Available online at www.sciencedirect.com

Chemical Engineering Research and Design

journal homepage: www.elsevier.com/locate/cherd

Physics-informed machine learning modeling for predictive control using noisy data

Mohammed S. Alhajeri[a,c], Fahim Abdullah[a], Zhe Wu[d], Panagiotis D. Christofides[a,b,*]

[a] Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095-1592, USA
[b] Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095-1592, USA
[c] Department of Chemical Engineering, Kuwait University, P.O.Box 5969, Safat13060, Kuwait
[d] Department of Chemical and Biomolecular Engineering, National University of Singapore, 117585, Singapore

ARTICLE INFO

Article history:
Received 6 June 2022
Received in revised form 21 July 2022
Accepted 22 July 2022
Available online 26 July 2022

Keywords:
Process control
Model predictive control
Nonlinear processes
Machine Learning
Recurrent neural networks
Aspen Plus Dynamics

ABSTRACT

Due to the occurrence of over-fitting at the learning phase, the modeling of chemical processes via artificial neural networks (ANN) by using corrupted data (i.e., noisy data) is an ongoing challenge. Therefore, this work investigates the effect of both Gaussian and non-Gaussian noise on the performance of process-structure based recurrent neural networks (RNN) models, which take the form of partially-connected RNN models in this work, that are used to approximate a class of multi-input-multi-outputs nonlinear systems. Furthermore, two different techniques, specifically Monte Carlo dropout and co-teaching, are utilized in the development of partially-connected RNN models. These two techniques are employed to reduce the over-fitting in ANNs when noisy data is used in the training process and, hence, to improve the open-loop accuracy as well as the closed-loop performance under a Lyapunov-based model predictive controller (MPC). Aspen Plus Dynamics, a well-known high-fidelity process simulator, is used to simulate a large-scale chemical process application in order to demonstrate the anticipated improvements in both open-loop approximation and closed-loop controller performance in the presence of Gaussian and non-Gaussian noise in the data set using physics-informed RNNs.

1. Introduction

The fundamental benefit of data-driven modeling techniques is that no prior knowledge of the process is required. However, historical data including key measured process variables is required to develop data-driven models. The effectiveness of data-driven modeling methods is based on the quality and amount of process data. Moreover, owing to their capability to analyze data in vast quantities from industrial processes, machine learning techniques have attracted significant attention in traditional engineering fields recently. Machine learning is a broad family of techniques including neural networks and their variants, which have been used successfully in regression and classification problems such as process modeling, process monitoring, and fault detection. Recurrent neural networks (RNN) and long short-term memory (LSTM) networks are two of the many types of neural networks that have gained popularity for modeling nonlinear dynamic systems from sequential data (i.e., time-series data), and have been used in advanced control strategies such as model predictive control (MPC) to predict the evolution of process states when first-principles process models are unavailable (e.g., Drgoňa et al., 2018; Wu et al., 2019; Chen et al., 2020). While many studies have been conducted on utilizing neural networks for modeling of chemical processes using clean/noise-free data, learning with noisy

* Corresponding author at: Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095-1592, USA.
E-mail address: pdc@seas.ucla.edu (P.D. Christofides).

https://doi.org/10.1016/j.cherd.2022.07.035

data is an essentially challenging task due to the high likelihood of neural networks ending up fitting the noisy data (i.e., over-fitting). Given that noise commonly affects sensor measurements in chemical plants during real-time operation, modeling of chemical processes via machine learning approaches utilizing industrial data (i.e., corrupted with noise) continues to be an important research topic.

To mitigate the effect of noise in data sets during modeling, several methods have been proposed in the literature. For linear dynamical systems, the famous Kalman filtering is one approach for dealing with noisy measurements. Furthermore, other approaches have been proposed such as unscented Kalman filter and moving horizon estimation (e.g., Patwardhan et al., 2012). An accurate explicit model representation is generally required in the state estimation technique in order to achieve a correct estimation, and also tuning of the co-variance matrices is required (Lima and Rawlings, 2011). The result of learning using raw vibration signals generated from water flow system in a laboratory-scale was recently investigated by Shah et al. (2020) using three different approaches; linear statistical learning approach, LSTM neural networks, and feed-forward neural network (FNN). According to their analyses, both the machine learning methods and the linear statistical model under-performed when utilizing raw vibration signals, and additional data treatment was required to enhance the developed models performance.

Since machine learning approaches were originally developed in the field of computer science, they generally either presume the access to high-fidelity data, or refer to mislabeling in classification tasks and numerical fallacies in regression tasks as "noisy data" (Han et al., 2018). As a result, the accuracy and effectiveness of these methods are typically evaluated using noise-free data sets. However, finding and collecting noise-free data in the domain of science and engineering, especially chemical engineering, remains a long-standing challenge. Thus, numerous works in the literature investigate machine learning methods with various types of noise. For instance, in the work of Hsu and Wang (2009), the robustness of an RNN model of Wiener type is evaluated by two types of noise: white noise and sinusoidal noise. Furthermore, a study by Krishnaiah et al. (2006) investigated the impact of Gaussian noise on an RNN modeling of chaotic systems represented by short time-series. Moreover, data quality can also be improved by using data preparation and smoothing techniques. For foaming control implementation in bio-processes using ensemble-based machine learning method, noisy and repetitive data are filtered in the work of Agarwal et al. (2019). In the same vein, Ma et al. (2020) conducted a data smoothing (i.e., pre-treatment) and dealt with incomplete data points by applying a third-order polynomial to the experimental data and then combined it with an ANN to create a deep reinforcement learning strategy to control a bio-reactor.

Many machine learning modeling algorithms can handle Gaussian noise. However, non-Gaussian noise can cause a degraded modeling performance between the input and the (noise-free) ground-truth output, and this is because of its over-fitting of the corrupted training data set's noisy behavior. In nonlinear processes modeling by machine learning algorithms exposed to industrial data noise with a non-Gaussian distribution, the work of Wu et al. (2021) utilized Monte Carlo dropout and co-teaching. The Monte Carlo dropout strategy in the neural network training process is an excellent way to minimize over-fitting to non-Gaussian noisy input without requiring any a prior process knowledge. As for the co-teaching technique, it's essentially using noise-free data generated from a first-principles model to mitigate the impact of noise in the training phase of machine learning models. Abdullah et al. (2022) used a training data set where 20% of the data set was noise-free to improve the overall modeling performance using co-teaching method.

Standard RNN models, also known as fully-connected RNN models, are a popular option for analyzing time-series data within a black-box modeling framework. Such a modeling methodology, however, may not always be preferable, particularly for large chemical processes due to the complex interactions among process variables. Hence, to improve RNN performance, several studies (e.g., Stephanopoulos and Han, 1996; Kahrs and Marquardt, 2007; Zhang et al., 2019) have looked into gray-box modeling, also referred to as hybrid modeling, which involves integration of a prior physical knowledge and expertise into the modeling of neural networks. Another method is to reflect physical relations among the given process inputs and outputs into the modeling of neural networks (i.e., known as partially-connected modeling) which was proposed in Wu et al. (2020). In this direction, Alhajeri et al. (2022) examined the partially-connected method by comparatively investigating open-loop and closed-loop simulations utilizing a fully-connected RNN model against a partially-connected RNN model on a large-scale complex chemical process modeled in Aspen Plus Dynamics. It was demonstrated that a partially-connected RNN model outperformed a fully-connected RNN model in terms of smoother state trajectories and lower computational burden under the MPC. Yet, to our knowledge, the performance of partially-connected RNN has not been studied in the presence of industrial noise.

Taking into consideration the preceding factors, the current study takes noise into account and attempts to evaluate the performance of a partially-connected RNN-based MPC through application on a large-scale nonlinear chemical process. Initially, we use the process simulators Aspen Plus and Aspen Plus Dynamics to create a simulation model of a chemical plant that produces Ethylbenzene via two continuous stirred tank reactors (CSTR) in series. Then, we carry out extensive open-loop simulations using Aspen dynamical model to construct a base data set, which will be corrupted with two types of noise (i.e., Gaussian and non-Gaussian) to obtain two separate data sets based on the noise type. Subsequently, we train a standard partially-connected RNN model as described in Alhajeri et al. (2022) using the noisy data. Then, two other models are developed by employing the co-teaching and Monte Carlo dropout techniques, respectively. Eventually, we evaluate both open-loop and closed-loop performance of each model, and show the benefits of using the two proposed approaches to overcome noisy data presence in a partially-connected RNN-based MPC framework.

The remainder of this manuscript is structured as follows: In Section 2, the class of nonlinear chemical process systems, mathematical notation, and stabilizing feedback control law assumptions are discussed. Next, the conceptualization and the development of partially-connected RNN models, and LSTM models are introduced in Section 3. Subsequently, Sections 4 and 5 introduce the concepts of co-teaching and Monte Carlo dropout techniques, respectively. The incorporation of a partially-connected RNN model into a model predictive controller with Lyapunov stability assumption is

proposed and discussed in Section 6. In Section 7, the open-loop as well as the closed-loop performances of MPCs using different RNN models that underwent different training procedures are assessed utilizing chemical process application modeled in Aspen Plus Dynamics simulator.

2. Preliminaries

2.1. Notations

Throughout this work, the Euclidean norm of a vector κ is represented by the notation $|\kappa|_2$. The standard Lie derivative is notated by $L_f h(x) = \frac{\partial h(x)}{\partial x} f(x)$. Set subtraction operator used in this work is " $-$ ", as in $A - B = \{x \in \mathbf{R}^n | x \in A, x \notin B\}$. A function $f(x)$ is regarded as of class C^1 if it is continuously differentiable in its domain.

2.2. Class of systems

We consider the class of nonlinear continuous-time multi-input multi-output (MIMO) systems with the following state-space form:

$$\dot{x} = F(x, u) := f(x) + g(x)u, \qquad x(t_o) = x_o \tag{1a}$$

$$y = x + \nu \tag{1b}$$

where $x = [x_1, \ldots, x_n]^T \in \mathbf{S}^n$ denotes the state vector, and the vector of manipulated variables (i.e., inputs) is denoted by $u = [u_1, \ldots, u_r]^T \in \mathbf{S}^r$. The term $y = [y_1, \ldots, y_n]^T \in \mathbf{S}^n$ represents the vector of state measurements that are continuously sampled, and the measurements noise vector is denoted by $\nu \in \mathbf{R}^n$. The input vector u is bounded by a lower bound $u^{\min}$ and an upper bond $u^{\max}$ and both are $\in \mathbf{R}^r$. $f(\cdot)$ is a vector function $\in \mathbf{R}^{n\times 1}$ and $g(\cdot)$ is a matrix function $\in \mathbf{R}^{n\times r}$, and both are assumed to be adequately smooth. We assume that the entire state vector is in deviation form from the steady state of the considered system, such that when $\nu(t) = 0$ and the function $F(0, 0)$ is equal to zero then the origin is a steady-state of the nominal system in Eq. 1a, i.e., $(x_s, u_s) = (0, 0)$, where the subscript "s" indicates the steady-state.

2.3. Stabilizability Assumption

For closed-loop stability considerations, a stabilizing feedback controller $u = \Phi(x) \in U$, where $U := \{u^{\min} \le u \le u^{\max}\}$ is assumed to exist. This controller is assumed to be able to enforce the steady state (i.e., the origin) of the system of Eq. 1 to be exponentially stable in a neighborhood around the origin. Such an assumption implies that a control Lyapunov function of class $\mathscr{C}^1$ exists and is represented by $V(x)$, such that for all x in an open neighborhood D around the origin, the following inequalities hold:

$$c_1|x|^2 \le V(x) \le c_2|x|^2, \tag{2a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \Phi(x)) \le -c_3|x|^2, \tag{2b}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \le c_4|x| \tag{2c}$$

where c_i are positive real numbers $\forall\, i \in \{1, \ldots, 4\}$. There are several methods to construct the controller $\Phi(x)$; for instance, a possible method is the universal Sontag's control law (Lin and Sontag, 1991). Other methods can also be applied, such as finding a well-tuned proportional control law (i.e., P-controller). Once the controller is chosen, subsequently, following Wu et al. (2019) the closed-loop stability region Ω_ρ is defined to be a level set of $V(x)$ within the region D that is characterized under the controller $u = \Phi(x) \in U$, i.e., $\Omega_\rho := \{x \in D | V(x) \le \rho, \rho > 0\}$.

3. Recurrent neural networks model (RNN)

RNN models, as noted in the introduction, are an effective tool for modeling time-series data. The hidden layer neurons' recursive action enables the RNN to retain the memory of earlier states, allowing it to adequately mimic a time-series dataset behaviour. RNN models are utilized to estimate the nonlinear system of Eq. (1a) using process operational data in this study is represented as follows:

$$\dot{\bar{x}} = F_{rnn}(\bar{x}, u) := A\bar{x} + \Theta^T \gamma \tag{3}$$

where the RNN state vector is $\bar{x} = [\bar{x}_1, \ldots, \bar{x}_n]$, and $u = [u_1, \ldots, u_r]$ is the vector of the manipulated inputs. The vector γ is based on $\bar{x}$ and u, and is defined as $[\gamma_1, \ldots, \gamma_n, \gamma_{n+1}, \ldots, \gamma_{n+r}] = [\alpha(\bar{x}_1), \ldots, \alpha(\bar{x}_n), u_1, \ldots, u_r] \in \mathbf{S}^{n+r}$. The notation $\alpha(\cdot)$ denotes the nonlinear activation function utilized for the activation of the hidden layers. Such activation functions include the sigmoid function and the hyperbolic tangent function. The diagonal matrix $A = diag[-a_1, \ldots, -a_n]$ consists of negative coefficients with each $a_i > 0$ with the intention that the states are kept stable in the sense of bounded-input-bounded-state stability. Regarding the matrix $\Theta = [\theta_1, \ldots, \theta_n] \in \mathbf{R}^{(n+r)\times n}$, it consists of a vector $\theta_i = b_i[w_{i1}, \ldots, w_{i(n+r)}]$, where the elements of each θ_i are b_i, which are constants, and w_{ij}, which is the weight on the interrelation that links the jth input to the ith neuron, where $j \in \{1, \ldots, (n+r)\}$, and $i \in \{1, \ldots, n\}$.

3.1. Partially-connected RNN

For constructing a dynamic model for a nonlinear process, a neural network model that takes all available process inputs and predicts the desired outputs is usually preferred. Using open-source machine learning software, a dynamic RNN model for such processes may be simply developed, and this model will be able to account for all conceivable correlations between each input and output of the underlying process. The general construction of such a fully-connected RNN with an input layer, hidden layers, and an output layer is depicted on the left side of Fig. 1. As a result, fully-connected RNN models are typically the best initial choice for modeling processes where no prior knowledge is available.

The term "process prior knowledge" is defined as a physical understanding of the process considered that exists before the derivation of the first-principles process model. It involves, but is not restricted to, the model's intended goal, each and every hard and soft physical constraint imposed on the process as a result of design considerations, as well as the process structure. We focus on process structure knowledge in the form of connections between process input and output variables in this paper. Physical connections between process variables can sometimes be straightforward, especially in the chemical process industry. Upstream processes, for example, have an effect on downstream processes, whereas the opposite effect may not exist. This relationship among upstream and downstream stages is frequently reflected explicitly in a first-principles mathematical model. As a result, as discussed in Thompson and

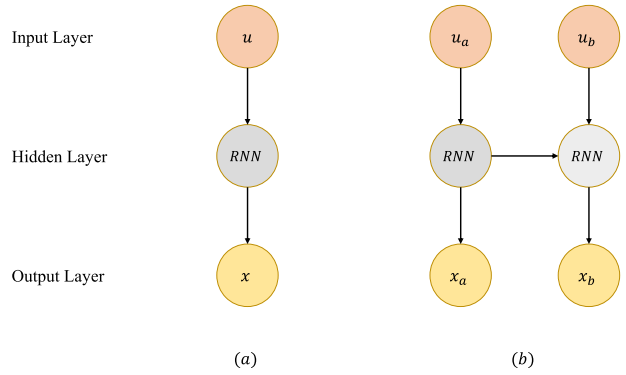

Fig. 1 – Schematic of (a) Standard and (b) Physics-informed RNN structures, with $u = [u_a, u_b]$ and $x = [x_a, x_b]$.

Kramer (1994), integrating process knowledge that defines physical relationships among the considered system's inputs and outputs into neural network modeling enhance its performance. This modeling approach is referred to as partially-connected neural network modeling.

We consider the system with $x = [x_a, x_b]$ and $u = [u_a, u_b]$, such that $x_a \in \mathbf{5}^{n_1}$, $x_b \in \mathbf{5}^{n_2}$, $u_a \in \mathbf{5}^{r_1}$, and $u_b \in \mathbf{5}^{r_2}$ where $r_1 + r_2 = r$ and $n_1 + n_2 = n$. It is assumed that the input vector u_a exclusively influences the state vector x_a, whereas x_b is influenced by both u_a and u_b. Physical knowledge is incorporated into the RNN modeling of the nominal system by modifying the RNN structure to explicitly disconnect the links between u_b and x_a. In due course, an enhanced model approximation is achieved as a result of implementing partially-connected RNN model as discussed and demonstrated in Alhajeri et al. (2022) and Wu et al. (2020).

The development framework for partially-connected RNN models is similar to that of fully-connected RNN models, but with additional specifications. Alhajeri et al. (2022) discussed the construction and training processes of partially-connected RNN models and provided the pseudocode of these modeling methods. Basically, a training/validation dataset can be collected from experimental and industrial sources, or through extensive open-loop simulations. The general principle of splitting the collected data set into 70% training and 30% validating can be followed, or more advanced techniques such as cross-validation may be employed. Prior to training RNN models, the input vectors u_a and u_b, as well as the output vectors x_a and x_b, should be defined, which is also referred to as data preprocessing.

To build and train the RNN models in this study, we used the Keras library, which is an open source Python library. Our models consist of four layers: an input layer, two hidden layers, and an output layer, where the hidden neurons are activated by nonlinear functions: hyperbolic tangent and sigmoid functions. Instead of feeding the full input vector u through one input layer, in partially connected RNN models the input vectors u_a and u_b are fed independently through different input layers according to the process structure, as illustrated in Fig. 1. The first hidden layer predicts the output vector x_a based on the input vector u_a. Subsequently, u_b and x_a are concatenated and then sent to the subsequent hidden layer to estimate x_b. Ultimately, the developed model will be able to estimate both the output vectors x_a and x_b given u_a and u_b.

3.2. *Long short term memory (LSTM)*

Long-short-term memory networks, or LSTMs in short, are a class of RNNs, and henceforth will be referred to as RNN-LSTM. Due to the design of three gates in the network structure, namely the input gate, the forget gate, and the output gate, RNN-LSTM networks are capable of capturing long-term dependencies in problems involving sequential prediction. Fig. 2 shows a diagram of an LSTM network. In this study, RNN-LSTM based models are developed in the framework of partially-connected modeling. When given control actions

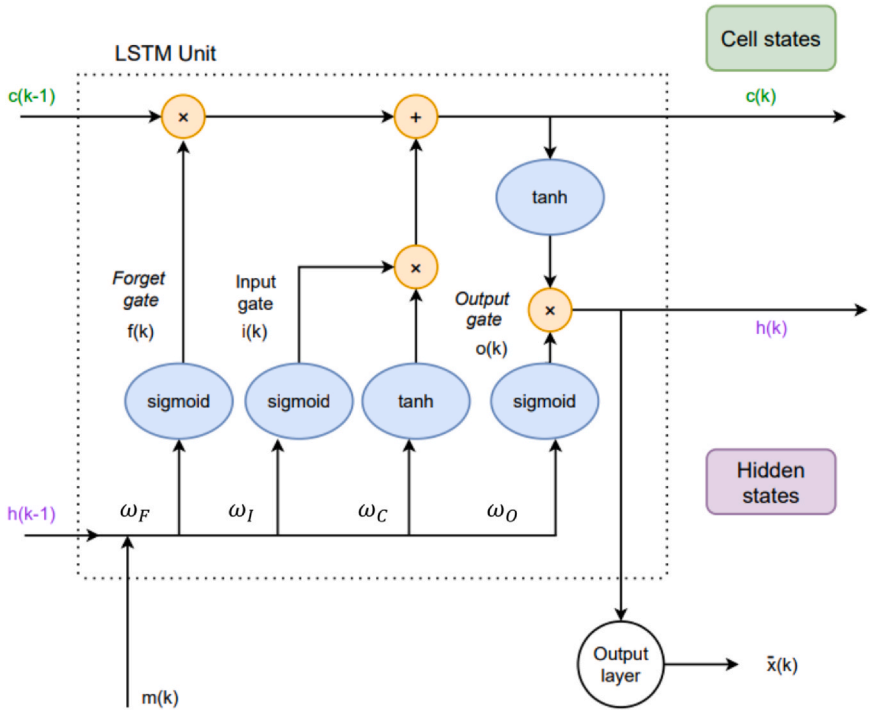

Fig. 2 – RNN-LSTM network schematic.

and previous noisy state measurements, the generated RNN-LSTM models are used to predict the states of Eq. 1a. In particular, considering the input sequence $m(k)$, where $k = 1, \ldots, T$ and T denotes the number of measured states of the nominal system in Eq. 1a, the approximate output sequence $\bar{x}(k)$ is computed using the following equation:

$$i(k) = \sigma(\omega_i^m m(k) + \omega_i^h h(k-1) + b_i) \tag{4a}$$

$$f(k) = \sigma(\omega_f^m m(k) + \omega_f^h h(k-1) + b_f) \tag{4b}$$

$$c(k) = f(k)c(k-1) + i(k)\tanh(\omega_c^m m(k) + \omega_c^h h(k-1) + b_c) \tag{4c}$$

$$o(k) = \sigma(\omega_o^m m(k) + \omega_o^h h(k-1) + b_o) \tag{4d}$$

$$h(k) = o(k)\tanh(c(k)) \tag{4e}$$

$$\bar{x}(k) = \omega_y h(k) + b_y \tag{4f}$$

where $m(k)$ denotes the input sequence, while $c(k)$ and $h(k)$ are the cell state and the internal state, respectively. Regarding the gates, $f(k)$ is the forget gate, $o(k)$ represents the output gate, and the output of the input gate is denoted by $i(k)$. The output sequences $\bar{x} \in \mathbf{5}^{n\times T}$ is the RNN-LSTM network output.

The weight matrices here are given by ω_p^l, where $p \in \{i, c, f, o\}$ represents the gate or the state, while l represents the associated vector (i.e., either m or h). Similarly, the bias term is denoted by b_p. Eventually, Eq. 4f is used to compute the predicted state of the RNN-LSTM, which the terms b_y and ω_y represent the bias vector and the output weight matrix, respectively. Because the RNN-LSTM model predicts future states using control actions and previous state measurements, the input sequence $m \in \mathbf{R}^{(n+r)\times T}$ consists of manipulated input $u \in \mathbf{R}^r$ along with previous measured states $x \in \mathbf{R}^n$ during a given time period (i.e., T). The nonlinear activation functions used in the LSTM model are $\sigma(\cdot)$ and $\tanh(\cdot)$, which are sigmoid and hyperbolic tangent functions, respectively.

4. Co-Teaching technique

Noisy data in classification problems could result in mislabelling (e.g., an image with a label "A" is mislabeled as "B"), and for regression problems noise in the data might result in a deviation from its ground truth value (i.e., the true value). In both cases, it is quite challenging for a machine learning model to fulfill the desired model accuracy with a noisy data set following the standard learning algorithms.

The co-teaching technique was originally proposed to improve the accuracy of ML-based models in image classification problems when the training data set is corrupted with noise (Han et al., 2018). Abdullah et al. (2022) utilized co-teaching in the context of regression to handle noisy data with an observed enhancement in model accuracy. In the same vein, to reduce the affect of noisy data on RNN-LSTM modeling, Wu et al. (2021) employed the co-teaching method, and achieved better closed-loop performance under MPC when applied to a reactor example in comparison with the standard RNN-LSTM model. To our knowledge, little attention has been paid to the implementation of the co-teaching method in solving regression problems. Hence, there is growing research into extending the co-teaching technique to regression problems using RNN-LSTM networks.

The idea behind the co-teaching technique comes from the fact that earlier in the training process, neural networks would employ a simple pattern to fit training data (Han et al., 2018). As a result, when evaluating the loss function value under a simple pattern that approximates the relationship between inputs and outputs of a neural network, noisy data typically have a large loss function value, and noise-free data typically have a small value. Therefore, merging the two data sets provides a possible way to train machine learning models in the presence of noisy data by benefiting from noise-free data. Such clean data can be generated from simulation of first-principles models. Therefore, mixing the two data sets in some ratio (i.e., noisy data set and noise-free data set) can improve the robustness of the RNN-LSTM model training process to over-fitting to noisy data. To apply this technique, after merging the two data sets (i.e., the noisy and the clean data) into one data set, then the new data set is to be shuffled first and then split into training and validating sets as a prerequisite for machine learning based model development. Subsequently, both partially-connected and fully-connected RNN models can be developed following the procedure proposed in Wu et al. (2020) and Alhajeri et al. (2022).

5. Dropout technique

Another candidate strategy to reduce over-fitting caused by noisy data, specifically non-Gaussian type (Wu et al., 2021), without having any prior process knowledge is to utilize a dropout technique in the neural network development process. In particular, as depicted in Fig. 3, a dropout strategy arbitrarily drops the associations between units in neighboring layers during training and provides an efficient method for combining various neural network structures to improve prediction accuracy.

Consider the RNN-LSTM model in its general form as in Eq. 3. For all RNN-LSTM layers, let $W = \{W_1, \ldots W_L\}$ denote the set of weight matrices that incorporate both weights and bias terms, where W_1 denotes the weight matrix associated with the first layer in the RNN-LSTM structure and L denotes the number of layers. The primary objective of the Monte Carlo (MC) dropout technique is to acquire the posterior distribution (i.e., an integration of the prior distribution as well as the likelihood function, which reveals what information the observed data contain) of the RNN-LSTM weights W, denoted by $p(W)$, from the training data (M, X) in which M and X are the input and output data matrices, respectively. The weight matrix W_i is described as follows, following the method in Gal and Ghahramani (2016a):

$$W_i = B_i \cdot Z_i \tag{5a}$$

where $Z_i = diag(z_i)$ and $i = 1, \ldots, L$. Each z_i is a set of binary variables that follows the well-known Bernoulli distribution and symbolizes the weights which are dropped out according to a particular user-defined probability, and B_i denotes the variational variables that shall be optimized. The RNN-LSTM predicted output distribution can be estimated by conducting Monte Carlo dropout at testing time after the RNN-LSTM model has been trained using the Monte Carlo dropout technique. Several realizations of the RNN-LSTM model are used to obtain the predicted distribution by taking the mean of the distribution as follows:

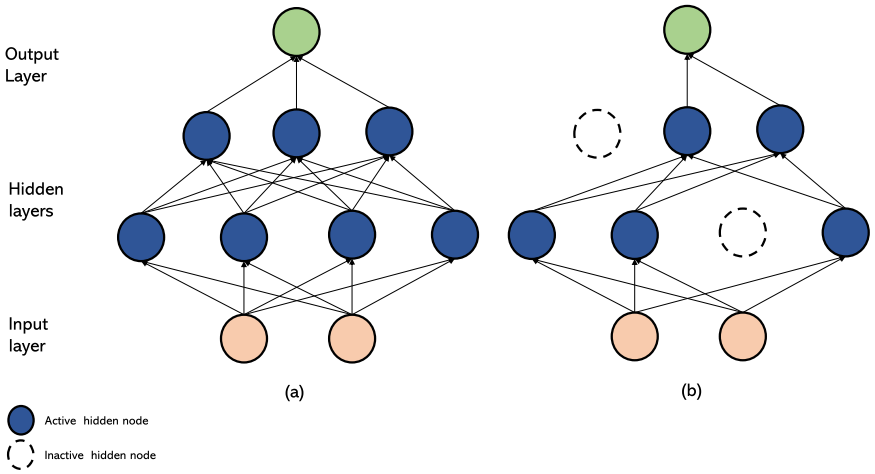

Fig. 3 – Fully-connected neural network layers (a) without dropout and (b) with dropout.

$$p(x^*|m^*,M,X) \approx \frac{1}{N_t}\sum_{i=1}^{N_t} p(x^*|m^*,W_i) \tag{6a}$$

where the RNN-LSTM input and output are m^* and x^* in the test set, respectively. The total number of Monte Carlo realizations at the testing stage is N_t. Given the same input, the RNN-LSTM output is no longer deterministic, because the RNN-LSTM model obtained utilizing the MC dropout technique is a probabilistic model, which is approximated by its mean value in this work, as shown in Eq. 6a. As a result of using Eq. 6a, a probabilistic distribution is obtained as an approximation of the model predictions' uncertainty. Furthermore, the ground-truth process dynamics can be roughly evaluated via the sample mean of all the model estimations.

The dropout rate (i.e., the likelihood of a neuron to be dropped out) plays a critical role in the model's performance, and is determined during model training to accomplish desired training/validation results. In particular, a small dropout value (e.g., 0.2–0.5) is suggested to begin with. Then, the user may increase the dropout rate value if over-fitting still occurs, or to decrease the value if the network fails to capture (i.e., learn) the dynamics of the underlying process.

Remark 1. Since the considered training data is corrupted with noise, our perception of the process dynamics via machine learning-based models utilizing the dropout strategy is expressed in a probabilistic manner. The Monte Carlo dropout method is an effective tool for modeling uncertain process dynamics and to estimate underlying nominal dynamics by a sequence of probabilistic forward passes. In particular, the RNN-LSTM model, developed via the Monte Carlo dropout technique, can be considered an uncertain process model, with the RNN-LSTM weights as uncertain variables.

Remark 2. The Monte Carlo dropout technique can be used to address over-fitting issues for different tasks with regards to chemical processes. For example, it can be employed to improve the machine learning-based model accuracy for state estimation, fault diagnosis, and other tasks in the presence of process noise, measurements noise, and other uncertainties associated with the process of interest. Furthermore, this technique may not only be used for regression but can also be applied to classification problems. The interested reader may refer to Gal and Ghahramani (2016a), Gal and Ghahramani (2016b), Srivastava et al. (2014), Wu and Zhao (2018), and Kwon et al. (2022) for details.

6. RNN-LSTM based model predictive control

In this section, we integrate an RNN-LSTM model into a Lyapunov-based model predictive controller (LMPC) formulation. In particular, the partially-connected modelling of RNN-LSTM is executed as discussed in Alhajeri et al. (2022) and then utilized as a predictive model to provide state estimation to solve the optimization problem of the LMPC, which is expressed in the following form:

$$\mathscr{J} = \min_{u\in S(\Delta)} \int_{t_k}^{t_k+P} L(\tilde{x}(t), u(t))dt \tag{7a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{rnn}(\tilde{x}(t), u(t)) \tag{7b}$$

$$u(t) \in U, \quad \forall \quad t \in [t_k, t_k + P) \tag{7c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{7d}$$

$$\dot{V}(x(t_k), u) \le \dot{V}(x(t_k)), \Phi_{nn}(x(t_k)), \quad \text{if} \quad x(t_k)\in\Omega_\rho - \Omega_{\rho_{nn}} \tag{7e}$$

$$V(\tilde{x}(t)) \le \rho_{nn}, \quad \forall \quad t \in [t_k, t_k + P), \quad \text{if} \quad x(t_k) \in \Omega_{\rho_{nn}} \tag{7f}$$

where $S(\Delta)$ denotes a set of piecewise constant functions with period Δ, $\tilde{x}$ is the state trajectory predicted by the RNN-LSTM model, and P is the prediction horizon expressed as a multiple of the sampling period (i.e., $P = N \times \Delta$, $N > 0$). The time-derivative of the Lyapunov function V in Eq. 7e is given as $\dot{V}(x, u)$, i.e., $\frac{\partial V(x)}{\partial x}(F_{rnn}(x, u))$. During the prediction horizon $t \in [t_k, t_k + P)$, the LMPC computes the optimum input sequence $u^*(t)$ and delivers the first control signal $u^*(t_k)$ to the system to be implemented for the following sampling period. After that, at the following sampling interval, the LMPC receives new data and is resolved with updated state estimations. Furthermore, the MPC optimization problem's goal is to minimize the integral of $L(\tilde{x}(t), u(t))$, given in Eq. 7a, which represents the cost function over the prediction horizon while satisfying the constraints of Eqs. 7b–7f. The RNN-LSTM model from Eq. 7b is used to forecast the evolution of the closed-loop state trajectory $\tilde{x}(t_k)$ under the MPC, and its initial conditions are updated according to Eq. 7d, where $x(t_k)$ is the last state measurement. The input constraints are expressed in Eq. 7c, and they are imposed across the prediction horizon.

To ensure the stability of the closed-loop system, when $x(t_k) \in \Omega_\rho - \Omega_{\rho_{nn}}$, where $\Omega_{\rho_{nn}}$ is the target region, the

condition of Eq. 7e is triggered. As a result of this constraint, the Lyapunov function of the closed-loop states declines, and the state approaches the steady-state within a finite period of time. Eventually, when the state $x(t_k)$ arrives to $\Omega_{\rho_{nn}}$, the predicted closed-loop state will be kept within this region for the duration of the prediction horizon. Following section 2.3, the controller $\Phi_{nn}(x)$ was developed with the intent of ensuring that the origin of the RNN-LSTM system of Eq. (3) is exponentially stable.

When using noise-free data for training, a well-conditioned RNN-LSTM model can be generated with adequate model accuracy. Therefore, by using the RNN-LSTM based LMPC of Eq. 7a-f to control a nonlinear system as that of Eq. 1a, the closed-loop state is assured to be bounded within the stability region Ω_ρ throughout the simulation time and eventually will converge to a small region around the origin under the condition that the modeling error, i.e., $|\nu| = |F(x, u) - F_{rnn}(\bar{x}, u)|$, is sufficiently small (Wu et al., 2019; Alhajeri et al., 2021).

7. Application to a chemical process using aspen plus simulator

In this section, we evaluate the proposed partially-connected RNN-based LMPC using a large-scale chemical process in the presence of industrial noise. First, we use Aspen Plus Dynamics V11 to create a dynamic model of a chemical process. Following that, a time-series data set of the process states and input variables is generated in order to train and test the RNN-LSTM models using extensive open-loop simulation. Subsequently, open-loop and closed-loop simulations using an RNN-model-based MPC are performed and discussed. Henceforth, PCRNN-LSTM will stand for partially-connected RNN-LSTM.

7.1. *Process description*

The process of producing Ethylbenzene (EB) from Ethylene (E) and Benzene (B) as reactive raw materials is used to demonstrate the performance of the proposed control strategy. There are three reactions in this process, which take place in two nonisothermal and well-mixed continuous stirred tank reactors (CSTR). The reaction scheme is shown below:

$$C_2H_4 + C_6H_6 \rightarrow C_8H_{10} \quad \text{(primary)} \tag{8a}$$

$$C_2H_4 + C_8H_{10} \rightarrow C_{10}H_{14} \tag{8b}$$

$$C_6H_6 + C_{10}H_{14} \rightarrow 2C_8H_{10} \tag{8c}$$

where the desired reaction is the second-order, exothermic, irreversible reaction labeled as “primary”.

The first-principles model for the two CSTRs is derived using mass and energy balances. Particularly, the dynamic model of the two reactors is given by the following system of ordinary differential equations (ODEs):

$$\frac{dC_{E_1}}{dt} = \frac{F_1 C_{Eo_1} - F_{out_1} C_{E_1}}{V_1} - r_{1,1} - r_{1,2} \tag{9a}$$

$$\frac{dC_{B_1}}{dt} = \frac{F_1 C_{Bo_1} - F_{out_1} C_{B_1}}{V_1} - r_{1,1} - r_{1,3} \tag{9b}$$

$$\frac{dC_{EB_1}}{dt} = \frac{-F_{out_1} C_{EB_1}}{V_1} + r_{1,1} - r_{1,2} + 2r_{1,3} \tag{9c}$$

$$\frac{dC_{DEB_1}}{dt} = \frac{-F_{out_1} C_{DEB_1}}{V_1} + r_{1,2} - r_{1,3} \tag{9d}$$

$$\frac{dT_1}{dt} = \frac{(T_{1_0} F_1 - T_1 F_{out_1})}{V_1} + \sum_{j=1}^{3} \frac{-\Delta H_j}{\rho_1 C_p} r_{1,j} + \frac{Q_1}{\rho_1 C_p V_1} \tag{9e}$$

$$\frac{dC_{E_2}}{dt} = \frac{F_2 C_{Eo_2} + F_{out_1} C_{E_1} - F_{out_2} C_{E_1}}{V_2} - r_{2,1} - r_{2,2} \tag{9 f}$$

$$\frac{dC_{B_2}}{dt} = \frac{F_2 C_{Bo_2} + F_{out_1} C_{B_1} - F_{out_2} C_{B_1}}{V_2} - r_{2,1} - r_{2,3} \tag{9 g}$$

$$\frac{dC_{EB_2}}{dt} = \frac{F_{out_1} C_{EB_1} - F_{out_2} C_{EB_2}}{V_2} + r_{2,1} - r_{2,2} + 2r_{2,3} \tag{9 h}$$

$$\frac{dC_{DEB_2}}{dt} = \frac{F_{out_1} C_{DEB_1} - F_{out_2} C_{DEB_2}}{V_2} + r_{2,2} - r_{2,3} \tag{9i}$$

$$\frac{dT_2}{dt} = \frac{(T_{2_0} F_2 + T_1 F_{out_1} - T_2 F_{out_2})}{V_2} + \sum_{j=1}^{3} \frac{-\Delta H_j}{\rho_2 C_p} r_{2,j} + \frac{Q_2}{\rho_2 C_p V_2} \tag{9j}$$

where the reaction rates are calculated by the following expressions:

$$r_{i,1} = k_1 e^{\frac{-E_1}{RT_i}} C_{E_i} C_{B_i} \tag{10a}$$

$$r_{i,2} = k_2 e^{\frac{-E_2}{RT_i}} C_{EB_i} C_{E_i}, \quad i = 1, 2 \text{ (reactor index)} \tag{10b}$$

$$r_{i,3} = k_3 e^{\frac{-E_3}{RT_i}} C_{DEB_i} C_{B_i} \tag{10c}$$

In this work, the two CSTRs are connected in series, so that the output of the first reactor affects the output of the second one, but not vice versa. Furthermore, the model of this process is created with Aspen Plus and Aspen Plus Dynamics V11, which are high-fidelity simulators that are used for steady-state and/or dynamics of complex chemical processes modeling. The process model is initially built in Aspen Plus, where steady-state simulation is carried out and solved using material and energy balances. Following that, we use Aspen Plus Dynamics to run a dynamic simulation of the underlying process to analyze and control its dynamical performance. The dynamic model is developed following the procedure described in Alhajeri et al. (2022), and the resulting flow sheet is shown in Fig. 4.

The flow rates F_1 and F_2 are the raw material feed to the first and second reactors, respectively. The concentrations of the species considered in this process are given as: C_E, C_B, C_{EB}, and C_{DEB} and they represent Ethylene, Benzene, Ethylbenzene, and Di-Ethylbenzene, respectively. Process parameters such as reactor temperature, mass density, and liquid volume of each CSTR are denoted, in the same order, by T_i, ρ_i, V_i where $i \in \{1, 2\}$ *and* refers to the CSTR index. The values of these parameters are listed in Table 1, which also includes the steady-state values and the liquid mixture's mass heat capacity, denoted as C_p which is considered constant in this work. The subscript “s” refers to steady-state value, and “o” represents the value at $t = t_o$. To control the reactors temperature, we added a cooling/heating jacket to each reactor that removes/provides heat to the reactor at a rate Q_i. The pressure is set to 15 bar initially for both reactors,

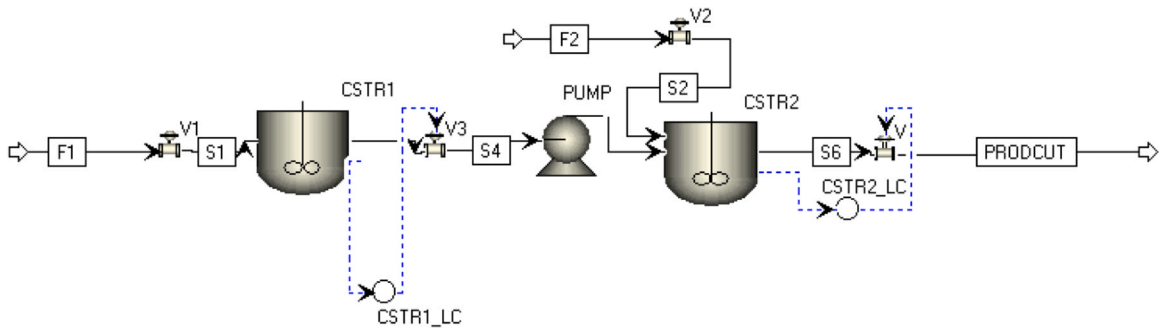

Fig. 4 – Aspen Plus model flow sheet of two chemical reactors in series.

Table 1 – Parameter values, steady-state values, and model configuration of the Aspen Plus model.

$T_{1_o} = 400$ K	$T_{1_s} = 310.523$ K
$T_{2_o} = 450$ K	$T_{2_s} = 430.542$ K
$F_1 = 43.2 m^3/hr$	$F_2 = 91.079 m^3/hr$
$C_{E_1} = 4.2455$ $kmol/m^3$	$C_{E_2} = 0.3254$ $kmol/m^3$
$C_{B_1} = 5.3532$ $kmol/m^3$	$C_{B_2} = 1.3841$ $kmol/m^3$
$C_{EB_1} = 0.1854$ $kmol/m^3$	$C_{EB_2} = 3.8744$ $kmol/m^3$
$C_{DEB_1} = 9.1426 \times 10^{-7}$ $kmol/m^3$	$C_{DEB_2} = 0.0058$ $kmol/m^3$
Heat transfer option	*Dynamics*
Medium temperature	298K
Temperature approach	77.33K
Heat capacity of coolant	4200J/kgK
Medium holdup	1000kg
$C_p = 2.411 kJ/kgK$	$\rho_1 = 639.1530 kg/m^3$
$V_1 = V_2 = 60 m^3$	$\rho_2 = 607.5040 kg/m^3$

while the temperatures of the first and second reactors at $t = t_o$ are set to $T_{1_0} = 400K$ and $T_{2_0} = 450K$, respectively. This choice of initial temperature, is made to keep both the reactants and products as liquids during the operation (i.e., simulation of the process). These values will be automatically adjusted by running Aspen Plus built-in steady-state simulations. After configuring the reactions in the two reactors, the steady-state simulation is carried on in order to analyze the behavior of the process. Moreover, the reactor geometry as well as thermodynamic parameters must be specified before exporting the steady-state model to Aspen Plus Dynamics. In this study, the vessels in the Aspen model are all vertical, the heads are flat, and each CSTR is ten meters long. Table 1 contains a list of the thermodynamic parameters that were used in the Aspen model. Finally, after running a pressure check, the steady-state model is exported to Aspen Plus Dynamics to generate and initiate the dynamic model. Moreover, to prepare the model for both open-loop and closed-loop simulation the flow rates F_1 and F_2 are fixed, and the two reactors' heating modes are switched to constant duty to enable external control over the manipulated variables (i.e., Q_1 and Q_2). By following the procedure outlined above, the dynamical model of the considered process is completed.

7.2. *Data generation and model training*

Data are required for the development of data-driven models and, generally given that the data are independent and identically distributed, the larger the data set size, the more accurate the model can be (Wu et al., 2022). However, excessive use of corrupted data (e.g., noisy, repeated, etc.) in the training data-set may lead to a less accurate RNN model, especially if the training is not carried out with potential pitfalls in mind. Therefore, this trade-off between data generation/collection and model accuracy should always be taken into account. In addition, large data sets are available from several sources, including industries, pilot plants, and computer-based simulations. However, in general, industrial data is not publicly accessible, and collecting data from pilot plants and laboratory experiments are expensive and time intensive. Therefore, in this work, we build our data set via extensive open-loop simulations utilizing an alternative approach.

The constructed dynamical model in Aspen Plus Dynamics is used in open-loop simulations with the input signals randomly generated via MATLAB and then implemented in a sample-and-hold fashion, such that the input remains fixed throughout each sampling time Δ. A local message passing interface (MPI) is created to connect Aspen Plus Dynamics and MATLAB, so that the Aspen dynamical model can automatically read the input signals from MATLAB. The MATLAB code, in particular, generates the manipulated variables in deviation form in relation to their steady-state values (i.e., $u_1 = Q_1 - Q_{1s}$ and $u_2 = Q_2 - Q_{2s}$). The two manipulated variables randomly vary within the lower bounds $[u_1^{min}, u_2^{min}] = [-1 \times 10^4 kW, -1.5 \times 10^4 kW]$ and the upper bounds $[u_1^{max}, u_2^{max}] = [1 \times 10^3 kW, 5 \times 10^3 kW]$. Both inputs are employed to the dynamic simulation in which the values are updated every five minutes (i.e., the sampling time). According to Aspen's manual, the default numerical integration method of the Aspen process dynamic model is the Implicit Euler scheme with an adaptive integration time step. In this study, we used a sampling time $\Delta = 5 min$ i.e., the integration scheme records the process state values every 5 min in process operation time. Note that the numerical integration time step while adaptive is always several orders of magnitude smaller that the sampling time. Other integration techniques in Aspen that are available for users to choose from include the 4th-order Runge-Kutta method, the explicit Euler scheme, and Gear methods.

We employ Aspen dynamic simulations to generate data sets for neural network training, since the Aspen dynamic model can be regarded as a high-fidelity process model for various complex chemical processes such as a system of CSTRs. To simulate typical sensor variability in chemical plants, industrial noise is incorporated into the state measurements. The normalized data noise obtained from Aspen public domain is shown in Fig. 5. The probability distribution of the normalized industrial noise in Fig. 5 is shown in Fig. 6, from which we confirm that the industrial noise has a non-

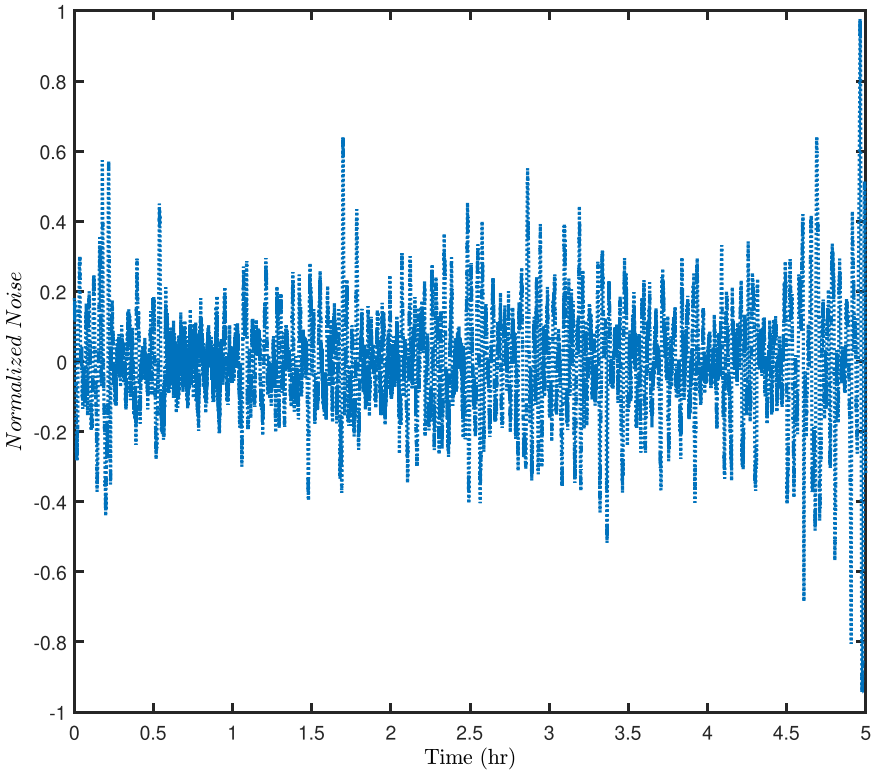

Fig. 5 – Normalized industrial noise from Aspen public domain data.

Gaussian distribution. With the random inputs generated from Matlab as discussed in the previous paragraph, and the normalized noise are amplified by six times and then added to the reactors temperature measurements. To create the training/validating data sets, all input values and output states (such as T, C_A, and C_B) are recorded as sequential time series data. For the Gaussian noise case, we generated normalized white noise with zero mean and standard deviation of 0.1, which was amplified six times and added to the temperature measurements. The dataset generated via this procedure for the Gaussian and non-Gaussian cases are denoted as $S_{noisy(G)}(x, u)$ and $S_{noisy(NG)}(x, u)$, respectively.

For the co-teaching method, noise-free data is required for the ML-based model development. This noise-free data set is non-trivial to collect in practice (i.e., not readily available from the laboratory/plant). First-principles (FP) models with simplifying assumptions are developed as candidates for generating such clean data for training/validating

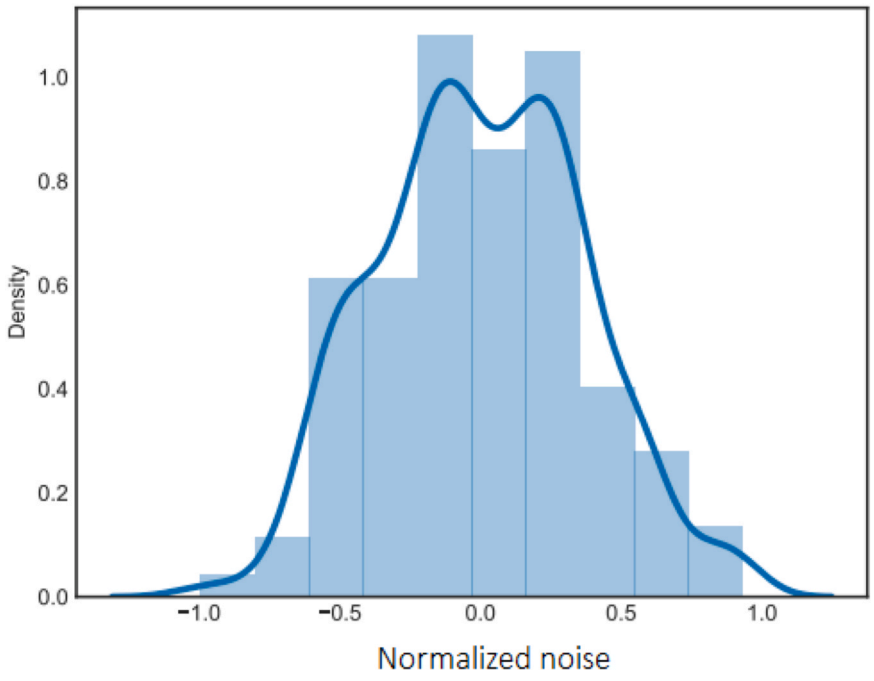

Fig. 6 – Probability density plot of normalized industrial noise introduced to the Aspen model.

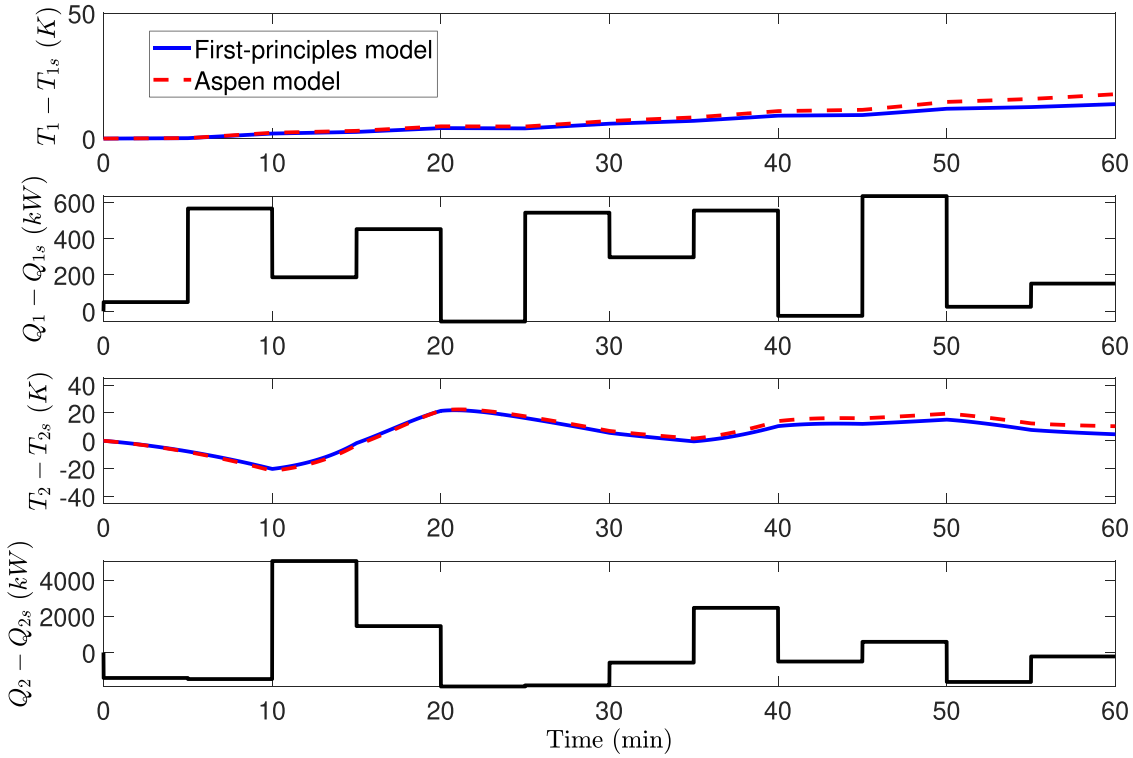

Fig. 7 – Open-loop state and manipulated inputs profiles for the process (noise-free).

Table 2 – Input and output states of the RNN-LSTM models.

Notation	State (in deviation form)	
x_1	Concentration of Ethane	1st CSTR
x_2	Concentration of Benzene	
x_3	Concentration of Ethylbenzene	
x_4	Concentration of Diethylbenzene	
x_5	Reactor's Temperature	
u_1	Heating/cooling duty	
x_6	Concentration of Ethane	2nd CSTR
x_7	Concentration of Benzene	
x_8	Concentration of Ethylbenzene	
x_9	Concentration of Diethylbenzene	
x_{10}	Reactor's Temperature	
u_2	Heating/cooling duty	

processes to solve this challenge. Therefore, in this work, we numerically solved the FP model developed in the previous section to generate noise-free data $S_{FP}(x, u)$ to train the PCRNN-LSTM model using the co-teaching method. The noise-free open-loop trajectories of the Aspen dynamical model and the first-principles model denoted by FP under time-varying inputs are shown in Fig. 7. Although the first-principles model may not fully capture the dynamics of the Aspen model under the different operating conditions, we will show that the co-teaching method using noisy data from the Aspen model and noise-free data from solving the first-principles model can still improve the LSTM model's prediction accuracy.

Table 3 – Open-loop prediction MSE results under Gaussian and non-Gaussian noise.

Dropout rate	Non-Gaussian	Gaussian
0.2	7.487×10^{-6}	2.02×10^{-6}
0.3	8.66×10^{-6}	2.87×10^{-6}
0.4	7.428×10^{-6}	5.08×10^{-6}
0.5	1.144×10^{-5}	6.75×10^{-6}

The generated data sets are used in the development of three different RNN-LSTM models, as illustrated in Fig. 8. Both the standard RNN-LSTM model and the dropout RNN-LSTM model use either $S_{noisy(G)}$ or $S_{noisy(NG)}$ according to the noise type, yet we use the dropout rate while training the dropout RNN model. Basically, we perform a grid search for the dropout rate in the range 0.2–0.5 that achieves the lowest training/validation loss (i.e., mean squared error (MSE)), and the results are summarized in Table 3. From the table, for the case of non-Gaussian noise and Gaussian noise, dropout rates of 0.2 and 0.4 are selected, respectively. As for the co-teaching model, it was trained typically as the standard model. Using

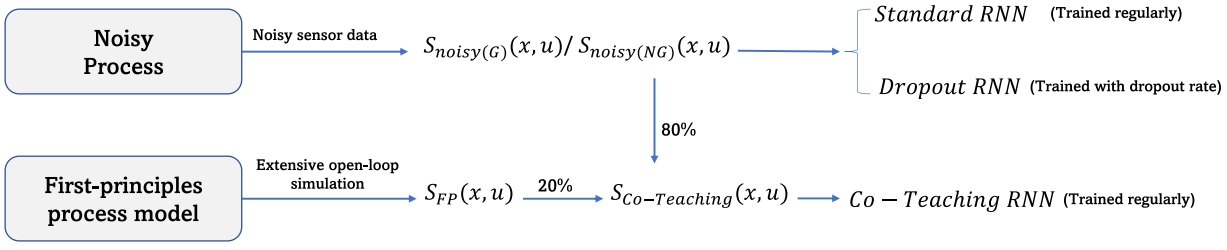

Fig. 8 – Partially-connected RNN model development methods.

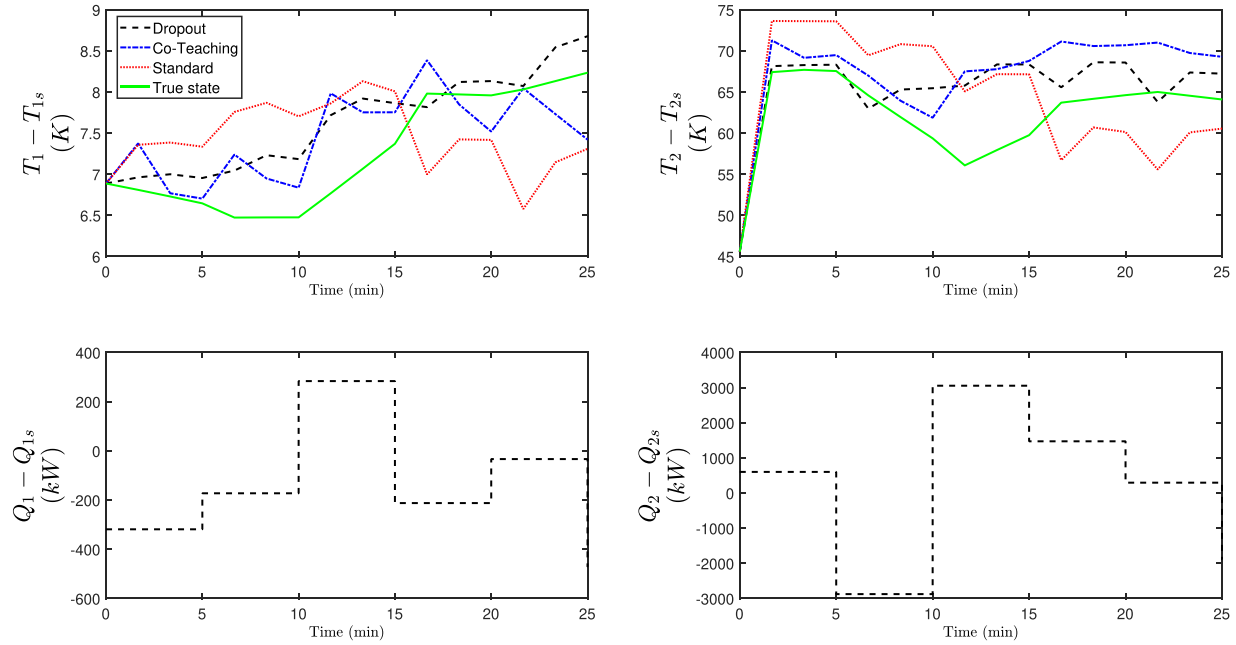

Fig. 9 – Open-loop state trajectory predicted by dropout LSTM, co-teaching LSTM, and standard LSTM, respectively, under the same time varying inputs in the presence of non-Gaussian noise.

the Keras library, the three RNN-LSTM models are constructed following the strategy discussed in section 3.1. The models are designed as follows: each neural network has two LSTM layers with thirty neurons in each, where we select the hyperbolic tangent functions (i.e., $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$) as the activation function. In addition, the output layer is to be activated by a linear activation function, and this layer will provide the estimated ten states, while the input layer will receive twelve inputs. The inputs to RNN-LSTM models are the states and the manipulated variables at t_k, where the model outputs are the states at $t = t_k + \Delta$, and all are defined in Table 2. Specifically, we predict the evolution of the states for the next five minutes (the equivalent of one sampling time) using input data from the previous five-minute sampling period. Rather than the traditional gradient descent optimization algorithm, we employ the Adam optimizer, which is a hybrid of two algorithms: gradient descent with momentum and RMSprop. Furthermore, to generate more robust models, we perform a five-fold cross-validation on the RNN-LSTM models and choose the models with the lowest validation MSE.

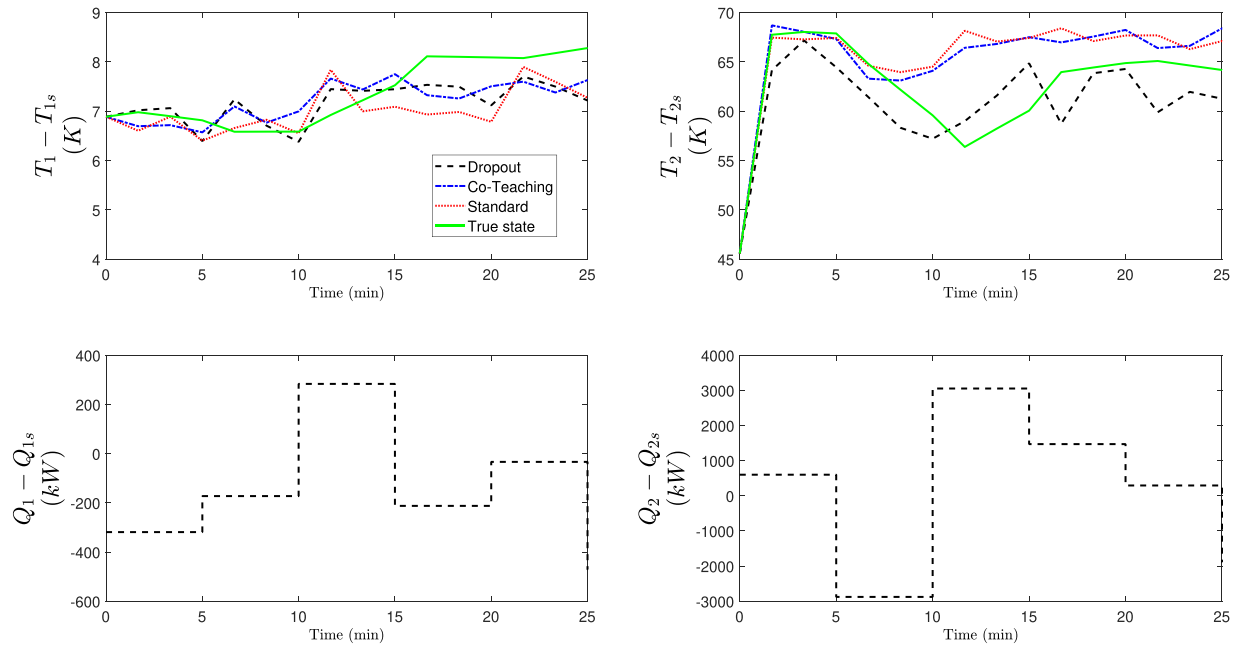

Fig. 10 – Open-loop state trajectory predicted by dropout LSTM, co-teaching LSTM, and standard LSTM, respectively, under the same time varying inputs in the presence of Gaussian noise.

Table 4 – Open-loop prediction results (MSE) by the three different models under non-Gaussian and Gaussian industrial noise.

Model	Non-Gaussian		Gaussian	
	x_5	x_{10}	x_5	x_{10}
Dropout	0.2445	24.197	0.27385	10.5538
Co-teaching	0.28458	35.56	0.265	19.255
Standard	0.9088	47.564	0.4485	21.968

After the development of the three RNN-LSTM models, we run two open-loop simulations considering the existence of Gaussian and non-Gaussian noise independently. Figs. 9 and 10 illustrate the open-loop prediction of the three models, with the process output (i.e., the Aspen dynamical model output) denoted as the ``True state'', in response to time varying inputs generated randomly from MATLAB, and starting from the exact same initial conditions. Both figures demonstrate the improvement in the prediction accuracy of the RNN-LSTM models utilizing the two proposed methods; the dropout and the co-teaching methods. The open-loop prediction MSEs of the two scenarios are listed in Table 4, with noticeable improvements in the approximation of the process outputs, since both the dropout and the co-teaching RNN-LSTM provided relatively lower MSE values compared to the standard RNN-LSTM model.

7.3. *Closed-loop simulation: Gaussian noise*

Following the open-loop simulations, we performed closed-loop simulations in the case of existence of Gaussian noise in the measurement of the two reactor temperatures under an

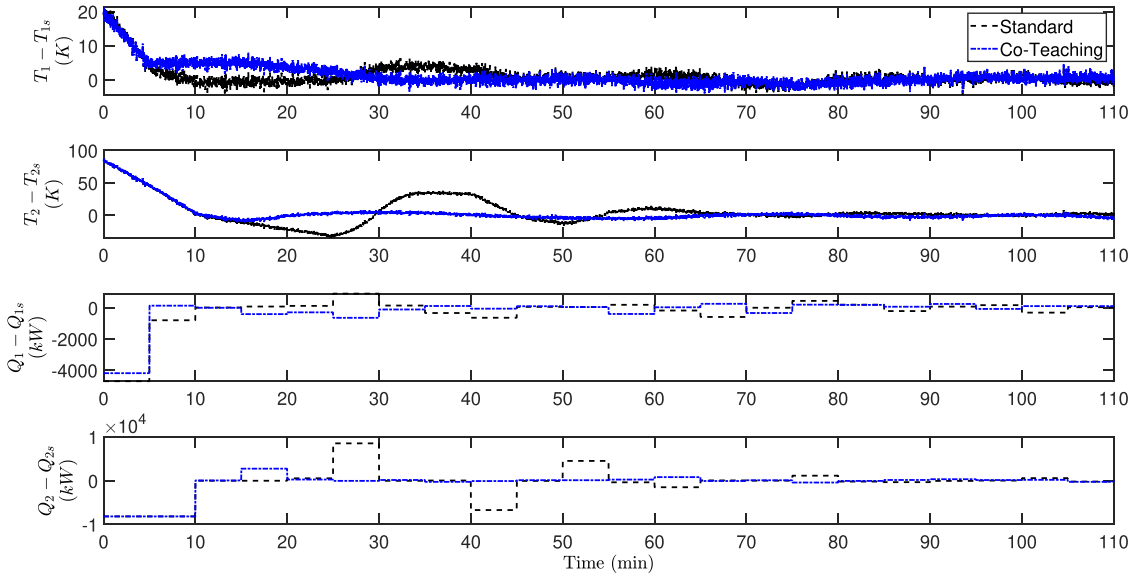

Fig. 11 – State and input profiles of the closed-loop simulation in the presence of Gaussian noise under the LMPC using PCRNN-LSTM models developed by: standard method (dashed line) and co-teaching method (dash-dotted line).

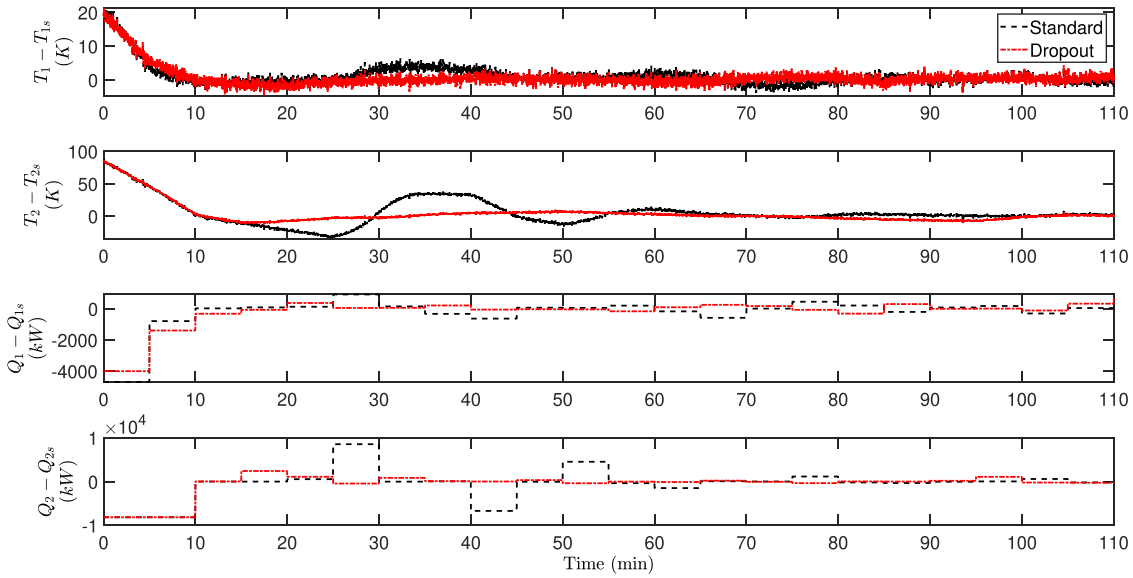

Fig. 12 – State and input profiles of the closed-loop simulation in the presence of Gaussian noise under the LMPC using PCRNN-LSTM models developed by: standard method (dashed line) and MC dropout method (dash-dotted line).

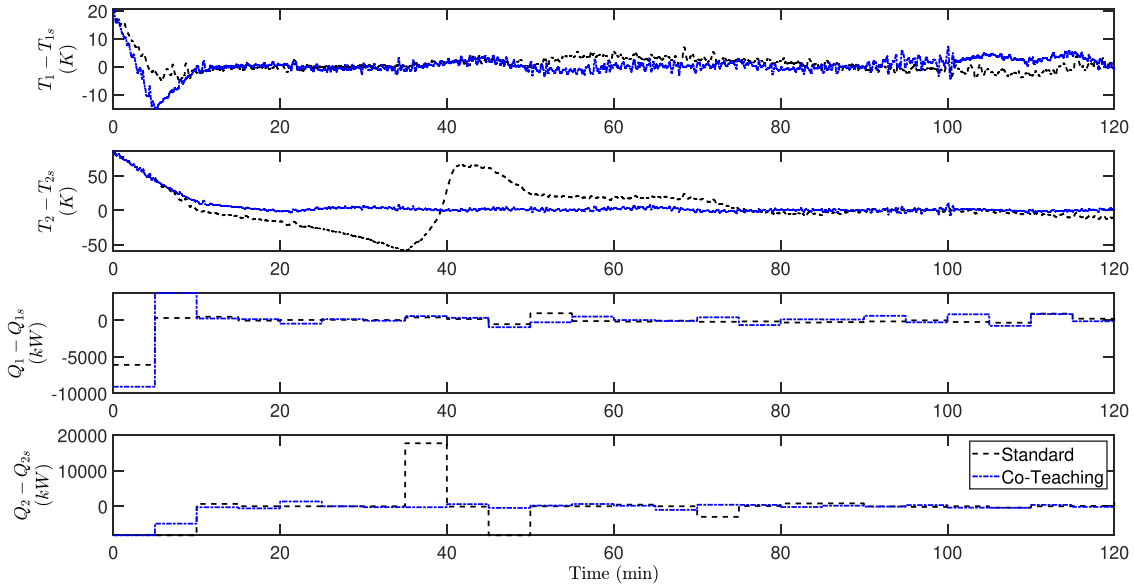

Fig. 13 – State and input profiles of the closed-loop simulation in the presence of non-Gaussian noise under the LMPC using PCRNN-LSTM models developed by: standard method (dashed line) and co-teaching method (dash-dotted line).

LMPC controller utilizing each of the three different PCRNN-LSTM models separately. The process dynamics corresponding to the PCRNN-LSTM model developed by the co-teaching method versus the standard PCRNN-LSTM model is presented in Fig. 11. Both LMPCs were able to derive the process to the desired steady-state and to stabilize the system around the origin even in the presence of Gaussian industrial noise. However, the improvement of the closed-loop performance when utilizing the co-teaching method is noticeable throughout the figure, as the temperatures trajectories are smoother and show less oscillation in comparison to the standard model.

Concurrently, a closed-loop simulation is performed using the dropout trained PCRNN-LSTM model. The results are plotted in Fig. 12. From the state trajectories and control actions in Fig. 12, similarly to the co-teaching strategy, the dropout method has observably enhanced the performance of the LMPC.

7.4. *Closed-loop simulation: non-Gaussian noise*

In this subsection, we discuss the results of closed-loop simulation in the existence of non-Gaussian type of noise. Fig. 13 shows the closed-loop inputs and state trajectories under the LMPC when using the PCRNN-LSTM model trained according to the standard strategy and the co-teaching strategy. It is notable that the standard PCRNN-LSTM model shows significant variation when the closed-loop state reaches the steady-state. This stems from the fact that the states predicted using the standard LSTM model are not sufficiently close to the true state values; hence, the LMPC is deceived to provide a solution that drives the process states in the wrong direction. In the same figure, the co-teaching based PCRNN-LSTM demonstrated enhanced model accuracy, and that the LMPC using the co-teaching method was successfully able to derive the state to the steady-state. As for the stability, the co-teaching method enabled the LMPC to

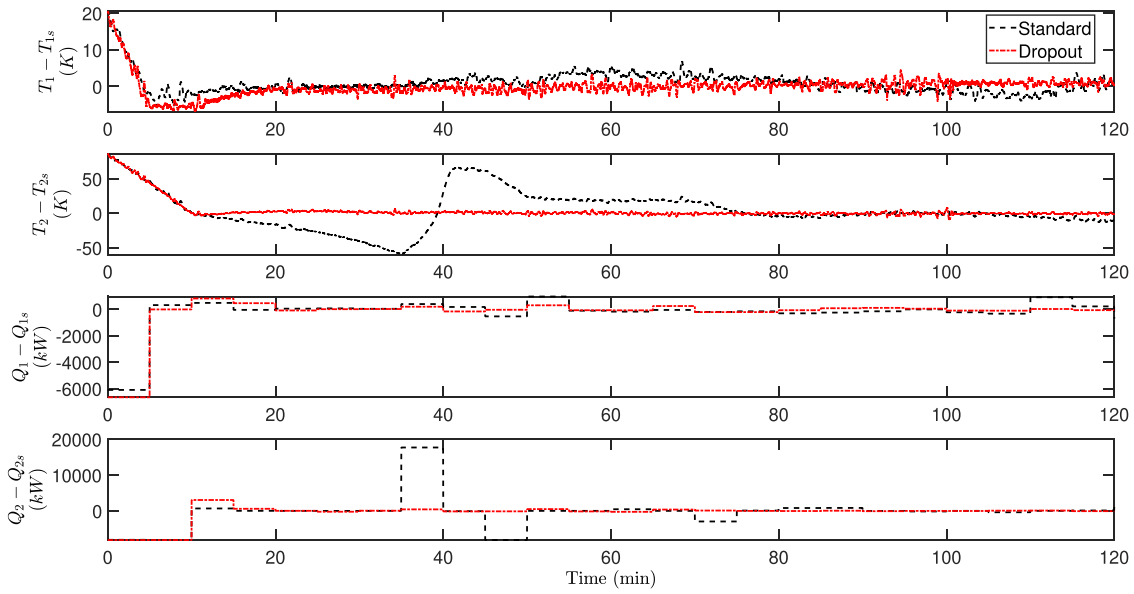

Fig. 14 – State and input profiles of the closed-loop simulation in the presence of non-Gaussian noise under the LMPC using PCRNN-LSTM models developed by: standard method (dashed line) and MC dropout method (dash-dotted line).

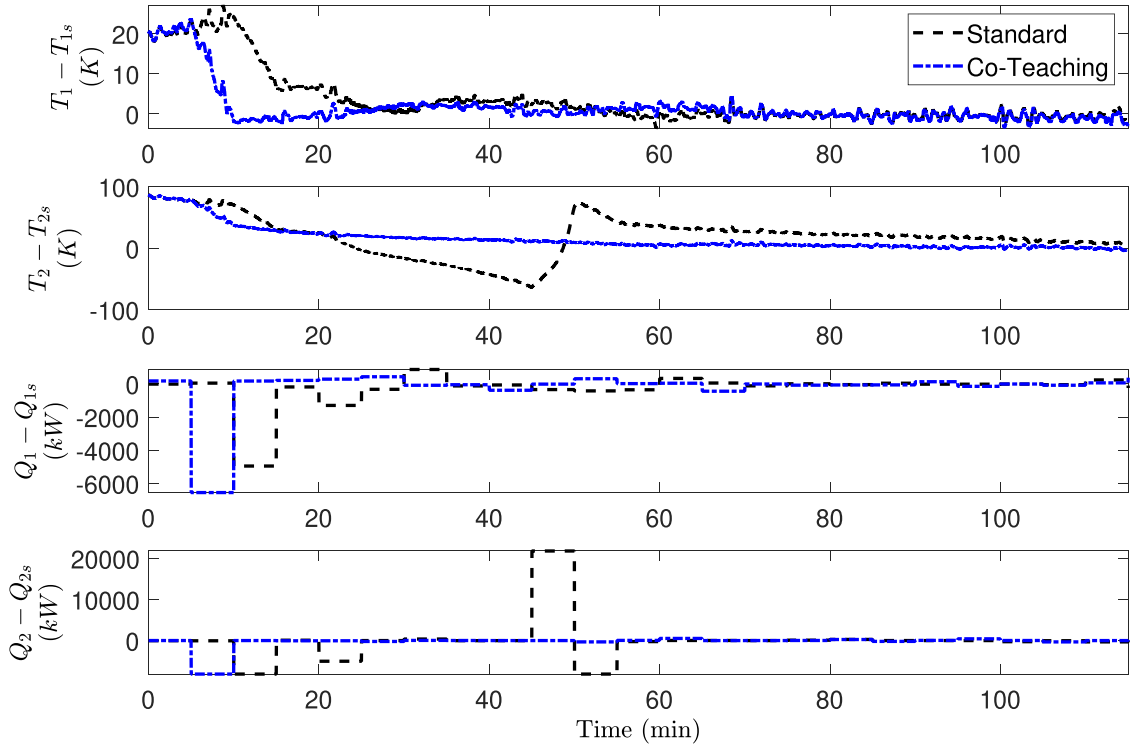

Fig. 15 – State and input profiles of the closed-loop simulation in the presence of non-Gaussian noise under the LMPC using FCRNN-LSTM models developed by: standard method (dashed line) and co-teaching method (dash-dotted line).

maintain the state in a small neighborhood around the steady-state more smoothly in comparison with the PCRNN-LSTM model built following the standard training algorithm. Note that the MPC will not be able to stabilize the system exactly at the steady state due to the sample-and-hold implementation of control actions and the model mismatch between the LSTM model and the actual nonlinear process. Therefore, if the state trajectory of the closed-loop system starting from the stability region Ω_ρ remains bounded in Ω_ρ and converges to a small compact set around the origin where it will be maintained thereafter, then the system is considered practically stable under the sample-and-hold implementation of MPC.

Similarly, the dropout strategy in training the PCRNN-LSTM model, referred to as the "dropout model", is used as the predictive model for LMPC. The closed-loop simulation

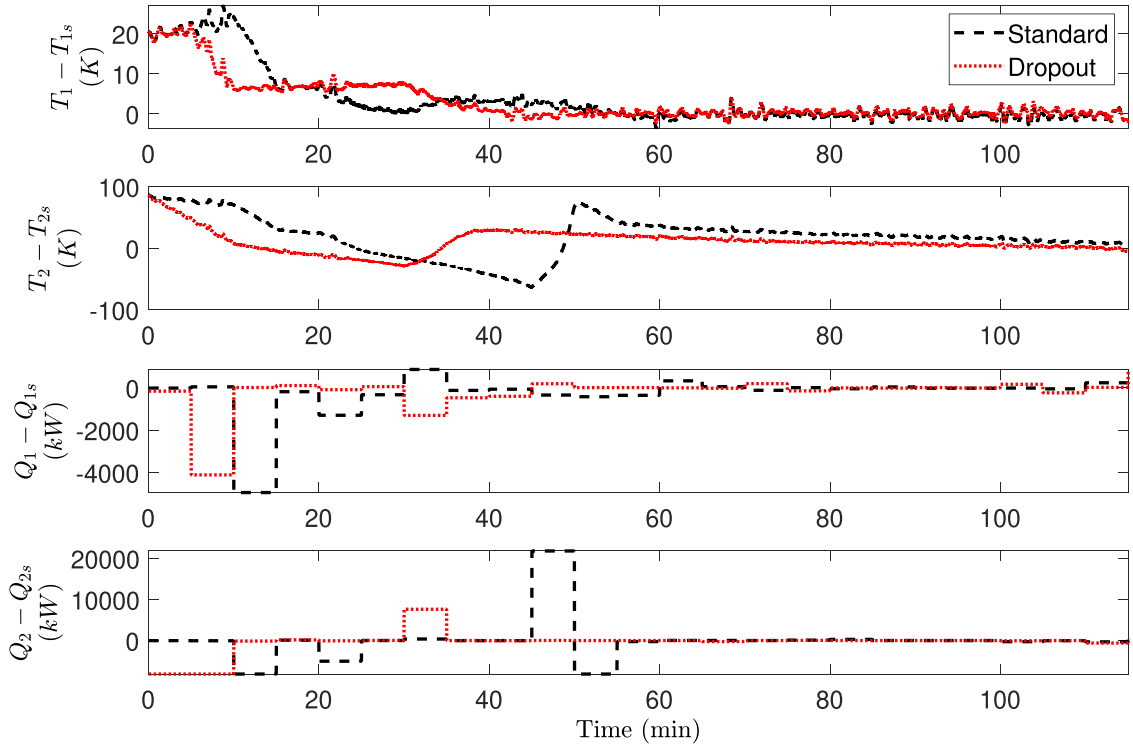

Fig. 16 – State and input profiles of the closed-loop simulation in the presence of non-Gaussian noise under the LMPC using FCRNN-LSTM models developed by: standard method (dashed line) and MC dropout method (dotted line).

Table 5 – The value of the time integral of the cost function using different models under two modeling architectures.

Model	Partially-connected	Fully-connected
Standard RNN-LSTM	1.01×10^9	1.6×10^9
Co-Teaching RNN-LSTM	6.01×10^8	6.8×10^8
Dropout RNN-LSTM	5.54×10^8	5.77×10^8

results of using the dropout model for the LMPC prediction is illustrated in Fig. 14. The figure demonstrates the superiority of the dropout model over the standard PCRNN-LSTM model in both metrics: (i) smoothness of the trajectories, and (ii) the ability to stabilize the underlying process within the stability region Ω_ρ as discussed previously.

In the same vein, we carried out a closed-loop simulation in the presence of non-Gaussian noise using fully-connected RNN-LSTM models (FCRNN-LSTM) trained according to the three strategies considered in this work. Figs. 15 and 16 show the process state under the FCRNN-LSTM-based LMPC. Specifically, Fig. 15 depicts the states and inputs trajectories by models trained by co-teaching and standard strategies. Once more, the co-teaching improved the LMPC performance more observably in the $T_2 - T_{2s}$ trajectory. Similarly, in Fig. 16, the trained FCRNN-LSTM model based on the dropout strategy provided a more favorable closed-loop dynamics in relation to the standard FCRNN-LSTM. The three FCRNN-LSTM model based LMPCs were also able to derive the system to steady-state values, but took longer time and experienced more oscillatory behaviour than the PCRNN-LSTM models.

In both fully-connected and partially-connected architectures, using co-teaching and dropout approaches lead to changes in the value of the integration of the cost function over the complete simulation duration against the standard method. In the partially-connected scenario, the cost function values for co-teaching and dropout approaches fell by 40.6 % and 45.05 %, respectively, when compared to the standard method, as shown in Table 5. Moreover, by using the partially-connected modeling over the fully-connected modeling the value of the integrated cost function associated with the controllers using the standard, co-teaching, and dropout based models are reduced by 37.5 %, 12 %, and 4 %, respectively.

8. Conclusion

In this paper, we used the Monte Carlo dropout and the co-teaching strategies to train PCRNN-LSTM models for predicting underlying process dynamics (ground truth) from noisy data. The Ethylbenzene production process by two CSTRs in series is considered, and modeled via the Aspen Plus dynamics simulator. The co-teaching and dropout strategies were applied to create PCRNN-LSTM models with two type of noise independently (i.e., Gaussian and non-Gaussian), where noisy and noise-free data were generated by extensive open-loop simulation of Aspen dynamical model and first-principles model, respectively. Subsequently, open-loop as well as closed-loop simulations were performed to illustrate the superiority of PCRNN-LSTM based models trained via co-teaching and dropout techniques over the standard PCRNN-LSTM modeling technique in terms of enhancing the accuracy of open-loop prediction as well as improving the closed-loop performance. In comparison to the standard approach, both co-teaching and dropout techniques obtained lower values of the cost function time integral in the two modeling methodologies (i.e., partially-connected and fully-connected), indicating faster convergence and lower energy.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The financial support from the National Science Foundation and the Department of Energy is appreciated. Mohammed Alhajeri acknowledges the support from Kuwait University through the KU-scholarship program.

References

Abdullah, F., Wu, Z., Christofides, P.D., 2022. Handling noisy data in sparse model identification using subsampling and co-teaching. Comput. Chem. Eng. 157, 107628.
Agarwal, A., Liu, Y., McDowell, C., 2019. 110th anniversary: ensemble-based machine learning for industrial fermenter classification and foaming control. Ind. Eng. Chem. Res. 58, 16719–16729.
Alhajeri, M., Luo, J., Wu, Z., Albalawi, F., Christofides, P.D., 2022. Process structure-based recurrent neural network modeling for predictive control: a comparative study. Chem. Eng. Res. Des. 179, 77–89.
Alhajeri, M.S., Wu, Z., Rincon, D., Albalawi, F., Christofides, P.D., 2021. Machine-learning-based state estimation and predictive control of nonlinear processes. Chem. Eng. Res. Des. 167, 268–280.
Chen, S., Wu, Z., Rincon, D., Christofides, P.D., 2020. Machine learning-based distributed model predictive control of nonlinear processes. AIChE J. 66, e17013.
Drgoňa, J., Picard, D., Kvasnica, M., Helsen, L., 2018. Approximate model predictive building control via machine learning. Appl. Energy 218, 199–216.
Gal, Y., Ghahramani, Z., 2016a. Dropout as a bayesian approximation: representing model uncertainty in deep learning. Proc. 33rd Int. Conf. Mach. Learn. 1050–1059.
Gal, Y., Ghahramani, Z., 2016b. A theoretically grounded application of dropout in recurrent neural networks. Adv. Neural Inf. Process. Syst. 29.
Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., Sugiyama, M., 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. Adv. Neural Inf. Process. Syst. 31.
Hsu, Y., Wang, J., 2009. A Wiener-type recurrent neural network and its control strategy for nonlinear dynamic applications. J. Process Control 19, 942–953.
Kahrs, O., Marquardt, W., 2007. The validity domain of hybrid models and its application in process optimization. Chem. Eng. Process.: Process.Intensif. 46, 1054–1066.
Krishnaiah, J., Kumar, C., Faruqi, M., 2006. Modelling and control of chaotic processes through their Bifurcation Diagrams generated with the help of Recurrent Neural Network models: Part 1-simulation studies. J. Process Control 16, 53–66.
Kwon, Y., Lee, D., Choi, Y., Kang, S., 2022. Uncertainty-aware prediction of chemical reaction yields with graph neural networks. J. Chemin.-. 14, 1–10.
Lima, F.V., Rawlings, J.B., 2011. Nonlinear stochastic modeling to improve state estimation in process monitoring and control. AIChE J. 57, 996–1007.
Lin, Y., Sontag, E.D., 1991. A universal formula for stabilization with bounded controls. Syst. Control Lett. 16, 393–397.

Ma, Y., Noreña-Caro, D.A., Adams, A.J., Brentzel, T.B., Romagnoli, J.A., Benton, M.G., 2020. Machine-learning-based simulation and fed-batch control of cyanobacterial-phycocyanin production in Plectonema by artificial neural network and deep reinforcement learning. Comput. Chem. Eng. 142, 107016.
Patwardhan, S.C., Narasimhan, S., Jagadeesan, P., Gopaluni, B., Shah, S.L., 2012. Nonlinear Bayesian state estimation: a review of recent developments. Control Eng. Pract. 20, 933–953.
Shah, D., Wang, J., He, Q.P., 2020. Feature engineering in big data analytics for IoT-enabled smart manufacturing-comparison between deep learning and statistical learning. Comput. Chem. Eng. 141, 106970.
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1929–1958.
Stephanopoulos, G., Han, C., 1996. Intelligent systems in process engineering: a review. Comput. Chem. Eng. 20, 743–791.
Thompson, M.L., Kramer, M.A., 1994. Modeling chemical processes using prior knowledge and neural networks. AIChE J. 40, 1328–1340.
Wu, H., Zhao, J., 2018. Deep convolutional neural network model based chemical process fault diagnosis. Comput. Chem. Eng. 115, 185–197.
Wu, Z., Alnajdi, A., Gu, Q., Christofides, P.D., 2022. Statistical machine-learning-based predictive control of uncertain nonlinear processes. AIChE J. 68, e17642.
Wu, Z., Rincon, D., Christofides, P.D., 2020. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. J. Process Control 89, 74–84.
Wu, Z., Rincon, D., Luo, J., Christofides, P.D., 2021. Machine learning modeling and predictive control of nonlinear processes using noisy data. AIChE J. 67, e17164.
Wu, Z., Tran, A., Rincon, D., Christofides, P.D., 2019. Machine learning-based predictive control of nonlinear processes. Part I: theory. AIChE J. 65, e16729.
Zhang, Z., Wu, Z., Rincon, D., Christofides, P.D., 2019. Real-time optimization and control of nonlinear processes using machine learning. Mathematics 7, 890.