Machine Learning Modeling and Predictive Control of Nonlinear Processes Using Noisy Data

Zhe Wu, David Rincon, Junwei Luo, Panagiotis D. Christofides*

August 19, 2020

Abstract

This work focuses on machine learning modeling and predictive control of nonlinear processes using noisy data. We use long short-term memory (LSTM) networks with training data from sensor measurements corrupted by two types of noise: Gaussian and non-Gaussian noise, to train the process model that will be used in a model predictive controller (MPC). We first discuss the LSTM training with noisy data following a Gaussian distribution, and demonstrate that the standard LSTM network is capable of capturing the underlying process dynamic behavior by reducing the impact of noise. Subsequently, given that the standard LSTM performs poorly on a noisy dataset from industrial operation (i.e., non-Gaussian noisy data), we propose an LSTM network using Monte Carlo dropout method to reduce the overfitting to noisy data. Furthermore, an LSTM network using co-teaching training method is proposed to further improve its approximation performance when noise-free data from a process model capturing the nominal process state evolution is available. A chemical process example is used throughout the manuscript to illustrate the application of the proposed modeling approaches and demonstrate

*Zhe Wu, David Rincon, and Junwei Luo are with the Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095, USA. Panagiotis D. Christofides is with the Department of Chemical and Biomolecular Engineering and the Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095, USA. Emails: wuzhe@g.ucla.edu, and pdc@seas.ucla.edu. Corresponding author: P. D. Christofides. Financial support from the National Science Foundation and the Department of Energy is gratefully acknowledged.

their open- and closed-loop performance under a Lyapunov-based model predictive controller with state measurements corrupted by industrial noise.

Keywords: Machine learning; Long short-term memory; Noisy data; Nonlinear systems; Model predictive control

1 Introduction

Data-driven modeling has historically received significant attention in the context of model predictive control (MPC), e.g.,.[2,4,6,11,17,20,26,34,39,44] Among many data-driven modeling approaches, machine learning modeling has proven to be successful in approximating nonlinear dynamical systems in[3,21,31,35] due to its fast capability of getting ready-to-use models capturing nonlinear behavior when first-principles models are unavailable. In order to develop a machine learning model with a desired prediction accuracy for MPC, a high-quality dataset that can be generated from industrial process sensors, lab experiments, or extensive computer simulations is required, from which supervised machine learning models can learn the nonlinear relationship between network inputs and outputs. However, real industrial measurements often involve noise stemming from different sources, such as sensors variability and common plant variance, which is a critical point in machine learning modeling that has not been addressed yet in the context of machine-learning-based MPC.

Many methodologies have been proposed for dealing with noisy measurements in the literature. For instance, the Kalman filter is one of the most well-known methodologies for state estimation of linear systems under Gaussian white noise ([24,42]). Similarly, state estimation techniques have been proposed for denoising industrial measurements for nonlinear systems, such as the extended Kalman filter, unscented Kalman filter, and moving horizon estimation ([24]). In general, these methodologies are built under assumptions on the type of noise and system structure ([24]). As a result, a priori process knowledge such as first-principles models are often needed, which restricts the use of these methods in real industrial systems ([12,42]). On the other hand, the Savitzky-Golay filter is one of the most popular methods for smoothing noisy data ([27]) without any knowledge of process model. This method fits a moving low-order polynomial on adjacent data points by tuning the polynomial order and the size of the data window. However, tuning the parameters could be time demanding and the computational cost is proportional to the window width, which limits its application in real-time control problems. Therefore, how to develop a data-driven process model that can directly predict process dynamic behavior from noisy measurement data remains a challenge.

In recent years, machine learning has attracted an increased level of attention in model identification.[5,29,33] Recurrent neural networks (RNN) have been one of the widely used modeling approaches due to their ability of representing temporal dynamic behavior through feedback loops in neurons, and have been successfully incorporated in model predictive control (MPC).[37,38] When the data training set is not very informative, a model based on principal component analysis (PCA) and RNN has been proposed and tested with MPC in which an alternative closed-loop model re-identification step is proposed ([14]). Long short-term memory network (LSTM) is a type of RNN model that has been used to maintain information in memory for long periods of time due to its special units in addition to standard units. LSTM has also been used as prediction model in MPC in,[7] and has been implemented with economic MPC in its encoder-decoder LSTM structure.[8] Additionally, machine learning techniques have also been improved by incorporating feature engineering ideas to efficiently develop models for large-scale systems. For example, in,[28] it was demonstrated that both machine learning and projection to latent structure (PLS) methods show poor performance on raw vibration signals from a laboratory-scale water flow system, and therefore, further treatment of raw datasets, such as feature-based monitoring that can significantly improve model prediction[15,16,28] is needed. An implementation of similar machine learning structures can be found for nonlinear systems in,[22,25,43] and their potential role in Industry 4.0 was also highlighted in (bio)chemical processes ([1]). While machine learning techniques have shown great potential in modeling big datasets, machine learning modeling using noisy data has been one of the key issues hampering their implementation to chemical plant data since most machine learning applications are still limited to deterministic cases (i.e., noise-free datasets) in the literature ([42]).

Many in-silico studies adopt Gaussian noise for testing the robustness of the proposed machine learning methods; however, more realistic conditions such as non-Gaussian noise have not been studied. Unlike Gaussian noise that can be handled by many standard machine learning modeling approaches, non-Gaussian noise may lead to incorrect mapping from input to its (noise-free) ground-truth output due to its overfitting to the noisy pattern of the training dataset corrupted by non-stationary noise. A simple, effective technique to reduce overfitting to non-Gaussian noisy data without having any a priori process knowledge is to employ a dropout method in the neural network training process. Specifically, a dropout method randomly drops the connections between units in adjacent layers during training, and provides an efficient way that approximately combines many different neural network architectures together to improve the prediction performance ([18,30]). Recent works[9,10] have extended dropout to approximate Bayesian inference with theoretical results that provide insights into the use of dropout in RNN models. Moreover, when noise-free process data

generated using a priori process knowledge are available (for example, computer simulations based on first-principle models that approximate real processes), the co-teaching method may provide a potential solution to further improve model performance by using the noise-free dataset. The co-teaching method was originally proposed to solve classification problems with noisy data (i.e., mislabeled data) in the machine learning community.[13] Specifically, the co-teaching method with a symmetric or asymmetric mode depending on the noise level of the dataset, is able to learn the noise-free pattern through noisy data by using two models that share clean information between each other during each training step ([13]). However, to our knowledge, little attention has been paid to the extension of the co-teaching method in regression problems.

Motivated by the above, this work studies machine learning modeling of nonlinear processes using noisy data via novel neural networks techniques with dropout layer and co-teaching methods. Specifically, we first investigate the standard LSTM models' capability of modeling noise-free process dynamics using process data with Gaussian noise. Then, we present the Monte Carlo dropout technique and demonstrate the implementation of dropout LSTM in handling noisy industrial-data sets from ASPEN that follows a non-Gaussian distribution. Lastly, we discuss the co-teaching method in the context of regression problems and demonstrate its improved modeling performance by further accounting for noise-free data in the training process. The rest of this article is organized as follows: in Section 2, the notations, the class of nonlinear systems considered, the long short term memory network and the formulation of LSTM-based model predictive controller are given. In Section 3, we use a chemical process example to study the de-noising capability of LSTM networks using a noisy training dataset of Gaussian distribution, and the dropout LSTM modeling approach for handling non-Gaussian noise. In Section 4, we propose the co-teaching scheme using both industrial noisy data and first-principles solutions to improve LSTM training performance. Open-loop and closed-loop simulations under a Lyapunov-based MPC using the aforementioned LSTM models are carried out to compare their performance.

2 Preliminaries

2.1 Notation

The notation $|\cdot|$ is used to denote the Euclidean norm of a vector. x^T denotes the transpose of x. The notation $L_f V(x)$ denotes the standard Lie derivative $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$. Set subtraction is denoted by "\", i.e., $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$. The function $f(\cdot)$ is of class $\mathcal{C}^1$ if it is continuously differentiable in its domain.

2.2 Class of Systems

The class of continuous-time nonlinear systems considered is described by the following system of first-order nonlinear ordinary differential equations:

$$\begin{aligned} \dot{x} &= F(x,u,w) := f(x) + g(x)u,\ x(t_0) = x_0 \\ y &= x + w \end{aligned} \tag{1}$$

where $x \in \mathbf{R}^n$ is the state vector, $u \in \mathbf{R}^m$ is the manipulated input vector, $y \in \mathbf{R}^n$ is the vector of state measurements that are sampled continuously, and $w \in \mathbf{R}^n$ is the noise vector. The control actions are constrained by $u \in U := \{u_i^{\min} \leq u_i \leq u_i^{\max}, i = 1, ..., m\} \subset \mathbf{R}^m$. $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$ and $n \times m$, respectively. Throughout the manuscript, we assume that the initial time t_0 is zero ($t_0 = 0$), and $f(0) = 0$ such that the origin is a steady-state of the nominal (i.e., $w(t) \equiv 0$) system of Eq. 1 (i.e., $(x_s^*, u_s^*) = (0, 0)$, where x_s^* and u_s^* represent the steady-state state and input vectors, respectively).

2.3 Stabilization via Control Lyapunov Function

Consider the nominal system of Eq. 1 with noise-free state measurement available (i.e., $y(t) = x(t)$ with $w(t) \equiv 0$). To guarantee that the closed-loop system is stabilizable, a stabilizing control law $u = \Phi(x) \in U$ that renders the origin of the nominal system of Eq. 1 (i.e., $w(t) \equiv 0$) exponentially stable is assumed to exist. Following converse Lyapunov theorems ([19]), there exists a $\mathcal{C}^1$ Control Lyapunov function $V(x)$ such that the following inequalities hold for all x in an open neighborhood D around the origin:

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

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

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

where c_1, c_2, c_3 and c_4 are positive constants. $F(x,u)$ represents the nonlinear system of Eq. 1. The universal Sontag control law ([23]) is a candidate controller for $u = \Phi(x)$. A set of states $\phi_u \in \mathbf{R}^n$ is first characterized where Eq. 2 is satisfied under $u = \Phi(x)$. Then a level set of the Lyapunov function inside ϕ_u is used as the closed-loop stability region Ω_ρ for the nonlinear system of Eq. 1 as follows: $\Omega_\rho := \{x \in \phi_u \mid V(x) \leq \rho\}$, where $\rho > 0$ and $\Omega_\rho \subset \phi_u$.

2.4 Long Short-Term Memory Network

Recurrent neural networks (RNN) have been utilized in numerous applications to model nonlinear dynamical systems with time-series process data due to their ability to represent temporal behavior using the feedback loops in the hidden layer. Among many types of RNN models, long short-term memory (LSTM) networks have received increasing attention due to their ability to model long-term sequential dependencies with the use of three gates (the input gate, the forget gate, and the output gate) that help avoid vanishing gradients during training.

In this work, we develop an LSTM model with the following general form to approximate the nonlinear system of Eq. 1:

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u) := A\hat{x} + \Theta^T z \tag{3}$$

where $\hat{x} \in \mathbf{R}^n$ is the LSTM state vector, and $u \in \mathbf{R}^m$ is the manipulated input vector. $z = [z_1 \cdots z_{n+m+1}]^T = [H(\hat{x}_1) \cdots H(\hat{x}_n) \;\; u_1 \cdots u_m \;\; 1]^T \in \mathbf{R}^{n+m+1}$ is a vector of both the network states $\hat{x}$ and the inputs u, where $H(\cdot)$ represents nonlinear activation functions in each LSTM unit, and "1" represents the bias term. $A = diag\{-\alpha_1 \cdots -\alpha_n\} \in \mathbf{R}^{n\times n}$ is a diagonal coefficient matrix, and $\Theta = [\theta_1 \cdots \theta_{n+m+1}] \in \mathbf{R}^{(n+m+1)\times n}$ with $\theta_i = \beta_i[\omega_{i1} \cdots \omega_{i(n+m)} \;\; b_i]$, $i = 1, ..., n$. α_i and β_i are constants, and ω_{ik} is the weight connecting the k^{th} input to the i^{th} neuron where $i = 1, ..., n$ and $k = 1, ..., (n + m)$, and b_i is the bias term for $i = 1, ..., n$. α_i are assumed to be positive constants such that the state vector $\hat{x}$ is bounded-input bounded-state stable. From the general form of the continuous-time LSTM networks of Eq. 3, it is clear that the evolution of LSTM states over time can be obtained based on the current state and manipulated inputs. However, considering that the neural network training datasets typically consist of sampled-data that are discrete in time, the following equations are practically utilized by the LSTM network to calculate the predicted output sequence $\hat{x}(k)$ from the input sequence $m(k)$, $k = 1, ..., T$:

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

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

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

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

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

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

where $m(k)$ denotes the k^{th} element in the input sequence $m \in \mathbf{R}^{(n+m)\times T}$ that contains the measured states $x \in \mathbf{R}^n$ and the manipulated inputs $u \in \mathbf{R}^m$ with a sequence length of T, and $\hat{x} \in \mathbf{R}^{n\times T}$

denotes the LSTM network output sequence. T is the number of measured states of the sampled-data system of Eq. 1. $h(k)$, $c(k)$, $i(k)$, $f(k)$, and $o(k)$ are the internal state, the cell state, the outputs from the input gate, the forget gate, and the output gate, respectively. $tanh(\cdot)$ is the hyperbolic tangent activation function and $\sigma(\cdot)$ is the sigmoid activation function. ω_i^m and ω_i^h represent the weight matrices for the LSTM input vector m, and the hidden state vector in the input gate, respectively. Similarly, ω_c^m, ω_c^h, ω_f^m, ω_f^h, ω_o^m, ω_o^h represent the weight matrices for the input vector m and the hidden state vector h in calculating the cell state c, the forget gate f, and the output gate o, respectively, with b_i, b_f, b_o, b_c representing the corresponding bias terms. The predicted state vector $\hat{x}$ given by the LSTM network is the linear combination of the internal state h, where ω_y and b_y denote the weight matrix and bias vector for the output, respectively. The schematic of LSTM network structure corresponding to Eq.4 can be found in,[7] and is omitted here.

The training and validation datasets for developing the LSTM model are generated using extensive open-loop simulations of the nonlinear system of Eq. 1 with various initial conditions $x_0 \in \Omega_\rho$ and control actions $u \in U$. Specifically, the control actions are fed into the nonlinear system of Eq. 1 in a sample-and-hold fashion, i.e., $u(t) = u(t_k)$, $\forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$ and Δ is the sampling period. The continuous-time nonlinear system of Eq. 1 is integrated using explicit Euler method with a sufficiently small integration time step $h_c < \Delta$; in practice, time-series process data may be used. In each simulation run, we choose an initial condition and a set of control actions, and carry out the open-loop simulation for the system of Eq. 1 over a fixed length of time $(P+1)\cdot\Delta$, where P is an integer and $P \geq 1$. The states x are measured continuously at each integration time step, and therefore, a total of $(P+1)\cdot\frac{\Delta}{h_c}$ measured states are saved in each simulation run. As a result, the LSTM network will use the measured states within the first P sampling periods (i.e., $y(t)$, $\forall t \in [0,\ P\cdot\Delta)$) and the manipulated inputs $u(t)$, $\forall t \in [\Delta,\ (P+1)\cdot\Delta)$ to predict the states in the second P sampling period (i.e., $y(t)$, $\forall t \in [\Delta,\ (P+1)\cdot\Delta)$), from which the predicted state in the last sampling period, i.e., $y(t)$, $t \in [P\cdot\Delta,\ (P+1)\cdot\Delta)$, is obtained based on the previous state measurements. In this case, the number of measured states T in the LSTM network of Eq. 4 is set to $P\cdot\frac{\Delta}{h_c}$, and the input sequence m to the LSTM network is formulated as follows:

$$m = [x(0), x(h_c)..., x(P\cdot\Delta - h_c);\ u(\Delta), u(\Delta + h_c), ..., u((P+1)\cdot\Delta - h_c)] \tag{5}$$

The LSTM dataset is then partitioned into training, validation and testing datasets, and the training process is carried out using the neural-network library, Keras, with an optimal LSTM structure (e.g., number of layers, units, and weight initialization approaches) to achieve a desired model accuracy that satisfies the closed-loop stability requirements of MPC that will be introduced in the next subsection. The LSTM model is developed to predict one sampling period forward; however, by

applying the LSTM model in a rolling horizon manner, it can predict future states over a much longer period of time, and therefore, can be used as the prediction model in MPC.

Remark 1. It should be noted that the LSTM inputs and outputs in this work are different from those in,[7] which only uses the state measurement $x(t)$ at the current sampling time, e.g., $t = t_k$, to predict the states in the following sampling period, i.e., $x(t)$, $\forall t \in (t_k, t_{k+1}]$. Although the proposed LSTM structure in[7] works well when noise-free state measurement is available, i.e., $w \equiv 0$ in Eq. 1, the state prediction based on the current state measurement may show significant deviation from the true state in the presence of sensor noise. As a result, we introduce a window of length P in the LSTM input and output sequences in this work to reduce the dependence of LSTM network on the current state measurement by accounting for past (noisy) state measurements over P sampling periods to provide better predictions.

2.5 Model Predictive Control Using LSTM models

The Lyapunov-based model predictive control (LMPC) scheme using the LSTM model is given by the following optimization problem:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u(t))dt \tag{6a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{6b}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{6c}$$

$$\dot{\hat{V}}(x(t_k), u) \leq \dot{\hat{V}}(x(t_k), \Phi_{nn}(x(t_k))), \ \text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{6d}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+N}), \ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{6e}$$

where $\tilde{x}$ is the predicted state trajectory, N is the number of sampling periods in the prediction horizon,and $S(\Delta)$ is the set of piecewise constant functions with period Δ. $\Phi_{nn}(x)$ is the stabilizing control law that renders the origin of the LSTM system of Eq. 3 exponentially stable. $\Omega_{\hat{\rho}}$ is the stability region for the closed-loop LSTM system, and $\Omega_{\rho_{nn}}$ is the target region that the state is ultimately driven into. $\dot{\hat{V}}(x, u)$ represents the time-derivative of the Lyapunov function $\hat{V}$, i.e., $\frac{\partial \hat{V}(x)}{\partial x}(F_{nn}(x, u))$. The LMPC computes the optimal input sequence $u^*(t)$ over the prediction horizon $t \in [t_k, \, t_{k+N})$, and send the first control action, $u^*(t_k)$, to the system to be applied for the next sampling period. Then, the LMPC horizon is rolled one sampling step forward, and is resolved at the next sampling time.

The optimization problem of Eq. 6 is to minimize the time-integral of the function $L(\tilde{x}(t), u(t))$ that has its minimum value at the steady-state $(x_s^*, u_s^*) = (0, 0)$ over the prediction horizon subject

to the constraints of Eqs. 6b-6e. The constraint of Eq. 6b is the LSTM model of Eq. 3 that is used to predict the states of the closed-loop system. Eq. 6c defines the input constraints applied over the entire prediction horizon. The constraint of Eq. 6d drives the closed-loop state towards the origin if $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$, in which the time-derivative of V in the constraint of Eq. 6d is approximated using a forward finite difference method. The constraint of Eq. 6e maintains the state within $\Omega_{\rho_{nn}}$ for the remaining time after the state enters $\Omega_{\rho_{nn}}$. Additionally, since the LSTM model is developed using sampled-data collected every integration time step h_c in open-loop simulation, the state is measured with the same integration time step in MPC and fed into the LSTM model of Eq. 6b at each sampling time. When the noise-free state measurement is available, closed-loop stability is guaranteed for the nonlinear system of Eq. 1 under LMPC in the sense that the closed-loop state is bounded in $\Omega_{\hat{\rho}}$ for all times and can be ultimately driven into $\Omega_{\rho_{nn}}$ for any initial condition $x_0 \in \Omega_{\hat{\rho}}$. The reader is referred to[37] for detailed closed-loop stability and feasibility analysis.

3 Neural Network Training Using Noisy Data

While LSTM networks developed using noise-free computational simulation data have been demonstrated to be able to predict process dynamics accurately, LSTM training with noisy data has been a challenging research problem when dealing with industrial process data that contain noise. Traditional approaches for reducing noise in time-series data include the use of a filter, e.g., Savitzky-Golay filter, to smoothen the data without distorting the signal tendency. However, the filtering performance relies on the sliding time window length, which also affects the computation time in real-time implementation. Therefore, in this section, we discuss LSTM training methods that directly use noisy time-series data to predict true states. Specifically, we will discuss two types of sensor noise, Gaussian noise and non-Gaussian noise in the following subsections.

3.1 LSTM Training With Gaussian Noise

We first consider a Gaussian white noise $w \sim \mathcal{N}(0, \sigma^2)$ in sensor measurement y of Eq. 1, and demonstrate that the LSTM network is able to predict the true states well using the noisy state measurements following Gaussian distribution. We use a chemical process example to illustrate the de-noising capability of LSTM network with Gaussian noise data. Specifically, a well-mixed, non-isothermal continuous stirred tank reactor (CSTR) where an irreversible second-order exothermic reaction takes place is considered. The reaction transforms a reactant A to a product B ($A \rightarrow B$). The inlet concentration of A, the inlet temperature and feed volumetric flow rate of the reactor are C_{A0}, T_0 and F, respectively. The CSTR is equipped with a heating jacket that supplies/removes

heat at a rate Q. The CSTR dynamic model is described by the following material and energy balance equations:

$$\frac{dC_A}{dt} = \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT}} C_A^2 \tag{7a}$$

$$\frac{dT}{dt} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} \tag{7b}$$

where C_A is the concentration of reactant A in the reactor, V is the volume of the reacting liquid in the reactor, T is the temperature of the reactor and Q denotes the heat input rate. The concentration of reactant A in the feed is C_{A0}. The feed temperature and volumetric flow rate are T_0 and F, respectively. The reacting liquid has a constant density of ρ_L and a heat capacity of C_p. ΔH, k_0, E, and R represent the enthalpy of reaction, pre-exponential constant, activation energy, and the ideal gas constant, respectively. Process parameter values can be found in,[38] and are omitted here.

We study the operation of CSTR under LMPC around the unstable steady-state $(C_{As},\ T_s) = (1.95\ kmol/m^3,\ 402\ K)$, and $(C_{A0_s}\ Q_s) = (4\ kmol/m^3,\ 0\ kJ/hr)$. The manipulated inputs are the inlet concentration of species A and the heat input rate, which are represented by the deviation variables $\Delta C_{A0} = C_{A0} - C_{A0_s}$, $\Delta Q = Q - Q_s$, respectively. The manipulated inputs are bounded as follows: $|\Delta C_{A0}| \leq 3.5\ kmol/m^3$ and $|\Delta Q| \leq 5 \times 10^5\ kJ/hr$. Therefore, the states and the inputs of the closed-loop system are $x^T = [C_A - C_{As}\ T - T_s]$ and $u^T = [\Delta C_{A0}\ \Delta Q]$, respectively, such that the equilibrium point of the system is at the origin of the state-space, (i.e., $(x_s^*, u_s^*) = (0, 0)$). The Lyapunov function $V(x) = x^T P x$ is designed with $P = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix}$. Then, the closed-loop stability region Ω_ρ for the CSTR is characterized as a level set of the Lyapunov function with $\hat{\rho} = 368$ inside the region ϕ_u, from which $(\Omega_{\hat{\rho}})$ the origin can be rendered exponentially stable under the controller $u = \Phi(x) \in U$.

The explicit Euler method with an integration time step of $h_c = 10^{-4}\ hr$ is used to numerically simulate the dynamic model of Eq. 7. The nonlinear optimization problem of the LMPC of Eq. 6 is solved using the python module of the IPOPT software package,[32] named PyIpopt with the sampling period $\Delta = 10^{-2}\ hr$. The LSTM models are generated following the standard data generation and learning algorithm in.[36,37]

We first carry out extensive open-loop simulations of Eq. 7 with various initial conditions and control actions to generate the clean and noisy datasets. Specifically, the state x is measured every integration time step h_c and saved in the clean datset without introducing any measurement noise, while a Gaussian noise $w_1 \sim \mathcal{N}(0\ kmol/m^3, 2.5 \times 10^{-3}\ (kmol/m^3)^2)$ and $w_2 \sim \mathcal{N}(0\ K, 25\ K^2)$ are added on concentration and temperature measurements, respectively in the noisy dataset. Then, we train the LSTM network using the noisy dataset following the standard LSTM training process

without accounting for the fact that the dataset is corrupted by Gaussian noise. Additionally, an LSTM network using the noise-free dataset is also generated as a baseline for comparison. The mean squared errors (MSE) between the LSTM predicted values and the actual noise-free value are calculated as follows to indicate the capability of the two LSTMs in approximating true state trajectories: 1) the MSE of x_1 and x_2 for the standard LSTM using clean dataset are 3.816×10^{-5} $(kmol/m^3)$ and 0.0837 (K), respectively, and 2) the MSE of x_1 and x_2 for the standard LSTM using noisy dataset are 1.275×10^{-4} $(kmol/m^3)$ and 1.053 (K), respectively. It can be seen that the LSTM trained with clean dataset achieves better MSE results than that using noisy dataset; however, both networks achieve sufficiently small MSEs, which implies that the standard LSTM formulation is able to suppress the contributions from noisy state measurements, and learn the noise-free pattern that captures the true process dynamics. Following the study in,[41] we provide an insight on the de-noising capability of LSTM by computing the relative contributions of the LSTM internal dynamics and of the noisy input (i.e., noisy state measurements). Fig. 1 shows the time evolution of the LSTM internal states (i.e., $\|\omega_i^h h\|_1$ in a black dotted line) and LSTM input network (i.e., $\|\omega_i^m m\|_1$ in a red dashed line) states for 20 data samples using the same noise level. From any of the data sample, it is observed that the internal state evolution is smoother than the input network and that the norm of the internal states is larger than that of the input network. Therefore, it is concluded that the internal states play a dominant role in the prediction of LSTM outputs when using a noisy dataset. This explains why the LSTM network is capable of predicting true states well even using noisy state measurements.

Additionally, we study the de-noising capability of LSTM networks for different noise levels. We calculate the root mean-square errors (RMSE) of the three LSTM networks trained against different noise levels (i.e., small, medium and large σ in Fig. 2) in predicting true states using the noisy datasets with various noise levels (i.e., noise level 1, 2, 3 and 4 in Fig. 2). Noise level 1, 2, 3, 4 correspond to the white Gaussian noise with $\sigma_1^{C_A} = 0.05$, $\sigma_1^T = 5$, $\sigma_2^{C_A} = 0.1$, $\sigma_2^T = 10$, $\sigma_3^{C_A} = 0.15$, $\sigma_3^T = 15$, and $\sigma_4^{C_A} = 0.2$, $\sigma_4^T = 20$ on $x_1 = C_A - C_{As}$ and $x_2 = T - T_s$, respectively, where the subscript number denotes the noise level, and the units are omitted here. The three LSTM networks trained against the small, medium and large noise levels represent the LSTM models using the training datsets with level 1, 2 and 3 Gaussian noise, respectively. It is shown in Fig. 2 that the LSTM model trained against the noise with a small σ perform better in small noise level, but is sensitive to the variation of noise level. On the other hand, the LSTM model trained against the noise with a large σ is less sensitive to the noise level since the network relies more on its internal dynamics as shown in Fig. 1. Therefore, Fig. 2 provides an insight on the optimal range of

noise levels where different LSTM networks can be applied. Additionally, it also explains why the neural networks trained on a noise-free dataset become extremely sensitive to small perturbations on testing data, and as a result, we sometimes intentionally introduce noise into neural network training process in order to improve its robustness in practical implementation.

3.2 LSTM Training With Non-Gaussian Noise

Through the investigation of the case of Gaussian noise in training datasets, we have demonstrated that the standard LSTM is able to reduce the impact of Gaussian noise by utilizing its internal dynamics. In this section, we further study the LSTM capability of handling non-Gaussian noise and introduce a dropout method that is used to reduce overfitting in LSTM when trained with a noisy dataset. While there are many types of non-Guassian noise that are difficult to formalize the distribution using an equation, we study a specific industrial noise in chemical processes to illustrate the application of LSTM networks.

The top figure in Fig. 3 shows the original dimensionless data from ASPEN (public domain data), from which it can be seen that the data noise follows a non-Gaussian distribution due to the irregular drifts. To extract the noise information, a Savitzky−Golay filter is used to approximate the underlying noise-free profile, which is represented by the red line in the top figure of Fig. 3. By subtracting the raw data from its noise-free trajectory and then performing normalization, the main feature of this industrial data noise is extracted, which is shown as the normalized noise in the bottom of Fig. 3. In the following simulations, the normalized industrial data noise will be added into LSTM datasets generated from open-loop simulations of the CSTR of Eq. 7 to mimic the industrial data for the CSTR process we considered.

3.3 LSTM Networks Using Dropout Layers

The dropout technique is widely used in neural networks to prevent overfitting by randomly masking network units.[30] Although the standard LSTM model is demonstrated to predict process dynamics well using the sensor measurements with Gaussian noise, it performs poorly on training data corrupted by non-Gaussian noise as shown in Fig. 4. In this section, we take advantage of Monte Carlo dropout method (MC dropout) that was recently proposed in,[9,10] in which RNN models with MC dropout layers were interpreted as probabilistic models. Specifically, the RNN weights are treated as random variables, and the posterior distribution of the RNN model weights is obtained by sampling the network with randomly dropped out weights at test time (termed Monte Carlo samples). It was demonstrated in[10] that by using dropout techniques, a large RNN model with a

sufficient number of neurons may lead to improved denoised results compared to small RNN models that were commonly used in the past to avoid overfitting.

To simplify the discussion, we present the Monte Carlo method using the general form of LSTM models in Eq. 3. However, the following discussion can be readily generalized to the detailed LSTM network of Eq. 4, and other RNN models, e.g., gated recurrent unit (GRU) as shown in.[10] Let $\mathbf{W} = \{W_i\}_{i=1}^{L}$ denote the weight matrix of the LSTM model of Eq. 3 including all the weights and bias terms to be optimized, where W_i is the weight matrix of dimension $K_i \times K_{i-1}$ for each LSTM layer i, and L is the number of layers. Given the LSTM training data that include the data pair of $(\mathbf{M}, \mathbf{X})$, where $\mathbf{M}$ and $\mathbf{X}$ represent the LSTM input and output matrices, respectively, the goal of the LSTM model using MC dropout layers (termed dropout LSTM) is to find the posterior distribution over the weights $p(\mathbf{W} \mid \mathbf{M}, \mathbf{X})$. We first define a binary variable $z_{i,j}$ of Bernoulli distribution following:[9]

$$z_{i,j} \sim \text{Bernoulli}(p_i) \tag{8}$$

where $z_{i,j} = 0$, $i = 1, ..., L$, $j = 1, ...K_{i-1}$ represents the j_{th} weight between layer $i-1$ and layer i being dropped out with probability $1 - p_i$, and $z_{i,j} = 1$ represents the weight remaining unchanged with probability p_i. Therefore, the weight matrix W_i can be represented as follows for $i = 1, ..., L$.

$$W_i = B_i \cdot \text{diag}(z_i) \tag{9}$$

where B_i are the variational variables to be optimized. Since the posterior distribution $p(\mathbf{W} \mid \mathbf{M}, \mathbf{X})$ is intractable in practice,[9] proposed to use the approximating distribution $q(\mathbf{W})$ of Eqs. 8-9 and minimize the Kullback-Leibler (KL) divergence between the full posterior and $q(\mathbf{W})$. Therefore, an approximate predictive distribution of LSTM output can be given as follows:

$$p(\mathbf{x}^* \mid \mathbf{m}^*,\ \mathbf{X},\ \mathbf{M}) = \int p(\mathbf{x}^* \mid \mathbf{m}^*,\ \mathbf{W}) q(\mathbf{W}) d\mathbf{W} \tag{10}$$

where $\mathbf{m}^*$ is the LSTM input in testing datasets, $\mathbf{x}^*$ is the corresponding LSTM predicted output, and $\mathbf{X}$, $\mathbf{M}$ are the LSTM inputs and outputs in training dataset. Additionally, Eq. 10 can be approximated by performing Monte Carlo dropout at test time and calculating the averaged results as follows:

$$p(\mathbf{x}^* \mid \mathbf{m}^*,\ \mathbf{X},\ \mathbf{M}) \approx \frac{1}{N_t} \sum_{k=1}^{N_t} p(\mathbf{x}^* \mid \mathbf{m}^*,\ \mathbf{W}_k) \tag{11}$$

where $\mathbf{W}_k \sim q(\mathbf{W})$, and N_t is the number of realizations in dropout LSTM. It is noted that unlike normal dropout approach that does not apply dropout at test time, the main difference of MC dropout is that it applies dropout at both training and testing phases, thereby the LSTM prediction

is no longer deterministic. Given a same testing input, by performing dropout LSTM prediction multiple times, we will be able to generate random predictions, from which an approximate probabilistic distribution of LSTM output is obtained. In this way, dropout LSTM provides a potential solution to learn the ground truth (i.e., nominal state trajectories) from sensor measurement data corrupted by a complex, non-Gaussian noise. The application of dropout LSTM is illustrated using the same CSTR example that has been introduced in the previous section, and the simulation results are presented below.

The open-loop prediction results for standard LSTM and dropout LSTM are summarized in Table 3, and will be discussed in detail after we introduce the co-teaching method in the next section. Overall, it can be concluded from Table 3 that the dropout LSTM achieves better prediction results in terms of smaller validation MSE. Fig. 4 shows one of the open-loop prediction results from the dropout LSTM and the standard LSTM using the same noisy dataset for training. Specifically, we ran dropout LSTM prediction 300 times to obtain the distribution of predicted state trajectories, from which we show the mean state trajectory (red line) and the 95% standard deviation interval (gray region) in Fig. 4. It is demonstrated that the mean state trajectory predicted by the dropout LSTM is much closer to the ground truth (i.e., the nominal state trajectory in black) compared to the state trajectory (yellow line) predicted by the standard LSTM.

Since dropout LSTM uses Monte Carlo simulation to obtain the prediction of mean state trajectory, parallel computing is utilized to reduce the computation time by running multiple realizations in different cores/nodes. The parallelization of dropout LSTM is particularly useful when it is incorporated in MPC for real-time control implementation. Table 1 summarizes the averaged computation time for predicting one state trajectory using standard LSTM, and dropout LSTM under serial and parallel computing, respectively with a testing dataset of 100 data samples. It is noted that in the parallel implementation of dropout LSTM, 60 nodes were reserved in UCLA Hoffman 2 computing cluster to carry out 60 simulation runs such that each simulation run was assigned to a single worker node. The mean state trajectory and standard deviation are computed in the host node after synchronization operation. It is clearly seen from Table 1 that the computation time increases dramatically as the number of simulation runs increase under serial computation of dropout LSTM. On the other hand, the parallel computation of dropout LSTM can instead significantly reduce the computation time to the level that is comparable to the standard LSTM.

Remark 2. It should be noted that the computation time for dropout LSTM using parallel computing reported in Table 1 is the best case one can achieve in general. In fact, when we simulate a large number of trajectories under dropout LSTM, for example, 300 simulation runs in generating Fig. 4,

it is impractical to reserve such a number of computing nodes to allow simulating one trajectory in each single worker node. Therefore, it is important to adjust the number of simulation runs to achieve a balance between the computational burden and optimal prediction results.

4 Co-teaching Method

Unlike the standard and dropout LSTM networks that use noisy data only for training, the co-teaching method that we will discuss in this section takes advantage of noise-free datasets that can be obtained from first-principles modeling and simulation of chemical processes to further improve LSTM prediction accuracy. Co-teaching has been originally proposed for the classification problem with noisy labels ([13,40]), for example, in image classification, one of the most popular applications of neural networks that classifies the images into one of a number of predefined classes. However, in many cases, the training dataset for image classification task is not totally clean in the sense that some images are mislabeled, where the term "noisy label" is often used to represent the image data with incorrect labels. Without any treatment on dataset, training neural networks using such a noisy dataset may lead to undesired model accuracy. However, cleaning noisy labels manually also appears impractical when dealing with a high-dimensional dataset.

Co-teaching is an algorithm that trains models with noisy labels by training two networks simultaneously using different datasets of the targeted system ([13]). Fig. 5 shows a schematic diagram of the co-teaching method with two networks: A and B. The intuition of co-teaching is straightforward and it is based on the observations that neural networks tend to fit simple pattern at the early stage of training process.[13] As a result, noise-free data will achieve a low loss function value, while noisy data typically has a high loss function value. Specifically, the co-teaching training method (Algorithm 1) works as follows: in each mini-batch during training epoch k, each model checks its data sequences (i.e., each pair of data labeled as input and output), and generates a small dataset with all the data that has a low loss function value. This new dataset can be approximately regarded as noise-free datasets, and will be shared between two networks. Subsequently, after receiving the new noise-free datasets from the peer network, the weights are updated and the training is resumed for one more epoch. The above process is repeated until the all the training epochs are completed.

Algorithm 1: Co-teaching Algorithm

D is the original mixed dataset, I_{max} is the maximum number of iterations, x is the data sequence, $loss(A, x)$ calculates the loss function value for data x under model A, $loss_T$ is the threshold for identifying small-loss data sequences, and η is the learning rate.

for $i = 0$ **to** I_{max} **do**

Select a mini-batch D_m from D

Obtain the small-loss data sequences from model A: $D_A = \{x \in D_m \mid loss(A, x) \leq loss_T\}$

Obtain the small-loss data sequences from model B: $D_B = \{x \in D_m \mid loss(B, x) \leq loss_T\}$

Update the weight matrix of model A: $\mathbf{W_A} = \mathbf{W_A} - \eta \nabla loss(A, D_B)$

Update the weight matrix of model B: $\mathbf{W_B} = \mathbf{W_B} - \eta \nabla loss(B, D_A)$

end

One of the main benefits of co-teaching method is that each network can filter noisy labels in different ways due to distinct learning abilities ([13]). In order for the algorithm to perform accurately, it is important to have a considerable number of clean sequences during the training step such that clean sequences can be detected and shared within the two networks. However, to our knowledge, little attention has been paid to the implementation of co-teaching method in solving regression problems. In this study, we take advantage of the idea of co-teaching method, and adapt it to LSTM modeling of nonlinear processes using noisy data. Specifically, we first develop a noise-free dataset from extensive first-principles model simulations. Although an accurate process model is generally unavailable for industrial chemical processes, a first-principles model based on well-known mass, energy, and momentum balances can be developed to approximate process dynamics. This noise-free dataset can then be used in co-teaching training to guide LSTM networks to learn the underlying (noise-free) dynamics. When using co-teaching method for solving regression problems, it is important to tune the LSTM structure, develop a high-quality dataset consisting of both noise-free and noisy data, and choose the threshold that can identify the clean data in the mixed dataset. Specifically, the number of units in each network should be carefully chosen in order to achieve a balanced performance between the noisy and clean pattern. For example, a large number of hidden units are not preferred since the network tends to fit both clean and noisy data at the same time (i.e., overfitting). Second, the ratio between clean data obtained from the approximated model and noisy data should also be carefully chosen. If clean data is insufficient, the networks are not able to share the noise-free process dynamics information during training phase; however, if too much clean data is included in the mixed dataset, then the network would simply learn the process dynamics from approximated first-principles model instead of the actual process dynamics from noisy data. Additionally, given a mixed dataset, it is also important to identify the clean data by evaluating loss function value as introduced at the beginning of the section. To better understand the role

of threshold in choosing clean data, we perform the following case study using the symmetry co-teaching method with model A and B that share clean data during the training step. Model B uses a dataset of 500 sequences in which the first 100 data are noisy and the following 400 data are clean sequences. Subsequently, at the end of each epoch, based on the threshold we pre-determined, we select the clean sequences by evaluating their contribution to the loss function. Table 2 shows the selected clean sequences for four thresholds $loss_T$ during the same epoch. it is demonstrated in Table 2 that under the threshold of 0.01, 306 sequences are selected to be shared with model A in which 267 are truly clean sequences (i.e., the accuracy is approximately 87%). By decreasing the threshold value, it is shown in Table 2 that fewer data sequences are selected as clean data; however, the accuracy is increasing (e.g., the ratio of true clean data is increasing to 95.6% for the threshold 0.0014). Therefore, the threshold for identifying clean data also plays an important role in the co-teaching training performance.

Another type of co-teaching framework uses an asymmetric structure (termed asymmetric co-teaching). Unlike the symmetric co-teaching that trains two models using the same dataset, asymmetric co-teaching methodology trains two models (i.e., A and B) using a noise-free and a noisy dataset, respectively. In this work, the noise-free dataset for model A comes from extensive open-loop simulation for the CSTR model of Eq. 7 under 2000 initial conditions that cover the entire stability region and 100 pairs of manipulated inputs. Based on the open-loop simulation dataset, the noisy dataset is generated by adding the industrial noise of Fig. 3 on state measurements. As a result, the two datasets have the same number of data sequences and the data is also organized in the same order. In other words, the k_{th} data sequence in the two datasets are using the same initial condition and the same control actions with the only difference that the industrial noise is added on state measurements. In asymmetric co-teaching framework, clean dataset is extracted and shared only from model A to model B, and not in the opposite direction. Similar to Algorithm 1, in each training epoch, model A sends a subset of clean data sequences to model B. To update the noisy labels during the co-teaching implementation in the asymmetric co-teaching method, the noisy labels are randomly updated using its noise-free counterpart from the clean dataset. It will be shown in the next section that LSTM training using asymmetric co-teaching method can achieve better model accuracy than the LSTM using standard training algorithm with the same mixed dataset of clean and noisy data. To implement the co-teaching method in Keras, the training process of the neural network is discretized by using a loop structure in order to update the noisy LSTM outputs between two consecutive epochs. After updating the outputs, the training is resumed and the noisy outputs will be updated again at the end of next training epoch.

4.1 Open-loop Simulation Results of Three LSTM Models

We carry out the open-loop simulations using standard LSTM, dropout LSTM, and co-teaching LSTM models trained with the same noisy dataset, and show the MSE results in Table 3. It should be noted that all the LSTM models in Table 3 are developed using the same structure in terms of neurons, layers, epochs, and activation functions. Additionally, the MSE is calculated as the difference between the predicted state trajectories and the underlying (noise-free) state trajectories since the goal is to capture the nominal process dynamics using noisy data. Therefore, an LSTM model with a low MSE value implies that the model is able to predict smooth state trajectories that are close to the ground-truth trajectories.

As shown in Table 3, we first train three standard LSTM models using the noise-free dataset (baseline case), the mixed dataset, and the noisy dataset, respectively, to demonstrate that the standard LSTM modeling approach cannot achieve a desired model accuracy without a high-quality dataset. However, compared with the standard LSTM using noisy data only (i.e., 1c in Table 3), it is observed that the LSTM using mixed data (i.e., 1b in Table 3) shows a slight improvement of model prediction with the use of noise-free data in training. The co-teaching LSTM and dropout LSTM models are then developed using the same noisy dataset. Specifically, the co-teaching LSTM starts with a noise dataset, and receives clean data sequences from the peer model as the training process evolves. As a result, the weight matrices of the co-teaching model are able to capture a balanced pattern that accounts for both noisy and clean data. The dropout LSTM is trained using the noisy dataset only and the final prediction results are obtained by averaging all Monte Carlo realizations that apply dropout at test time. It is shown in Table 3 that the dropout LSTM outperforms the standard LSTM model in terms of a lower MSE, and the co-teaching LSTM achieves the best performance among all the models.

4.2 Closed-loop Simulation Results under LMPC

We run closed-loop simulations on the CSTR system of Eq. 7 under the LMPC of Eq. 6 using different LSTM models (i.e., standard LSTM, co-teaching LSTM, and dropout LSTM). The LMPC receives noisy measurements at each integration time step, and solves for the optimal control actions at every sampling time. The control objective is to operate the CSTR at the unstable equilibrium point ($C_{As},\ T_s$) by manipulating the heat input rate ΔQ and the inlet concentration ΔC_{A0} under LMPC. The objective function of the LMPC is designed with the following form that has its

minimum value at the steady-state:

$$L(x,u) = |x|^2_{Q_1} + |u|^2_{Q_2} \tag{12}$$

where Q_1 and Q_2 are coefficient matrices that balance the contributions of state convergence and of control actions (i.e., energy and reactant). Table 4 summarizes the closed-loop performance of different LSTM models by using the objective function of Eq. 12 as an indicator. Specifically, industrial noise of four different levels (i.e., tiny, small, medium and large levels) that have been discussed in Section 3.1 are added on closed-loop state measurements for each LSTM model. Each level of noise is tested in closed-loop simulation using five different initial conditions in the stability region. Then, we use the LMPC objective function value to indicate its closed-loop performance in the way that a large objective function value implies a slow convergence and a high energy consumption. We calculate the objective function value over time for each closed-loop state trajectory and compute the averaged result, i.e., $\frac{1}{N_s}\sum_{i=1}^{N_s}\int_{t=0}^{t=t_s} L(x,u)dt$ for each LSTM model, where N_s is the number of simulation runs ($N_s = 5$ corresponding to five different initial conditions in this study), and t_s is the closed-loop simulation time. The mean objective function value is finally normalized with respect to its biggest value in all simulation runs to eliminate the impact of initial conditions on the analysis of closed-loop performance.

As shown in Table 4, the standard LSTM has the worst closed-loop performance in terms of the largest objective function value among all three models. Additionally, co-teaching and dropout LSTMs achieve a significant improvement (around 20%) for the small and medium noise levels, while the improvement is less than 10% for tiny and large noise levels. Specifically, it is observed from closed-loop simulations that under a tiny noise level, the LMPCs using the three LSTM models all drive the closed-loop state to the steady-state quickly. This is consistent with the fact that the machine-learning-based LMPC is robust to a sufficiently small disturbance as shown in.[37] Additionally, all the LSTM models perform poorly under a large noise level since the noisy state measurements deviate far from its true state in the presence of a large noise, which makes it challenging for LSTM to predict the true states. As a result, LMPC is unable to solve for the optimal control actions that can drive the closed-loop state towards the steady-state. In fact, all the closed-loop state trajectories show significant oscillations under a large noise level. Practically, such a large noise implies that the problem is beyond the scope of sensor noise, and equipment maintenance may be needed before bringing controller system on-line. In the case of a small and a medium noise level, it is shown in Table 4 that closed-loop performances under the co-teaching and dropout LSTMs are much improved compared to the standard LSTM. This implies that the proposed LSTM modeling approaches are preferred in handling noise in such a range of noise levels.

In Fig. 6 and Fig. 7, we show one of the closed-loop simulation results under a medium noise level. It is observed in state profiles of Fig. 6 that the standard LSTM model shows significant variation when the closed-loop state approaches the steady-state. This is due to the fact that states predicted by the standard LSTM are far from the true states, which mislead the LMPC to give a solution that drives the state in a wrong direction. However, since the co-teaching LSTM and dropout LSTM models have been demonstrated to achieve a desired model accuracy in Section 4.1, it is expected that the LMPC using these two models can maintain the state in a small neighborhood around the steady-state more smoothly.

Remark 3. It should be noted that closed-loop stability under LMPC is not guaranteed since a sufficiently high model accuracy as required for machine learning models in[37] may be unachievable for the proposed LSTM models that receive noisy measurements to predict true states. However, from extensive closed-loop simulation runs, it is demonstrated that with a sufficiently small noise, the state trajectories for all types of LSTM models successfully converge to the steady-sate from different initial conditions. As the noise level increases, the closed-loop performance gets worse for all LSTM models in the sense that most of the state trajectories can still converge to the steady-state but showing oscillation over time. In this case, the proposed dropout and co-teaching LSTM models can be used to improve closed-loop performance when dealing with a handleable noise level.

5 Conclusion

In this work, we presented machine learning approaches that can predict underlying nonlinear process dynamics from noisy data. We initially investigated the case of Gaussian noise and demonstrated that the standard LSTM using noisy dataset was able to achieve a desired model accuracy since the internal states of the neural network played a dominant role in prediction. Then, we investigated the case of a non-Gaussian noise from ASPEN industrial-data sets, for which the standard LSTM networks showed poor prediction performance. To handle industrial noisy data, dropout LSTM and co-teaching LSTM schemes were proposed to learn process dynamics using noisy data only, and using both noisy data and noise-free first-principles model simulation data, respectively. The chemical process example was utilized to demonstrate the improved model accuracy achieved by the dropout and co-teaching LSTM models through both open-loop and closed-loop simulations.

Literature Cited

1. I. A. Udugama, C. L. Gargalo, Y. Yamashita, M. A. Taube, A. Palazoglu, B. R. Young, K. V. Gernaey, M. Kulahci, and C. Bayer. The role of big data in industrial (bio) chemical process operations. *Industrial & Engineering Chemistry Research*, 2020.

2. E. Aggelogiannaki and H. Sarimveis. Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models. *Computers & Chemical Engineering*, 32:1225–1237, 2008.

3. J. M. Ali, M. A. Hussain, M. O. Tade, and J. Zhang. Artificial intelligence techniques applied as estimator in chemical process systems–a literature survey. *Expert Systems with Applications*, 42:5915–5931, 2015.

4. S. Aumi, B. Corbett, T. Clarke-Pringle, and P. Mhaskar. Data-driven model predictive quality control of batch processes. *AIChE Journal*, 59:2852–2861, 2013.

5. C. M. Bishop. *Pattern recognition and machine learning*. Springer-Verlag New York, 2006.

6. D. Chaffart and L. A. Ricardez-Sandoval. Optimization and control of a thin film growth process: A hybrid first principles/artificial neural network based multiscale modelling approach. *Computers & Chemical Engineering*, 119:465–479, 2018.

7. S. Chen, Z. Wu, D. Rincon, and P. D. Christofides. Machine learning-based distributed model predictive control of nonlinear processes. *AIChE Journal*, in press, 2020.

8. M. J. Ellis and V. Chinde. An encoder-decoder LSTM-based EMPC framework applied to a building HVAC system. *Chemical Engineering Research and Design*, 160:508–527, 2020.

9. Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the International Conference on Machine Learning*, pages 1050–1059, 2016.

10. Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1019–1027, 2016.

11. A. Garg and P. Mhaskar. Utilizing big data for batch process modeling and control. *Computers & Chemical Engineering*, 119:228–236, 2018.

12. F. Hamilton, T. Berry, and T. Sauer. Ensemble kalman filtering without a model. *Physical Review X*, 6:011021, 2016.

13. B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 8527–8537, 2018.

14. H. Hassanpour, B. Corbett, and P. Mhaskar. Integrating dynamic neural network models with principal component analysis for adaptive model predictive control. *Chemical Engineering Research and Design*, 161:26–37, 2020.

15. Q. P. He and J. Wang. Statistical process monitoring as a big data analytics tool for smart manufacturing. *Journal of Process Control*, 67:35–43, 2018.

16. Q. P. He, J. Wang, and D. Shah. Feature space monitoring for smart manufacturing via statistics pattern analysis. *Computers & Chemical Engineering*, 126:321–331, 2019.

17. L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

18. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

19. H. K. Khalil. *Nonlinear systems*, volume 3. Prentice Hall, Upper Saddle River, NJ, 2002.

20. G. Kimaev and L. A. Ricardez-Sandoval. Nonlinear model predictive control of a multiscale thin film deposition process using artificial neural networks. *Chemical Engineering Science*, 207:1230–1245, 2019.

21. E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. A. Ioannou. High-order neural network structures for identification of dynamical systems. *IEEE Transactions on Neural Networks*, 6:422–431, 1995.

22. J. H. Lee, J. Shin, and M. J. Realff. Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, 114:111–121, 2018.

23. Y. Lin and E. D. Sontag. A universal formula for stabilization with bounded controls. *Systems and Control Letters*, 16:393–397, 1991.

24. S. C. Patwardhan, S. Narasimhan, P. Jagadeesan, B. Gopaluni, and S. L. Shah. Nonlinear bayesian state estimation: A review of recent developments. *Control Engineering Practice*, 20:933–953, 2012.

25. S. J. Qin and L. H. Chiang. Advances and opportunities in machine learning for process data analytics. *Computers & Chemical Engineering*, 126:465–473, 2019.

26. M. M. Rashid, N. Patel, P. Mhaskar, and C. L. Swartz. Handling sensor faults in economic model predictive control of batch processes. *AIChE Journal*, 65:617–628, 2019.

27. A. Savitzky and M.J.E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36:1627–1639, 1964.

28. D. Shah, J. Wang, and Q. P. He. Feature engineering in big data analytics for IoT-enabled smart manufacturing–comparison between deep learning and statistical learning. *Computers & Chemical Engineering*, page 106970, 2020.

29. A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

30. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15:1929–1958, 2014.

31. A. P. Trischler and G. M. D'Eleuterio. Synthesis of recurrent neural networks for dynamical system simulation. *Neural Networks*, 80:67–78, 2016.

32. A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

33. H. Wang, D. Chaffart, and L. A. Ricardez-Sandoval. Modelling and optimization of a pilot-scale entrained-flow gasifier using artificial neural networks. *Energy*, 188:116076, 2019.

34. Y. Wang, M. Fang, X. Jiang, B. W. Bequette, and H. Xie. Intensive insulin therapy for critically ill subjects based on direct data-driven model predictive control. *Journal of Process Control*, 24:493–503, 2014.

35. W. Wong, E. Chee, J. Li, and X. Wang. Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing. *Mathematics*, 6:242, 2018.

36. Z. Wu and P. D. Christofides. Economic machine-learning-based predictive control of nonlinear systems. *Mathematics*, 7:494, 20 pages, 2019.

37. Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine learning-based predictive control of nonlinear processes. part I: Theory. *AIChE Journal*, 65:e16729, 2019.

38. Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine-learning-based predictive control of nonlinear processes. Part II: Computational implementation. *AIChE Journal*, 65:e16734, 2019.

39. W. Xie, I. Bonis, and C. Theodoropoulos. Data-driven model reduction-based nonlinear MPC for large-scale distributed parameter systems. *Journal of Process Control*, 35:50–58, 2015.

40. F. Yang, K. Li, Z. Zhong, Z. Luo, X. Sun, H. Cheng, X. Guo, F. Huang, R. Ji, and S. Li. Asymmetric co-teaching for unsupervised cross-domain person re-identification., 2020.

41. K. Yeo. Short note on the behavior of recurrent neural network for noisy dynamical system. *arXiv preprint arXiv:1904.05158*, 2019.

42. K. Yeo and I. Melnyk. Deep learning algorithm for data-driven simulation of noisy dynamical system. *Journal of Computational Physics*, 376:1212–1231, 2019.

43. Y. Yu, X. Si, C. Hu, and J. Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31:1235–1270, 2019.

44. J. Zeng, C. Gao, and H. Su. Data-driven predictive control for blast furnace ironmaking process. *Computers & Chemical Engineering*, 34:1854–1862, 2010.

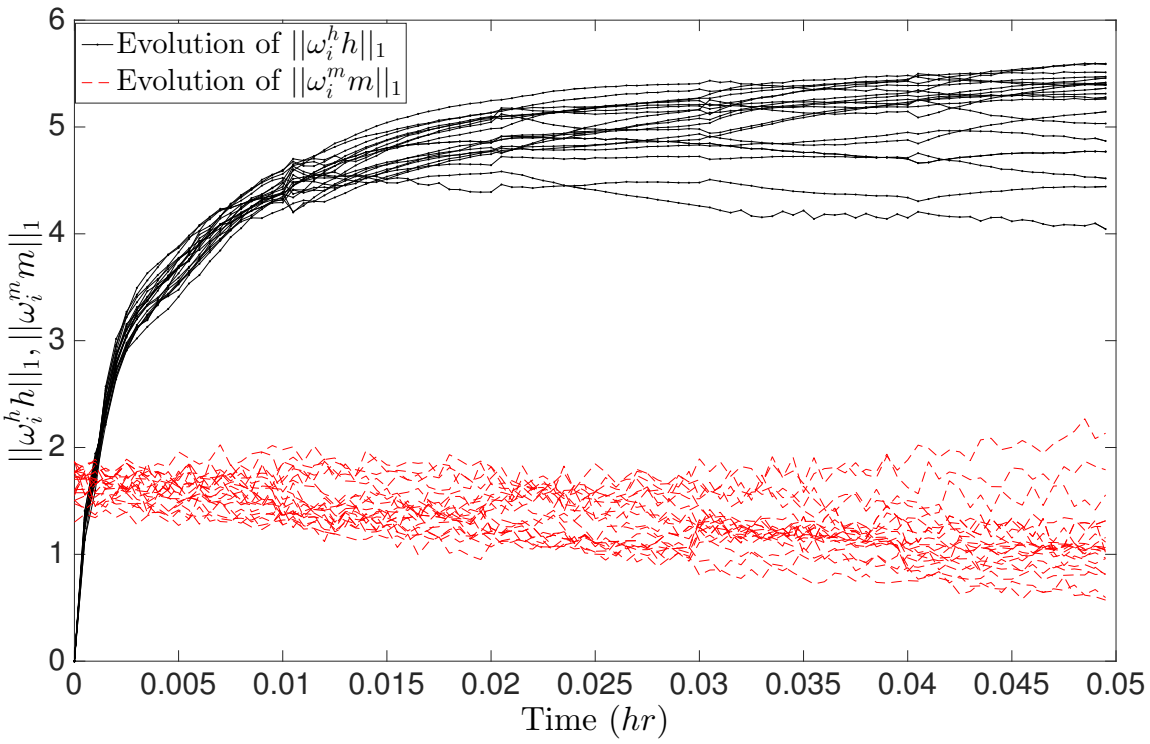

Figure 1: Time evolution of the LSTM internal states $\|\omega_i^h h\|_1$ and the input states $\|\omega_i^m m\|_1$ under Gaussian noise for multiple data sequences.

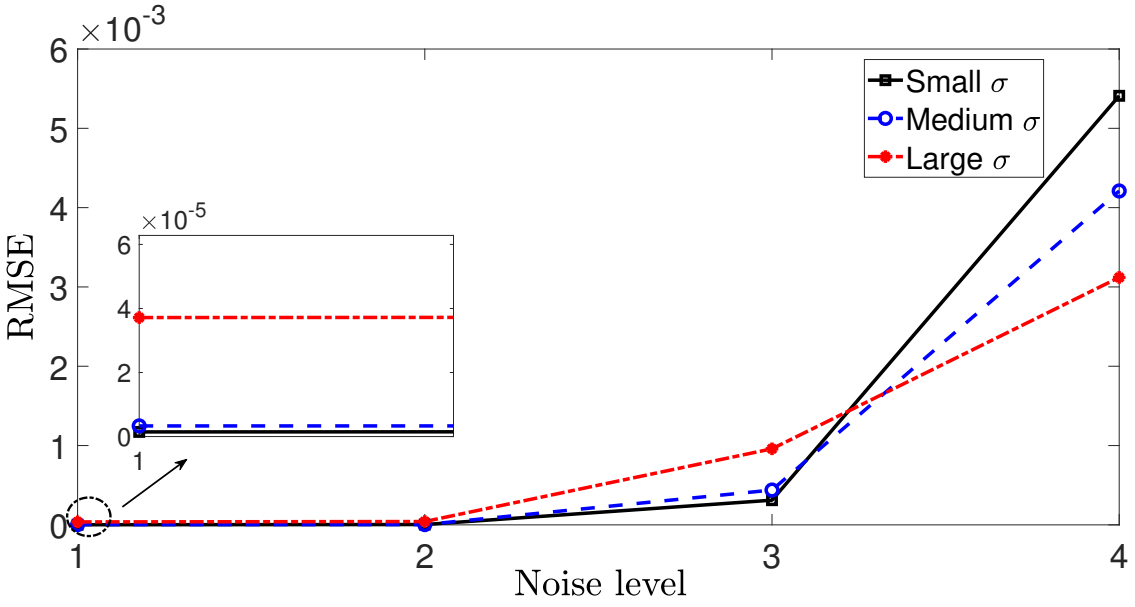

Figure 2: RMSE in terms of the noise level for LSTM models trained against different noise levels.

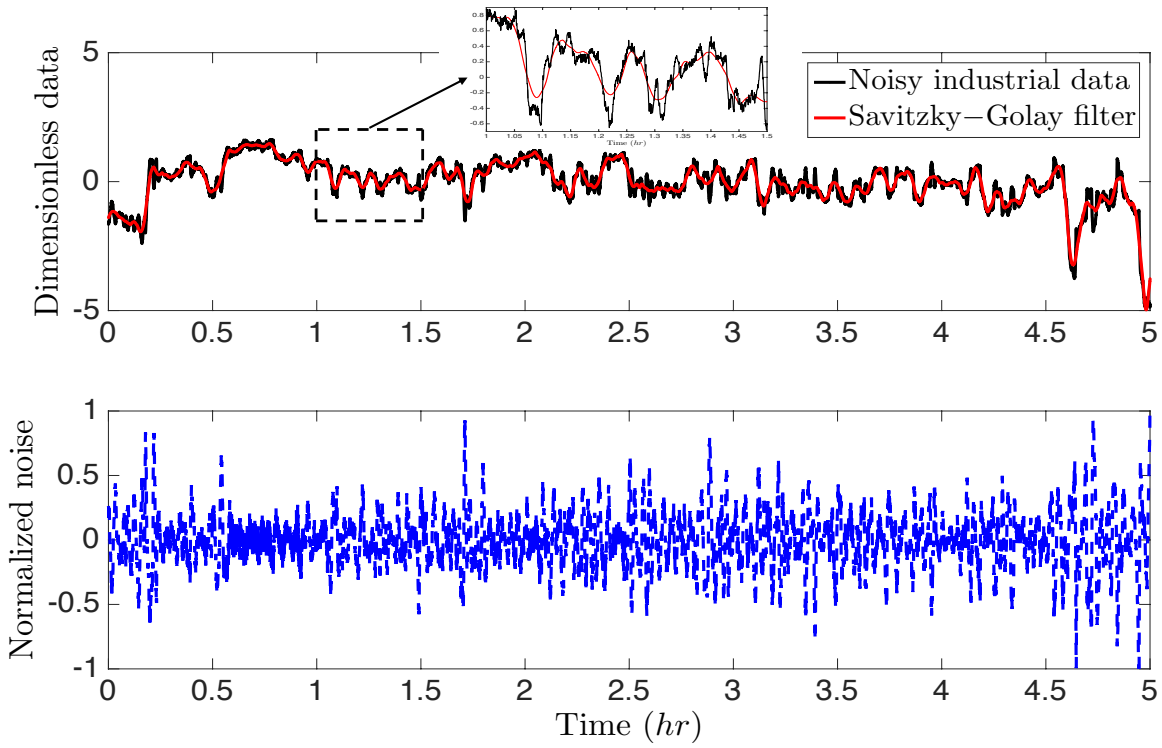

Figure 3: Noisy industrial-data sets from ASPEN (black line in the top figure), the denoised result using Savitzky−Golay filter (red line in the top figure), and the extracted (normalized) noise from ASPEN industrial data (bottom figure).

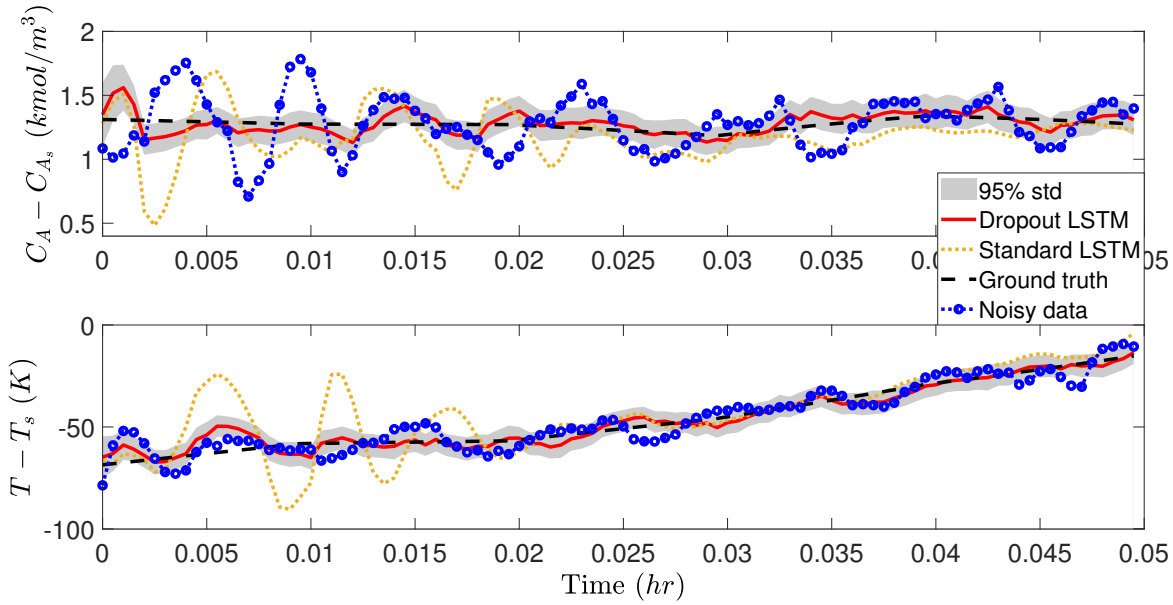

Figure 4: State profiles predicted by the dropout LSTM and the standard LSTM, where the red line is dropout LSTM, the black, dashed line is the ground truth, the yellow line is the standard LSTM, and the blue, dotted line is the noisy state measurement.

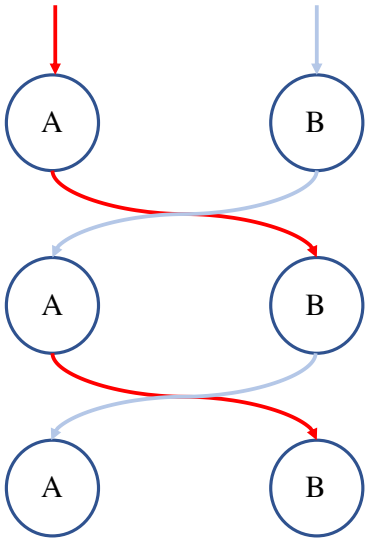

Figure 5: The symmetric co-teaching framework that trains two networks (A and B) simultaneously.

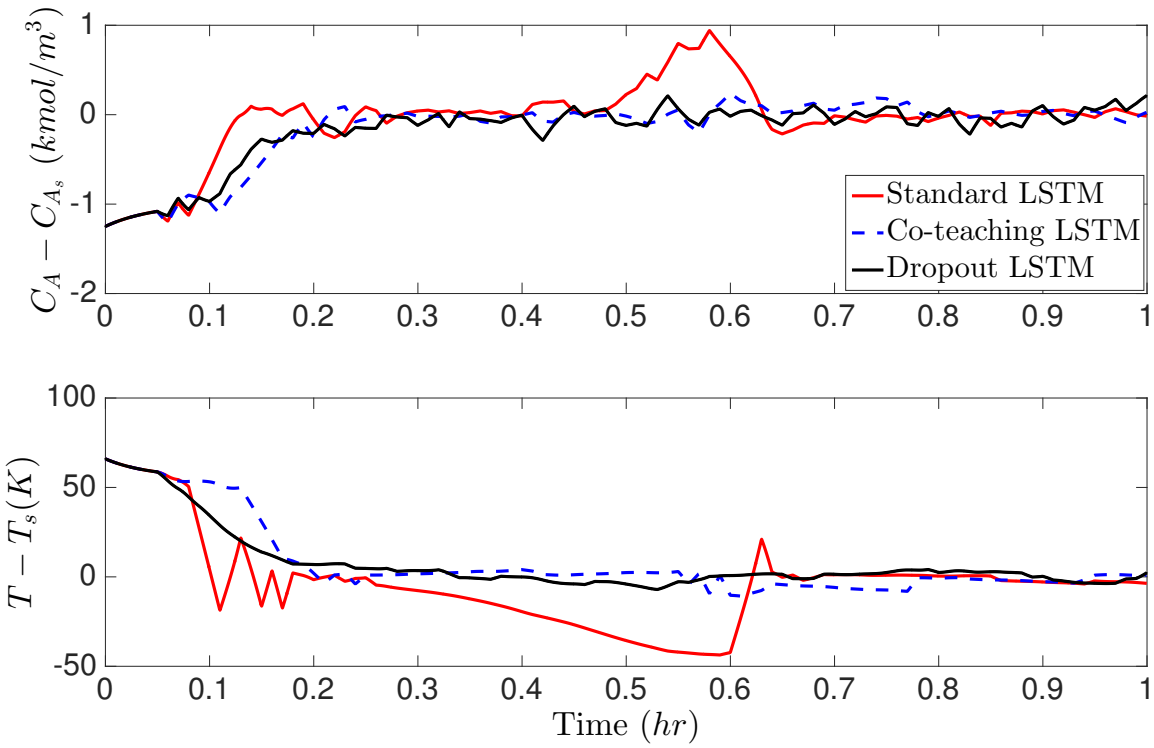

Figure 6: Closed-loop state profiles for the initial condition (-1.25, 66) under LMPC using standard LSTM (red), co-teaching LSTM (blue), and dropout LSTM (black).

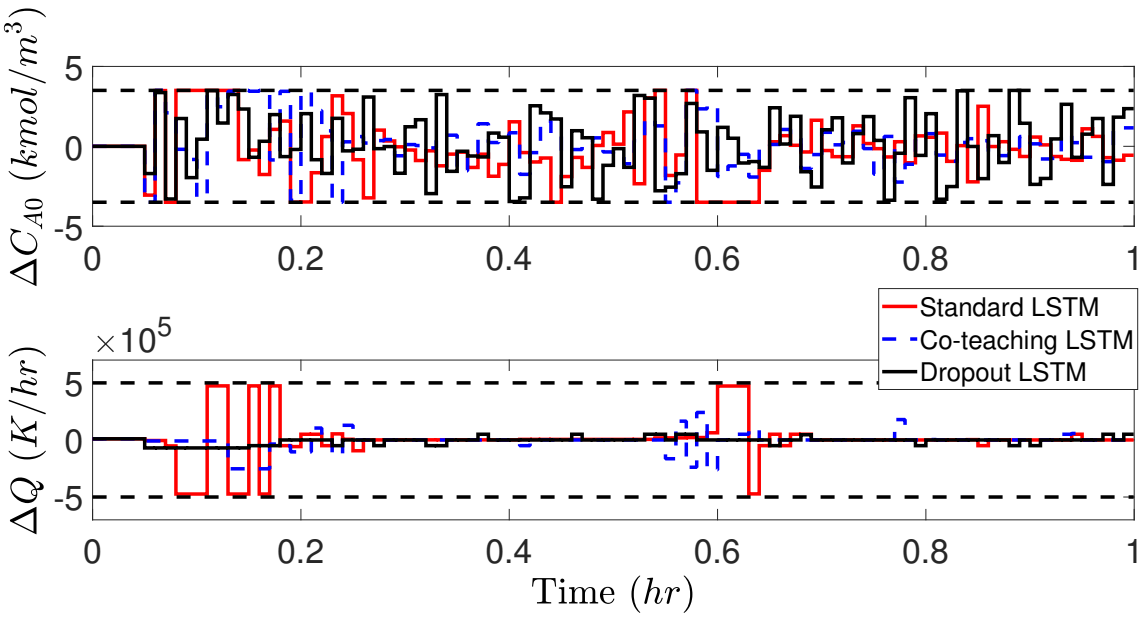

Figure 7: Manipulated input profiles ($u_1 = \Delta C_{A0}$, $u_2 = \Delta Q$) for the initial condition (-1.25, 66) under LMPC using standard LSTM (red), co-teaching LSTM (blue), and dropout LSTM (black).

Table 1: Comparison of computation time under the standard LSTM and dropout LSTM networks.

Model	**Computation time (seconds)**
1) Standard LSTM	0.0165
2) Dropout LSTM with serial computing	0.0536
3) Dropout LSTM with parallel computing	0.0183

Table 2: Tuning the threshold for the symmetry co-teaching methodology.

Threshold	**Clean seqs./Total seqs.**
0.01	(267 / 306)
0.003	(96 / 102)
0.002	(37 / 38)
0.0014	(22 / 23)

Table 3: Statistical analysis of the open-loop predictions under non-Gaussian noise.

Methods	**MSE** x_1	**MSE** x_2
1a) LSTM : noise-free data only	0.0011	8.2056
1b) LSTM : mixed data	0.0258	22.1795
1c) LSTM: noisy data only	0.0328	29.5571
2) Co-teaching LSTM	0.0053	7.2123
3) Dropout LSTM	0.0052	19.2123

Table 4: Statistical analysis of the closed-loop simulation results under non-Gaussian noise using four noise levels.

Methods	**Tiny**	**Small**	**Medium**	**Large**
1) Standard LSTM	**0.8786**	**0.9796**	**0.9430**	**0.8768**
2) Co-teaching LSTM	0.8110	0.7587	0.7749	0.7850
3) Dropout LSTM	0.7162	0.6889	0.8078	0.7928