Partial Weights Update Based ANN for State Estimation in Presence of Faults

A Dissertation

Submitted in Partial Fulfillment of

the requirements for the degree of

Master of Technology

by

Mishra Digvijay

(Roll No. 203020005)

Under the Guidance of

Prof. Mani Bhushan

Department of Chemical Engineering

Indian Institute of Technology Bombay

Mumbai 400076 (India)

21st June 2022

Acceptance Certificate

Department of Chemical Engineering

Indian Institute of Technology, Bombay

The Dissertation entitled “Partial weights update based ANN for state estimation in presence of faults” submitted by Digvijay Mishra (Roll No. 203020005) may be accepted for being evaluated.

Mani B

Date: 21st June 2022

Prof. Mani Bhushan

Approval Sheet

This Dissertation entitled "Partial weights update based ANN for state estimation in presence of faults" by Digvijay Mishra is approved for the degree of Master of Technology.

Examiner

Examiner

Supervisor

Chairman

Date: 21st June 2022

Place: Mumbai

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will cause for disciplinary action by the Institute and can evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: 21st June 2022

Digvijay Mishra
(Roll No. 203020005)

Abstract

In the process industry, many advanced process operations tasks, such as state estimation and advanced control, must be conducted in real time. In their decision-making, these techniques frequently use a process model. For many systems, a detailed first principles model can be too cumbersome to be used by these approaches as it can make these approaches infeasible for online implementation.

With the increasing availability of either historical plant data or data from detailed plant simulators (a first principles model), it is possible to train a data driven model which is more suitable for advanced control or estimation applications. Several works in literature have demonstrated that artificial neural networks (ANNs) can provide a generic nonlinear model for processes with highly nonlinear relationships between the output and input variables. Using the ANN model with advanced estimators such as Extended Kalman Filter (EKF), accurate state estimates can be obtained at lesser computational load than using the detailed plant simulator with EKF. However, efficient updation of neural network to capture time varying behaviour of various plant parameters remains a challenge. In principle, all the weights of the neural network can be updated (retrained) to capture the plant behaviour in presence of parametric variations (faults). However, this can be computationally tedious and also will require significant amount of training data.

To deal with this problem, in the current work, partial weight training of the artificial neural network (ANN) model is proposed. Towards this end, changes in various parameters of the first principles model are mapped to weights of the neural networks. In particular, a subset of weights is identified which is sensitive to changes in the parameter. During online implementation, depending on the changed parameter, only the corresponding subset of weights are updated to ensure that the updated neural network captures the plant behaviour despite the parameter variation. The strategy was implemented and compared with existing conventional training method (involving full weights update) on a reactor system using simulation studies. The results demonstrate that a partially trained neural network is capable of accurately capturing system dynamics and, when combined with EKF, can provide adequate performance for state estimation in presence of faults with lower computing costs.

Table of content

List of figures

List of tables

Chapter 1

Introduction

Chemical plants are extremely complicated in nature, and to regulate these processes, we employ a huge number of sensors and devices that gather data for measured output variables for process control and monitoring. Over the last few decades, researchers have come up with predictive models which are useful in predicting the process variables that are important for the overall plant operations, and they are assessed using either the online low frequency sampling rate or offline methodologies.

These models are based on a mathematical model of the process. As industrial processes are quite complex, the problem of finding an exact or approximate model for the dynamic system frequently occurs in engineering applications [1]. The mechanistic model generated for these complex systems often ends up with substantial computational cost due to complex coupled differential equations and is thus not feasible to be used online. As a result, data-driven approaches that employ measured process variables to estimate plant dynamics can be applied. With the upcoming advancements in AI and computer capacity, we can create such a model using deep neural networks. Once trained, we may utilise this model in conjunction with multiple techniques to efficiently estimate plant states. Several researches have demonstrated that artificial neural networks (ANNs) can provide a generic nonlinear solution to processes with a highly nonlinear relationship between the output and input variables. Using the ANN model with highly efficient estimators such as Moving Horizon Estimation (MHE) and Extended Kalman Filter (EKF) [2] will overcome the major challenges encountered by the chemical sector.

1.1 Motivation

Although Artificial neural networks (ANN) are highly efficient in capturing Nonlinear behaviours of Chemical industries and many other areas of work have already begun to adopt Deep learning methods to improve process efficiency and operational cost. If training for artificial neural networks (ANN) is done on measured data at some moment and implemented for process state estimation, the model's accuracy will decrease over time as fault occurs in

process parameters operating predetermined values (e.g., catalyst deactivation). As a result, the demand to train the model from scratch will increase with time, requiring a significant amount of time, monetization, and training data. This is one of the primary issues that chemical industries confront when applying deep learning methods such as artificial neural networks (ANNs).

As a result, the concept of partial weight training of the artificial neural network (ANN) model is offered as a solution. The derivation of a relationship between the process parameters of the chosen case study and the weights of the artificial neural network model was used to select these partial weights. Only weights that show a deviation greater than the chosen threshold will be trained in this strategy, whereas other weights that remain constant or show very little fluctuation in their values while training the model on multiple datasets will be left untrained. As a result, by not having to train the complete artificial neural network model from the beginning, this strategy saves time, money, and computational cost. Overall approach is as given presented in Figure 1-1. In Chapter 6, we discuss the entire strategy for determining the link between process parameters and ANN model weights in detail, and in Chapter 7 we discuss how to execute partial weight training using the relationship between process parameters and ANN weights.

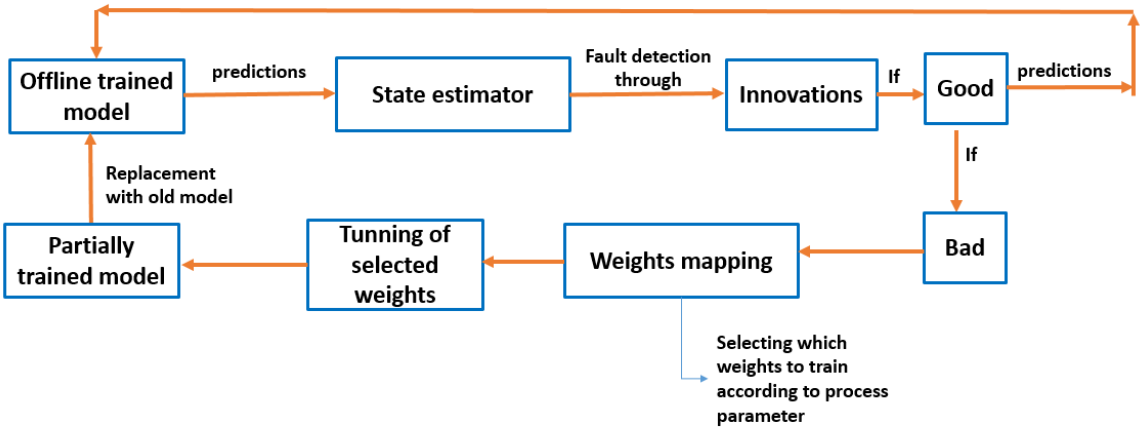

Figure 1-1: Structure of overall approach

1.2 Report Organization

A study of a different way to generate a model of the system instead of a physics-based approach and combination it with a state estimator has been done from existing literature on Chapter 2. Background work that is EKF algorithm and modelling of the system

using artificial neural network has been discussed in Chapter 3. The proposed method that is combining ANN with EKF and its various permutations has been described in Chapter 4 Trained ANN-EKF results for taken case study have been discussed in Chapter 5. approach for deriving relationship between process parameter and ANN model has been given in Chapter 6. Step by step procedure to do partial weight training of ANN model with results have been given in Chapter 7 Conclusion of the results from the proposed method has been reported in Chapter 8.

Chapter 2

Literature Survey

There are some existing methods that can be used to model the system to reduce the calculation complexity, which has been discussed in Chapter 3. In the existing literature, two possible ways to approximate system dynamics have been found which can be used by the soft sensor to solve the state estimation problems [3].

- First principal modelling (theoretical approach)
- Data-based modelling (data-driven approach)

2.1 First Principal Modelling

Many additional names for the first principal modelling include white box models and mechanistic modelling. The development of the first principal model necessitates a methodical approach and is based on the following:

- Material and energy balances of the whole process plant
- Physical laws and constitutive relationships within the process variables
- Kinetic and thermodynamic models supporting the process plant
- Heat and mass transfer models

2.2 Data-based modelling

Data-based modelling is done with the help of the measured variables database. There are two ways to collect the data for the data-driven models:

2.2.1 Experimental methods

Experiments are undertaken to generate data. The data collected by them is thought to be ideal for modelling since it contains extremely little error and noise, allowing for greater precision. However, doing the trials is either forbidden or too expensive.

2.2.2 Plant data or historic data

Chemical operations are provided with a variety of sensors that continuously monitor the numerous process factors. The process database's accumulation of data over time can be a helpful source of data for model development. Plant modelling with process data is typically inexpensive, but the data contains several impurities that must be removed before it can be used for modelling. As a result, before using the data set for modelling, it must undergo suitable data pre-processing.

Mechanistic models are frequently inefficient, costly, and time-consuming. They can also be computationally demanding, necessitating repetitive calculations that are inappropriate for online applications. As a result, shifting to efficient data-driven modelling methodologies will aid in the resolution of this issue. Multilayer networks have proven highly successful in pattern recognition problems [4]. As a result, they can tackle the challenge of modelling large chemical plants, which are difficult to represent in traditional approaches. There have been various papers supporting this idea. [5] discussed a systematic approach to design a non-linear observer to estimate the state vector of a non-linear dynamic system. The author used two neural networks to solve the state estimation problem in the research. The first neural network was used to forecast the model, while the second neural network generated filtered state values. As a result, the state estimation problem was handled entirely by neural networks in the aforementioned research. In addition, the paper includes a case study to assess the efficacy of the planned task. In [6], a neural network-based state estimator for a general class of non-linear dynamical systems is presented. The proposed state estimator employs the cascading of a recurrent neural network structure (RNN) that learns the dynamical system's internal behaviour and a feedforward neural network structure (FFNN) that learns the system's measuring relations from the input-output data by minimising prediction error. The recurrent neural network has been trained with a dynamic learning method. A similar analogy like above is also shown in [7] and many more. As a result, employing neural networks for system identification has been researched and acknowledged in numerous research articles, such as the ones mentioned above, and may therefore be utilised for system modelling.

One approach to tackling the state estimation problem is to use neural networks to alone drive the model and estimator. Second, in the current research, a

combination of the neural network model with classic state estimators such as Kalman filter, Extended Kalman Filter, and so on has been documented. A typical feedforward neural network with an estimator may be used for model predictions, and these approaches have been widely addressed in the literature. A few publications that address the notion of using ANN as a model with traditional estimators are included below in the report. [8] paper addressed the problem of estimating the states of the system using an artificial neural network as the model and EKF as the state estimator. The EKF requires a state-space representation of the model, hence this paper describes a method for transforming the ANN into a state-space model. Relevant parameters of the system, such as $\emptyset$ and γ, were determined using this state-space model and were used by the estimator to identify filtered values of the states. In addition, case examples were included in the study to demonstrate the efficacy of the suggested task.

As a result of the research reviewed above, alternative solutions for solving the state estimate problem have been given in Chapter 4 All of the solutions are based on the notion of modelling the system using an artificial neural network and estimating states with an EKF that is solely driven by an ANN (ANN-EKF). Using such methodologies can help save a lot of computing time, as demonstrated by the implementation of the suggested work on a case study given in Chapter 5.

2.3 Conclusion

Two possible ways to approximate system dynamics are available 1. Theoretical approach 2. Data-driven approach. Theoretical approaches are time consuming and computationally heavy so as a solution data-driven approaches can be used. Multilayer ANN has proven highly successful in pattern recognition problems. As a result, they can tackle the challenge of modelling large chemical plants, which are difficult to represent in traditional approaches. Some paper describes a method for transforming the ANN into a state-space model and Relevant parameters of the system, such as $\emptyset$ and γ, were determined using this state-space model and were used by the estimator to identify filtered values of the states.

Chapter 3

Background of Neural Network and State Estimator

As literature survey is given in Chapter 2 about ANN and EKF, in this chapter explanation of how the artificial neural network works, how the EKF state estimator works with mathematical representation is given and at the end information about ADAM optimizer is mentioned, which has been used in ANN model training and also in partial weight training.

3.1 Artificial Neural Network

Artificial Neural Networks (ANNs) can be used as a tool for modelling non-linear process systems. They provide a black-box approach in which ANNs don't require prior knowledge about the system. If some data is available from the system, then ANNs have potential to learn the intrinsic relationship between the input and variables of the system and after training, it is capable of predicting states based on given inputs.

An artificial neural network is one of the most popular and basic techniques used for approximating complex nonlinear systems. In [9], it is mentioned that a basic ANN structure is a combination of several non-linear processors which are also known as nodes, these nodes are classified as input, hidden or output layers. Every node is connected with all the nodes of the next layer. These links between nodes have different importance and it is decided by weights assigned to that particular link and these weights define the knowledge acquired by the network. Input variables are scaled to a standard form which is between 0 and 1 if we have used sigmoid as an activation function. Each node in the input layer distributes its scaled values to every node in the next layer. In the hidden layer and output layer, for the j^{th} node in the i^{th} layer, the input is the sum of outputs from all the previous nodes which is indicated by $z_{i,j}$:

$$z_{i,j} = b_{i,j} + \sum_{k=1}^{n_{i-1}} w_{i,j,k}\ O_{i-1,k} \qquad 3\text{-}1$$

Where $b_{i,j}$ is a bias and $w_{i,j,k}$ are the weights in the connection between the j^{th} node in layer i and the k^{th} node in layer i-1. This weighted sum is then transformed to each node by using

activation functions like tanh, sigmoid, ReLu, etc. The output of sigmoid activation function for j^{th} node in the i^{th} layer is given by:

$$O_{i,j} = \frac{1}{1 + e^{-z_{i,j}}} \qquad 3\text{-}2$$

Thus, every node generates an output between 0 and 1 for the sigmoid activation function and we have a linear activation function at the output layer which combines all the output of previous layers to give a desired output of the whole network.

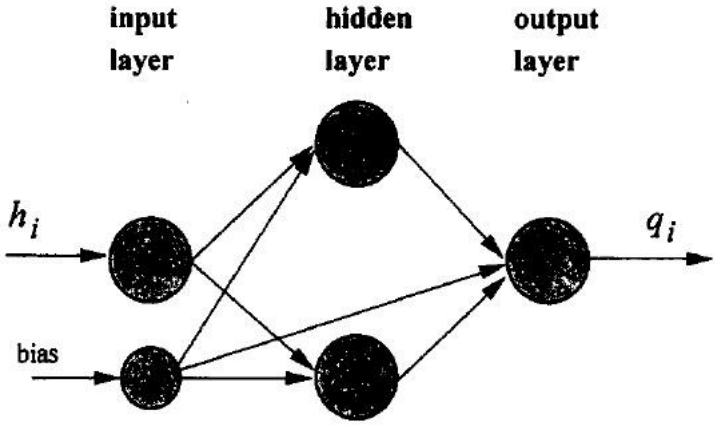

Figure 3-1:Structure of the ANN model

Backpropagation is used to get the weight value that minimises network error. Outputs generated by the ANN network are compared to the corresponding values in the training data set and it generates errors for output nodes. Then these error values are used to update weights on the links to the nodes, this process is called the backpropagation technique. However, to capture highly dynamic plants, a neural network may have some memory of past calculations which can be captured easily with special types of neural networks called Recurrent Neural Networks (RNN). Many other neural networks like Radial basis Function (RBFNN), FRNN (Fully connected neural networks) and DRNN (diagonal recurrent neural network) can also be used for approximating plant dynamics depending on the complexity of the plant.

3.2 Extended Kalman Filter

Consider a nonlinear process dynamics and nonlinear measurement equation can be represented as

$$X(k+1) = F\big(X(k), u(k), d(k)\big) \qquad 3\text{-}3$$

With $d(k) \sim N(0, Q)$

$$Y(k) = h\big(X(k)\big) + v(k) \quad \text{3-4}$$

With $v(k) \sim N(0, R)$

In Eq. (3-3) X(k)$\in R^n$ represent the states of the system, u(k)$\in R^n$ represent the manipulated inputs at the time instant k and d(k)$\in R^n$ represent a zero mean gaussian white noise with covariance matrix Q. in Eq. (3-4) Y(k)$\in R^n$ represent the measurement of nonlinear function of the state and v(k)$\in R^n$ represent zero mean white gaussian noise with covariance R. Further, d(k) and v(k) are assumed to be independent and uncorrelated of each other. Noise at different time instant is uncorrelated. The steps of EKF are discussed in [10]. The first order approximation of Taylor's series of the nonlinear system equation around states estimated at current instant is used by EKF.

$$X(k+1) = F\left(\hat{X}(k|k), u(k), d(k)\right) + \emptyset(k)(X(k) - \hat{X}(k|k)) \quad \text{3-5}$$

$$where\ A(k) = \frac{\partial f}{\partial x}|_{(\hat{X}(k|k), u(k), d(k))} \quad \text{3-6}$$

$$\emptyset(k) = \exp(A(k)t_s) \quad \text{3-7}$$

Prediction step in EKF:

$$\hat{X}(k+1) = F\left(\hat{X}(k|k), u(k), d(k)\right) \quad \text{3-8}$$

$$P(k+1|k) = \emptyset(k)P(k|k)\emptyset(k)^T + \gamma_d Q {\gamma_d}^T \quad \text{3-9}$$

$$Y(k) = h\big(X(k)\big) + v(k) \quad \text{3-10}$$

The correction step of EKF is given by:

$$K(k+1) = P(k+1|k)C(k)^T(C(k)P(k+1|k)C(k)^T + R)^{-1} \quad \text{3-11}$$

$$\hat{X}(k+1|k+1) = \hat{X}(k+1|k) + K(k+1)e(k) \quad \text{3-12}$$

$$where\ \ e(k) = y(k+1) - h(\hat{x}(k+1|k)) \quad \text{3-13}$$

$$P(k+1|k+1) = P(k+1|k) - K(k+1)C(k)P(k+1|k) \quad \text{3-14}$$

$$where\ C(k) = \frac{\partial h}{\partial x}|_{(\hat{X}(k+1|k))} \quad 3\text{-}15$$

In Eq. (3-7) t_s represents sample time of the process. In Eq. (3-7) and Eq. (3-15), $\emptyset(k)$ and C(k) are the Jacobian matrices, which are required to calculate predicted covariance, P(k+1|k) and Kalman gain matrix, K(k+1) in the Eq. (3-9) and Eq. (3-12) respectively.

3.3 Adam Optimizer

Before looking at Adam optimizer we have to understand different optimizers like Gradient Decent (SGD), Momentum GD and RMSprop because Adam is having essence of all these optimizers and has proven to be the best optimizer in many case studies [11].

1) Gradient Descent: It is the most basic optimizer and optimization formula is

$$W_{i,t} = W_{i,t-1} - \delta(\frac{\partial L}{\partial W_i}) \quad 3\text{-}16$$

Where, $\delta = learning\ Rate, L = Loss\ function, W_{i,t} = weight\ at\ time\ \ t$

2) Momentum Gradient Descent: SGD provides data one by one to model and weights get updated but due to that it creates a lot of noise so to reduce the noise we include momentum in GD and because of that process becomes faster compared to before.

$$W_{i,t} = W_{i,t-1} - \delta(V_{dw,t-1}) \quad 3\text{-}17$$

$$b_{i,t} = b_{i,t-1} - \delta(V_{db,t-1}) \quad 3\text{-}18$$

Where $b_{i,t} = bias\ in\ the\ neural\ network$

$$V_{dw,t} = \beta_1 V_{dw,t-1} + (1-\beta_1)\frac{\partial L}{\partial W_i} \quad 3\text{-}19$$

$$V_{db,t} = \beta_1 V_{db,t-1} + (1-\beta_1)\frac{\partial L}{\partial b_i} \quad 3\text{-}20$$

Where, β_1 is constant and its values are between 0 and 1. The learning rate is different for biases and weights but β_1 value is same for both.

3) RMSProp or Adadelta: In this optimizer initially learning rate takes a large value and as the process proceeds, learning rate reduces so that's how it takes less time compared to fix learning rate.

$$W_{i,t} = W_{i,t-1} - \delta'(\frac{\partial L}{\partial W_i}) \quad 3\text{-}21$$

Where $\delta' = \frac{\delta}{\sqrt{\gamma}_t + \varepsilon}$, $\varepsilon = small\ numeric\ value$

$$\gamma_{w,t} = \beta_2 \gamma_{w,t-1} + (1 - \beta_2)(\frac{\partial L}{\partial W_i})^2 \quad 3\text{-}22$$

$$\gamma_{b,t} = \beta_2 \gamma_{b,t-1} + (1 - \beta_2)(\frac{\partial L}{\partial b_i})^2 \quad 3\text{-}23$$

Where, in $\gamma_{w,t}$, w represent that it is for weights and in $\gamma_{b,t}$, b represent that it is for biases

4) ADAM Optimizer: In this optimizer, we have momentum + adaptive learning rate and that's why it is best till now compare to other optimizers.

$$W_{i,t} = W_{i,t-1} - \frac{\delta}{\sqrt{\gamma}_{w,t} + \varepsilon}(V_{dw}) \quad 3\text{-}24$$

$$b_{i,t} = b_{i,t-1} - \frac{\delta}{\sqrt{\gamma}_{b,t} + \varepsilon}(V_{db}) \quad 3\text{-}25$$

For training of ANN and for partial weights training, we are using ADAM optimizer and in that, we have to take initial values for parameters like $V_{db}, V_{dw}, \gamma_{w,t}, \gamma_{b,t}$ to start the process and these values are arbitrary.

3.4 Conclusion

ANN is a data-driven modelling and its mathematical representation is in linear algebra. ANNs are straightforward to train due to their simple structure. During training, the model makes predictions, which are then compared to actual data to determine errors. This error information is utilised to adjust weights on the links to the nodes. We employ Adam as an optimizer in the error reduction process to improve the weight updating strategy. Following model predictions, EKF is used to filter the predictions based on the plant's measured data.

Chapter 4

Approach for Model training and EKF implementation

Estimators such as EKF, UKF, MHE, and others are used to tackle the state estimation issue in nonlinear systems. The estimator requires a system model to anticipate the states. Capturing the dynamics of a nonlinear system via a mechanistic method is time-consuming. As mentioned in Section 3.1, artificial neural networks can be employed for the same purpose. Thus, the following methodologies were utilised to solve the state estimation issue, and the findings are presented in this chapter:

- The proposed approach is ANN with EKF (ANN-EKF)
- Relu as an activation function for ANN with EKF

4.1 System Identification using ANN

Because every estimator requires a model, the suggested technique uses an artificial neural network to capture the system's dynamics. This section contains a full explanation of the ANN's training. Different modelling methodologies for capturing system dynamics using ANN are offered in the various literatures. Most commonly used are [12]:

1. Parallel mode
2. Series-parallel mode

In parallel mode, the neural network's current instant output is forecasted using system inputs and previous outputs anticipated by the neural network. In the series-parallel mode, the network's current forecast is formed utilising the system's manipulated inputs and historical plant outputs. The series-parallel model is preferable to parallel mode as it generates stable adaptive results [13]. Hence series-parallel methods have been used for training the ANN model.

4.1.1 Data Generation and Pre-processing

To begin modelling the system, open-loop simulation in MATLAB was utilized to produce data that would be used to train the neural network. As a result, a system with a

differential equation that represents the plant is required. Furthermore, the sample time of the system is determined by considering the system's dynamics. In MATLAB, two simulations are run to generate data. The data used for ANN training was created without the addition of any noise, i.e., neither process nor measurement noise was employed to generate the data.

Subsequently, one additional simulation that included process and measurement noise was employed to obtain data that corresponded to the actual plant data. This data is used by the EKF to create filtered state values. The system's inputs (manipulated inputs) and output variables (measured and unmeasured) are saved as distinct CSV files from each simulation. Because we are using a series-parallel model, the input to our model will be the manipulated variables of the systems and previous output values from the plant. As a result, in the input CSV file, past time instant true state values are supplemented at each current time instant to eventually form the input CSV, which contains all of the variables to be utilised. At the same time, the output CSV contains the system's process states, which are utilised as the target variable. The input fed to the ANN is in the following form:

$$S \times F \tag{4-1}$$

Where,

- S: Number of Samples,
- F: Number of features

This data is split into the following:

- Training dataset
- Validating dataset
- Test dataset

The data was split into the above 3 sections in the ratio of 80:10:10. In which training data set was used for training of the network. Whereas, the rest of 2 sections are used to check the performance of the model.

4.1.2 Hyperparameter Tunning of the ANN

The term "hyperparameter tuning" refers to fine-tuning the hyperparameters of a neural network in order to improve its performance. Though there is no perfect analytical

method for determining the ideal network design, it is usual practice to start with a network with one hidden layer, a few neurons, and a user-specified activation function. If the results aren't satisfactory, the layer's number of neurons is raised. If the results are still unsatisfactory, an additional hidden layer is added, and the process is repeated as can be seen in Figure 4-1. This is repeated until the model's accuracy stabilises or begins to decline.

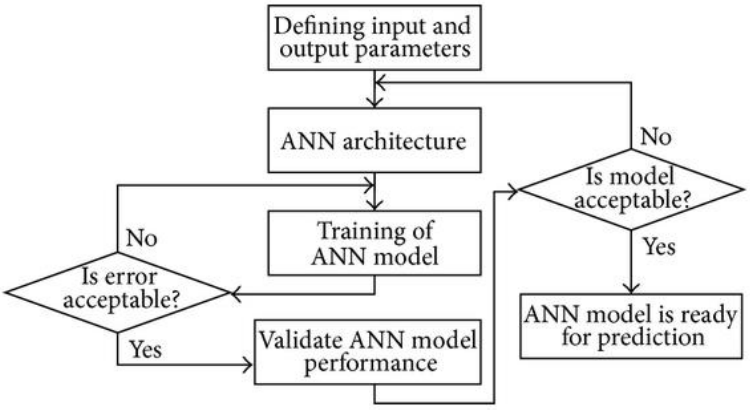

Figure 4-1: Training flowchart of ANN model

In addition to modifying network parameters, an appropriate optimizer and loss function is selected based on the task. In most cases, an Adam optimizer with a small learning rate is used, and the loss function is determined by the issue that the ANN is attempting to solve. The mean squared error is used as a loss function for regression issues, whereas the cross-entropy function is utilised for classification problems. The grid search approach is used to find the best values for the parameters. Each parameter is given a different range of values, and the model's accuracy is assessed using a grid made up of different combinations of these values. The field is then limited down to a short region where the model produces the highest accuracy. This process returns the optimal value for each parameter, completing the network's hyperparameter tuning. The model's performance is evaluated using the validated and test datasets after hyperparameter adjustment.

4.2 ANN with EKF (ANN-EKF)

EKF has been used to solve the state estimation problem. The ANN trained in section 4.1 was used as a model in the prediction step of the EKF, as every estimator requires a model. Furthermore, the ANN has been used to derive the necessary parameters for the EKF's correction step, which are discussed further down.

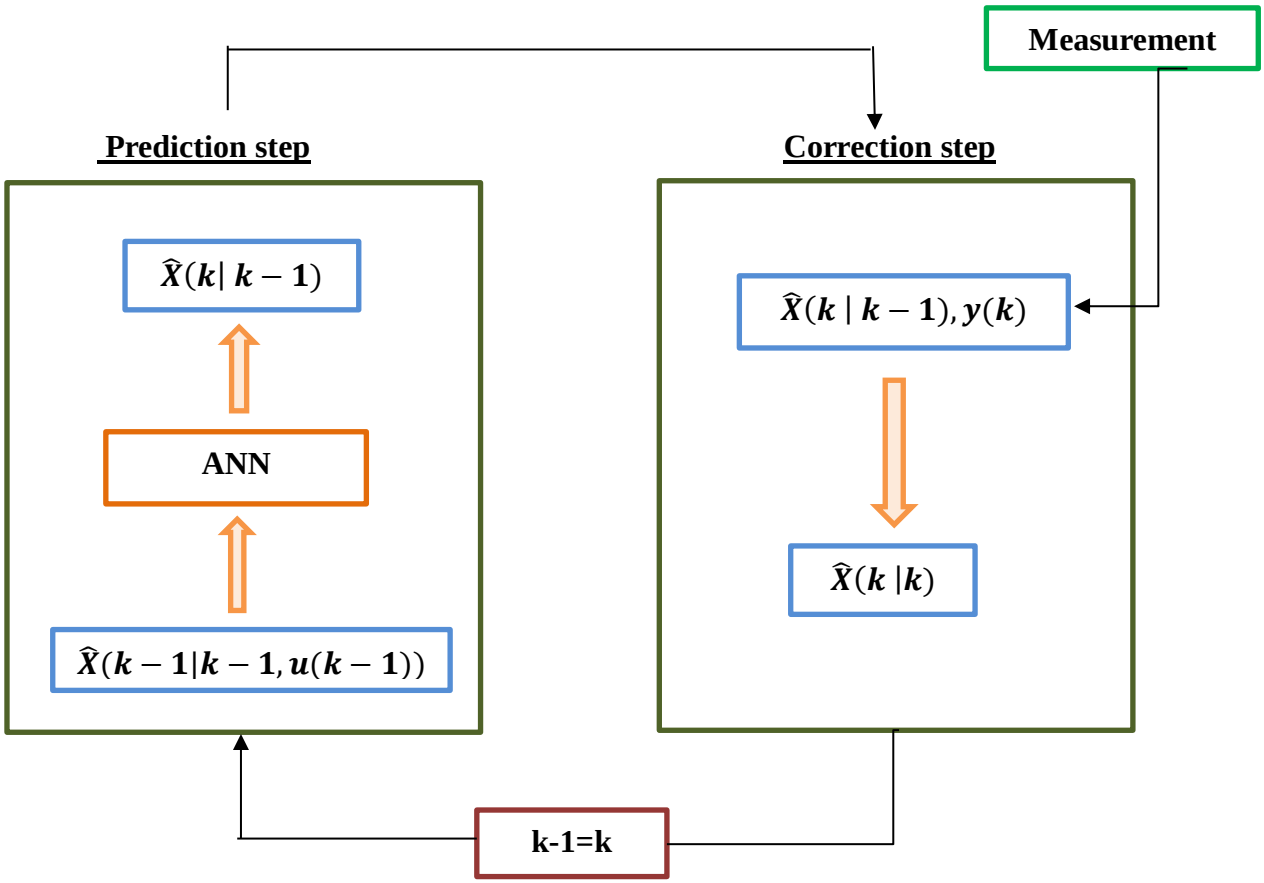

Figure 4-2 : flow chart for ANNEKF

4.2.1 EKF combined with ANN

Using the system's model generated in the above section 4.1, the state estimation problem has been solved using EKF. Filtered values of the state of the system are predicted using EKF as described in Chapter 3. The prediction step in EKF Eq. (3-8) can be replaced by the algebraic equation obtained from the artificial neural network model of the system as shown in Eq. (3-1). Further, the parameters required from the model like $\emptyset$ used in Eq. (3-9) can be derived from the algebraic ANN equations. The working of the ANN-EKF strategy has been shown in Figure 4-2. The way to derive the parameters has been taken from [14] and has been modified to use it for the ANN model as method given is for RNN. Method is explained in detail as below:

The algebraic equation that we get from the ANN as stated in Eq. (3-1) can be re written as below in terms of control relevant terminology:

$$x_{h1}(k+1) = f[W^u u(k) + W^y x^y(k) + b^1] \qquad 4\text{-}2$$

$$x_{h2}(k+1) = g[W^d f[W^u u(k) + W^y x^y(k) + b^1] + b^2] \quad 4\text{-}3$$

$$x^y(k+1) = W^x g[W^d f[W^u u(k) + W^y x^y(k) + b^1] + b^2] + b^3 \quad 4\text{-}4$$

Table 4-1: parameter for Artificial neural network

Parameters	**Definition**
$x_{h1}(k)$	Output vector of hidden layer 1
$x_{h2}(k)$	Output vector of hidden layer 2
$x^y(k)$	Output vector of the neural network at time k
u(k)	Vector of manipulated variables of the system
W^u	Weight matrix connecting the hidden units with the manipulated inputs u(k)
W^y	Weight matrix connecting the hidden units with the state inputs x^y (k-1)
W^d	Weight matrix for connection between hidden layer 1 & 2
W^x	Weight matrix connecting the $x_{h2}(k)$ with output layer of neural network
b^1	Input biases
b^2	Hidden layer biases
b^3	Output layer biases
$f(.)$	Activation function for hidden unit
$g(.)$	Activation function for output unit

Considering the ANN stated in equations (4-2), (4-3) and (4-4), to derive ∅ from these equations, we formulate an augmented state model in the sequel. We will now generate ∅ from the ANN for the CSTR system. Below are the equations obtained after differentiating Eq. (4-2), (4-3) and (4-4) with the augmented states x_{h1}, x_{h2} and x^y.

$$f \xrightarrow{\max(x,0)} 0 \quad 4\text{-}5$$

$$f \xrightarrow{\max(x,0)} \begin{cases} \frac{\partial f}{\partial x^y} = W^y \\ \frac{\partial f}{\partial x_{h1}} = 0 \\ \frac{\partial f}{\partial x_{h2}} = 0 \end{cases} \quad 4\text{-}6$$

$$g \xrightarrow{\max(x,0)} 0 \quad 4\text{-}7$$

$$g \xrightarrow{\max(x,0)} \begin{cases} \frac{\partial g}{\partial x^y} = W^d W^y \\ \frac{\partial g}{\partial x_{h1}} = 0 \\ \frac{\partial g}{\partial x_{h2}} = 0 \end{cases} \quad 4\text{-}8$$

$$Z \xrightarrow{Linear} \begin{cases} \frac{\partial z}{\partial x^y} = W^x W^d W^y \\ \frac{\partial z}{\partial x_{h1}} = 0 \\ \frac{\partial z}{\partial x_{h2}} = 0 \end{cases} \quad 4\text{-}9$$

Hence using differentiating values from Eq. (4-6), (4-8) and (4-9) $\emptyset$ matrix is generated:

$$\emptyset = [W^x W^d W^y]_{2\times 2} \quad 4\text{-}10$$

Here in Eq. (4-6), (4-8) and (4-9), Relu is used as a activation function and following is the derivation of Relu activation function

$$R(z) = \max(0, z) \quad 4\text{-}11$$

$$R(z)' = \begin{cases} 1 \; if \; z \geq 0 \\ 0 \; is \; z < 0 \end{cases} \quad 4\text{-}12$$

While generating initial $\emptyset$ as given in Eq. 4-10, it is assumed that all the weights are positive and while implementation of $\emptyset$ matrix in EKF we update the $\emptyset$ matrix at each

instant by making negative weights zero so that we can also consider the Relu activation function while differentiating the algebraic equations of ANN model. As in algebraic equations of ANN model (Eq. (4-2), (4-3) and (4-4)) there is no disturbance term is present (Ca0 is disturbance) so the γ_d term in EKF becomes zero and due to that whole $\gamma_d Q {\gamma_d}^T$ becomes zero. Hence using the above analogy, the final output of the EKF is filtered values of the states of the system and values of the augmented states, which were the hidden nodes of the ANN are used to update $\emptyset$ at every iteration. Filtered values of the system states are fed back to the ANN, which gives prediction for the system states, which are again fed to EKF with values of hidden nodes, and this cycle is repeated.

4.2.2 Activation function used for ANN-EKF

Various activation functions can be employed in ANN, and the following Relu activation function has been investigated for the ANN-EKF strategy. In this strategy, ANN-EKF used the Relu activation function inside the ANN to capture the physics of the system. The Rectified Linear Unit (ReLU) is represented theoretically as Eq. (4-13) and graphically as Figure 4-3. Many neural networks now use it as their default activation function.

$$R(z) = \max(0, z) \qquad 4\text{-}13$$

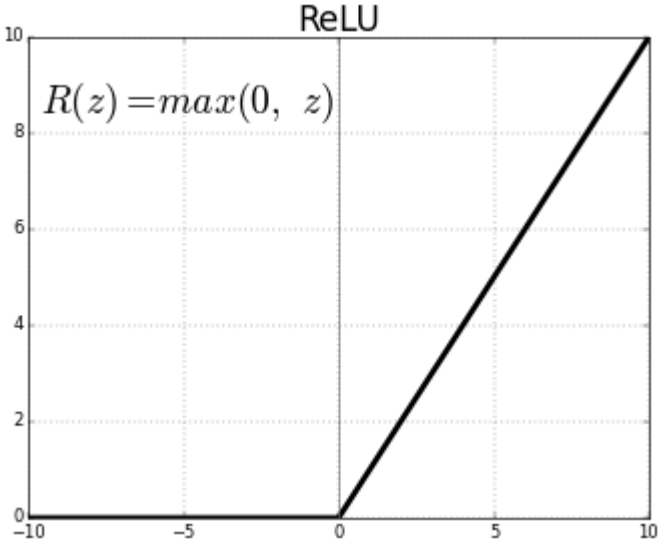

Figure 4-3: Rectified Linear Unit

4.3 Conclusion

A series-parallel strategy was employed to create a dataset for the ANN model, in which manipulated inputs with previous output states of the plant were used as input to the next time instant. Differential equations are used to create data in MATLAB. The ANN model dataset has been partitioned into train, validation, and test datasets in the ratio of 80:10:10 for training and validation purposes. After the hyperparameter tuning, the ANN model with two hidden layers and eight neurons produced good results. EKF was applied using an ANN model to tackle the state estimation problem, and a technique is also provided to find ∅ from ANN.

Chapter 5

Case Study

After discussing the approach for model training and EKF implementation with ANN in Chapter 4 Now in this chapter training of an artificial neural network model for the CSTR and how well the ANN model performed after being trained (e.g., Error values, number of epochs used for training and learning rate) is discussed. This chapter compares the computing time of the ANN model with the physics-based model, as well as the graphical representation of trained model predictions on training, validation, and test datasets. The predictions of the ANN model using the EKF state estimator are reviewed at the end.

5.1 Continuous Stirred Tank Reactor (CSTR)

This case study consists of nonlinear process dynamics and measurement dynamics is linear in nature. Considering the irreversible first-order reaction as given in Eq. (5-1)

$$\mathrm{A} \rightarrow B \tag{5-1}$$

From the first principles [15], the model of the reactor can be obtained as

$$\frac{dX_1}{dt} = f_1(X_1, X_2, U_1, U_2, D)$$

$$\frac{dX_1}{dt} = \frac{U_2}{V}(D - X_1) - k_0 X_1 \exp\left(-\frac{E}{RX_2}\right) \tag{5-2}$$

$$\frac{dX_2}{dt} = f_2(X_1, X_2, U_1, U_2, D)$$

$$\frac{dX_2}{dt} = \frac{U_2}{V}(T_0 - X_2) + \frac{(-\Delta H_r)k_0}{\rho C_p} X_1 \exp\left(-\frac{E}{RX_2}\right) - \frac{Q}{V\rho C_p} \tag{5-3}$$

$$\mathrm{Q} = \frac{a(U_1)^{b+1}}{U_1 + \left(\frac{a(U_1)^b}{2\rho_c C_{pc}}\right)}(X_2 - T_{cin}) \tag{5-4}$$

Following are the variables used In the ODE given above:

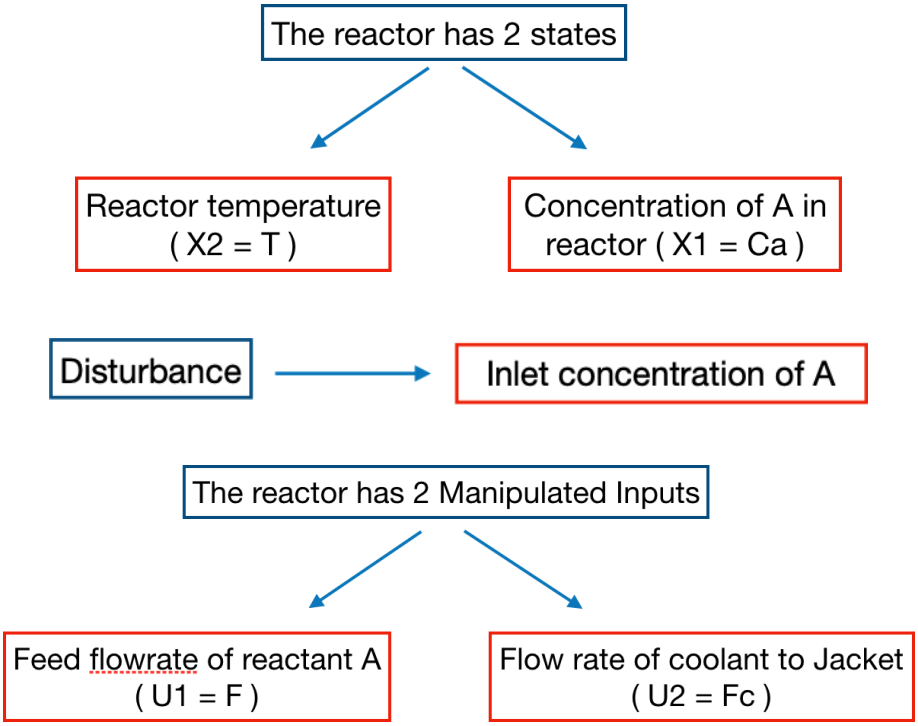

Figure 5-1: CSTR states description

As illustrated in Figure 5-1, all the states, Disturbance and Manipulative inputs have been discussed. State and measurement vectors are defined as follows:

$$x = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \quad y = I_{2\times 2}\, x \tag{5-5}$$

Non-isothermal CSTR reaction is a continuous system, which is discretized with a sampling interval (Ts) of 0.1 min. Following are the model parameter values used for the system.

Equilibrium Operating Condition of CSTR

Table 5-1: Equilibrium operating condition of CSTR

Parameter (↓) Operating point (→)		**Stable**
Inlet flow rate of A ($U_1 \equiv F$)	m^3/min	1
Coolant flow rate ($U_2 \equiv F_c$)	m^3/min	15
Reactor concentration of A ($X_1 \equiv C_A$)	$kmol/m^3$	0.265
Reactor temperature ($X_2 \equiv T$)	K	393.954
Inlet concentration of A ($D \equiv C_{A0}$)	$kmol/m^3$	2.0

Model Parameters of CSTR System

Table 5-2: Model parameters of the CSTR system

Parameter (↓) Operating point (→)		**Stable**
Reaction rate constant (k_0)	min^{-1}	10^{10}
Density of the reagent A (ρ)	g/m^3	10^6
Specific heat capacity of A (C_p)	cal/g^0C	1
Heat of reaction (ΔH_r)	$cal/kmol$	$-130 * 10^6$
Density of the Coolant (ρ_c)	g/m^3	10^6
Specific heat capacity of coolant (C_{pc})	cal/g^0C	1
Volume of the CSTR (V)	m^3	1
Inlet Temperature of the coolant (T_{cin})	K	365
Inlet Temperature of A (T_0)	K	363
a		$1.678 * 10^6$
Reaction rate parameter (E/R)	K^{-1}	8330
b		0.5

This case study consists of nonlinear process dynamics and measurement dynamics are linear in nature. For the data generation, simulation for non-isothermal CSTR reaction has been done with MATLAB version R2020a online. Where, 10% PRBS step changes were introduced in the manipulated inputs U_1 and U_2 for capturing effective dynamics of the system. Root Mean Square Error (RMSE) is used for measuring performance:

$$\mathbf{RMSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(X(k)-\hat{X}(k))^T(X(k)-\hat{X}(k))} \qquad 5\text{-}6$$

Where $X(k)$ and $\hat{X}(k)$ represent true state vector for k^{th} time instant and estimated state for k^{th} time instant respectively. Where, N represents the number of samples.

5.2 ANN model with EKF for CSTR

In this proposed approach the modelling of the system is being done with an artificial neural network and further, the states of the system are estimated by applying EKF on the model as described in section 4.2. The results have been reported below.

5.2.1 Modelling of CSTR using ANN

As discussed in section 4.1, datasets of the CSTR system have been generated and required pre-processing is done. Thus, for the training and testing of the neural network 16,666 points were generated in MATLAB using a mechanistic model of CSTR (Eq. 5-2, 5-3, 5-4). Below are the graphs of the manipulated input and states obtained from the simulation:

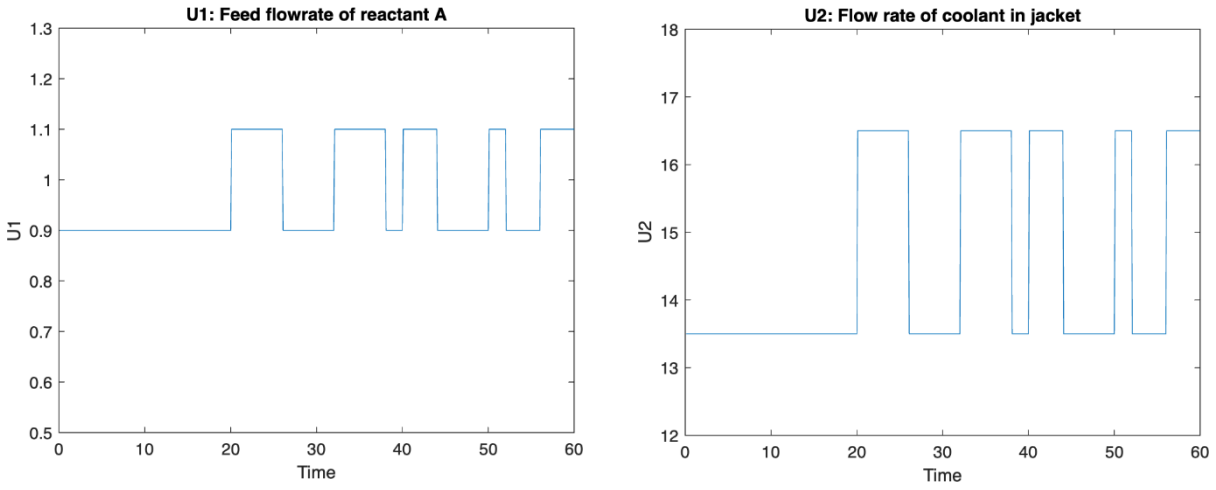

Figure 5-2: Manipulated input vs Time:
a) Feed flowrate vs time b) Coolant flowrate vs time

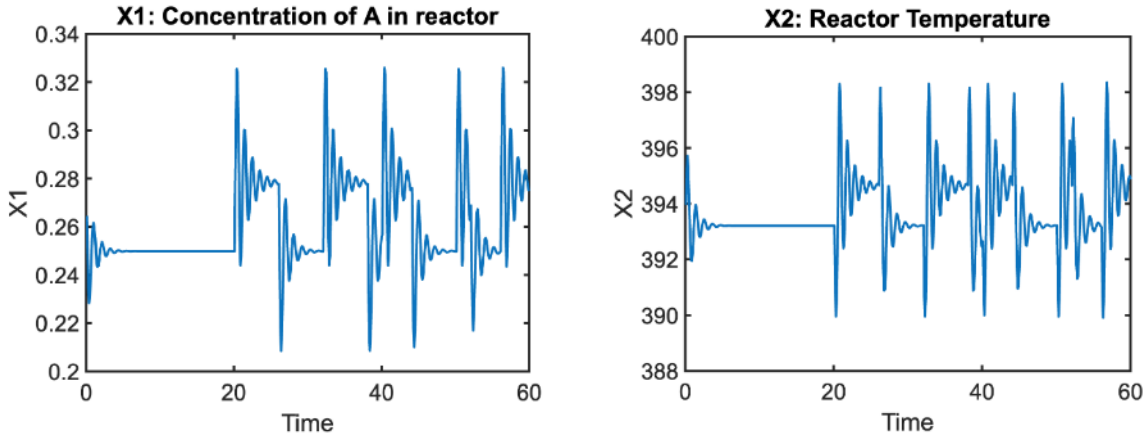

Figure 5-3: States input vs time
a) Concentration vs time b) Temperature vs time

As shown in Figure 5-3, the range of both the states is different hence, to incorporate the loss associated with concentration effectively concentration was scaled by a factor of 100 for training. Then this dataset was divided into 3 sets:

- Training dataset (13332 samples)
- Validating dataset (1666 samples)
- Test dataset (1668 samples)

The training dataset was used for the training of the artificial neural network and further for checking the performance of the network, validation and test dataset was used. MSE loss function was used to train the network as shown by Eq. (5-7) and Adam as the optimizer mentioned in section 3.3.

$$\frac{1}{N}\sum_{k=1}^{N}(X(k)-\hat{X}(k))^{T}(X(k)-\hat{X}(k)) \qquad 5\text{-}7$$

Finally, after hyperparameter tuning of the ANN following are the parameters that were used for the prediction of the states:

Table 5-3: Tuned Hyperparameters for CSTR ANN

Learning rate	Neurons in each hidden layer	Number of hidden layers	Epochs
0.00025	8	2	210000

5.2.1.1 Loss plots for CSTR-ANN

The values of the loss function with epochs are shown in Figure 5-4.

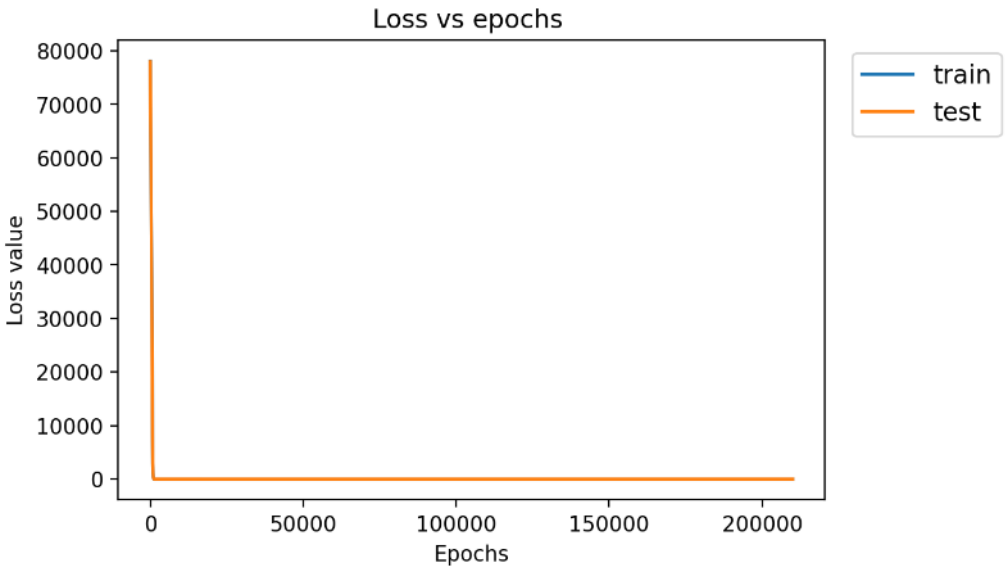

Figure 5-4: ANNEKF Loss vs Epochs

Further, due to the high range of loss values, Figure 5-5 has been plotted, which represents the loss values for <1500 epochs.

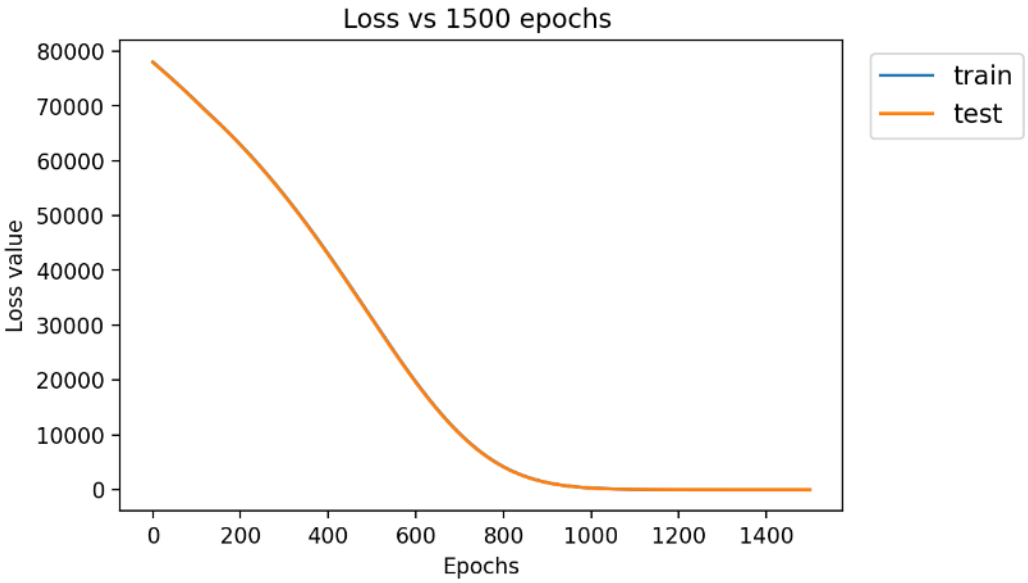

Figure 5-5: Loss vs 1500 Epochs

5.2.1.2 Predictions for CSTR-ANN

In this section, the predictions of ANN model are shown after training part for the datasets like training, validation and test datasets. We have considered to show only 2000 samples out of 13332 samples for the training dataset, so that we can see the predictions clearly.

- **<u>Predictions on Training dataset</u>**

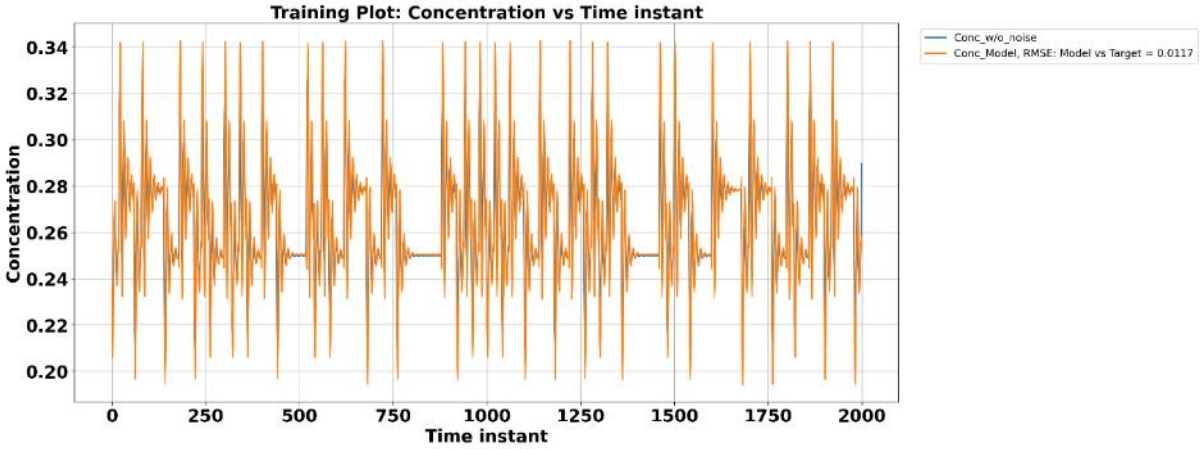

Figure 5-6: Concentration Vs Time instant for Train dataset

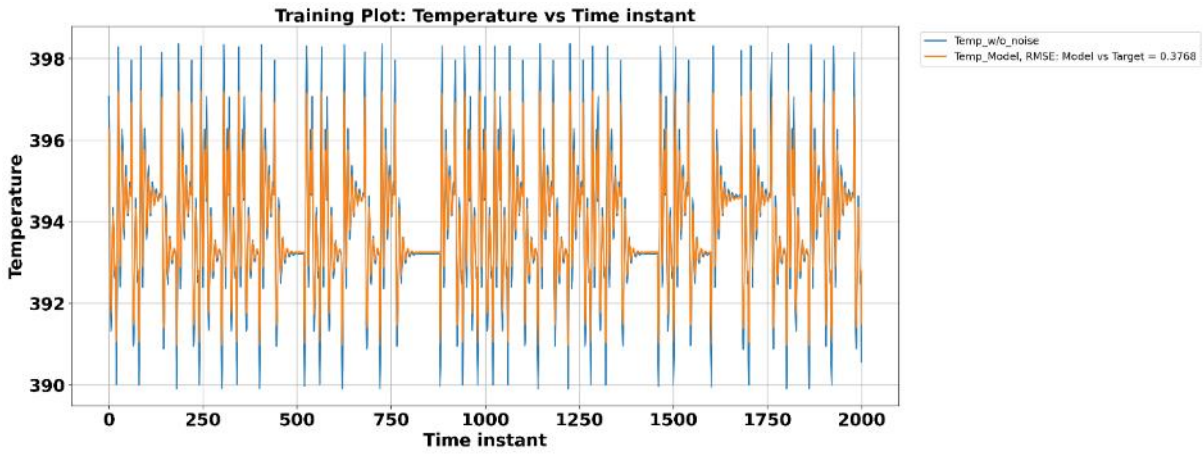

Figure 5-7: Temperature Vs Time instant for Train dataset

- **<u>Predictions on Validating dataset</u>**

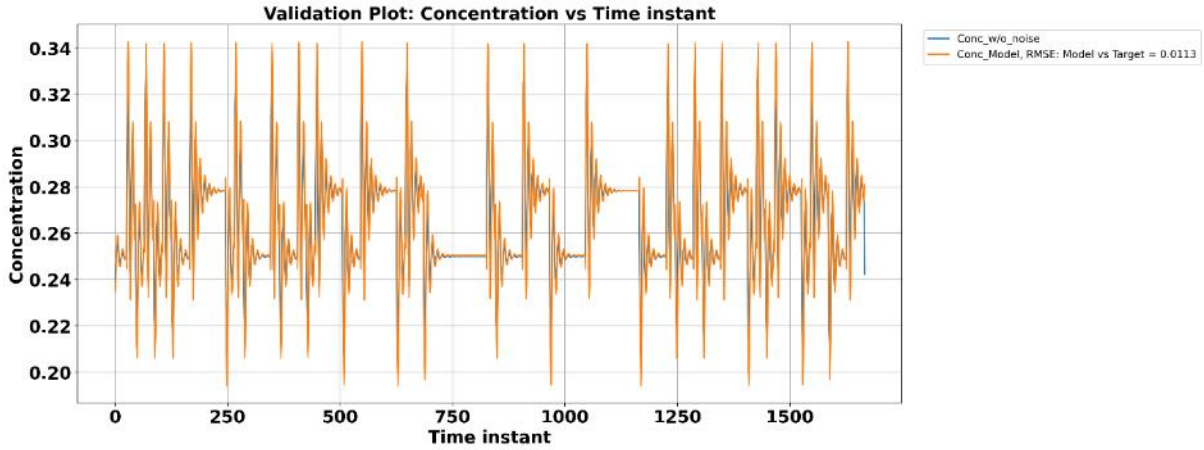

Figure 5-8: Concentration Vs Time instant for Validation dataset

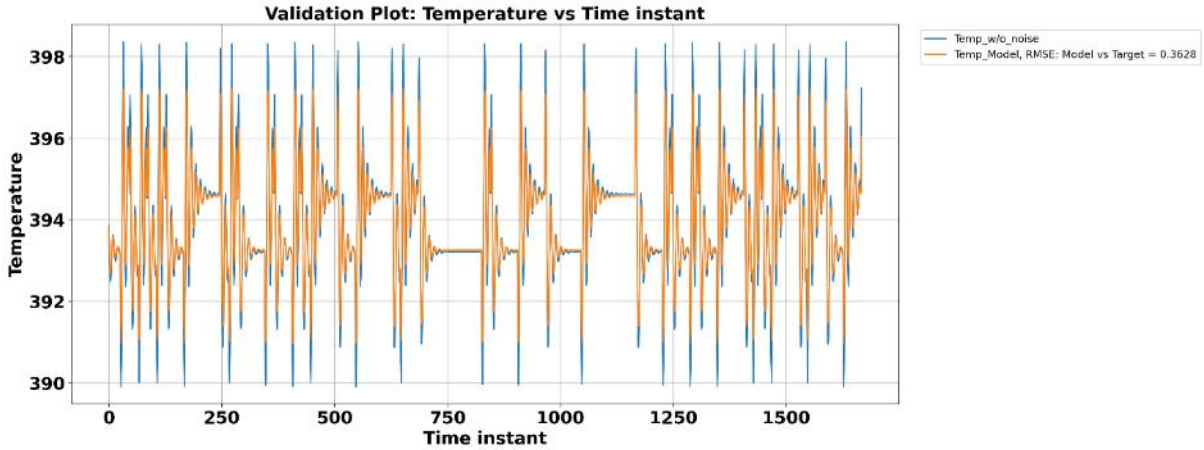

Figure 5-9: Temperature Vs Time instant for Validation dataset

- **Predictions on Test dataset**

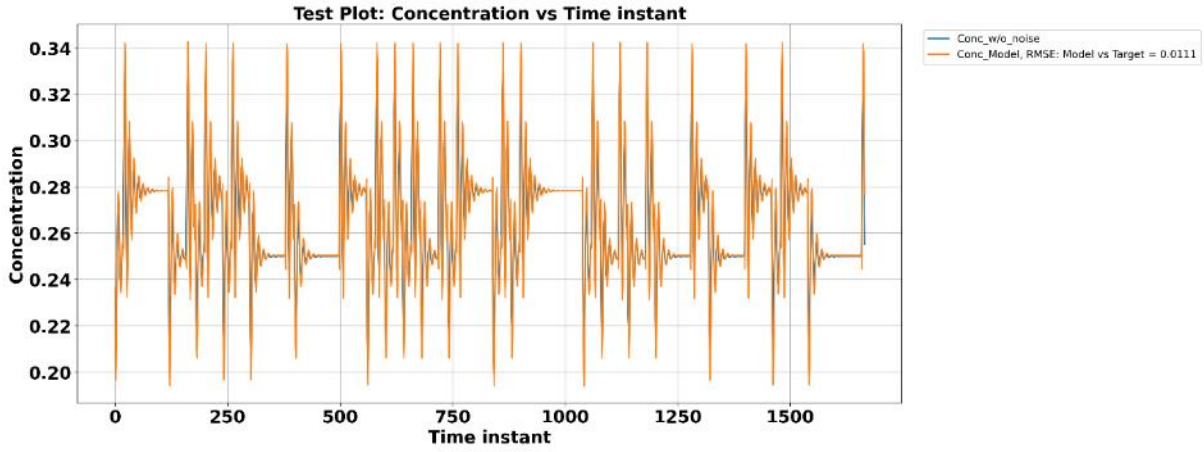

Figure 5-10: Concentration Vs Time instant for Test dataset

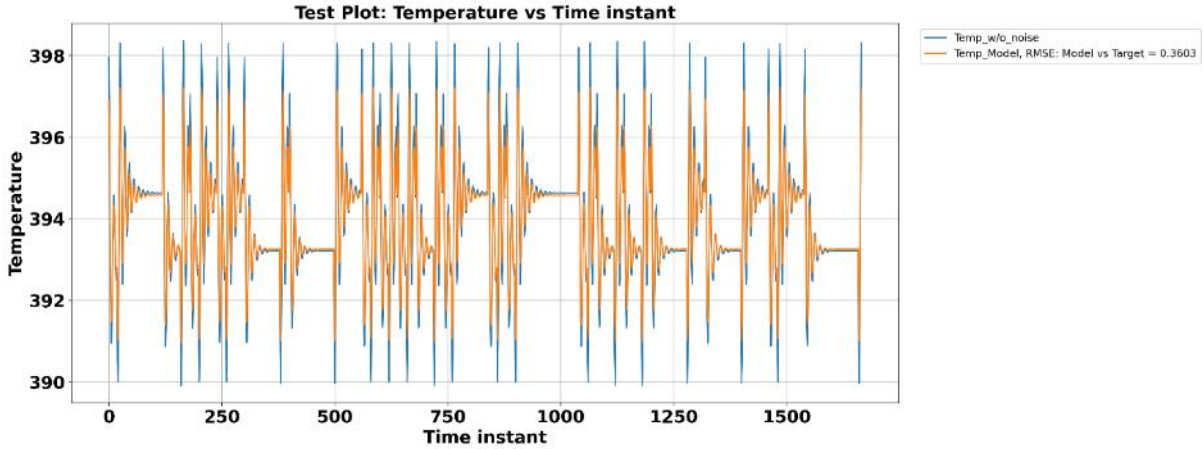

Figure 5-11: Temperature Vs Time instant for Test dataset

Summary of Trained Artificial neural network for CSTR is given in Table 5-4, in which the loss values for train dataset and test dataset with other specifications are mentioned.

Table 5-4: Summary of ANN model of CSTR

Learning rate	Neurons in each hidden layer	Number of hidden layers	Epochs	Train dataset MSE	Test dataset MSE
0.00025	8	2	210000	0.0711	0.0659

5.2.2 States estimated using ANN-EKF strategy for CSTR

In this strategy concentration and temperature both are taken as measured states. Using the model summarized in Table 5-4, filtered values of the states of the system are estimated using EKF as an estimator. The values of Q, R, C and P(0|0) have been taken from [16]. Following initial values of the states and covariance matrix were taken for the estimation:

$$x(0|0) = \hat{x}(0|0) = \begin{bmatrix} 0.265 \\ 393.954 \end{bmatrix} \quad 5\text{-}8$$

$$P(0|0) = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix} \quad 5\text{-}9$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad 5\text{-}10$$

Plant system process and measurements are incorporated with white Gaussian noise with zero mean and covariance matrix Q and R as:

$$Q = \begin{bmatrix} 0.012^2 & 0 \\ 0 & 0.012^2 \end{bmatrix} \quad 5\text{-}11$$

$$R = \begin{bmatrix} 0.01^2 & 0 \\ 0 & 0.15^2 \end{bmatrix} \quad 5\text{-}12$$

As discussed in section 4.1.2, the state-space model of CSTR has been generated from ANN. More detailed derivation for generation ∅ from ANN for the CSTR system has been explained in section 4.2.1. Thus, EKF was applied with the ANN model, and the results of combined ANN with EKF have been reported below:

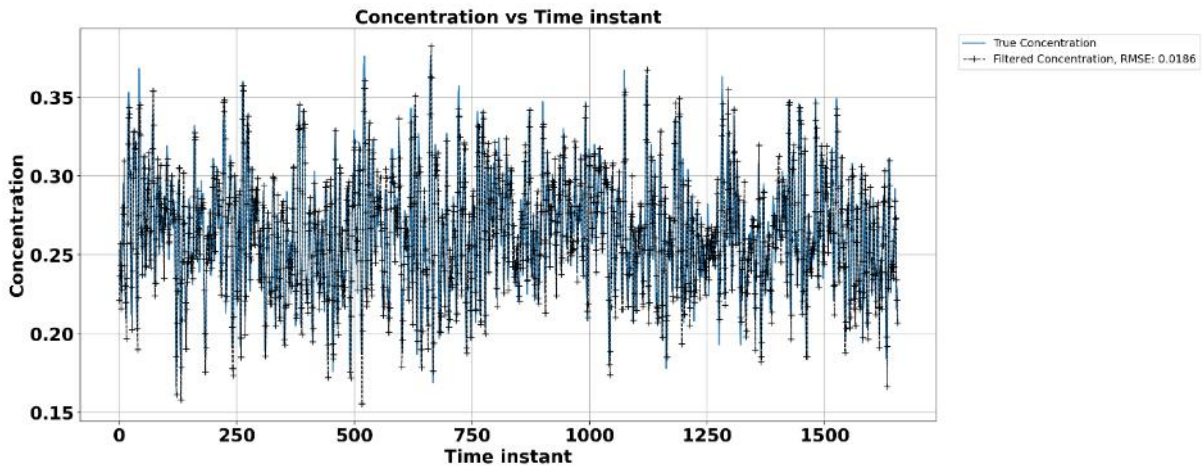

Figure 5-12:Concentration Vs Time by ANNEKF

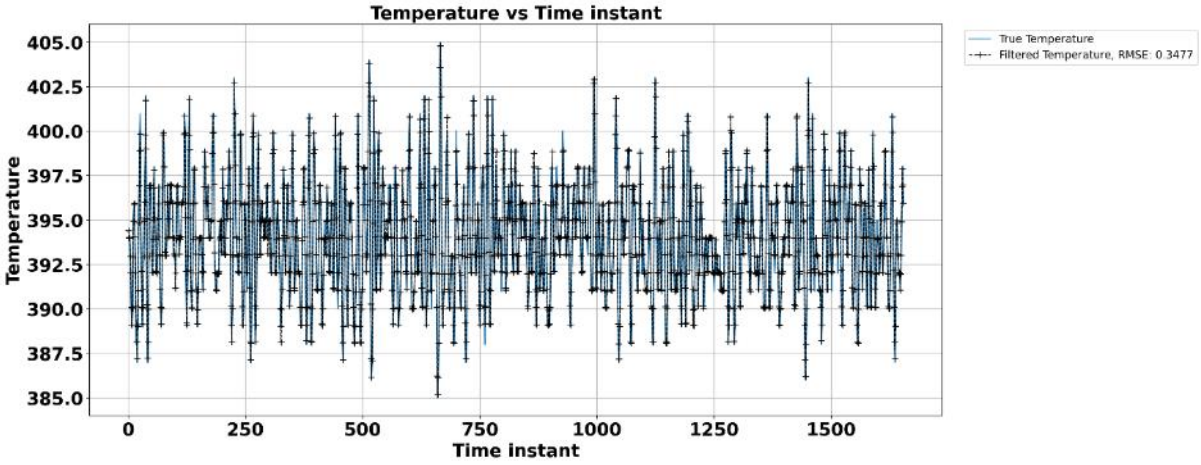

Figure 5-13: Temperature Vs Time by ANNEKF

Figure 5-12 and Figure 5-13 give the values of the filtered values of state from the EKF, which was derived from the ANN itself. The proposed idea of employing ANN as the model and combining it with EKF yields good results for the CSTR state estimation problem with low RMSE values, based on these findings. Physics based EKF is implemented on MATLAB and coupled ODE equations are solved using ode45 solver. Calculation of computational time has been done for physics based EKF in MATLAB itself. Calculation of time taken by ANN-EKF has been done on python and comparison has been between both of them. Table 5-5 shows that the RMSE of ANN-EKF and traditional physics-based EKF for both states are almost similar. Physics-based EKF took 12.45s to complete the computation whereas ANN-EKF took 1.16s to compute, which is 90.68 % faster than traditional EKF. As a result, ANN-EKF may be a better option than existing techniques because it requires less modelling time.

Table 5-5: summary of ANN-EKF and conventional EKF on CSTR

States	Proposed Work	RMSE	Computation time(s)
Temperature	ANN-EKF	0.347	1.16
Concentration	ANN-EKF	0.0186	1.16
Temperature	Physics EKF	0.2419	12.45
Concentration	Physics EKF	0.0064	12.45

5.3 Conclusion

CSTR operating conditions are represented using differential equations. The ANN modelling section includes loss vs epoch charts to visualise the model's training process. Plots of model predictions on train, validation, and test data are also presented. Finally, the results of the ANN model with the EKF estimator are shown, along with a comparison of the ANN-EKF with a physics-based model. In which we discovered that ANN-EKF is 90.68% faster than traditional EKF.

Chapter 6

Artificial Neural Network Weights Behaviour with CSTR Process Parameters

In this chapter brief information is given about the model which is trained in section 5.2.1. Also provided the structure of ANN model in which information about the distribution of weights and biases in ANN model is illustrated. Also, a discussion on behavioural changes of weights and biases with different process parameters of CSTR when a deviation in process parameter values is provided. This chapter also consists of categorization of weights based on their behaviour while training with different process parameters.

6.1 Weights and Biases distribution throughout the ANN Model

There is a total of 112 weights and 18 biases in the ANN model. The distribution of weights & biases is given as follows:

i) Between input layer and 1st hidden layer: There is a total of 32 weights connecting each input neuron to 8 neurons of 1st hidden layer (4*8=32) and there is a total of 8 biases for 1st hidden layer neurons.

ii) Between 1st hidden layer and 2nd hidden layer: There is a total of 64 weights connecting each neuron of 1st hidden layer to 8 neurons of 2nd hidden layer (8*8=64) and there is a total of 8 biases for 2nd hidden layer neurons.

iii) Between 2nd hidden layer and output layer: There are 16 weights connecting each 2nd hidden layer neuron to 2 output layer neurons (8*2=16) and there is a total of 2 biases for output layer neurons.

In Figure 6-1 structure of our Artificial neural network model with 2 hidden layers and 8 neurons in each hidden layer is shown. Also, there are 4 input neurons, among which 2 are manipulated inputs U1 and U2 and the other 2 are Temperature and Concentration which are outputs of a plant at a previous time instant. Apart from that ANN has 2 output neurons which are the final output of ANN model and they are Concentration and Temperature at the current time instant.

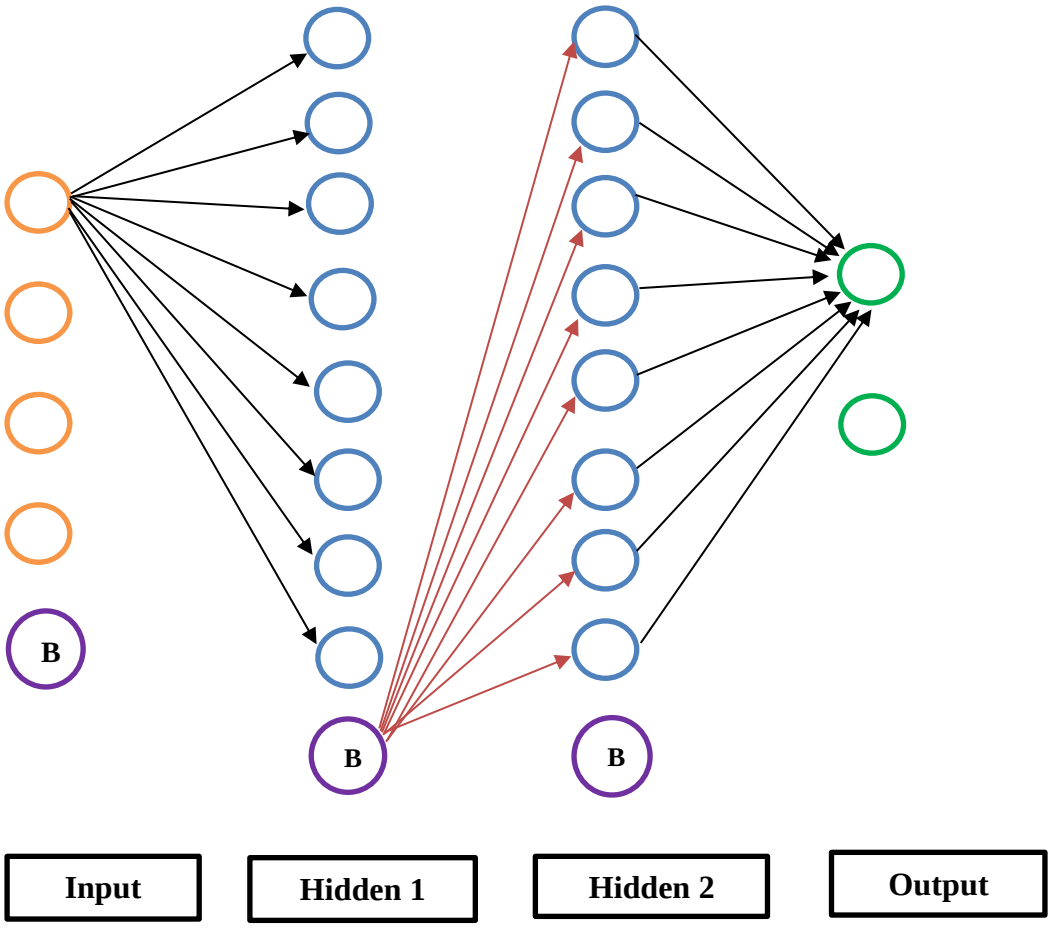

Figure 6-1: ANN Model Structure in detail

Where,

- Purple neurons represent the Bias neurons in ANN structure
- Orange neurons represent the Input neurons in ANN structure
- Blue neurons represent the Hidden layer neurons in ANN structure
- Green neurons represent the Output neurons in ANN structure

In Figure 6-1, an attempt to explain the number of weights and biases associated with each layer is made. So, each black arrow leading from 1st input neuron to all the hidden layer neurons contains weight and that is how ANN has 8 weights between 1st input neuron and 1st hidden layer neurons and another 24 weights between the other 3 input neurons and 1st hidden layer neurons, giving ANN a total of 32 weights between the input layer and the 1st hidden layer. Similarly, 64 weights can be defined between 1st and 2nd hidden layer neurons, with black arrows leading from all 2nd hidden layer neurons to the 1st output neuron, and ANN has 8 weights associated with that connection, as well as another 8 weights for the 2nd output, for a total of 16 weights between 2nd hidden layer and output layer.

Now, in Figure 6-1, one attempt is made to illustrate connections between the bias of the 1st hidden layer and the neurons of the 2nd hidden layer using red arrows, and as a result, the network contains eight biases between the 1st and 2nd hidden layers. In the same way, there are 8 biases between the input and the 1st hidden layer, and two biases between the 2nd hidden layer and the output layer in a network.

6.2 Weights and Biases behaviour with Heat of Reaction (ΔH_r)

A case study on CSTR has been given in Chapter 5, there in Table 5-2 Heat of reaction operating value is given as 130×10^6. For analysing the behaviour of weights and biases with changing values of Heat of Reaction, 100 different datasets have been generated using Heat of Reaction values in the range of $\pm 10\%$ of 130×10^6 which comes out in the range of (117×10^6 to 143×10^6) and trained the ANN model using each dataset from the collection of 100 different datasets. visualization of all the 112 weights and 18 biases for all the 100 different training sets have been done as explained distribution layer by layer in section 6.1. After examining the plots, the conclusion came out as, some of the weights and biases do not change at all for all 100 distinct values of Heat of reaction, while others show a lot of variation in their values after the model has been trained. There is a total of 17 plots for all of the weights and biases, but only 6 plots are presented here because they're sufficient to serve the purpose.

6.2.1 Analysis 1: between 1st input neuron and neurons of 1st hidden layer

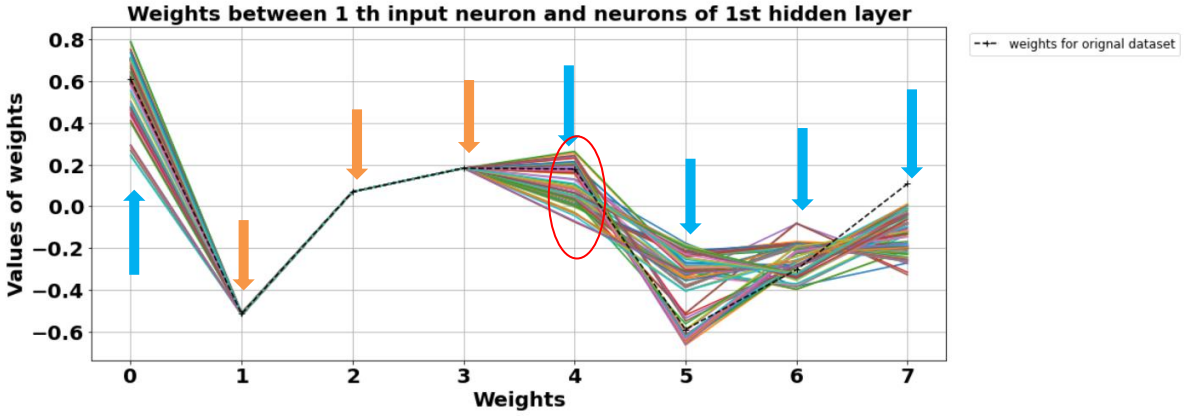

Figure 6-2: weights behaviour plot for connection between 1st input layer neuron and all the neurons of 1st hidden layer for heat of reaction

Where,

- Yellow arrows show that weights have remained constant for all the different values of Heat of Reaction
- Blue arrows show that weights have changed their values with changing values of Heat of Reaction

Figure 6-2 shows that the number of weights between the 1st input layer neuron and all of the neurons in the 1st hidden layer is 8, as indicated in section 6.1. As a result of the graph, it can be deduced that some weights, such as weights 2, 3, and 4 (in yellow arrows), remain constant even when the value of Heat of Reaction is changed, while others, such as weights 1, 5, 6, 7, and 8, (in blue arrows), change their values as the value of Heat of Reaction changes. Weight number 5 is pointed out through a circle because it has changed its behaviour with different process parameters, as shown in section 6.4.

6.2.2 Analysis 2: biases between the input layer and 1st hidden layer

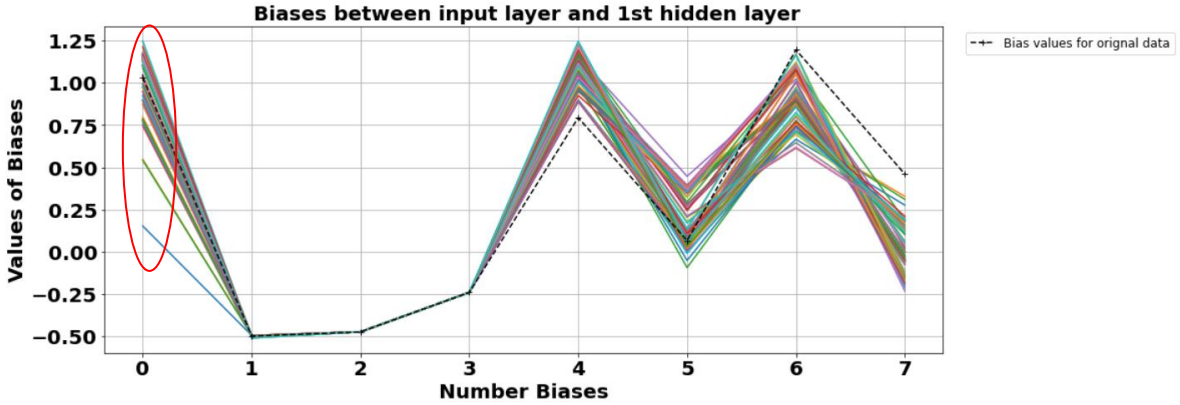

Figure 6-3: Biases behaviour plot for connection between input layer bias and 1st hidden layer neurons for heat of reaction

In section 6.1, a discussion has been carried out about how ANN has biases between any 2 layers, so that's how ANN has 8 biases between 1st hidden layer and input layer (refer to Figure 6-1 for more understanding). As discussed in analysis 1 about constant weights and changing weights, the same scenario is applicable here also. Bias 1 is pointed out in Figure 6-3 because it has shown a change in the behaviour when we have trained the ANN model for different parameter values (Rate constant of CSTR) and behavioural change can be seen in section 6.4.

6.2.3 Analysis 3: between 2nd neuron of 1st hidden layer and neurons of 2nd hidden layer

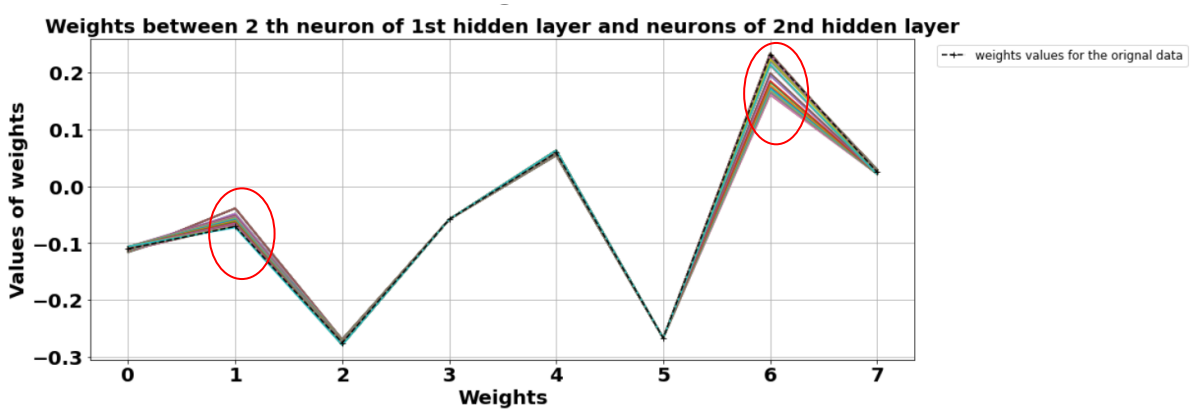

Figure 6-4: Weights behaviour plot for the connection between 2nd neuron of 1st hidden layer and all the neurons of 2nd hidden layer for heat of reaction

There are 8 weights between the Connection of 2nd neuron of 1st hidden layer and all the neurons of 2nd hidden layer as specified in section 6.1. In Figure 6-4, weight number 2 and weight number 7 are circled because they have shown a change in their behaviour when the ANN model is trained on different process parameter values.

6.2.4 Analysis 4: between 5th neuron of 1st hidden layer and neurons of 2nd hidden layer

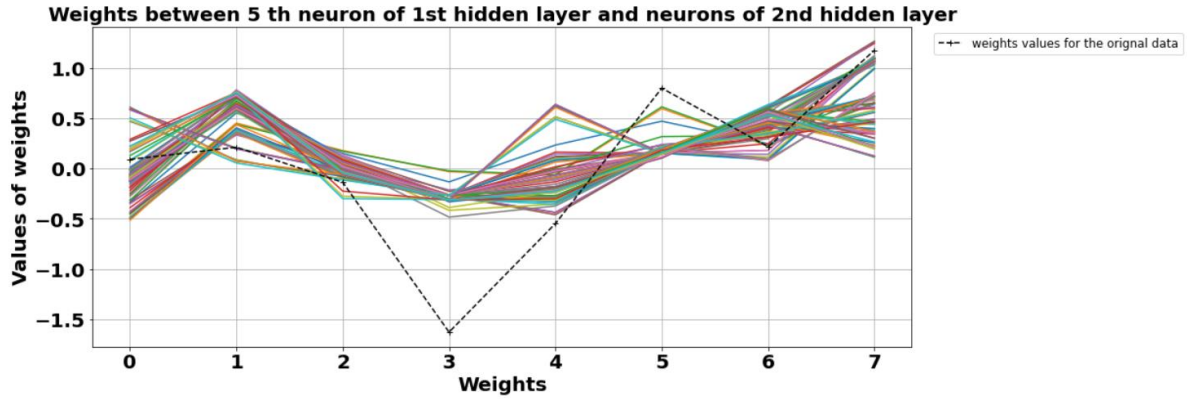

Figure 6-5:Weights behaviour plot for the connection between 5th neuron of 1st hidden layer and all the neurons of 2nd hidden layer for the heat of reaction

In Figure 6-5, all the weights have changed their values as the value of heat of reaction is changed. Many neurons of ANN model have shown this type of behaviour while training with different values of heat of reaction.

6.2.5 Analysis 5: between 2nd output neuron and 2nd hidden layer neurons

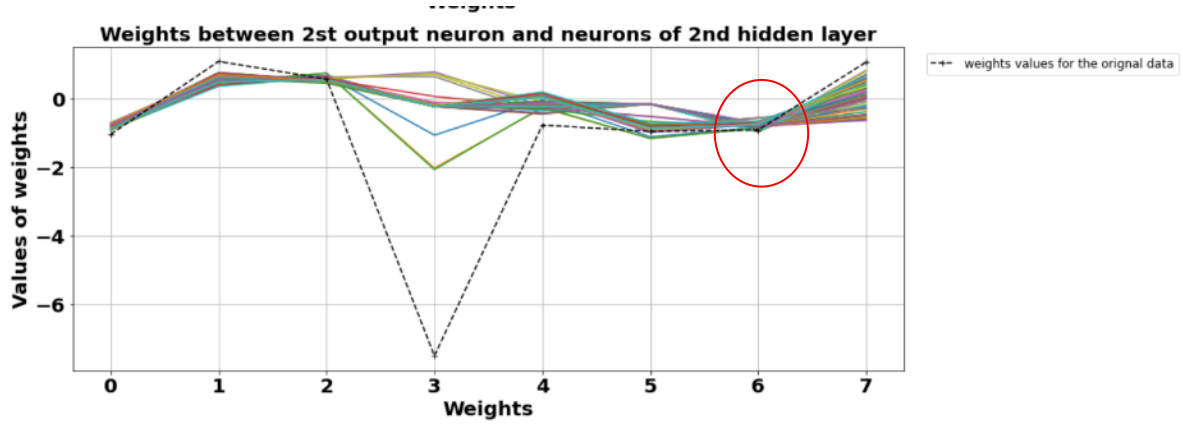

Figure 6-6: weights behaviour plot for the connection between all the neurons of 2nd hidden layer and 2nd neuron of output layer for the heat of reaction

Weight number 7 of 2nd hidden layer is circled in Figure 6-6 because that weight has changed the behaviour when the ANN model is trained on datasets that are prepared using different process parameter values. comparison will be carried out in upcoming sections.

6.2.6 Analysis 6: between 3rd neuron of 1st hidden layer and 2nd hidden layer neurons

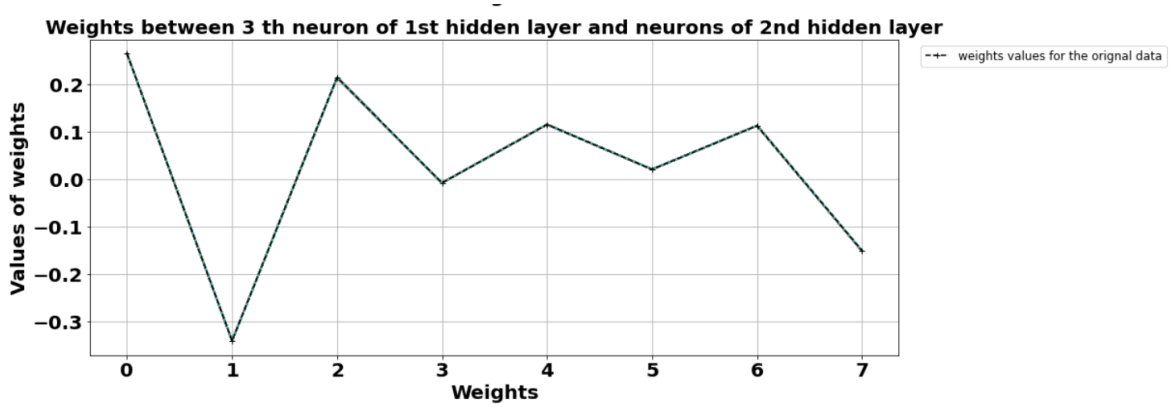

Figure 6-7: weights behaviour plot for the connection between 3rd neuron of 1st hidden layer and all the neurons of 2nd hidden layer for the heat of reaction

Figure 6-7 is presented just to throw light on some of the neuron's behaviour as all the weights remain constant for all the different values of heat of reaction. These kinds of

neurons have shown the same kind of behaviour as other process parameters also and all these behavioural data will be used when we partially train our ANN model.

6.3 Categorization of Weights and Biases for heat of reaction

Weights and biases are classified into three groups based on analysis. First, the weights and biases values have been normalized. To do this, all weights and biases trained on different values of heat of reaction are divided by the weights and biases obtained after training the ANN model on the original operating condition value of heat of reaction. So now there are 100 different values for each weight and bias because the ANN model was trained 100 times for 100 different training datasets. After normalization, 100 normalized values for any particular weight and bias were received, and maximum and minimum values were taken from that set of 100 normalized values. After that, weights and biases categorization was done based on those maximum and minimum values (**e.g**., let's say the threshold for category 1 is decided as 1.1 to 0.9, so if any particular weights maximum and minimum normalized value lies between 1.1 to 0.9 then that weight is classified in category 1).

Based on normalization we created 3 categories as follows:

i) **Category 1**: $\pm 10\%$ change in values of weights and biases $\{N \in (0.9,1.1)\}$

ii) **Category 2**: $\pm\ (10\%\ to\ 25\%)$ change in values of weights and biases $\{N \in (0.75,1.25)\}$

iii) **Category 3**: Greater than $\pm 25\%$ change in values of weights and biases {N<0.75 & N>1.25}

Where, N = Normalization value

Table 6-1: specification about weights name and their meaning

• Weights between output layer and 2nd hidden layer	**output HL2**
• Weights between input layer and 1st hidden layer	**input HL1**
• Weights between 1st hidden layer and 2nd hidden layer	**HL1 HL2**
• Biases from input to 1st hidden layer	**bias for HL1 neuron**
• Biases from 1st hidden to 2nd hidden layer	**bias for HL2 neuron**
• Biases from 2nd hidden layer to output layer	**bias for output**

Table 6-2: category 1 weights and biases for Heat of Reaction

Category 1 Weights & Biases	
'input HL1 W12'	'HL1 HL2 W32'
'input HL1 W13'	'HL1 HL2 W33'
'input HL1 W14'	'HL1 HL2 W34'
'input HL1 W22'	'HL1 HL2 W35'
'input HL1 W23'	'HL1 HL2 W36'
'input HL1 W24'	'HL1 HL2 W38'
'input HL1 W33'	'HL1 HL2 W41'
'input HL1 W34'	'HL1 HL2 W42'
'input HL1 W42'	'HL1 HL2 W43'
'input HL1 W43'	'HL1 HL2 W44'
'input HL1 W44'	'HL1 HL2 W45'
'HL1 HL2 W21'	'HL1 HL2 W46'
'HL1 HL2 W23'	'HL1 HL2 W47'
'HL1 HL2 W24'	'HL1 HL2 W48'
'HL1 HL2 W25'	'bias for HL1 neuron 2'
'HL1 HL2 W26'	'bias for HL1 neuron 3'
'HL1 HL2 W31'	'bias for HL1 neuron 4'
'HL1 HL2 W37'	

Table 6-3:category 2 weights and biases for Heat of Reaction

Category 2 Weights & Biases	
'input HL1 W32'	'HL1 HL2 W28'

Table 6-4:category 3 weights and biases for Heat of Reaction

Category 3 Weights & Biases	
'input HL1 W11'	'HL1 HL2 W72'
'input HL1 W15'	'HL1 HL2 W73'
'input HL1 W16'	'HL1 HL2 W74'
'input HL1 W17'	'HL1 HL2 W75'
'input HL1 W18'	'HL1 HL2 W76'
'input HL1 W21'	'HL1 HL2 W77'
'input HL1 W25'	'HL1 HL2 W78'
'input HL1 W26'	'HL1 HL2 W81'
'input HL1 W27'	'HL1 HL2 W82'
'input HL1 W28'	'HL1 HL2 W83'
'input HL1 W31'	'HL1 HL2 W84'
'input HL1 W35'	'HL1 HL2 W85'
'input HL1 W36'	'HL1 HL2 W86'
'input HL1 W37'	'HL1 HL2 W87'
'input HL1 W38'	'HL1 HL2 W88'
'input HL1 W41'	'output HL2 W11'
'input HL1 W45'	'output HL2 W12'
'input HL1 W46'	'output HL2 W13'
'input HL1 W47'	'output HL2 W14'
'input HL1 W48'	'output HL2 W15'
'HL1 HL2 W11'	'output HL2 W16'

'HL1 HL2 W12'	'output HL2 W17'
'HL1 HL2 W13'	'output HL2 W18'
'HL1 HL2 W14'	'output HL2 W21'
'HL1 HL2 W15'	'output HL2 W22'
'HL1 HL2 W16'	'output HL2 W23'
'HL1 HL2 W17'	'output HL2 W24'
'HL1 HL2 W18'	'output HL2 W25'
'HL1 HL2 W22'	'output HL2 W26'
'HL1 HL2 W27'	'output HL2 W27'
'HL1 HL2 W51'	'output HL2 W28'
'HL1 HL2 W52'	'bias for HL1 neuron 1'
'HL1 HL2 W53'	'bias for HL1 neuron 5'
'HL1 HL2 W54'	'bias for HL1 neuron 6'
'HL1 HL2 W55'	'bias for HL1 neuron 7'
'HL1 HL2 W56'	'bias for HL1 neuron 8'
'HL1 HL2 W57'	'bias for HL2 neuron 1'
'HL1 HL2 W58'	'bias for HL2 neuron 2'
'HL1 HL2 W61'	'bias for HL2 neuron 3'
'HL1 HL2 W62'	'bias for HL2 neuron 4'
'HL1 HL2 W63'	'bias for HL2 neuron 5'
'HL1 HL2 W64'	'bias for HL2 neuron 6'
'HL1 HL2 W65'	'bias for HL2 neuron 7'
'HL1 HL2 W66'	'bias for HL2 neuron 8'
'HL1 HL2 W67'	'bias for output1',
'HL1 HL2 W68'	'bias for output2'
'HL1 HL2 W71'	

In category 1, there are 35 weights + biases that show a $\pm 10\%$ deviation, in category 2 there is a total of 2 weights + biases that shown a $\pm 10\%$ to 25% deviation, and in category 3 there are total 93 weights + biases that show a deviation more than $\pm 25\%$. Section 6.4 will demonstrate how these weights change their behaviour with another process parameter which is rate constant of CSTR.

6.4 Weights and Biases behaviour with Rate Constant (K_0)

In Chapter 5, there is a case study on CSTR, and the rate constant operating value is stated as 10×10^9 in Table 5-2. For analysing the behaviour of weights and biases with changing values of rate constant, 100 different datasets of measurement states, true states, and model training datasets have been generated using rate constant values in the range of $\pm 10\%$ of 10×10^9 which comes out in the range of (9×10^9 to 11×10^9) and trained the ANN model using each dataset from the collection of 100 different datasets.

visualization of all the 112 weights and 18 biases for all the 100 different training sets have been done as explained distribution layer by layer in section 6.1. After examining the plots, the conclusion came out as, some of the weights and biases do not change at all for all 100 distinct values of rate constant, while others exhibit a significant shift in their values after

the model has been trained. There is a total of 17 plots for all of the weights and biases, but only 6 plots are presented here because they are sufficient to serve the purpose.

6.4.1 Analysis 1: between 1st input neuron and neurons of 1st hidden layer

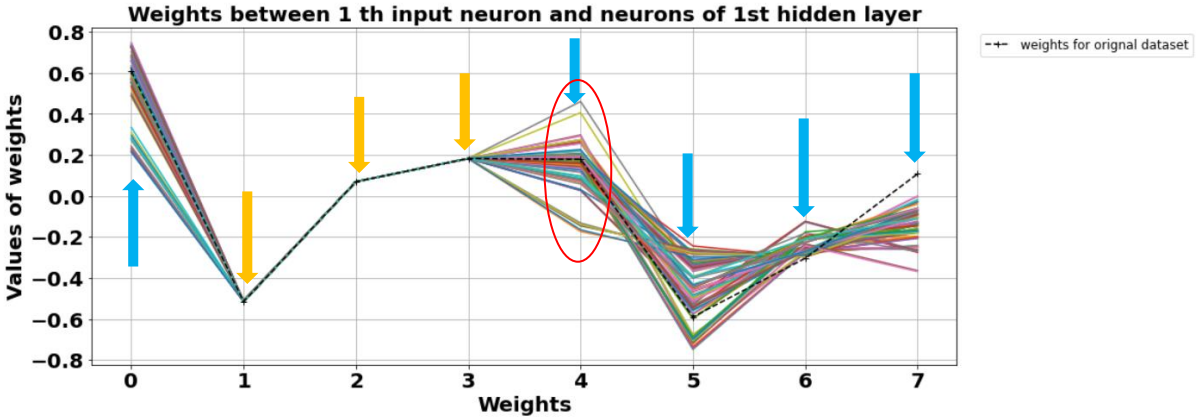

Figure 6-8: weights behaviour plot for connection between 1st neuron of input layer and all the neurons of 1st hidden layer for rate constant

Where,

- Yellow arrows show that weights have remained constant for all the different values of Heat of Reaction
- Blue arrows show that weights have changed their values with changing values of Heat of Reaction

Figure 6-8 shows that the number of weights between the 1st input layer neuron and all of the neurons in the 1st hidden layer is 8, as indicated in section 6.1. As a result of the graph, it can be deduced that some weights, such as weights 2, 3, and 4 (in yellow arrows), remain constant even when the value of rate constant is changed, while others, such as weights 1, 5, 6, 7, and 8, (in blue arrows), change their values as the value of rate constant changes.

Here comparison can be made for circled weight in Figure 6-8 with the same circled weight in Figure 6-2 and it's showing more deviation as we trained the ANN model on Rate constant datasets compared to our ANN model was trained on Heat of reaction datasets and other weights are showing moreover same kind of behaviour for both the parameters

6.4.2 Analysis 2: biases between the input layer and 1st hidden layer

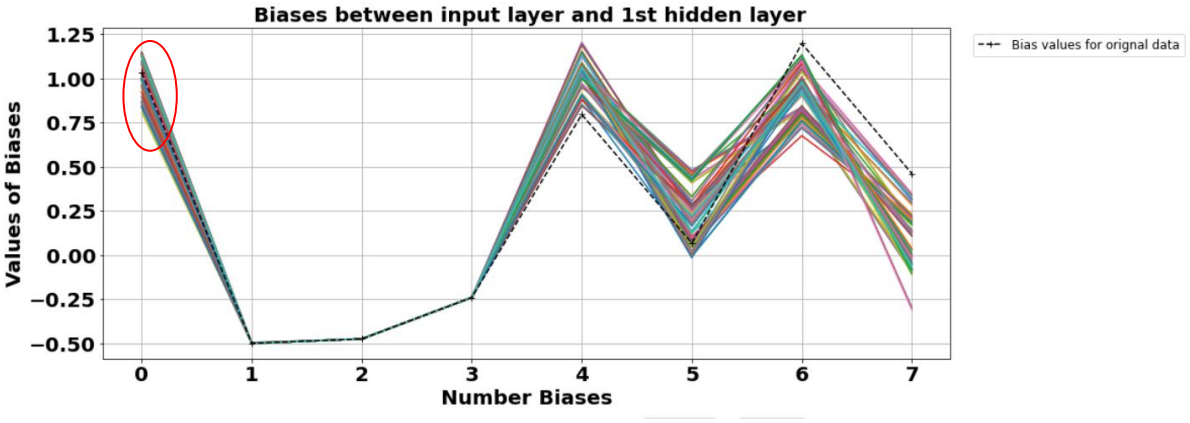

Figure 6-9: biases behaviour plot for the connection between input layer bias and all the neurons of 1st hidden layer for the rate constant

In section 6.1, a discussion has been carried out about how ANN has biases between any 2 layers, so that's how ANN has 8 biases between 1st hidden layer and input layer (refer to Figure 6-1 for more understanding). As discussed in analysis 1 about constant weights and changing weights, the same scenario is applicable here also. Bias 1 is pointed out in Figure 6-9 because it has shown a change in the behaviour when we have trained the ANN model for rate constant of CSTR. From this comparison, we can say that bias 1 has shown less deviation for rate constant compared to the heat of reaction and other biases have shown moreover same kind of behaviour for both the parameters.

6.4.3 Analysis 3: between 2nd neuron of 1st hidden layer and neurons of 2nd hidden layer

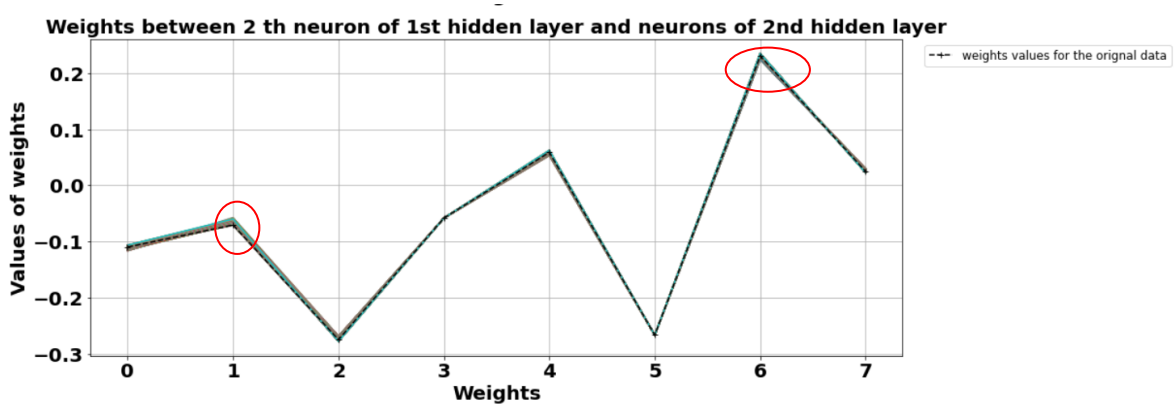

Figure 6-10: weights behaviour plot for connection between 2nd neuron of 1st hidden layer and all the neurons of 2nd hidden layer for rate constant

Weight number 2 and weight number 7 in Figure 6-10 are circled because changes can be seen in behaviour as ANN model is trained with rate constant datasets compared to Figure 6-4 where the ANN model is trained with the heat of reaction datasets. From both the figures, a conclusion can be made that the weights number 2 & 7 have shown less deviation with rate constant compared to heat of reaction and other weights have shown moreover same kind of behaviour for both the parameters.

6.4.4 Analysis 4: between 5th neuron of 1st hidden layer and neurons of 2nd hidden layer

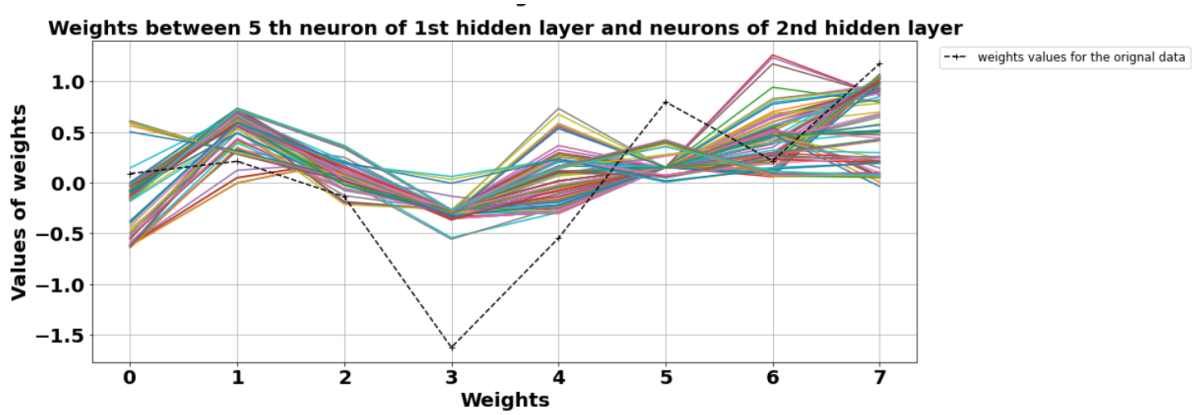

Figure 6-11: weights behaviour plot for connection between 5th neuron of 1st hidden layer and neurons of 2nd hidden layer for rate constant

Same as Figure 6-5, here also all the weights are changing with values of rate constant and no weight is showing constant behaviour.

6.4.5 Analysis 5: between 2nd output neuron and 2nd hidden layer neurons

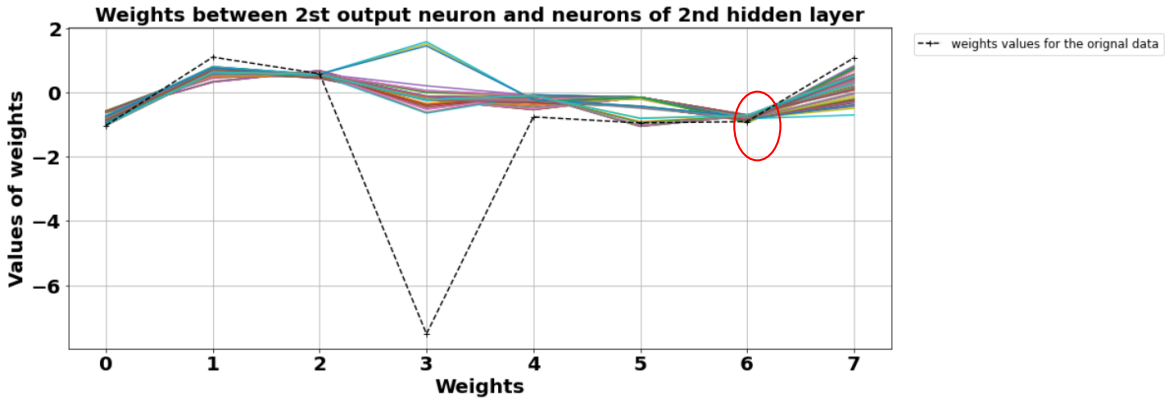

Figure 6-12: weights behaviour plot for the connection between all the neurons of 2nd hidden layer and 2nd neuron of output layer for the rate constant

Weight number 7 is circled in Figure 6-12 because changes can be seen in behaviour, as ANN model is trained with rate constant datasets compared to Figure 6-6, where the ANN model is trained with heat of reaction datasets. From both the figures, a conclusion can be made that the weight number 7 has shown less deviation with rate constant compared to heat of reaction and other weights have shown moreover same kind of behaviour for both the parameters.

6.4.6 Analysis 6: between 3rd neuron of 1st hidden layer and 2nd hidden layer neurons

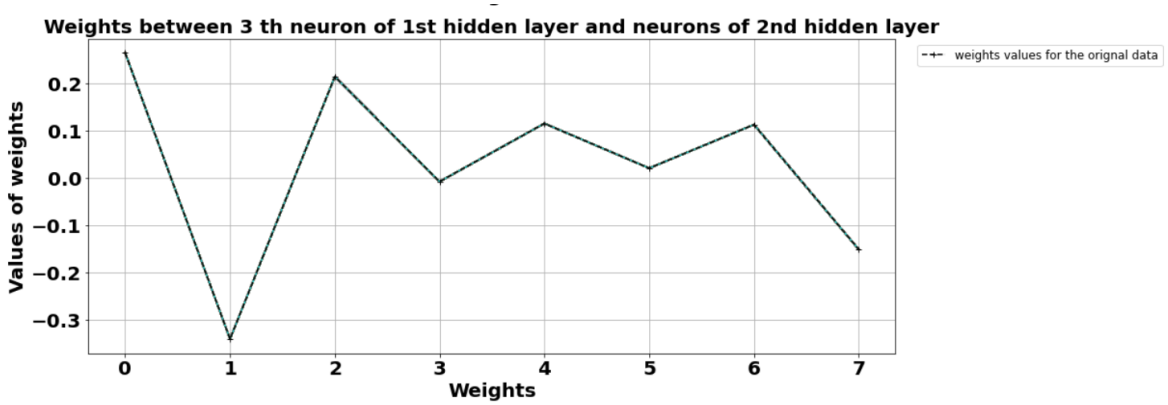

Figure 6-13: weights behaviour plot for the connection between 3rd neuron of 1st hidden layer and all the neurons of 2nd hidden layer for the rate constant

Here all the weights are showing the same behaviour as same as in Figure 6-7 for the heat of reaction.

6.5 Categorization of Weights and Biases for rate constant

The categorization strategy for rate constant is same as mentioned in section 6.3. Specifications about weight names and their meanings are mentioned in Table 6-1.

Table 6-5: category 1 weights and biases for Rate Constant

Category 1 Weights & Biases	
'input HL1 W12', 'input HL1 W13', 'input HL1 W14', 'input HL1 W22', 'input HL1 W23', 'input HL1 W24', 'input HL1 W33',	'HL1 HL2 W32', 'HL1 HL2 W33', 'HL1 HL2 W34', 'HL1 HL2 W35', 'HL1 HL2 W36', 'HL1 HL2 W37', 'HL1 HL2 W38',

'input HL1 W34', 'input HL1 W42', 'input HL1 W43', 'input HL1 W44', 'HL1 HL2 W21', 'HL1 HL2 W23', 'HL1 HL2 W24', 'HL1 HL2 W25', 'HL1 HL2 W26', 'HL1 HL2 W27', 'HL1 HL2 W31',	'HL1 HL2 W41', 'HL1 HL2 W42', 'HL1 HL2 W43', 'HL1 HL2 W44', 'HL1 HL2 W45', 'HL1 HL2 W46', 'HL1 HL2 W47', 'HL1 HL2 W48', 'bias for HL1 neuron 2', 'bias for HL1 neuron 3', 'bias for HL1 neuron 4'

Table 6-6: category 2 weights and biases for Rate Constant

Category 2 Weights & Biases	
input HL1 W32',	input HL1 W32',

Table 6-7: category 3 weights and biases for Rate Constant

Category 3 Weights & Biases	
'input HL1 W11', 'input HL1 W15', 'input HL1 W16', 'input HL1 W17', 'input HL1 W18', 'input HL1 W21', 'input HL1 W25', 'input HL1 W26', 'input HL1 W27', 'input HL1 W28', 'input HL1 W31', 'input HL1 W35', 'input HL1 W36', 'input HL1 W37', 'input HL1 W38', 'input HL1 W41', 'input HL1 W45', 'input HL1 W46', 'input HL1 W47', 'input HL1 W48', 'HL1 HL2 W11', 'HL1 HL2 W12', 'HL1 HL2 W13', 'HL1 HL2 W14', 'HL1 HL2 W15', 'HL1 HL2 W16', 'HL1 HL2 W17', 'HL1 HL2 W18', 'HL1 HL2 W51', 'HL1 HL2 W52', 'HL1 HL2 W53', 'HL1 HL2 W54', 'HL1 HL2 W55',	'HL1 HL2 W72', 'HL1 HL2 W73', 'HL1 HL2 W74', 'HL1 HL2 W75', 'HL1 HL2 W76', 'HL1 HL2 W77', 'HL1 HL2 W78', 'HL1 HL2 W81', 'HL1 HL2 W82', 'HL1 HL2 W83', 'HL1 HL2 W84', 'HL1 HL2 W85', 'HL1 HL2 W86', 'HL1 HL2 W87', 'HL1 HL2 W88', 'output HL2 W11', 'output HL2 W12', 'output HL2 W13', 'output HL2 W14', 'output HL2 W15', 'output HL2 W16', 'output HL2 W17', 'output HL2 W18', 'output HL2 W21', 'output HL2 W22', 'output HL2 W23', 'output HL2 W24', 'output HL2 W25', 'output HL2 W26', 'output HL2 W28', 'bias for HL1 neuron 5', 'bias for HL1 neuron 6', 'bias for HL1 neuron 7',

'HL1 HL2 W56', 'HL1 HL2 W57', 'HL1 HL2 W58', 'HL1 HL2 W61', 'HL1 HL2 W62', 'HL1 HL2 W63', 'HL1 HL2 W64', 'HL1 HL2 W65', 'HL1 HL2 W66', 'HL1 HL2 W67', 'HL1 HL2 W68', 'HL1 HL2 W71',	'bias for HL1 neuron 8', 'bias for HL2 neuron 1', 'bias for HL2 neuron 2', 'bias for HL2 neuron 3', 'bias for HL2 neuron 4', 'bias for HL2 neuron 5', 'bias for HL2 neuron 6', 'bias for HL2 neuron 7', 'bias for HL2 neuron 8', 'bias for output1', 'bias for output2'

In category 1, there are 36 weights + biases that show a $\pm 10\%$ deviation, in category 2 there is a total of 2 weights + biases that show a $\pm 10\% \; to \; 25\%$ deviation, and in category 3 there are total 92 weights + biases that show a deviation more than $\pm 25\%$. In Chapter 7, it is demonstrated how all of these categories of Rate Constant and Heat of Reaction are going to be used while partial weights training of the ANN model.

6.6 Conclusion

The trained ANN model's structure and weight distribution are given. The relationship between weights and process parameters such as heat of reaction and rate constant is determined using this trained ANN model, and three groups are created based on weight deviation. In Chapter 7, these classifications will be utilised to choose weights that will be trained as partial weights, while others will be left alone.

Chapter 7

Partial Weight Training of the ANN Model

In this chapter, predictions, innovation plots, and estimation error plots are presented of trained ANN model with EKF as mentioned in section 5.2.1 where temperature and concentration both are considered as measured states, for the dataset created at operating conditions of CSTR parameters as mentioned in Table 5-2. Same ANN model is used as mentioned in section 5.2.1 for the comparison of predictions, innovation plots and estimation error plots for following 3 different cases:

- Case I – CSTR parameters at operating condition
- Case II – heat of reaction operating value reduced by 40%
- Case III – rate constant operating value reduced by 40%

Finally, presented a brief overview of the methodology for doing partial weight training on the ANN-EKF model, along with predictions, innovation plots, and estimation error plots.

7.1 Manually coded Artificial neural network model

A model which was created using inbuilt libraries of pytorch (python library for deep learning) has been implemented to produce various result plots with and without EKF implementation as stated in section 5.2. For training purposes also, an inbuilt ADAM optimizer was employed. As per project need, partial weight training of the ANN model has to be accomplished and we were not able to do that with the inbuilt ANN model. So, to overcome this problem, the development of our own ANN model with multiple loops and functions has been prepared. To compare the similarities between the inbuilt model and the manually coded model, we applied the learned weights of the inbuilt ANN model to our manually coded ANN model, and the results matched exactly with the inbuilt ANN model.
In this chapter, we utilised a manually coded ANN model with EKF to generate all of the results, and we also used a manually coded ANN model for partial weight training.

7.2 Results for case I

In this section, results are shown when the trained ANN model is implemented on initial data which was created at operating condition of parameters and in upcoming sections model predictions are shown when dataset is changed due to a reduction in process parameter operating values so that we can carry out the performance comparison of trained ANN model with different datasets. The model is trained as described in section 5.2.1 and the results for ANN model predictions for training dataset are as shown in Figure 5-6 and Figure 5-7 and predictions for testing datasets are as shown in Figure 5-10 and Figure 5-11.

7.2.1 ANN-EKF results

The extended Kalman filter state estimator is combined with the trained model discussed in section 7.2 to produce results of true states vs filtered values as can be seen in Figure 5-12 and Figure 5-13 and we considered concentration and temperature both as measured states, also the innovation and estimation error plots for concentration and temperature are included just to see the behaviour when data is at operating condition.

7.2.1.1 Innovation and estimation error plots

For innovation and estimation error plots, only 400 samples out of 1655 are considered so that the visualisation could be seen more clearly.

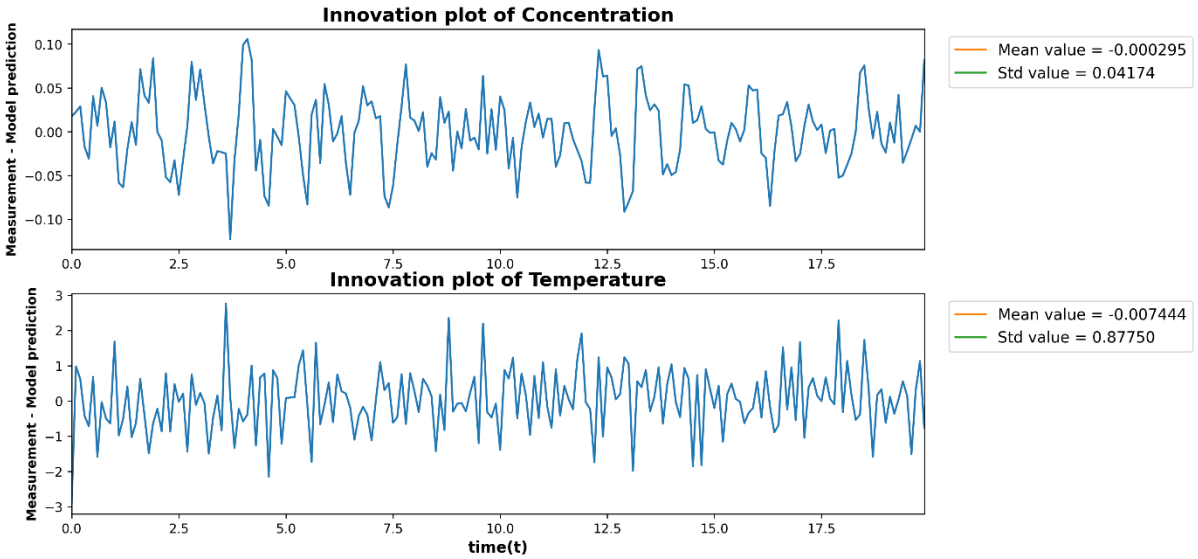

Figure 7-1: Innovation plots at operating condition

Table 7-1: Innovation plot summary at operating condition

Process states	Mean	Standard deviation
Concentration	-0.000295	0.04174
Temperature	-0.007442	0.87751

As shown in Figure 7-1 and Figure 7-2, the estimation error and innovation plots for concentration and temperature are fluctuating around zero, and the mean and standard deviation values of these fluctuations are presented in Table 7-1 and Table 7-2 and from this, a conclusion can be made about the good performance of ANN-EKF.

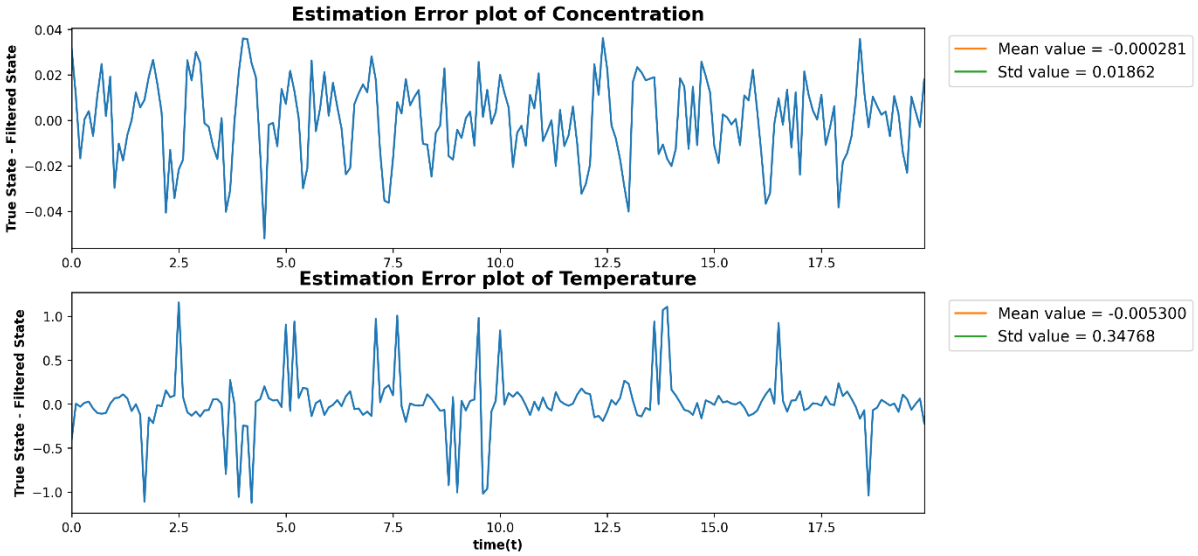

Figure 7-2: Estimation error plots at operating condition

Table 7-2: Summary of estimation errors at operating condition

Process states	Mean	Standard deviation
Concentration	-0.000281	0.01862
Temperature	-0.0053	0.34768

7.3 Results for case II

Just to examine how a trained ANN model (section 7.2) predicts in a chemical plant when some parameters alter their operating value over a time period. A dataset with a 40% drop in heat of reaction operating value was Created and employed the same trained model which was trained on normal operating condition values of parameters. This section presents

ANN and ANN-EKF predictions along with innovation and estimation error plots for comparison with section 7.2 when the plant was working normally.

7.3.1 ANN model results for reduced heat of reaction

A trained model as mentioned in section 5.2.1 is utilized and the following are the results for ANN model predictions for training and testing datasets. Only 300 samples are considered to show in plots so that we can clearly see the prediction curves with target value curves.

7.3.1.1 ANN predictions for training dataset and test dataset

As demonstrated in Figure 7-3 and Figure 7-4, the predictions of pretrained ANN model are not very good when heat of reaction value is reduced by 40%.

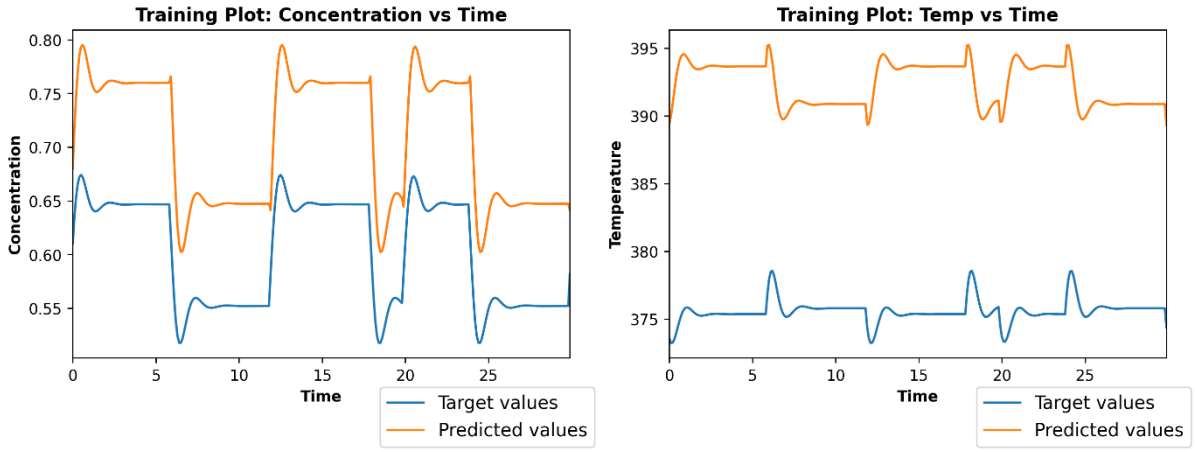

Figure 7-3: ANN predictions for training data at 40% reduction in heat of reaction value

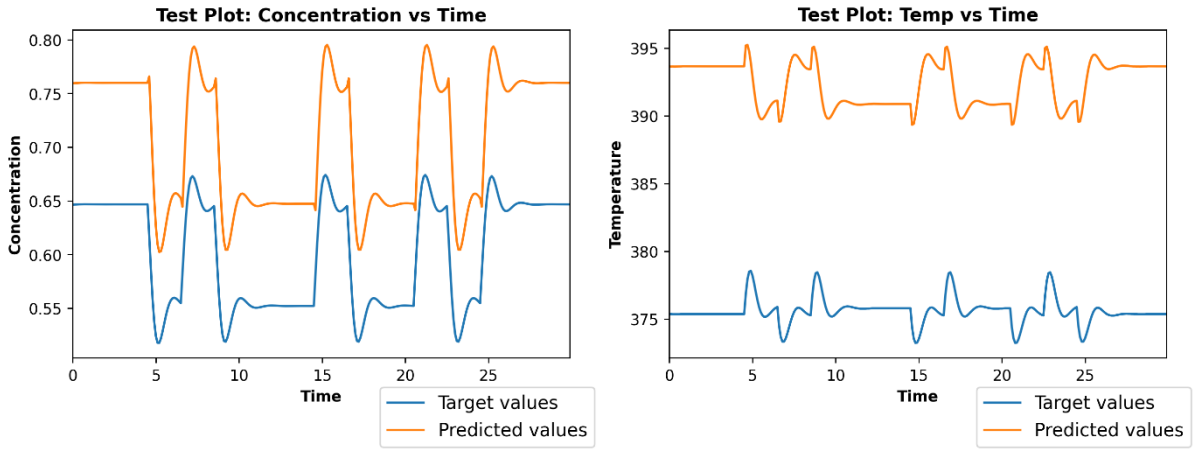

Figure 7-4: ANN predictions for test data at 40% reduction in heat of reaction value

7.3.2 ANN-EKF results for reduced heat of reaction

Extended Kalman filter state estimator is combined with the trained model mentioned in section 7.2 to produce results of true states vs filtered values when the operating condition value of heat of reaction is reduced by 40%. From Figure 7-5, it is clear that the ANN-EKF performance is not good for both states.

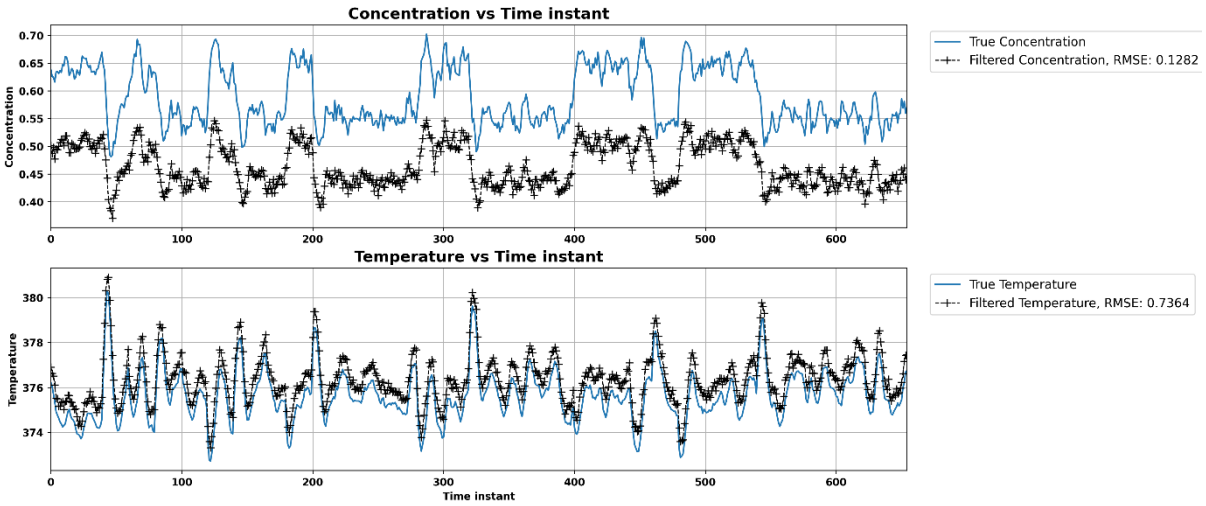

Figure 7-5:True Vs filtered plots for 40% reduction in heat of reaction

7.3.2.1 Innovation and estimation error plots for reduced heat of reaction

As shown in Figure 7-6 and Figure 7-7, the innovation and estimation error plots for temperature and concentration are not fluctuating around zero which indicates the poor performance of ANN-EKF for both the state variables.

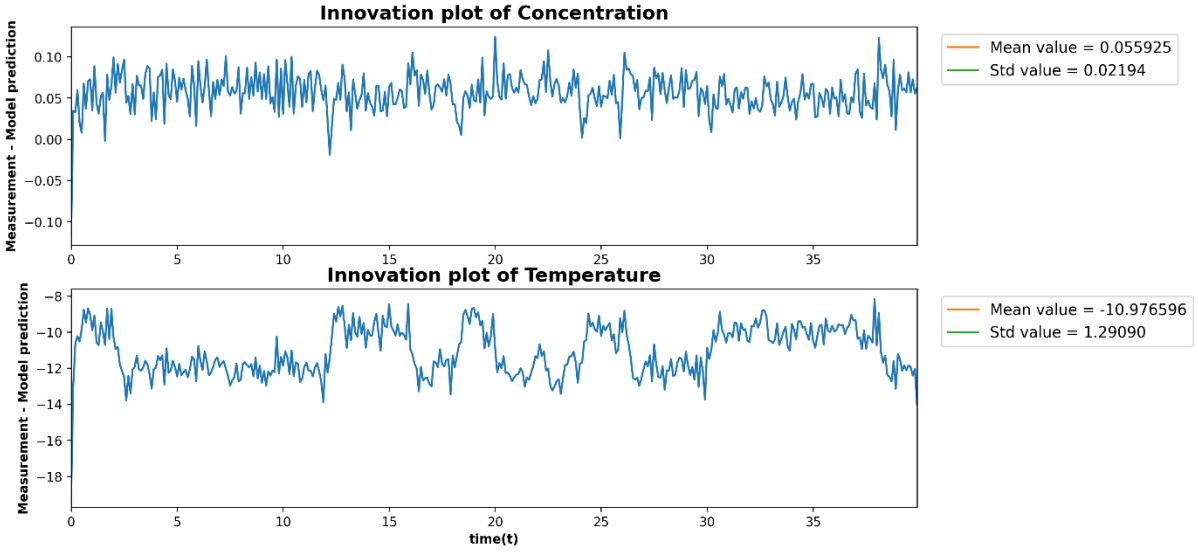

Figure 7-6: Innovation plots at 40% reduction in heat of reaction

Table 7-3: Summary of innovations at reduced heat of reaction

Process states	Mean	Standard deviation
Concentration	0.055925	0.02194
Temperature	-10.9765	1.2909

The innovations and estimation errors Mean and standard deviation values are given in Table 7-3 and Table 7-4 when the heat of reaction value is reduced by 40% and model remains the same as before.

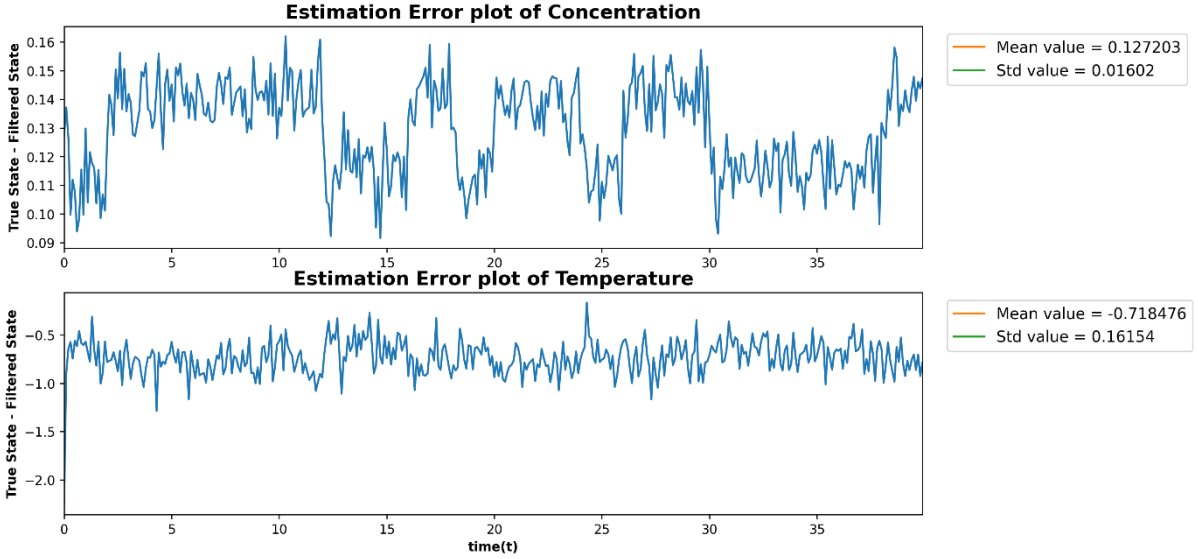

Figure 7-7: Estimation error plots at 40% reduction in heat of reaction

Table 7-4: Summary of estimation errors at reduced heat of reaction

Process states	Mean	Standard deviation
Concentration	0.1272	0.01602
Temperature	-0.7184	0.16154

7.3.3 Approach for partial weight training of ANN model

As demonstrated in Section 7.3.1 and 7.3.2, the ANN model is incapable of making correct predictions for both train and test data. Predictions improved after combining the ANN model with the EKF state estimator, although the improvement was not significant, as seen in Figure 7-5. As a conventional solution, an ANN model must be trained from scratch, but as discussed in Chapter 2, this approach is time-consuming and expensive. As a novel

aspect of this project, the concept of partial weight training has been developed in order to make it more efficient and less time-consuming.

Weights were classified into three types based on their behaviour with CSTR process parameters in Chapter 6. There are a total of 130 weights (bias also counts as a weight) in the artificial neural network model, some of which will be trained and others will be left alone. Weights will now be selected based on these three categories for partial training of ANN model. Weights are included in category 1 and have a deviation of less than 10%, whereas weights in the remaining two categories have a deviation of more than 10%. As a result, for partial weight training, category 2 and 3 weights will be chosen, while category 1 weights will be excluded.

7.3.4 Partial weight training of ANN model for reduced heat of reaction

As discussed in Chapter 6, 35 weights fall into category 1 and 95 weights fall into category 2 and category 3. As a result, according to the approach outlined in section 7.3.3, we must train only 95 of the 130 weights for partial weight training of the ANN model. After the hyperparameter tuning of the ANN model for partial weight training, the best-matched values for the number of epochs and learning rate are as follows.

Table 7-5: summary of ANN partial weight training for heat of reaction

Learning rate	Total weights	Trained weights	Initial train MSE	Initial test MSE	After train MSE	After test MSE	Number of epochs
0.00015	130	95	139.02	140.42	0.0086	0.0081	110000

7.3.4.1 Partially trained ANN model predictions for training and test datasets

In section 7.3.1.1, the identical plots for training and test datasets are presented for ANN model without training, and as shown in Figure 7-8 and Figure 7-9, predictions are significantly improved, and these results are accomplished by only training 73% weights of existing ANN model.

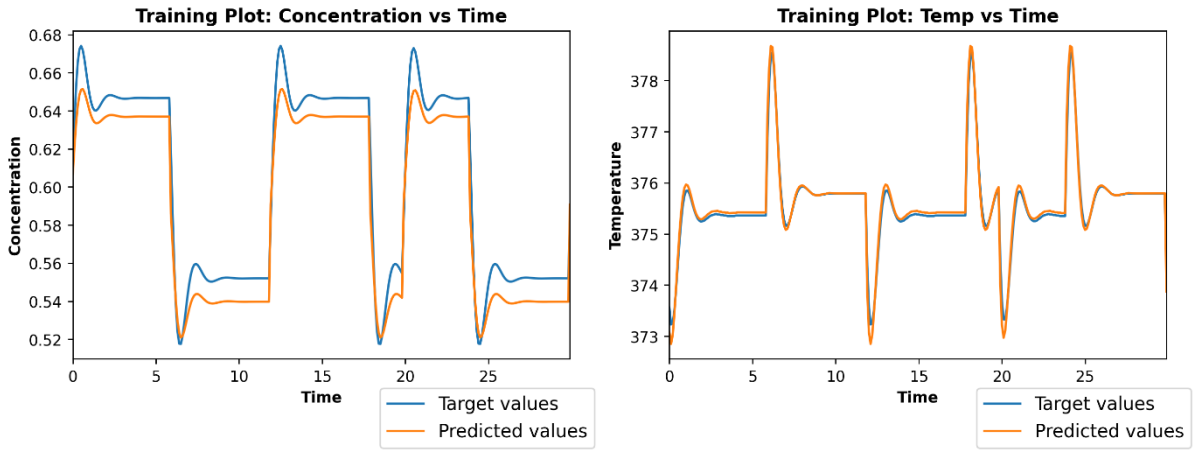

Figure 7-8: Partially trained ANN predictions for reduced heat of reaction training data

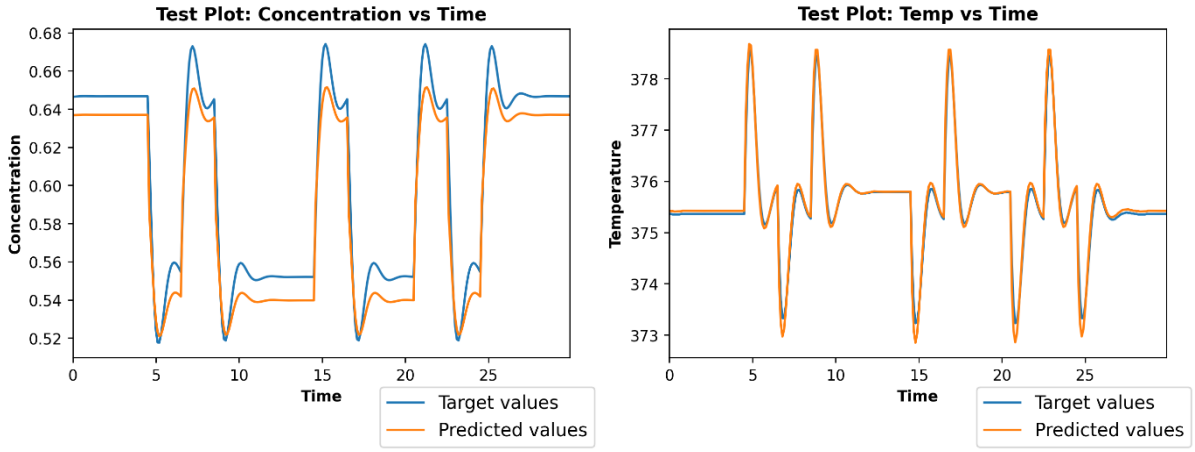

Figure 7-9:Partially trained ANN predictions for reduced heat of reaction test data

7.3.4.2 Partially trained ANN-EKF results for reduced heat of reaction

After the results of training dataset and test dataset were produced using a partially trained ANN model, now it is combined with the EKF state estimator for predictions of true concentration and temperature. The ANN-EKF predictions were not good without partial training as shown in Figure 7-5, and as demonstrated in Figure 7-10, After partial weight training, the ANN model produces exceptionally accurate true state predictions for both states.

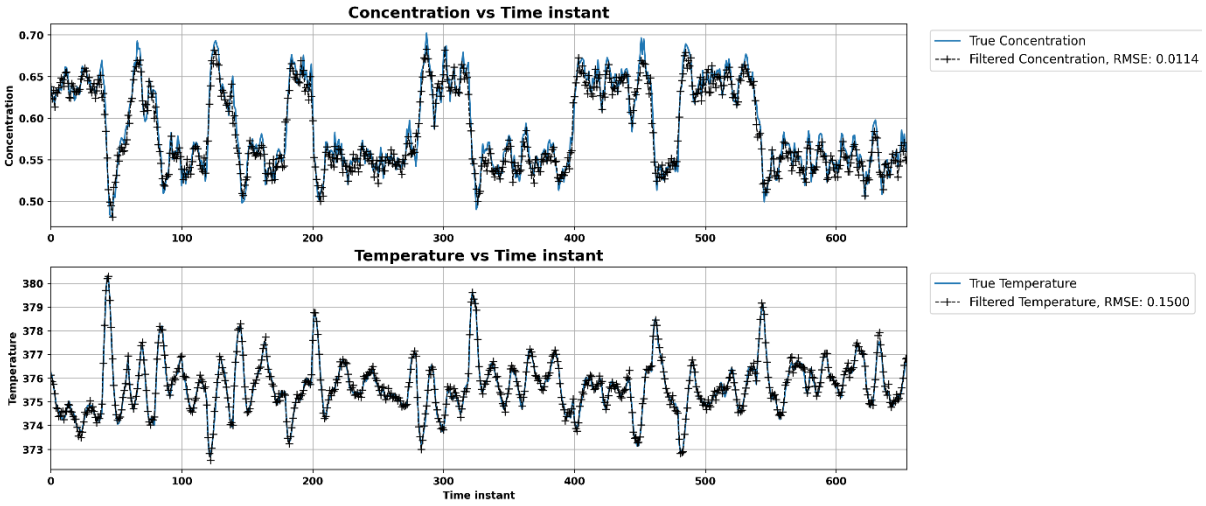

Figure 7-10: Partially trained ANN-EKF results for reduced heat of reduction

Table 7-6 compares different root mean square error values for a model before training and a model after partial weight training.

Table 7-6: RMSE Comparison of ANN before and after partial model training

Model used	**Concentration predictions by EKF**	**Temperature predictions by EKF**	**Concentration predictions by the model**	**Temperature predictions by the model**
ANN initially	0.128	0.736	0.058	11.05
ANN after partial training	0.0114	0.149	0.0211	0.3901

7.3.4.3 Innovation and estimation error plots for partially trained ANN

To facilitate comparison, innovation and estimation error plots are constructed in such a way that we can notice a distinct difference in errors between the initial ANN model without partial training and the ANN model after partial training. As illustrated in Figure 7-11 and Figure 7-12, initially errors are presented in blue colour for data at an operating value of heat of reaction, then errors are presented in red colour for which we introduced the 40% reduction in heat of reaction operating value and finally, green colour errors are

presented when ANN model is partially trained and applied for the data which was created with 40% reduction in heat of reaction operating value.

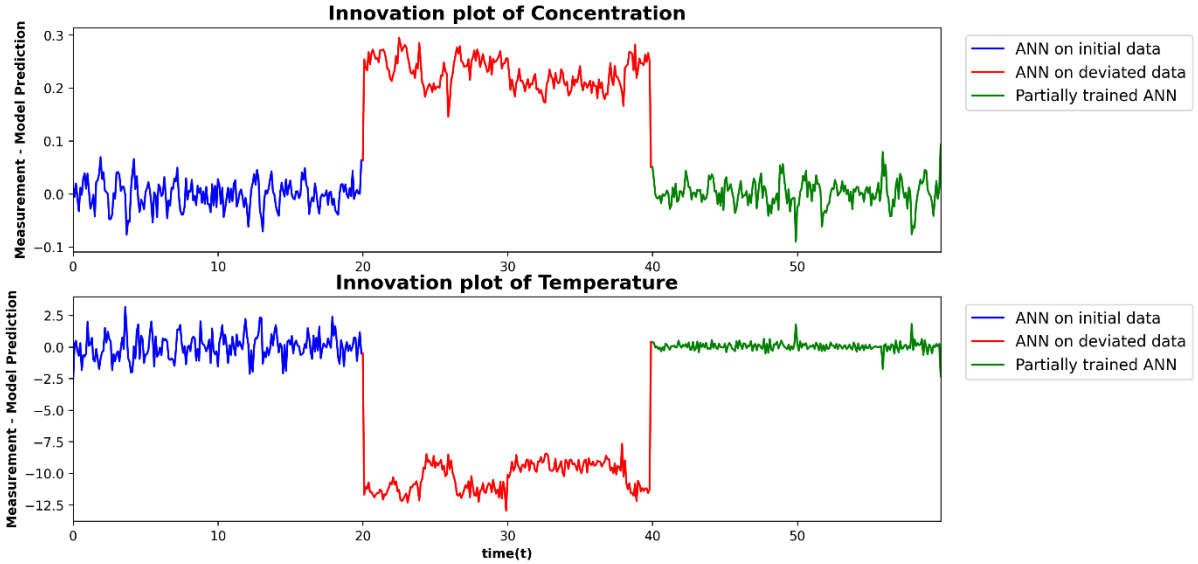

Figure 7-11: Comparable partially trained innovation plots for reduced heat of reaction

Both graphs clearly demonstrate that post partial weight training on deviating data, innovations reverted to oscillate around zero, just as they did for data at operational conditions.

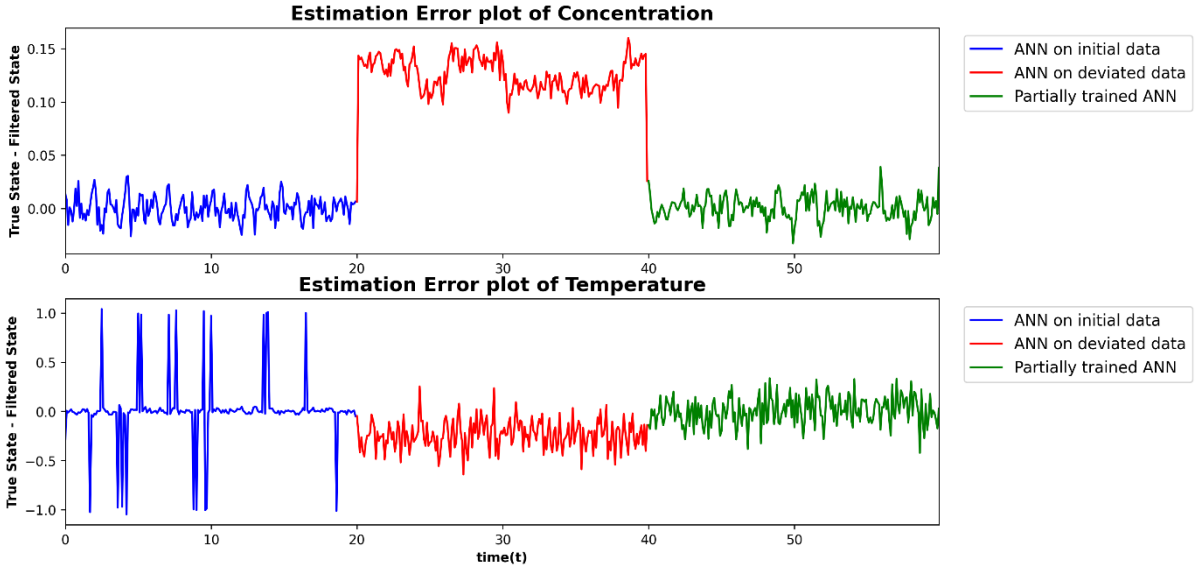

Figure 7-12: Comparable partially trained estimation error plots for reduced heat of reaction

The same investigation is carried out with the rate constant parameter of CSTR in the following section to test the performance of the partially trained model.

7.4 Results for case III

Just to examine how a trained ANN model (section 7.2) predicts in a chemical plant when some parameters alter their operating value over a time period. A dataset with a 40% drop-in rate constant operating value was Created and employed the same trained model which was trained on normal operating condition values of parameters. This section presents ANN and ANN-EKF predictions along with innovation and estimation error plots for comparison with section 7.2 when the plant was working normally.

7.4.1 ANN model results for the reduced rate constant

A trained model as we mentioned in section 5.2.1 is utilized and the following are the results for ANN model predictions for training and testing datasets. Only 300 samples are considered to show in plots so that we can clearly see the prediction curves with target value curves.

7.4.1.1 ANN predictions for training and test datasets

As demonstrated in Figure 7-13 and Figure 7-14, the predictions of pretrained ANN model are not very good when a 40% reduction is introduced in the operating value of rate constant process parameter.

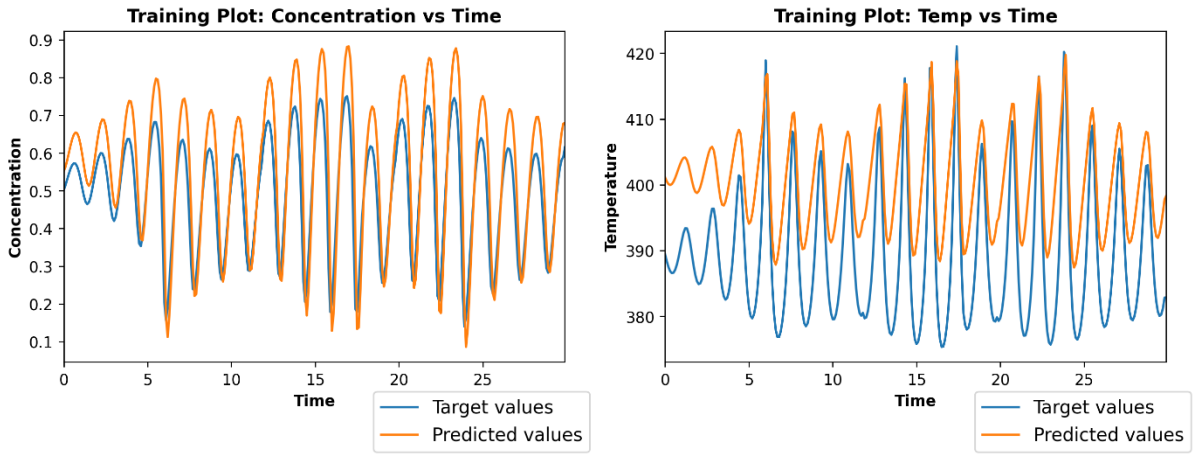

Figure 7-13:ANN predictions for training data at 40% reduction in rate constant value

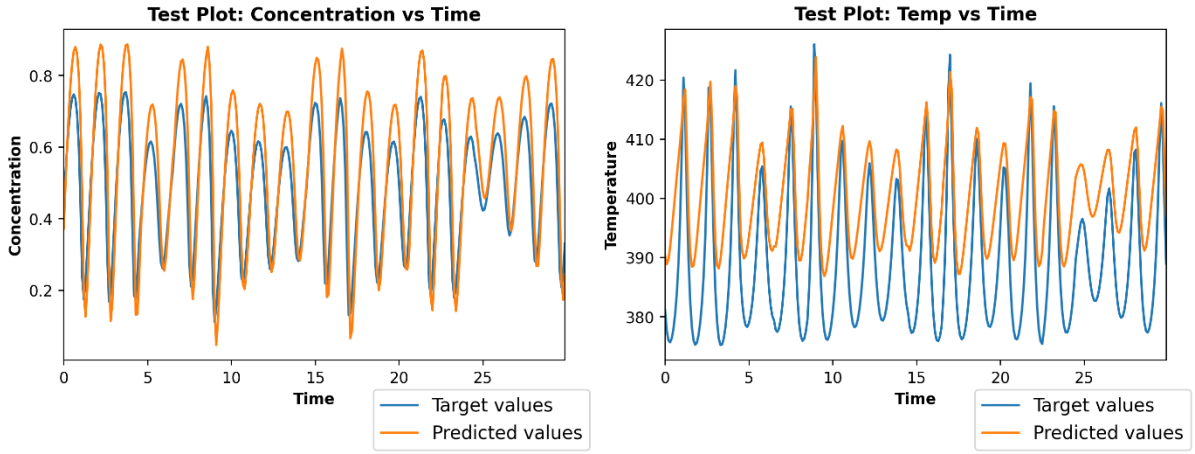

Figure 7-14:ANN predictions for test data at 40% reduction in rate constant value

7.4.2 ANN-EKF results for the reduced rate constant

Extended Kalman filter state estimator is combined with the trained model mentioned in section 7.2 to produce results of true states vs filtered values when the operating condition value of rate constant is reduced by 40%. From Figure 7-15, it is clear that the ANN-EKF performance is poor for both states.

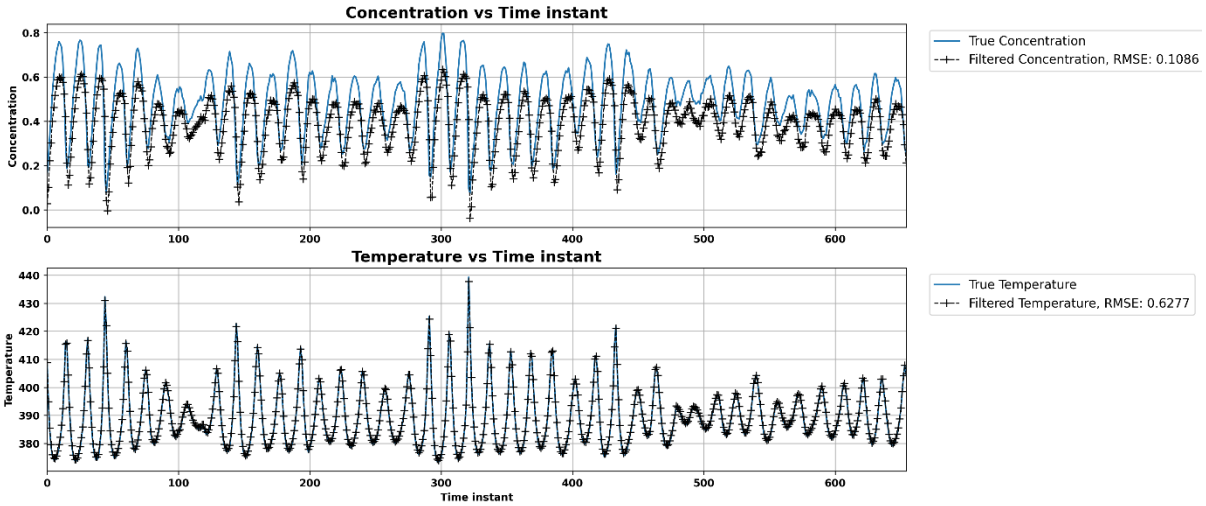

Figure 7-15:True Vs filtered plots for 40% reduction in rate constant

7.4.2.1 Innovation and estimation error plots for reduced rate constant

As shown in Figure 7-16 and Figure 7-17, the innovation and estimation error plots for temperature and concentration are not fluctuating around zero which indicates the poor performance of ANN-EKF for both the state variables when a 40% reduction is introduced in the operating value of rate constant.

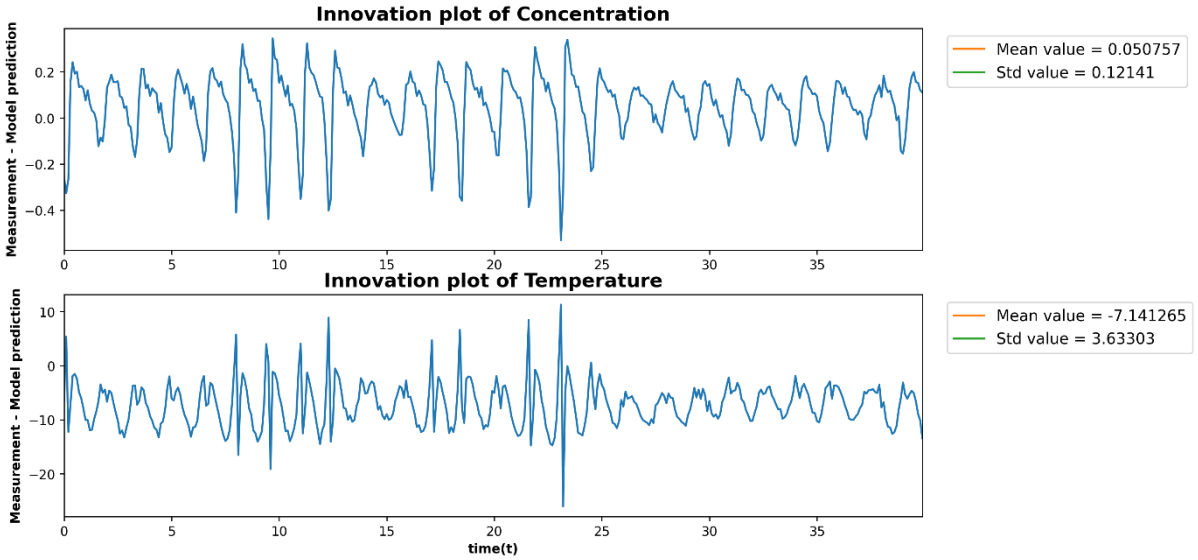

Figure 7-16: Innovation plots at 40% reduction in rate constant

Table 7-7: Summary of innovations at reduced rate constant

Process states	Mean	Standard deviation
Concentration	0.05075	0.12141
Temperature	-7.1412	3.63303

The innovations and estimation errors Mean and standard deviation values are given in Table 7-7 and Table 7-8 when the rate constant value is reduced by 40% and model remains the same as before.

Table 7-8: Summary of estimation errors at reduced rate constant

Process states	Mean	Standard deviation
Concentration	0.088243	0.06338
Temperature	-0.4982	0.38171

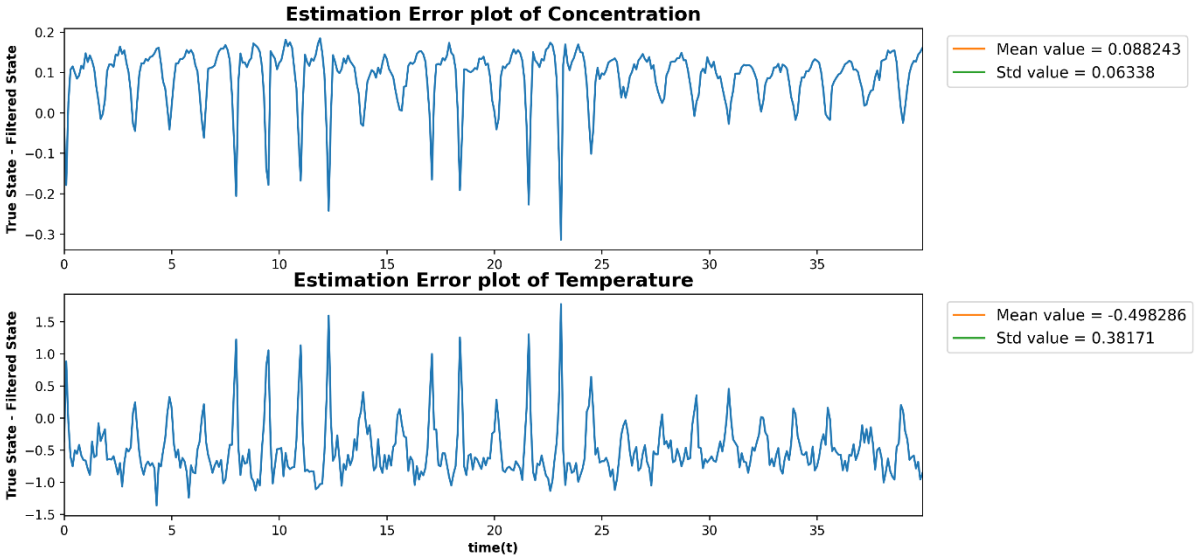

Figure 7-17: Estimation error plots at 40% reduction in rate constant

7.4.3 Partial weight training of ANN model for reduced rate constant

As discussed in Chapter 6, 36 weights fall into category 1 and 94 weights fall into category 2 and category 3. As a result, according to the approach outlined in section 7.3.3, we must train only 94 of the 130 weights for partial weight training of the ANN model. After the hyperparameter tuning of the ANN model for partial weight training, the best-matched values for the number of epochs and learning rate are as follows.

Table 7-9:summary of ANN partial weight training for heat of reaction

Learning rate	Total weights	Trained weights	Initial train MSE	Initial test MSE	After train MSE	After test MSE	Number of epochs
0.00015	130	94	75.09	76.47	3.8694	4.0694	80000

7.4.3.1 Partially trained ANN model predictions for training and test dataset

In section 7.4.1.1, the identical plots for training and test dataset are presented for ANN model without training, and as shown in Figure 7-18 and Figure 7-19, all the predictions are significantly improved, and these results are accomplished by only training 72.3% weights of existing ANN model.

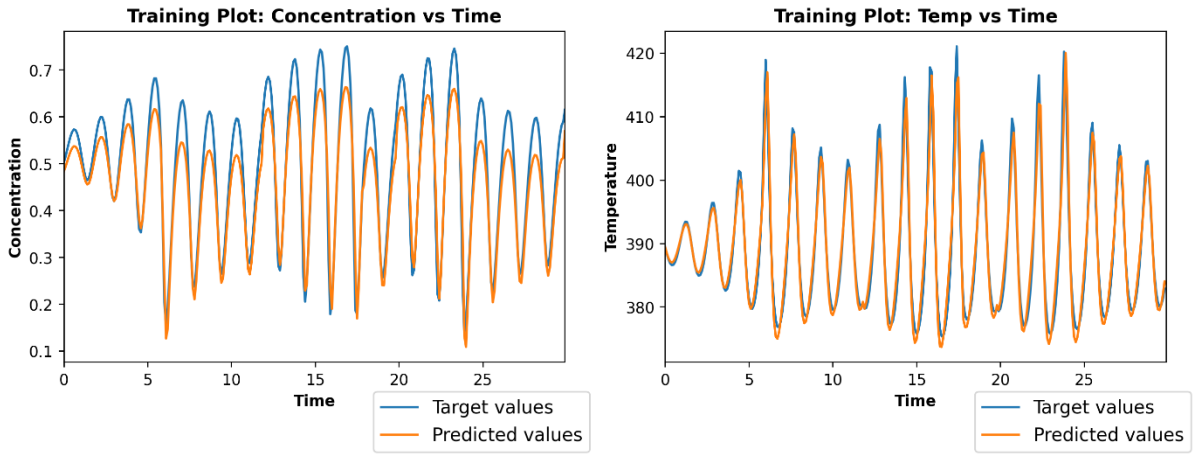

Figure 7-18: Partially trained ANN predictions for reduced rate constant training data

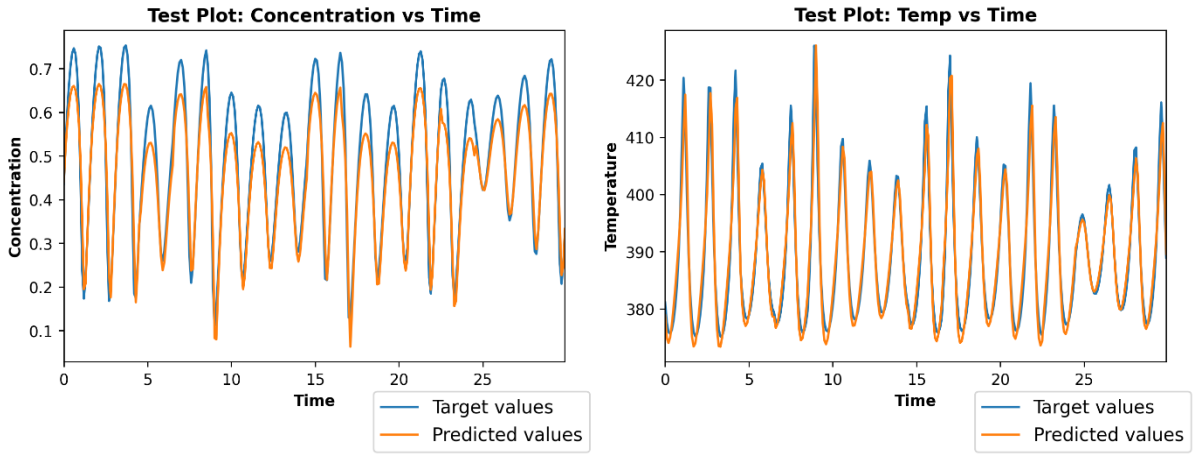

Figure 7-19:Partially trained ANN predictions for reduced rate constant test data

7.4.3.2 Partially trained ANN-EKF results for reduced rate constant

After the results of the training dataset and test dataset are produced using a partially trained ANN model, now it is combined with the EKF state estimator for predictions of true concentration and temperature. The ANN-EKF predictions were not good without partial training as shown in Figure 7-15, and as demonstrated in Figure 7-20, After partial weight training, the ANN model produces exceptionally accurate true state predictions for both states.

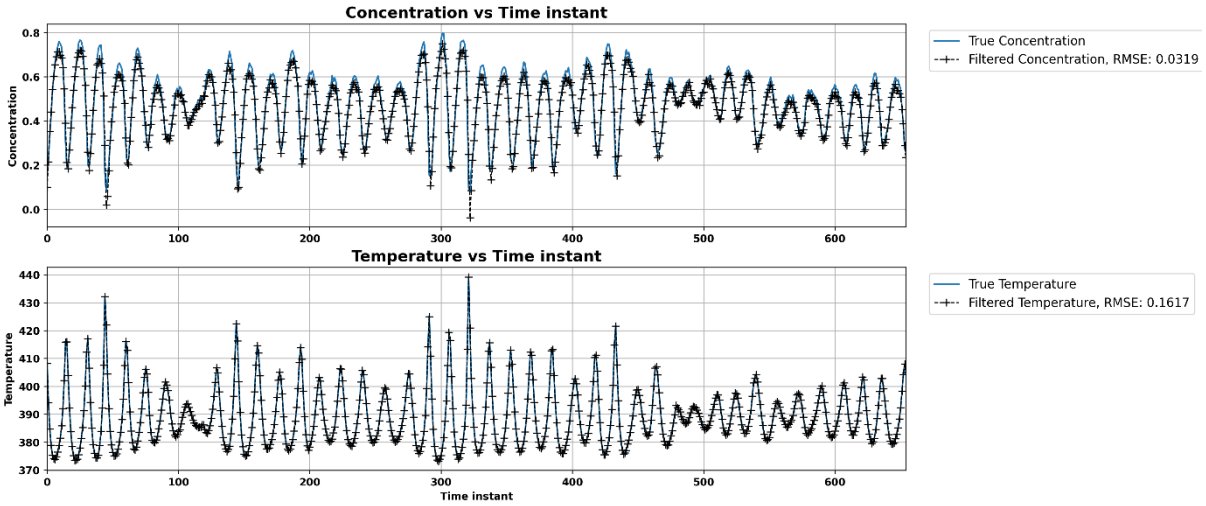

Figure 7-20: Partially trained ANN-EKF results for reduced rate constant

Table 7-10 compares different root mean square error values for the model before training and the model after partial weight training.

Table 7-10: RMSE Comparison of ANN before and after partial model training

Model used	**Concentration predictions by EKF**	**Temperature predictions by EKF**	**Concentration predictions by the model**	**Temperature predictions by the model**
ANN initially	0.1086	0.627	0.1306	8.009
ANN after partial training	0.0319	0.161	0.0651	3.153

7.4.3.3 Innovation and estimation error plots for partially trained ANN

To facilitate comparison, innovation and estimation error plots are constructed in such a way that we can notice a distinct difference in errors between the initial ANN model without partial training and the ANN model after partial training. As illustrated in Figure 7-21 and Figure 7-22, initially errors are presented in blue colour for data at an operating value of heat of reaction, then errors are presented in red colour for which we introduced the 40% reduction in heat of reaction operating value and finally, green colour errors are presented when ANN model is partially trained and applied for the data which was created with 40% reduction in heat of reaction operating value.

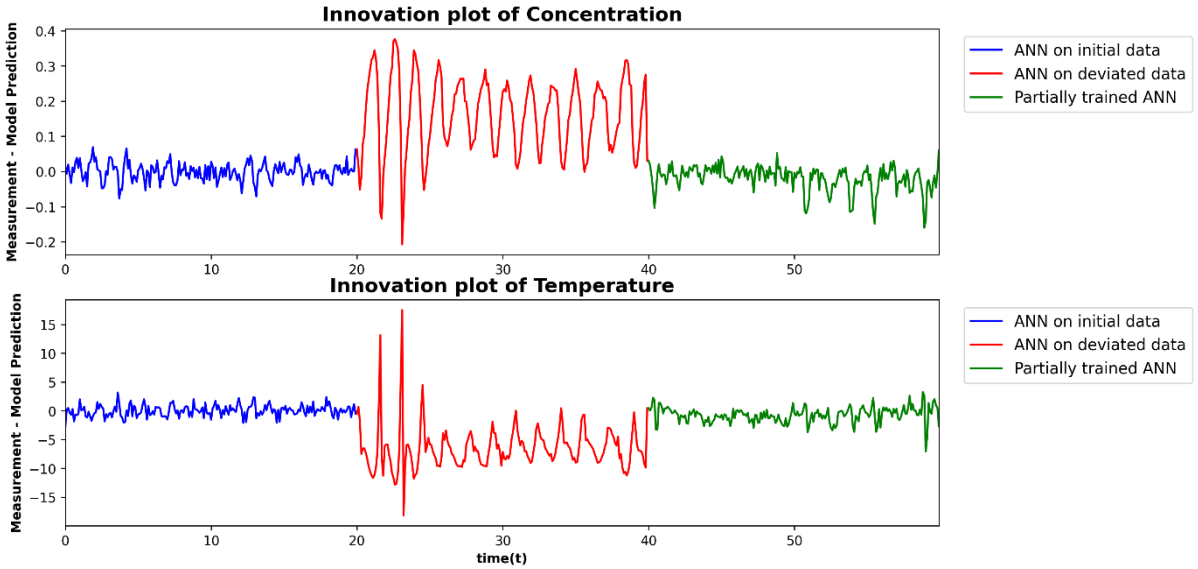

Figure 7-21: Comparable partially trained innovation plots for reduced rate constant

Both graphs clearly demonstrate that post partial weight training on deviating data, innovations reverted to oscillate around zero, just as they did for data at operational conditions.

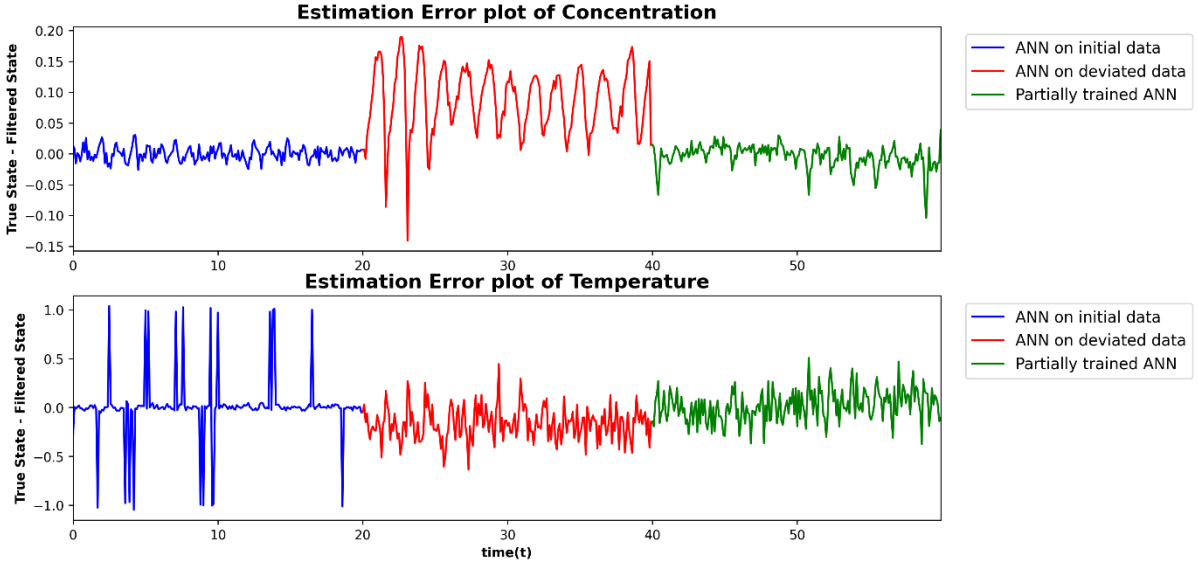

Figure 7-22: Comparable partially trained estimation error plots for reduced rate constant

7.5 Partial ANN training Vs complete ANN training

The complete ANN model has been trained for a 40% reduction in heat of reaction operating value and a 40% reduction in rate constant operating value so that a performance comparison can be made between complete ANN training and partial ANN training.

7.5.1 Comparison for the reduced value of heat of reaction

When Table 7-11 for full ANN training is compared to Table 7-5 for partial ANN training, it is clear that there are insignificant differences in mean square error values for train and test datasets when the ANN model is trained partially and fully, and from that conclusion can be made about the effective performance of partial ANN training with less computation load.

Table 7-11: Summary of full ANN weight training for heat of reaction

Learning rate	Total weights	Trained weights	Initial train MSE	Initial test MSE	After train MSE	After test MSE	Number of epochs
0.00015	130	95	139.02	140.42	0.0086	0.0079	110000

7.5.2 Comparison for the reduced value of rate constant

When comparing Table 7-12 for full ANN training to Table 7-9 for partial ANN training, it is clear that mean square error values for train and test datasets are higher when the ANN model is fully trained, so for some parameters like rate constant, we need to put some extra computational power in comparison to partial ANN training to get better results. Following can be the reasons for higher error values for complete ANN training:

- Hyperparameter tunning of the model deepens on number of neurons and number of hidden layer so the learning rate and number of epochs we selected for partial training not necessarily have to work well as it worked for partial weight training.
- As we can see in Figure 7-18 and Figure 7-19, data is oscillatory and due to unsuitable hyperparameter tunning, model couldn't perform well.

Table 7-12:Summary of full ANN weight training for rate constant

Learning rate	Total weights	Trained weights	Initial train MSE	Initial test MSE	After train MSE	After test MSE	Number of epochs
0.00015	130	94	75.09	76.47	5.769	6.113	80000

As shown in Table 7-13, when the full ANN model is trained for 1 lakh 50 thousand epochs, the MSE error values are lower than in Table 7-12 where we trained the full ANN model for 80 thousand epochs.

Table 7-13: Summary of full ANN weight training for rate constant with more epochs

Learning rate	Total weights	Trained weights	Initial train MSE	Initial test MSE	After train MSE	After test MSE	Number of epochs
0.00015	130	94	75.09	76.47	4.726	5.002	150000

7.5.3 ANN-EKF comparison for reduced heat of reaction

As shown in Table 7-14, the performance of the partially trained ANN model is almost similar to the fully trained ANN model, when ANN is combined with EKF for state estimation.

Table 7-14: RMSE Comparison of partially and fully trained ANN for heat of reaction

Model used	Concentration predictions by EKF	Temperature predictions by EKF	Concentration predictions by the model	Temperature predictions by the model
Fully trained ANN	0.01167	0.1504	0.0237	0.3981
Partially trained ANN	0.0114	0.149	0.0211	0.3901

7.5.4 ANN-EKF comparison for reduced rate constant

As shown in Table 7-15, the performance of the partially trained ANN model is better than the fully trained ANN model (Table 7-12) even after ANN is combined with EKF for state estimation.

Table 7-15: RMSE Comparison of partially and fully trained ANN for rate constant

Model used	**Concentration predictions by EKF**	**Temperature predictions by EKF**	**Concentration predictions by the model**	**Temperature predictions by the model**
Fully trained ANN	0.0641	0.4274	0.1084	3.2221
Partially trained ANN	0.0319	0.161	0.0651	3.153

7.6 ANN-EKF results when temperature is the only measured state

Till now in ANN-EKF strategy, concentration and temperature both are considered to be the measured states as mentioned in section 5.2.2. In this section, we have shown the results when the only temperature is considered as a measured state for ANN-EKF strategy and also carried out the performance comparison with section 5.2.2. So, all the variables and initial values are same as mentioned in section 5.2.2 except following variables:

$$C = [0 \quad 1] \tag{7-1}$$

$$R = 0.15^2 \tag{7-2}$$

From the comparison between Figure 5-12, Figure 5-13, and Figure 7-23, the conclusion can be made that the ANN-EKF performance is almost similar whether the concentration is taken as a measured state or not.

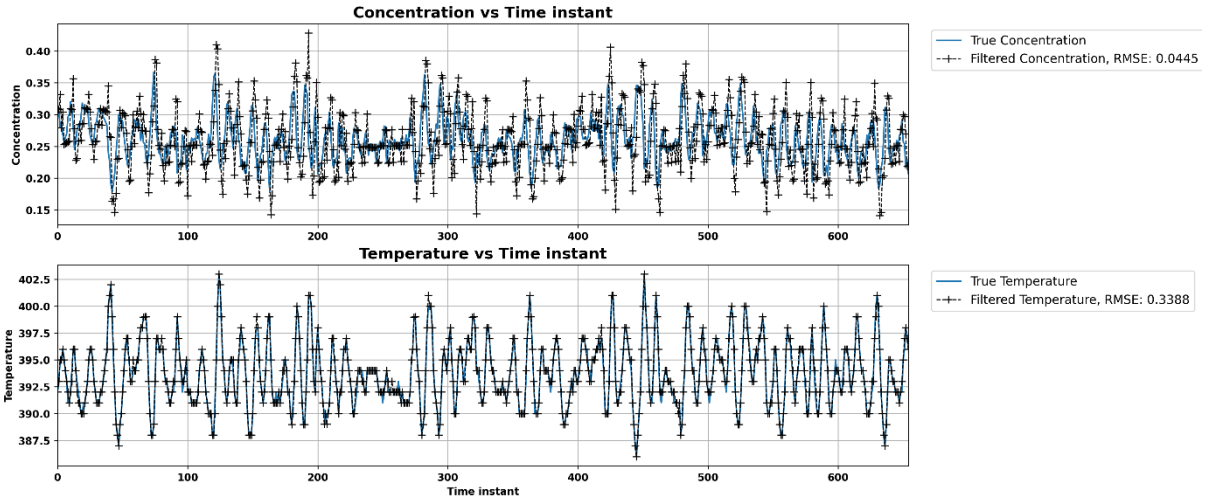

Figure 7-23: ANN-EKF results when temperature is a measured state

Comparison in root mean squared error values are given in Table 7-16 between both as measured states and only temperature as a measured state.

Table 7-16: RMSE comparison for different measured states

Measured states	**Concentration predictions by EKF**	**Temperature predictions by EKF**	**Concentration predictions by the model**	**Temperature predictions by the model**
Temperature	0.0445	0.338	0.0679	1.88
Temperature and concentration	0.0186	0.347	0.04	0.872

Different results of partial weight training for reduced values of heat of reaction and rate constant are presented in Appendix A.

7.7 Conclusion

A Manually coded artificial neural network model has been created for partial weight training. Using information from Chapter 6, partial weight training has been done when the rate constant and heat of reaction operating values were reduced by 40%. According to the material in Chapter 6, only category 2 and category 3 weights are trained in partial weight training since they have a high deviation in their values. According to the predictions,

innovation plots and estimation error plots of a partially trained ANN model with EKF, partial weight training of the model can be a suitable alternative for chemical industries.

Chapter 8

Conclusion and Future work

8.1 Conclusion

Different combinations of the model and estimator that can efficiently handle the state estimation problem were discussed in this study. Mechanistic models are frequently considered to be costly, time-consuming, and reliant on repetitive calculations. As a result, they are unfit for use in online applications. Switching to efficient data-based modelling methods can thereby solve this issue. Instead of using traditional physics-based models, an artificial neural network with dynamic mapping capabilities and the ability to be a universal approximator can be used. In the proposed strategies, the same concept has been implemented. Various case studies are discussed in the report in order to determine the performance of the recommended solutions. The root mean squared error was employed as a quantitative measure in each case study. Table 5-4 and Table 5-5 show that the proposed approach produced low RMSE values and was able to accurately predict the states. Furthermore, while comparing the computation times of ReLU-ANNEKF was discovered to be **90.68%** faster than the conventional method. As a result, it can be inferred that the proposed work eliminated a substantial amount of calculation complexity while not affecting the state estimator's performance.

To achieve the project's final goal, the correlation between the weights of the artificial neural network model and CSTR process parameters was performed, as described in Chapter 6. To obtain the link between weights and process parameters, an artificial neural network model was trained 100 times for 100 distinct datasets using 100 different values of process parameters generated in the range of -10% to +10% of their operational values (section 6.2 and section 6.4). From the visualizations given in Figure 6-2 to Figure 6-13 weights are classified into category 1 ($\pm$10% deviation), category 2 ($\pm$25% deviation), and category 3 (more than $\pm$25% deviation) as shown in

Table 6-2 to Table 6-7. From all of these plots and data, it can be concluded that Artificial neural network weights have some correlation with process parameters.

The results obtained in Chapter 5 demonstrate that neural network models can be used to model plant dynamics; however, the chemical industries with these existing neural network model approach encounter issues when any process parameter changes its value over time. To address this main challenge, the concept of partial weight training is introduced in Chapter 7. Sections 7.3 and 7.4 give two case studies to demonstrate the proposed idea of partial weight training. The results reported in sections 7.3.1, 7.3.2, 7.4.1, and 7.4.2 shows that as any process parameter changes its operational value, model performance diminishes. After performing the partial weight training of the ANN model, for the training dataset root mean square error (RMSE) value is reduced significantly by **99.99%** and **94.85%** and for the test dataset it decreased by **99.99%** and **94.58%** as can be seen in Table 7-5 and Table 7-9 for the heat of reaction and rate constant respectively. With the application of partially trained ANN-EKF, RMSE is reduced by **91.03%** and **70.44%** for true state concentration prediction, as can be seen in Table 7-6, and RMSE is reduced by **79.71%** and **63.95%** for true state temperature prediction as can be seen in Table 7-10 for the heat of reaction and rate constant respectively. Various innovation and estimation error plots associated with the performance of the proposed method have also been attached to the report.

To ensure the effectiveness of partial weight training, a comparison has been conducted in section 7.5 between partial ANN training Vs complete ANN training, and results for various case studies are presented in Table 7-11, Table 7-12, Table 7-13, Table 7-14 and Table 7-15 and from these results conclusion can be made that partially weight training is as efficient as complete network training. To ensure the realistic performance of ANN-EKF strategy, results are presented in section 7.6 and Appendix A when the only temperature is considered as a measured state. From the results presented in Figure 7-23 and Table 7-16, we can ensure the good performance of ANN-EKF even if there is only temperature as a measured state.

Based on the outcomes of the proposed strategy, it can be inferred that this method may be utilised to address the state estimate problem with less computing time and better accuracy than traditional training approaches of artificial neural network models, which are time-consuming and costly. This provides tremendous motivation to learn more about this method, and future studies will therefore tend to expand on the planned work.

8.2 Future work

Following are the things that can be explored taking the proposed work as a basis;

1. Implementation of an advanced estimator, such as MHE, using RNN as a model. Replacing the physics-based model with an RNN is projected to save a significant amount of time when solving the optimization problem within the MHE framework.
2. Parallel computing can be used to reduce the computation time of the suggested work by accessing a high GPU system and utilising each core inside the system to reduce the computation time by several folds.
3. Experimental validation of the proposed approach on systems like quadruple tanks and temperature control lab can be done.
4. In the future proposed approach can become automatic, in which partial training of the model happens according to a fault detected in process parameter values.
5. On-line adaptation of the developed model for state estimation using EKF in presence of process parameter variation to facilitate state estimation.
6. Explore more about approaches like RNNMSE, which works purely on artificial neural networks and can reduce computation time even further as the computations taken by the conventional estimator would be saved from these approaches

Appendix A.

ANN-EKF results when temperature is the only measured state

In this appendix, results are presented when temperature is the only measured state in ANN-EKF strategy and also comparison has been made with the case when concentration and temperature are both considered as measured states (section 7.3 and section 7.4).

A1 ANN-EKF results for 40% reduced heat of reaction

ANN model's results are given in section 7.3.1 for reduced heat of reaction. In Figure A 1, ANN-EKF predictions are given when temperature was the only measured state and heat of reaction value was reduced by 40%.

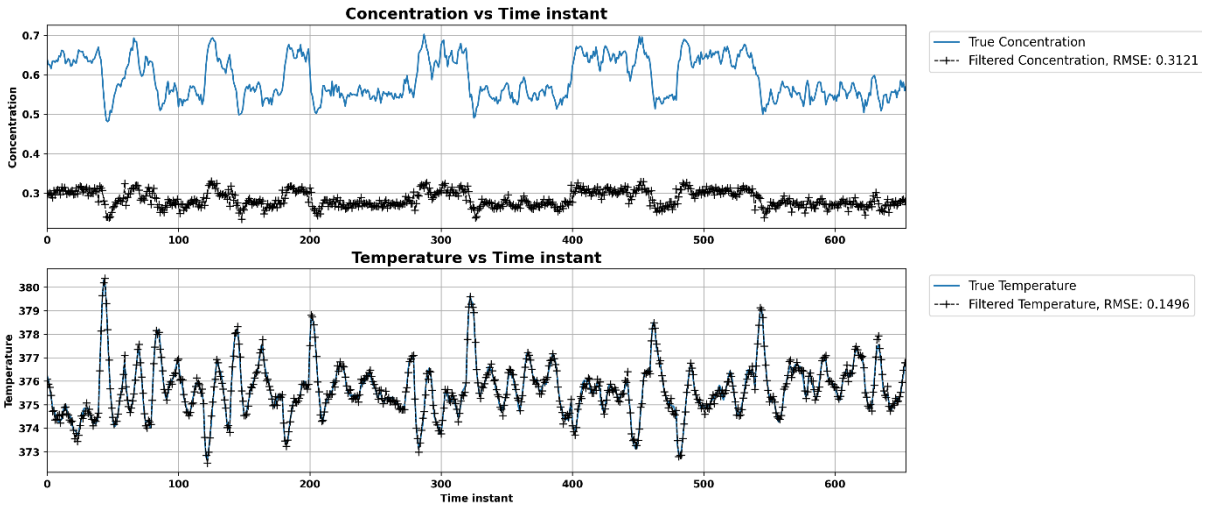

Figure A 1: ANN-EKF predictions for reduced heat of reaction

A2 Innovation and estimation error plots for reduced heat of reaction

As compared to Figure 7-6, in Figure A 2 we can see that the innovations are fluctuating around zero when we take temperature as a measured state.

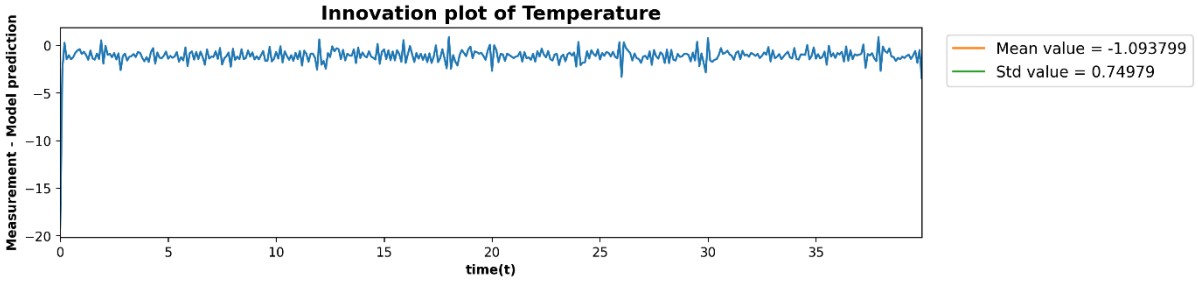

Figure A 2: Innovation plot for reduced heat of reaction

If we compare Figure 7-7 and Figure A 3, we can see that the estimation error values are increased for concentration and decreased for temperature.

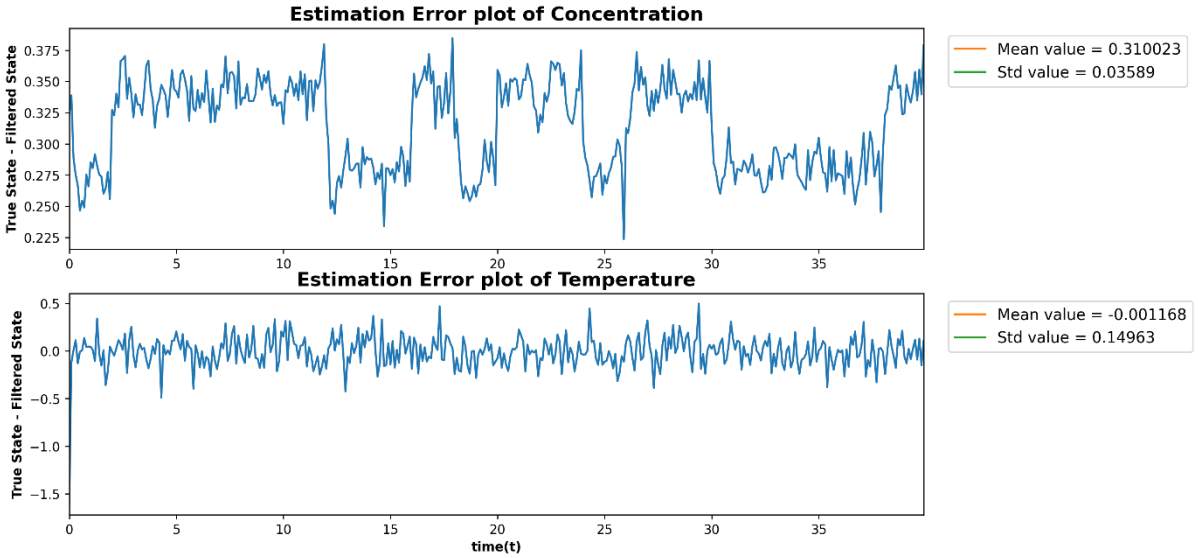

Figure A 3: ANN-EKF estimation error plots for reduced heat of reaction

A3 Partially trained ANN-EKF results for reduced heat of reaction

After the results of training dataset and test dataset were produced using a partially trained ANN model (section 7.3.4.1), now it is combined with the EKF state estimator for predictions of true concentration and temperature. The ANN-EKF predictions were not good without partial training as shown in Figure A 1, and as demonstrated in Figure A 4, After partial weight training, the ANN model produces exceptionally accurate true state predictions for both states.

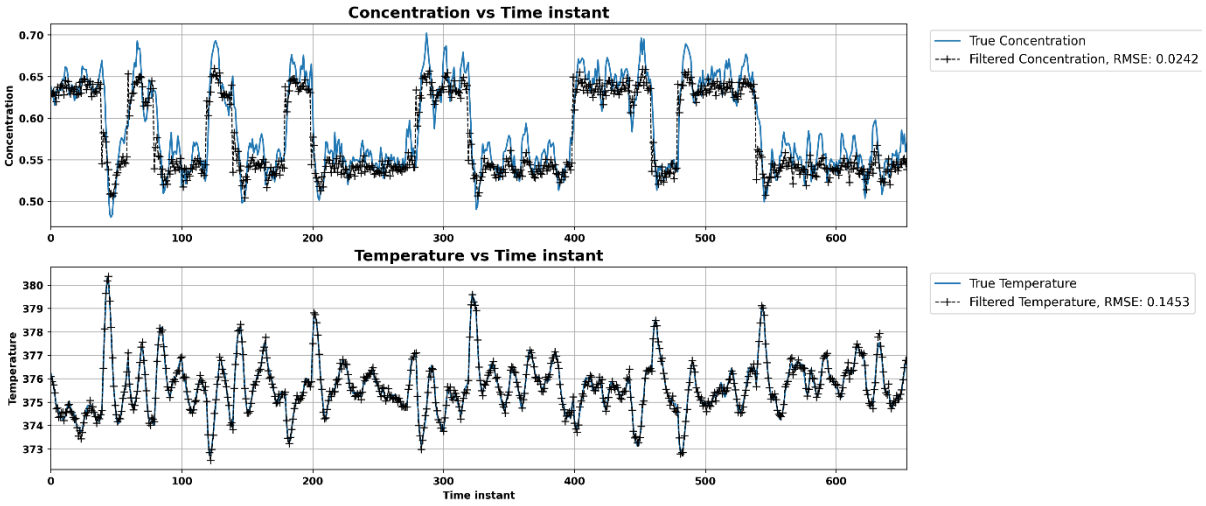

Figure A 4: Partially trained ANN-EKF results for reduced heat of reaction

If we compare Figure 7-10 with Figure A 4, there is an insignificant difference in the predictions of the true states even though temperature is the only measured state. If we compare Table 7-6 with Table A 1, it can be concluded that root mean square error values are moreover similar in both cases.

Table A 1: RMSE comparison for reduced heat of reaction

Model used	**Concentration predictions by EKF**	**Temperature predictions by EKF**	**Concentration predictions by the model**	**Temperature predictions by the model**
ANN initially	0.312	0.149	0.285	1.32
ANN after partial training	0.0241	0.145	0.0269	0.539

A4 Estimation error plots by partially trained ANN-EKF

If we compare Figure 7-12 when concentration and temperature both were considered as measured state with Figure A 5 when only the temperature is considered as a measured state, so we say that both the graphs are likely to be similar and show good performance after partial ANN training.

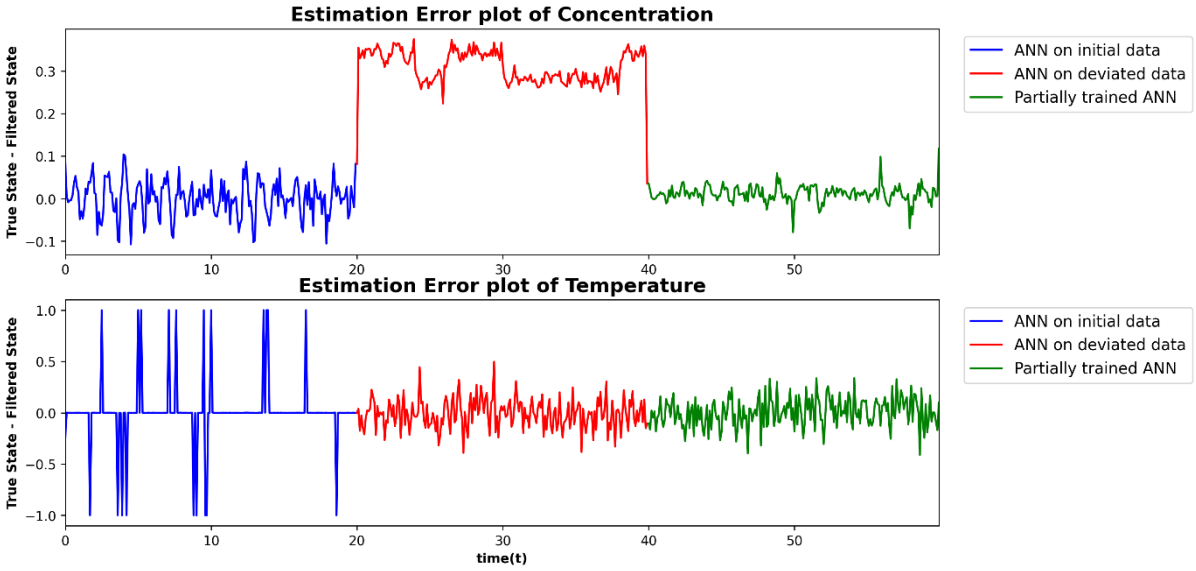

Figure A 5: Partially trained ANN-EKF estimation error plots for reduced heat of reaction

B1 ANN-EKF results for 40% reduced rate constant

ANN model's results are given in section 7.4.1 for reduced rate constant. In Figure B 1, ANN-EKF predictions are given when temperature was the only measured state and rate constant value was reduced by 40%.

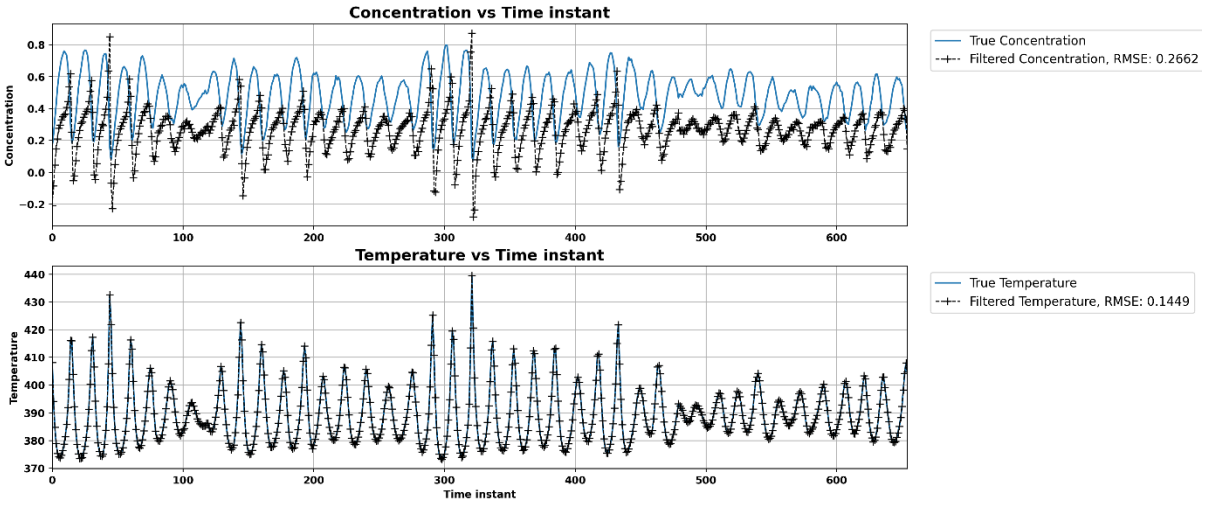

Figure B 1: ANN-EKF predictions for reduced rate constant

B2 Innovation and estimation error plots for the reduced rate constant

As compared to Figure 7-16, in Figure B 2we can see that the innovations are fluctuating around zero when we take temperature as a measured state.

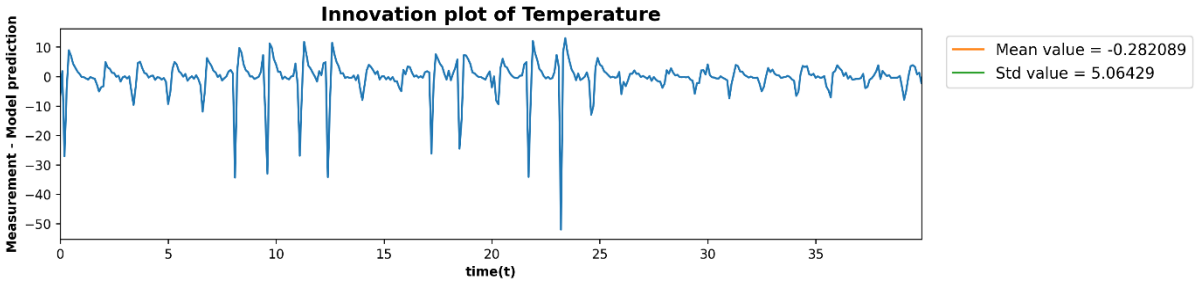

Figure B 2: Innovation plot for reduced rate constant

If we compare Figure 7-17 and Figure B 3, we can see that the estimation error values are increased for concentration and decreased for temperature.

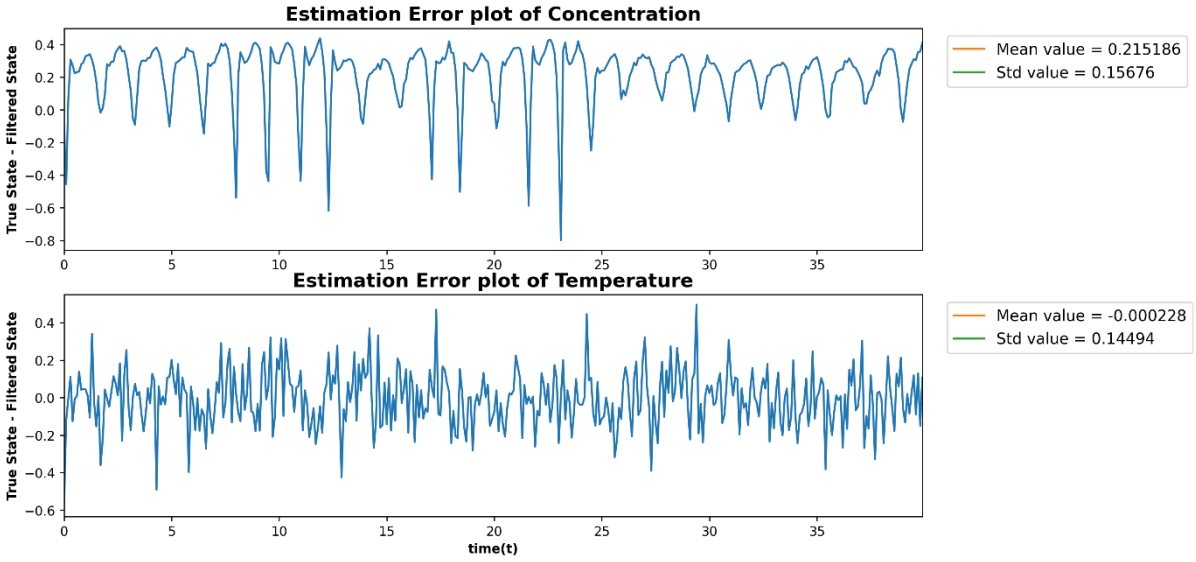

Figure B 3: Estimation error plots for reduced rate constant

B3 Partially trained ANN-EKF results for the reduced rate constant

After the results of training dataset and test dataset were produced using a partially trained ANN model (section 7.4.3.1), now it is combined with the EKF state estimator for predictions of true concentration and temperature. The ANN-EKF predictions were not good

without partial training as shown in Figure B 1, and as demonstrated in Figure B 4, After partial weight training, the ANN model produces exceptionally accurate true state predictions for both states.

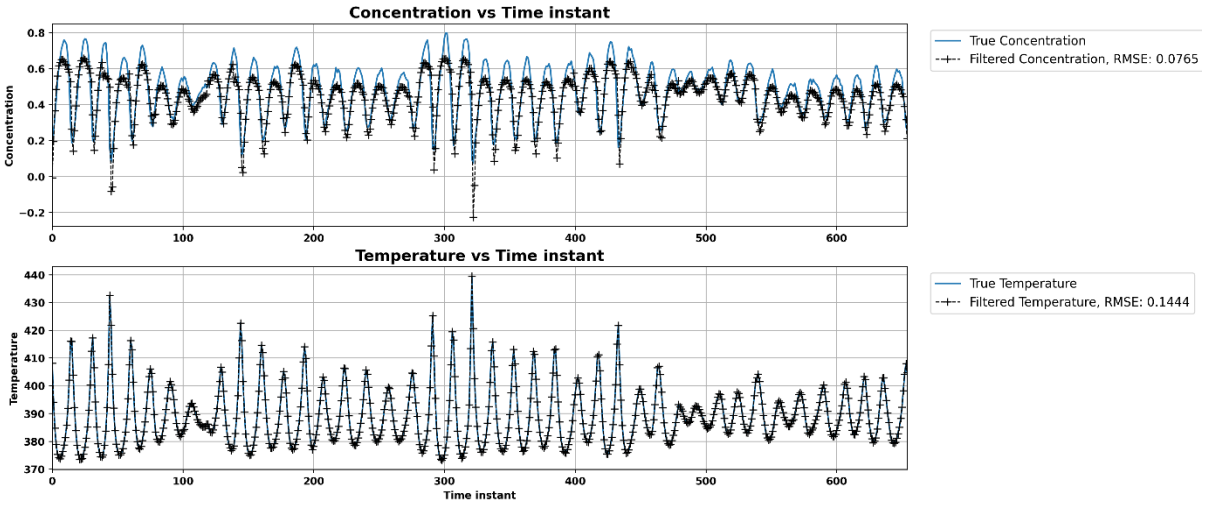

Figure B 4: Partially trained ANN-EKF results for reduced rate constant

If we compare Figure 7-20 with Figure B 4, there is an insignificant difference in the predictions of the true states even though temperature is the only measured state. If we compare Table 7-10 with Table B 1, it can be concluded that root mean square error values are moreover similar in both cases.

Table B 1: RMSE comparison for reduced rate constant

Model used	**Concentration predictions by EKF**	**Temperature predictions by EKF**	**Concentration predictions by the model**	**Temperature predictions by the model**
ANN initially	0.266	0.144	0.3021	5.065
ANN after partial training	0.0765	0.144	0.0804	4.005

B4 Estimation error plots by partially trained ANN-EKF

If we compare Figure 7-22 when concentration and temperature both were considered as measured state with Figure B 5 when only temperature is considered as a measured state,

so we say that both the graphs are likely to be similar and show good performance after partial ANN training.

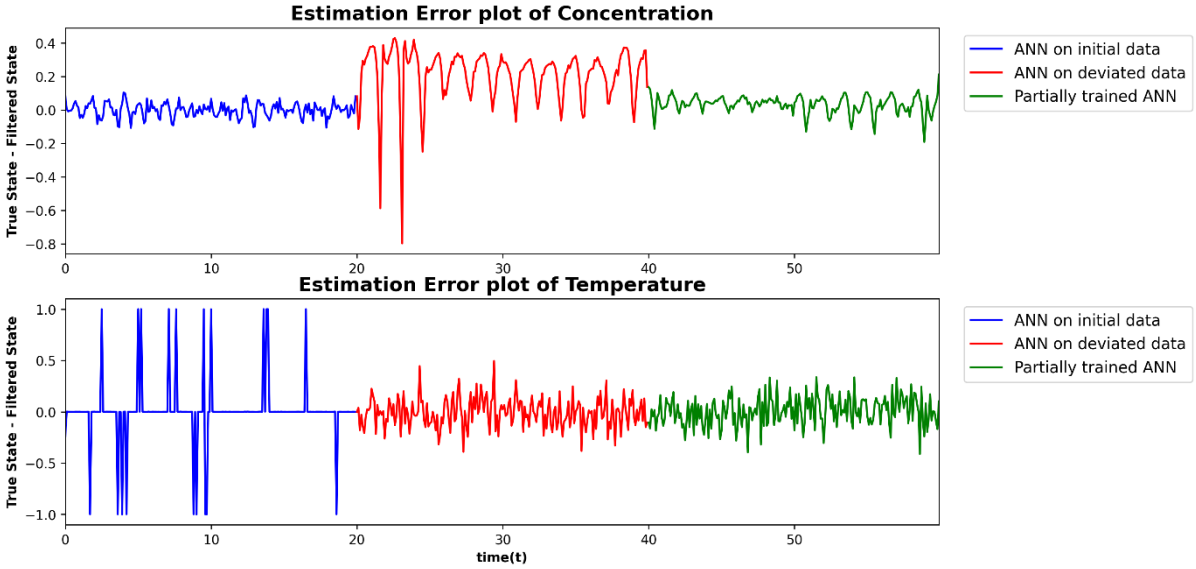

Figure B 5: Estimation error plots by partially rained ANN-EKF for reduced rate constant

References

[1] B. Alemayehu Ambaw and D.-I. Berhanu Assefa Dr-Ing Nurelegne Tefera, "addis ababa university school of graduate studies faculty of technology department of chemical engineering modeling chemical engineering processes using artificial neural networks a thesis submitted to the school of graduate studies of addis ababa university in partial fulfillment of the requirements for the degree of masters of chemical engineering," 2005.

[2] R. Archana, A. Unnikrishnan, and R. Gopikakumari, "An improved EKF based neural network training algorithm for the identification of chaotic systems driven by time series," 2012. doi: 10.1109/EPSCICON.2012.6175233.

[3] R. Grbić, D. Slišković, and P. Kadlec, "Adaptive soft sensor for online prediction based on moving window gaussian process regression," in *Proceedings - 2012 11th International Conference on Machine Learning and Applications, ICMLA 2012*, 2012, vol. 2, pp. 428–433. doi: 10.1109/ICMLA.2012.160.

[4] K. S. Narendra Fellow and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," 1990.

[5] K. Srinivasan and J. Prakash, "Non-linear state estimation for continuous stirred tank reactor using neural network state filter," 2006. doi: 10.1109/INDCON.2006.302760.

[6] D. Łuczak and A. Wójcik, "Neural State Estimator for Complex Mechanical Part of Electrical Drive: Neural Network Size and Performance of State Estimation," *Power Electronics and Drives*, vol. 3, no. 1, pp. 205–216, Dec. 2018, doi: 10.2478/pead-2018-0017.

[7] C. C. Ku and K. Y. Lee, "Diagonal Recurrent Neural Networks for Dynamic Systems Control," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 144–156, 1995, doi: 10.1109/72.363441.

[8] H. J. Sena, F. V. da Silva, and A. M. F. Fileti, “ANN model adaptation algorithm based on extended Kalman filter applied to pH control using MPC,” *Journal of Process Control*, vol. 102, pp. 15–23, Jun. 2021, doi: 10.1016/j.jprocont.2021.04.001.

[9] J. A. Wilson and M. Zorzetto, “A generalised approach to process state estimation using hybrid artificial neural network/mechanistic models,” 1997.

[10] Prof. Mani Bhushan and Prof. Sachin Patwardhan, *state estimation part 1: optimization based formulations*. 2018.

[11] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980

[12] Y. Wang, “A new concept using LSTM Neural Networks for dynamic system identification,” in *Proceedings of the American Control Conference*, Jun. 2017, pp. 5324–5329. doi: 10.23919/ACC.2017.7963782.

[13] W. der Chang, “Recurrent neural network modeling combined with bilinear model structure,” *Neural Computing and Applications*, vol. 24, no. 3–4, pp. 765–773, Mar. 2014, doi: 10.1007/s00521-012-1295-5.

[14] IEEE Neural Networks Council. and Institute of Electrical and Electronics Engineers., *The 1997 IEEE International Conference on Neural Networks, June 9-12, 1997, Westin Galleria Hotel, Houston, Texas, USA*. Institute of Electrical and Electronics Engineers, 1997.

[15] T. E. Marlin, “Process Control Designing Processes and Control Systems for Dynamic Performance 2 nd Edition,” 2015.

[16] prof. Sachin Patwardhan, *Advanced process control*. 2018.

Acknowledgment

I wish to express my deep gratitude to my supervisor, Prof. Mani Bhushan, for his noble guidance and valuable suggestions. I'm thankful to him for mentoring me throughout, with a lot of patience and constant support. Your continuous guidance and encouragement throughout the year have given me the confidence and strength to complete the M.Tech thesis in the best possible manner.

I would also like to express my gratitude to Prof. Sharad Bhartiya, Prof. Sachin Patwardhan for clearing my doubts and guiding me in the right direction whenever I needed. I would also like to thank my senior colleague, Vinayak Agrawal, Richa katare, Pravin jhaveri, Aman garg, Preet suntharia for their valuable input and advice. Also, I would like to thank Janak, Vishal, Sai darshan, Bhargav, Darshak and Kaushal for always helping me out with every problem whether it is academic or personal and giving me suggestions on valuable moments.

I am incredibly grateful to my parent, Mrs. Pratima mishra, for their love, prayers, caring, and sacrifices to educate and prepare me for my future. I would also like to express my thanks to my sister, Mrs. Mrunalini jagdev, for believing in me and supporting me, especially during hard times. I would like to thanks my grandfather Mr. Premsagar Awasthi and my uncle Mr. Siddharth Awasthi, who was there when I needed the financial and emotional support.

I am grateful to IIT Bombay for giving me access to good books and my friends for always offering me an impartial and honest perspective. Finally, I want to express my gratitude to everyone who has helped me accomplish the research work, whether directly or indirectly.

Mishra Digvijay
IIT Bombay
21st June 2022