is computed in two steps (Fig. 2.9). First, a sigmoid layer is utilized to select the relevant portions of the cell state to be transmitted to the output (Eq. (2.7)):

$$o_t = \sigma(x_t U^0 + h_{t-1} W^0) \quad (2.7)$$

.

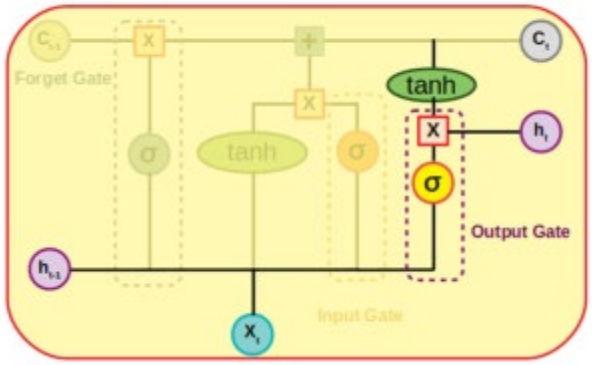


Figure 11: Step 4 in LSTM modeling.

The cell state is then passed via tanh (for normalizing values within the range (−1 and 1) and multiplied by the output of the sigmoid gate; thus we keep only the portions we selected to output (Eq. (2.8)):

$$h_t = tanh(C_t).o_t. \qquad (2.8)$$

At the end of the cycle, the Ht hidden layer units which represent the output of the cycle and the Ct memory state are ready for the next cycle usage. In summary, in the LSTM model, the three gates are trained for learning what information can be maintained in the memory, how long it can be stored, and when it can be read out. Combining several memory cells into blocks permits
them to share the same gates, and hence the number of adaptive parameters is reduced.

Stacked LSTM

Stacked LSTM, which is a deep neural network with multiple hidden layers, have been shown to perform well in highly non-linear sequential data. The structure of stacked LSTM is shown in Fig 2.10. Stacked LSTM consists of multiple LSTM layers and each LSTM layer contains multiple connected LSTM cells. The hidden state of the first LSTM layer is the input to the second LSTM layer and so on. Therefore, the hidden state in one LSTM layer is both propagated through time and passed to the next layer compared with vanilla LSTM.

By Stacking LSTM layers, the neural network becomes deeper, and is generally better at learning sequential data. We can use a stacked LSTM network as the fault diagnosis model by building a classification model.

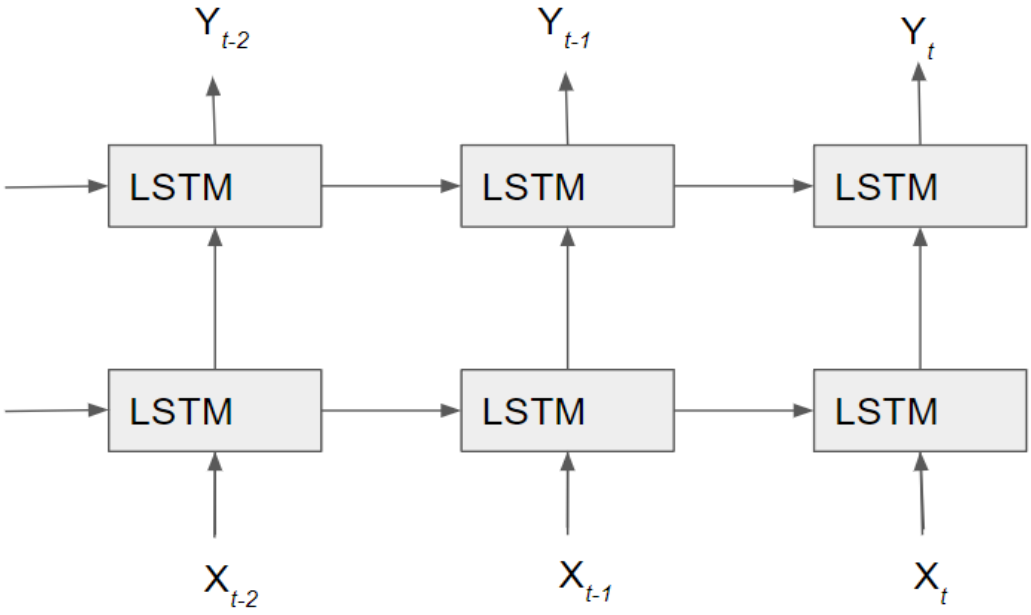


Figure 12: Many-to-many Stacked-LSTM model.

LOSS Function

Entropy is the number of bits required to transmit a randomly selected event from a probability distribution. A skewed distribution has a low entropy, whereas a distribution where events have equal probability has a larger entropy.There is binary cross entropy loss and multi-class cross entropy loss. Cross-entropy builds upon the idea of entropy from information theory and calculates the number of bits required to represent or transmit an average event from one distribution compared to another distribution.Cross-entropy loss is used when adjusting model weights during training. The aim is to minimize the loss, i.e, the smaller the loss the better the model. A perfect model has a cross-entropy loss of 0. Normally it serves for multi-class and multi-label classifications.

Optimization

It is done through gradient descent Adam optimizer. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iteratively based on training data. Adam is different to classical stochastic gradient descent. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training. A learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds.

2.3 Benchmark Dataset: Tennessee Eastman Process (TEP)

In order to demonstrate and illustrate the practical applicability of the process monitoring/Fault detection and diagnosis methods, various benchmark datasets and simulators of various industrial processes have been made for the study of performance of process monitoring and control strategies and are available online [1,2,3]. Among them, TEP is a very widely used simulation based on a real chemical process and has become an important benchmark in evaluation of various process monitoring and FDD strategies and is still used for new emerging models. [4,12,13].

The TEP is a benchmark created by the Eastman Chemical Company to provide a realistic industrial process for process monitoring and control studies to test out the effectiveness of various methods for monitoring and control strategy [2]. Since data-driven process monitoring and FDD only requires the labeled descriptive datasets from the process. Various datasets have been published specifical for the purpose of process monitoring based on data collected from simulations of the TEP [1].

We adopt the dataset provided online at [1] which is frequently used for monitoring. The process contains 52 variables in total, 41 sensor measurements (XMEAS(1)-XMEAS(41)) and 11 manipulated variables (XMV(1)-XMV(11)) each at an interval of 3 minutes and 21 different fault scenarios (IDV(1)-IDV(21)). The process contains five major units (Reactor, Condenser, Compressor, Vapor Liquid Separator and Stripper) and eight components (A, B, C, D, E, F, G and H). TEP is a useful benchmark for process monitoring for a wide range and variety of deviations from NOC. Among the eight components the five gaseous inputs: A, C, D and E are fed into the reactor along with the inert component B, where two liquid products G and H along with a by-product F are formed.

We adopt the dataset provided online at [1] which is frequently used for monitoring. The process contains 52 variables in total, 41 sensor measurements (XMEAS(1)-XMEAS(41)) and 11 manipulated variables (XMV(1)-XMV(11)) each at an interval of 3 minutes and 21 different fault scenarios (IDV(1)-IDV(21)). The process contains five major units (Reactor, Condenser, Compressor, Vapor Liquid Separator and Stripper) and eight components (A, B, C, D, E, F, G and H). TEP is a useful benchmark for process monitoring for a wide range and variety of deviations from NOC. Among the eight components the five gaseous inputs: A, C, D and E are fed into the reactor along with the inert component B, where two liquid products G and H along with a by-product F are formed.

$$A\,(g) + C\,(g) + D\,(g) \quad \rightarrow \quad G(liq)\ \text{(Product 1)},$$

$$A\,(g) + C\,(g) + E\,(g) \rightarrow H(liq)\ (\text{Product 2}),$$

$$A\,(g) + E\,(g) \rightarrow F(liq)\ (\text{Byproduct}),$$

$$3D\,(g) \rightarrow 2F(liq)\ (\text{Byproduct})$$

Table 1: process variable measurements and the manipulated variable

Block	Description	Tag number in Fig: TEP	Variable
Input feed	A feed (stream 1)	1	XMEAS(1)
	D feed(stream 2)	2	XMEAS(2)
	E feed(stream 3)	3	XMEAS(3)
	A and C feed(stream 4)	4	XMEAS(4)
Reactor	Reactor feed rate(stream 6)	6	XMEAS(5)
	Reactor pressure	7	XMEAS(6)
	Reactor level	8	XMEAS(8)
	Reactor temperature	9	XMEAS(9)
Separator	Separator temperature	11	XMEAS(11)
	Separator level	12	XMEAS(12)
	Separator pressure	13	XMEAS(13)
	Separator underflow	14	XMEAS(14)
Stripper	Stripper level	15	XMEAS(15)
	Stripper pressure	16	XMEAS(16)
	Stripper underflow	17	XMEAS(17)
	Stripper temperature	18	XMEAS(18)
	Stripper steam flow	19	XMEAS(19)
Miscellaneous	Recycle flow(stream 8)	5	XMEAS(5)
	Purge rate(stream 9)	10	XMEAS(10)
	Compressor work	20	XMEAS(20)
	Reactor water temperature	21	XMEAS(21)
	Separator water temperature	22	XMEAS(22)

Reactor feed analysis	Component A	23	XMEAS(23)
	Component B	24	XMEAS(24)
	Component C	25	XMEAS(25)
	Component D	26	XMEAS(26)
	Component E	27	XMEAS(27)
	Component F	28	XMEAS(28)
Purge gas analysis	Component A	29	XMEAS(29)
	Component B	30	XMEAS(30)
	Component C	31	XMEAS(31)
	Component D	32	XMEAS(32)
	Component E	33	XMEAS(33)
	Component F	34	XMEAS(34)
	Component G	35	XMEAS(35)
	Component H	36	XMEAS(36)
Product analysis	Component D	37	XMEAS(37)
	Component E	38	XMEAS(38)
	Component F	39	XMEAS(39)
	Component G	40	XMEAS(40)
	Component H	41	XMEAS(41)

Table 2: Process manipulated variable:

Description	Tag number in Fig: TEP	Variable
D feed flow	42	XMV(1)
E feed flow	43	XMV(2)
A feed flow	44	XMV(3)
A and C feed flow	45	XMV(4)
Compressor recycle valve	46	XMV(5)
Purge valve	47	XMV(6)

Separator pot liquid flow	48	XMV(7)
Stripper liquid product flow	49	XMV(8)
Stripper steam valve	50	XMV(9)
Reactor cooling water flow	51	XMV(10)
Condenser cooling water flow	52	XMV(11)

Table 3: Process faults:

Description	Type	Fault number
A/C feed ratio, B composition constant	Step	IDV(1)
B composition, A/C ratio constant	Step	IDV(2)
D feed temperature	Step	IDV(3)
Reactor cooling water inlet temperature	Step	IDV(4)
Condenser cooling water inlet temperature	Step	IDV(5)
A feed loss	Step	IDV(6)
C header pressure loss-reduced availability	Step	IDV(7)
A, B and C feed composition	Random variation	IDV(8)
D feed temperature	Random variation	IDV(9)
C feed temperature	Random variation	IDV(10)
Reactor cooling water inlet temperature	Random variation	IDV(11)
Condenser cooling water inlet temperature	Random variation	IDV(12)
Reaction kinetics	Slow drift	IDV(13)
Reactor cooling water valve	Sticking	IDV(14)
Condenser cooling water valve	Sticking	IDV(15)
Unknown	Unknown	IDV(16)
Unknown	Unknown	IDV(17)
Unknown	Unknown	IDV(18)
Unknown	Unknown	IDV(19)
Unknown	Unknown	IDV(20)
Valve fixed at the steady state position	Constant Position	IDV(21)

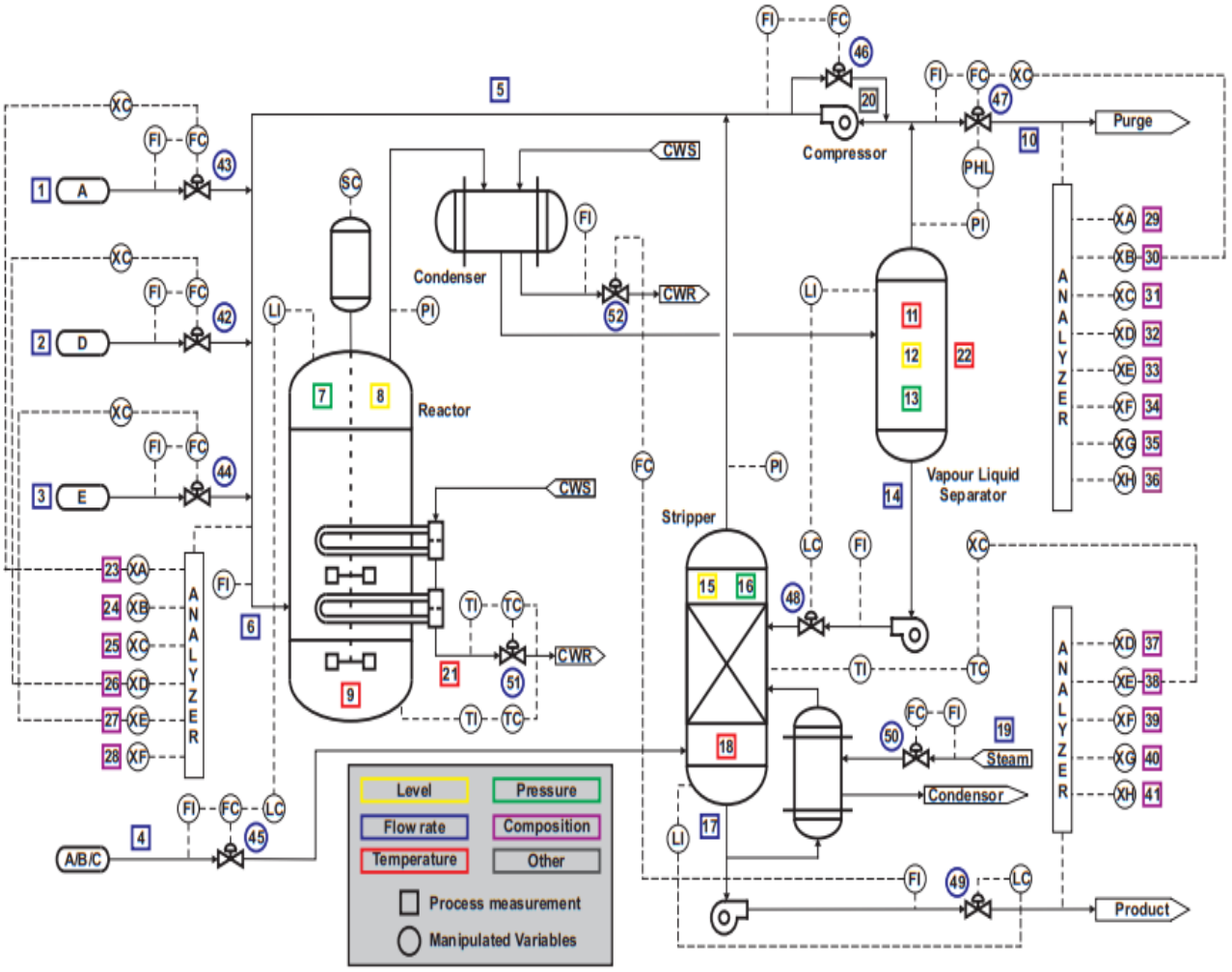


Figure 13: TEP: Process Schematic of the Tennessee Eastman Process (TEP)

2.4 Results

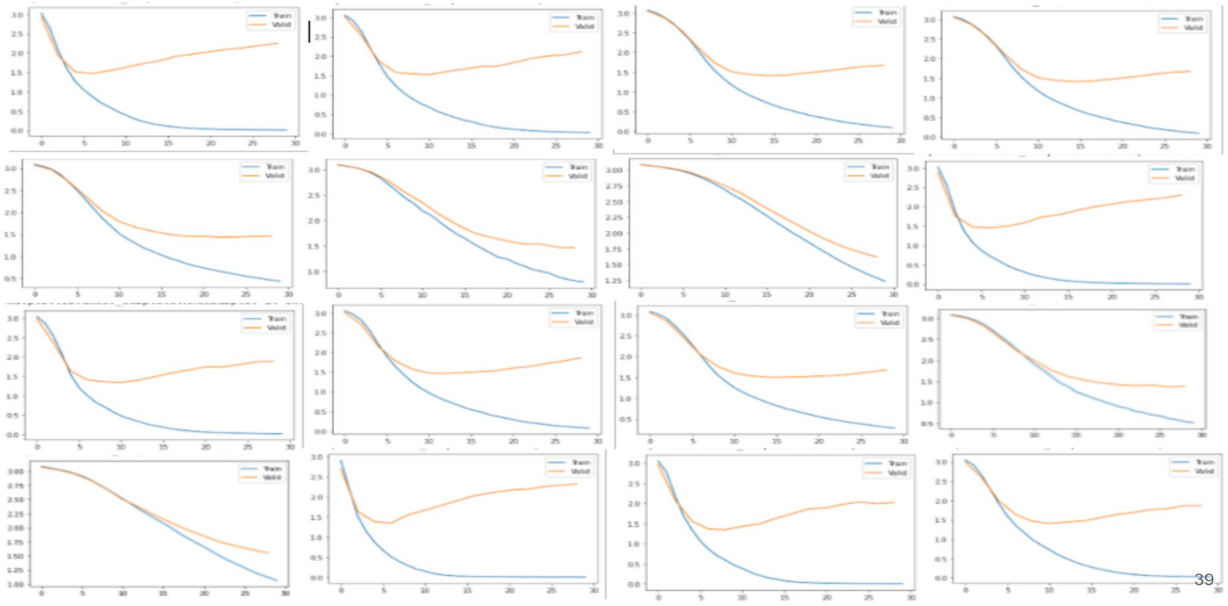
Fault detection and diagnosis method based on LSTM :

Deep neural network architectures have shown to be powerful at learning complex nonlinear feature representation. Further specialized neural network architectures such as LSTM are very powerful for processing sequential data[9,10]. For this report we will be using LSTM for directly classifying any process instance thus this model will be simultaneously performing fault detection and fault diagnosis through the means of end-to-end learning.

Performance Comparison with different key parameters:

In order to analyze the performance of the model for different hyperparameters, we created a smaller random sample of the dataset from the entire dataset. We then proceeded to preprocess the data by normalizing it using the standard deviation and mean from the normal operating condition dataset. This helps in speeding up the learning and leads to faster convergence. After this we tested the performance on different hyperparameters and finally used the hyperparameters that yielded the best performance. To minimize the computational time we had used a small dataset of 300 samples to create the epoch vs loss graph.

All these computation was done using google colaboratory on a Tesla K80 GPU using the the deep learning package pytorch



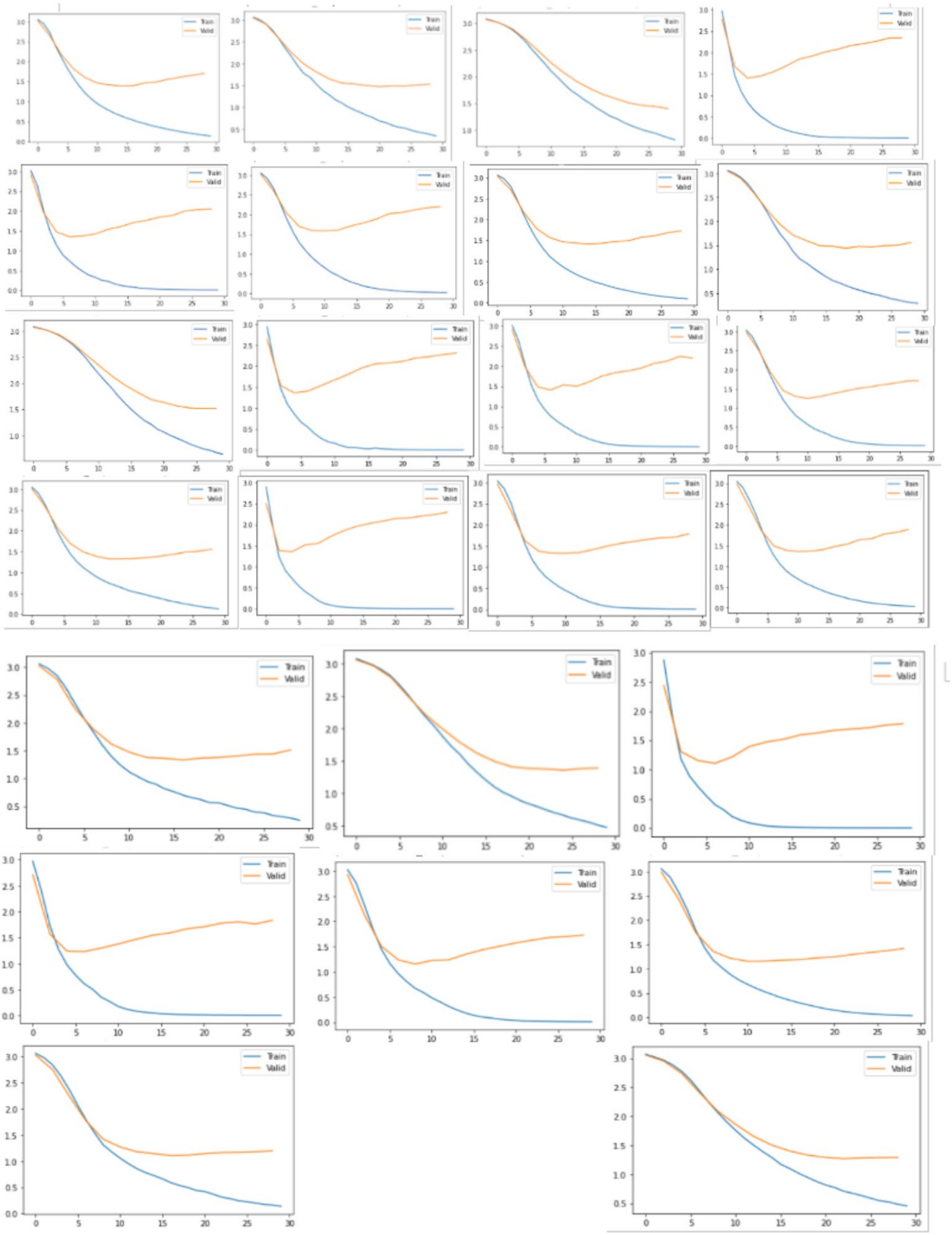


Figure 14: Epoch vs Loss Graph for Training and Validation loss for various hyperparameters

We decided to train the main LSTM model having 2 hidden LSTM layers and 40 nodes for each hidden layer. The training was done with the learning rate set to 0.01, batch size to 64 and the input sequence length to 8, i.e at each instance 7 previous sequence of data are fed to the LSTM model as well so the function is generalized as $F(X_{t-7}$ to $X_t)=Y_t$.

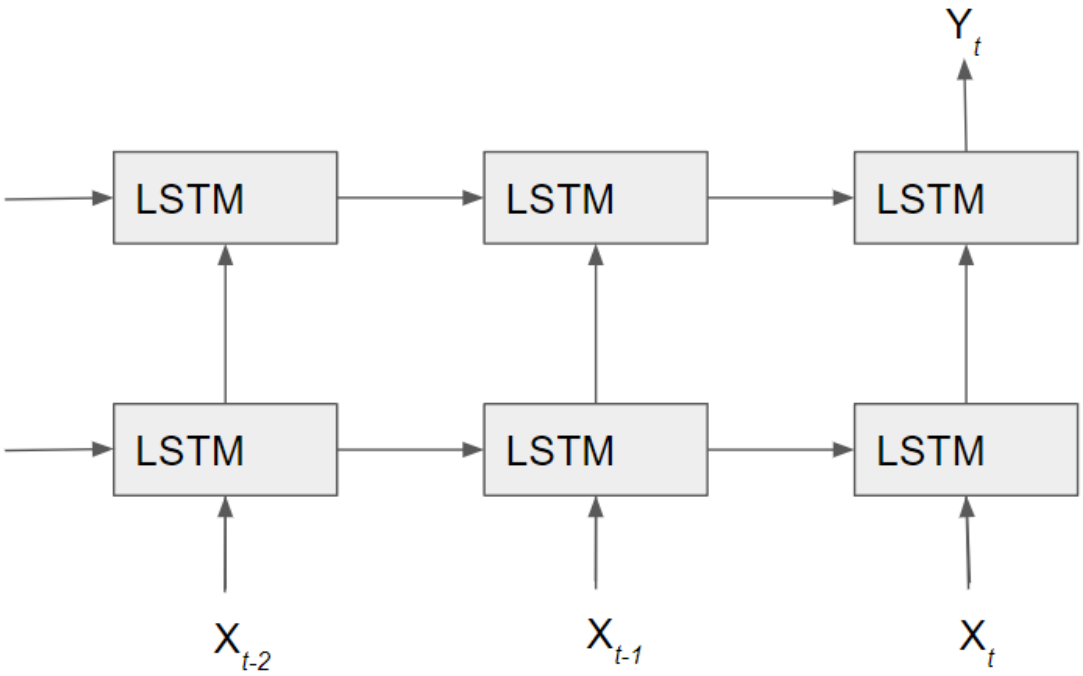


Figure 15: Two hidden LSTM layer (Many to one architecture)

The performance of the final trained model is measured using a confusion matrix and the ROC curve. The confusion matrix and the ROC curve for the training dataset is given in figure 3.2 and figure 3.3. Similarly the Confusion matrix and the ROC curve for the testing dataset is given in the figure 3.4 and figure 3.5

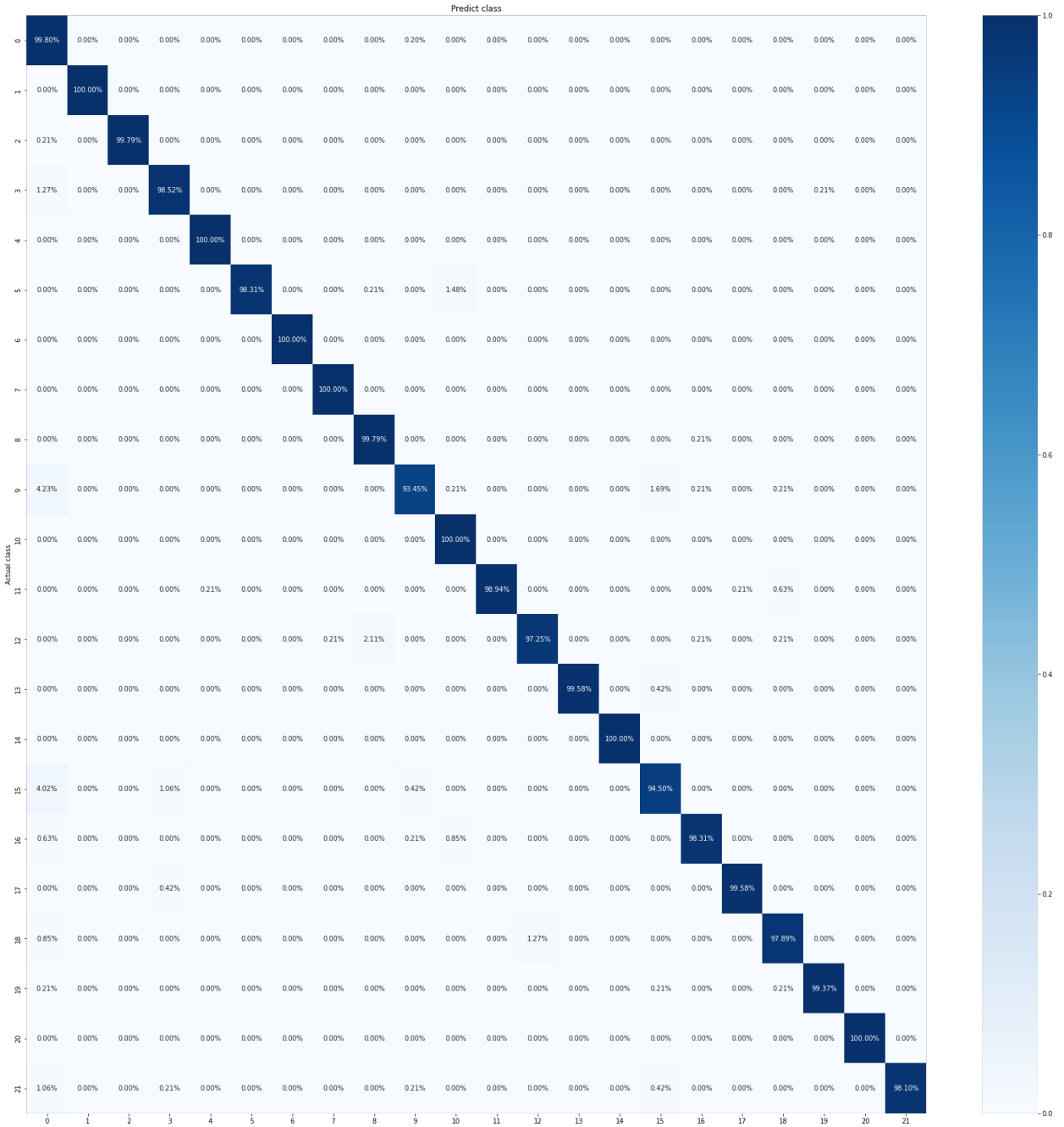


Figure 16: Confusion matrix for training data

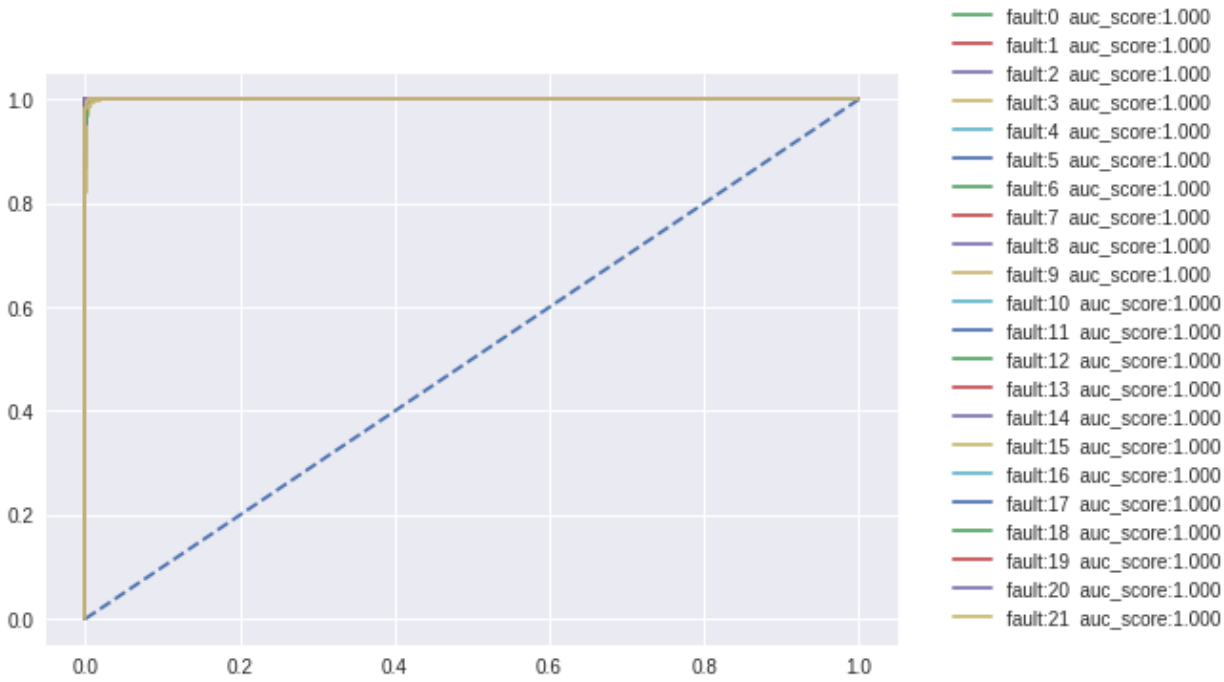


Figure 17: ROC curve for training data

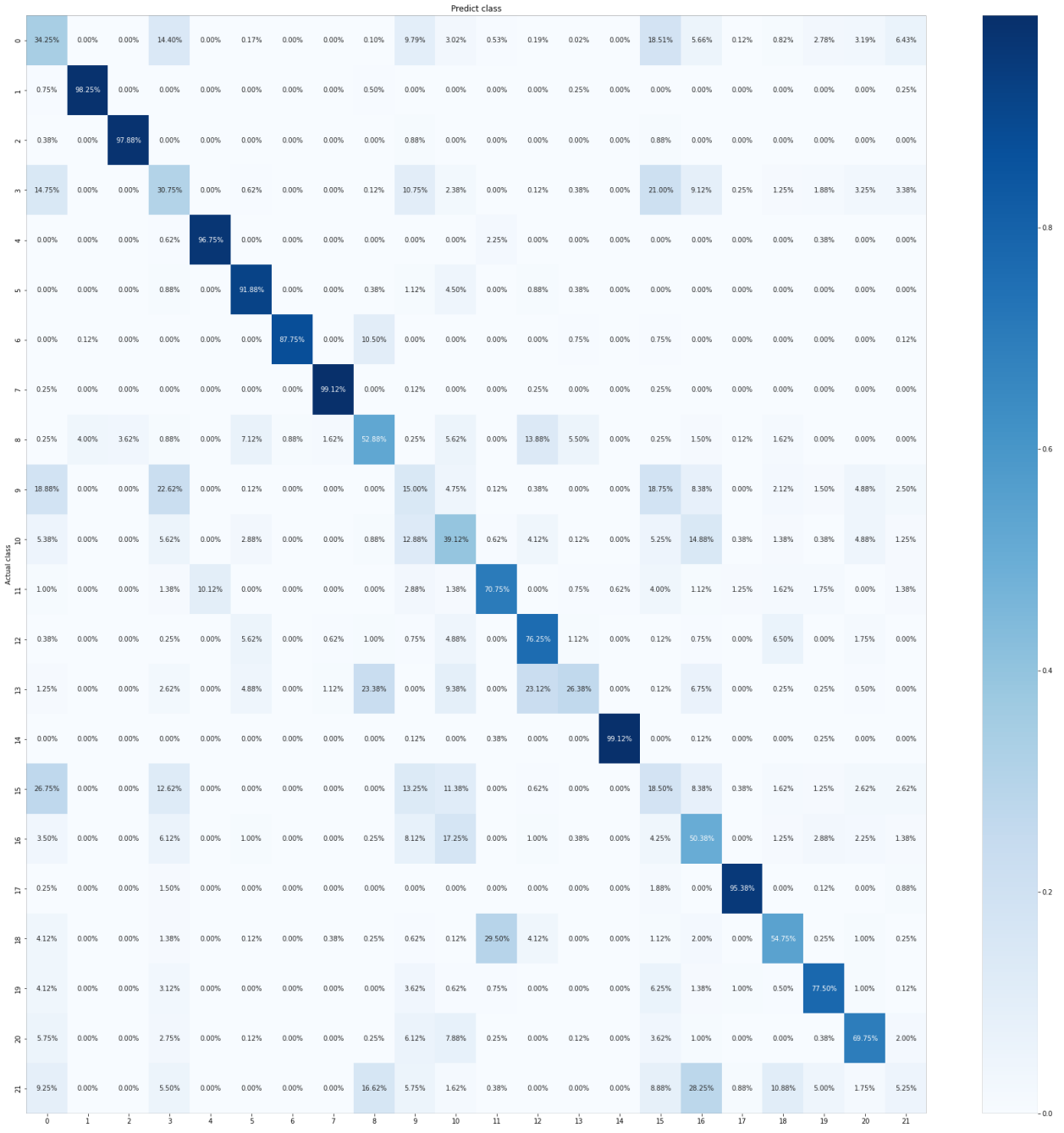


Figure 18: Confusion matrix for testing data

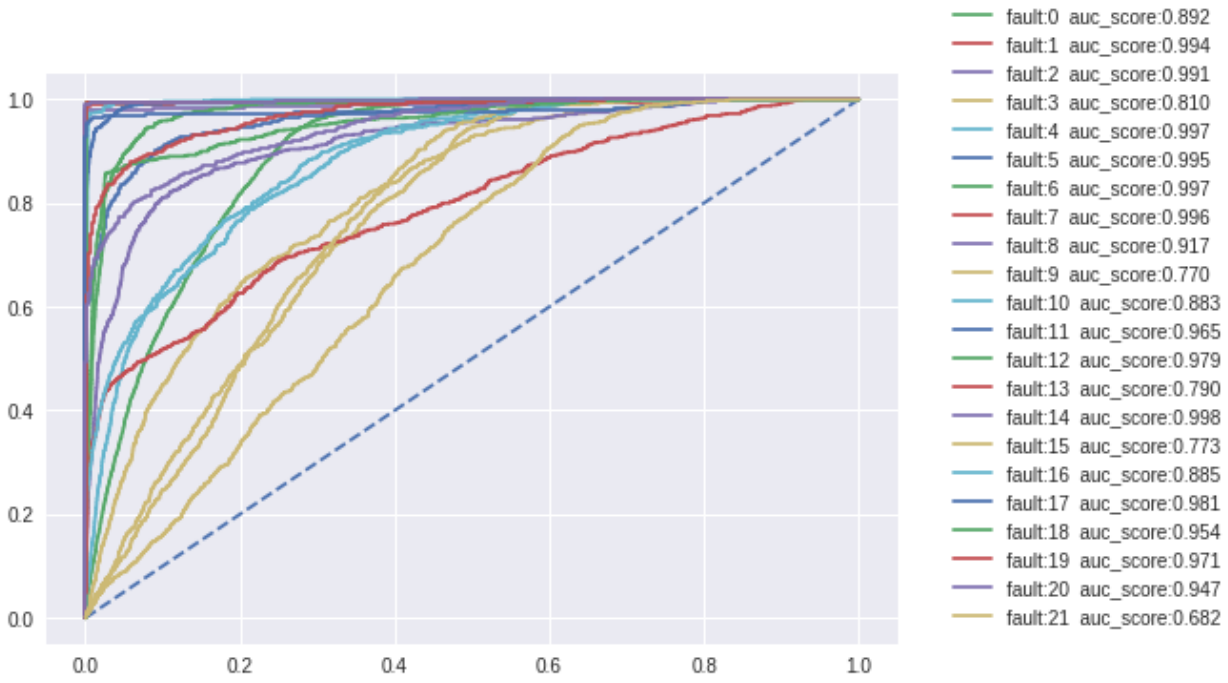


Figure 19: ROC curve for training data

The overall accuracy of the model in correctly identifying the class on the testing dataset was found to be 58.447%, whereas the accuracy on the training dataset was 98.72%. The model was able to correctly diagnose fault type 1, 2, 4, 5, 7, 14, 17 with accuracy greater than 90%, whereas the model performed poorly in correctly diagnosing NOC, fault type 3, 9, 10, 13, 15 and 21. The model however misclassified NOC into faults more than 60% which would significantly increase the false alarm rate.

The model requires more hyperparameter tuning to find the optimal settings for it to perform as intended. The accuracy can definitely be increased as evidently shown from[9, 10], where LSTM based models have shown to perform great.

CONCLUSION

The performance of process data analytics depends highly on the quality and quantity of training data. High diversity in data quality and quantity is commonly observed in manufacturing processes. Many applications have only limited samples from a manufacturing process because of time and cost constraints. Moreover, disturbances and biases are also present in the process measurements. Also, certain key process variables might not be recorded directly. In those situations, regardless of what analytical approach has been used, the performance of process data analytics will be affected. So, care must be taken when choosing which process variables to record and labeling the data.

Fault detection plays an important role in high-cost and safety-critical processes. Early detection of faults can help avoid abnormal event progression.

Particularly for the benchmark dataset, (TEP dataset) due to constraints on computing time and computing power, training on a broad set of hyperparameters became impossible and attempts to see the performance on a wide range of hyperparameters on larger dataset was discarded. The problem of overfitting can be addressed by regularization techniques such as including dropout during the training process which might increase the overall performance of the model.

As of now the performance of the model on the benchmark dataset wasn’t too satisfactory as improvements could be further made in the future.

REFERENCES

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

[2]Downs, J. J., & Vogel, E. F. (1993). A plant-wide industrial process control problem. Computers & chemical engineering, 17(3), 245-255.

[3]Yin, S., Ding, S. X., Haghani, A., Hao, H., & Zhang, P. (2012). 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.

[4]Kano, M., Hasebe, S., Hashimoto, I., & Ohno, H. (2001). A new multivariate statistical process monitoring method using principal component analysis. Computers & chemical engineering, 25(7-8), 1103-1113.

[5]Choi, S. W., Lee, C., Lee, J. M., Park, J. H., & Lee, I. B. (2005). Fault detection and identification of nonlinear processes based on kernel PCA. Chemometrics and intelligent laboratory systems, 75(1), 55-67.

[6]Russell, E. L., Chiang, L. H., & Braatz, R. D. (2000). Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. Chemometrics and intelligent laboratory systems, 51(1), 81-93.

[7]Zhang, K. (2016). Performance assessment for process monitoring and fault detection methods. Springer Fachmedien Wiesbaden.

[8]Ding, S., Zhang, P., Ding, E., Naik, A., Deng, P., & Gui, W. (2010). On the application of PCA technique to fault diagnosis. Tsinghua Science and Technology, 15(2), 138-144.

[9]Zhao, H., Sun, S., & Jin, B. (2018). Sequential fault diagnosis based on LSTM neural network. Ieee Access, 6, 12929-12939.

[10]Zhang, Q., Zhang, J., Zou, J., & Fan, S. (2020). A novel fault diagnosis method based on stacked lstm. IFAC-PapersOnLine, 53(2), 790-795.

[11]Wu, D., & Zhao, J. (2021). Process topology convolutional network model for chemical process fault diagnosis. Process Safety and Environmental Protection, 150, 93-109.

[13]Huang, T., Zhang, Q., Tang, X., Zhao, S., & Lu, X. (2022). A novel fault diagnosis method based on CNN and LSTM and its application in fault diagnosis for complex systems. Artificial Intelligence Review, 55(2), 1289-1315.