Review

Zhe Wu*, Panagiotis D. Christofides*, Wanlu Wu, Yujia Wang, Fahim Abdullah, Aisha Alnajdi and Yash Kadakia

A tutorial review of machine learning-based model predictive control methods

https://doi.org/10.1515/revce-2024-0055
Received July 26, 2024; accepted September 25, 2024;
published online December 10, 2024

Abstract: This tutorial review provides a comprehensive overview of machine learning (ML)-based model predictive control (MPC) methods, covering both theoretical and practical aspects. It provides a theoretical analysis of closed-loop stability based on the generalization error of ML models and addresses practical challenges such as data scarcity, data quality, the curse of dimensionality, model uncertainty, computational efficiency, and safety from both modeling and control perspectives. The application of these methods is demonstrated using a nonlinear chemical process example, with open-source code available on GitHub. The paper concludes with a discussion on future research directions in ML-based MPC.

Keywords: machine learning; neural networks; model predictive control; nonlinear systems; chemical processes

1 Introduction

Model predictive control (MPC) is an advanced control technique that has seen extensive use in industrial applications since the 1980s. Unlike traditional control methods, MPC uses a dynamic model of the system of interest to predict future behavior and compute optimal control actions. Upon receiving information on the system's current state, the MPC will generate a sequence of optimal control actions over a predefined time horizon, based on the predictions made by the dynamic model. The receding horizon feature of MPC implies that only the first control input in the optimal control action sequence is implemented, and a new sequence is calculated at the next time step with updated system information. As the MPC relies heavily on the predictions of the dynamic model for computation, a highly accurate model is vital for satisfactory control performance. Traditionally, these models are derived predominantly from fundamental theories. For example, the dynamic behaviors of chemical processes are theoretically described using mass and energy balance equations.

However, the derivation of these first-principles models can often be tedious and costly, especially for complex nonlinear systems. Empirical methods that construct dynamical models from data pose as an alternative to the theoretical approach. Although the use of machine learning (ML) tools such as artificial neural networks (ANNs) has been applied to chemical systems over the past decades (Hoskins and Himmelblau 1988; Nascimento et al. 2000), the recent success of deep learning models has reignited interest in adopting ML models for MPC applications. Moreover, with increasing access to industrial data and enhanced computational capabilities, a paradigm shift from a theory-based modeling approach to a data-centric approach has been observed. In particular, recurrent neural networks (RNNs), a subbranch of deep learning models, known for its ability to process time series data, have shown promising results in modeling the dynamics of complex chemical systems for MPC applications (Limon et al. 2017; Su et al. 1992; Terzi et al. 2021; Wu et al. 2019d; You and Nikolaou 1993). In addition to simulation studies, ML-based MPC has been successfully applied to real-life systems such as a paper machine (Lanzetti et al. 2019), an experimental electrochemical reactor (Luo et al. 2023), a yeast biofermentation bioreactor (Nagy 2007), and a continuous pharmaceutical manufacturing process (Wong et al. 2018). Despite these advancements, the implementation of ML-based MPC in industrial settings is still far from being realized. In a strengths, weaknesses, opportunities and threats (SWOT) analysis performed by

***Corresponding authors: Zhe Wu**, Department of Chemical and Biomolecular Engineering, National University of Singapore, Singapore, Singapore, E-mail: wuzhe@nus.edu.sg. https://orcid.org/0000-0002-2923-149X; and **Panagiotis D. Christofides**, Department of Chemical and Biomolecular Engineering, Department of Electrical and Computer Engineering, University of California, Los Angeles, CA, USA, E-mail: pdc@seas.ucla.edu
Wanlu Wu and Yujia Wang, Department of Chemical and Biomolecular Engineering, National University of Singapore, Singapore, Singapore
Fahim Abdullah, Aisha Alnajdi and Yash Kadakia, Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA, USA

Dobbelaere et al. (2021), on the use of ML tools in chemical engineering, the authors identified the lack of interpretability of black-box ML models and the challenges in obtaining sufficient and reliable data as the main obstacles that prevent the application of ML models. In addition, controller robustness, operation safety, and system stability are some of the common issues that have been raised in the discussion of ML-based MPC (Bonassi et al. 2022; Brunke et al. 2022; Hewing et al. 2020; Nian et al. 2020; Schweidtmann et al. 2021). In this review article, we consolidate some of the common challenges faced in the industrial implementation of ML-based MPC and categorize them according to their theoretical and practical aspects.

The theoretical challenge of ML-based MPC lies in the mathematical guarantee of closed-loop stability under ML-based MPC (Berberich et al. 2020). Closed-loop stability is essential to ensure safe, efficient, and reliable operation of control systems. ML models are typically developed using a set of training data that is representative of the underlying data distribution. Even with sufficient training, some ML models may struggle to generalize to new, unseen data beyond the training set. This may result in poor controller performance and stability issues in ML-based MPC. Thus, understanding the generalization performance of ML models is a key challenge in guaranteeing closed-loop stability.

On the other hand, the practical challenges of implementing ML-based MPC in industrial settings are significantly more diverse, arising from the different stages in the development of ML-based MPC. The formation of an ML-based MPC can be divided chronologically into three phases: data collection, modeling, and execution phases. Each phase presents a unique set of challenges that need to be addressed to ensure the feasibility of ML-based MPC. Data scarcity and data corruption are common issues that plague the data collection process in industrial settings. As ML models are highly dependent on the quantity and quality of the data used for training, the question of how to develop accurate ML models under such circumstances poses a major concern in the application of ML-based MPC (Thebelt et al. 2022). Additionally, practical challenges often arise when modeling large-scale systems in industries. The curse of dimensionality, a phenomenon in which an increase in data dimensions results in an exponential growth in data requirement and a reduction in the efficiency and effectiveness of ML algorithms, presents a significant challenge in modeling large-scale systems. Thus, how to effectively capture the dynamics of complex large-scale systems and bypass the curse of dimensionality is another key challenge in the development of ML-based MPC for industrial applications. The heavy computational burden and sluggish processing speed of ML-based MPC is a well-acknowledged problem (Wu et al. 2019d). In addition, as noted by Mesbah et al. (2022), model uncertainty and process disturbances are unavoidable issues in controller implementation, how to speed up ML-based MPC calculations and improve the robustness of ML-based MPC are valid concerns that require attention during the execution phase of ML-based MPC. Overall, safety is an overarching concern that applies to all stages of the development of ML-based MPC (Brunke et al. 2022; Hewing et al. 2020). Safe data collection, safe modeling, and safe implementation are vital to ensure safe operation under ML-based MPC. Finally, the lack of interpretability of black-box ML models raises a general concern among global community (Bonassi et al. 2022; Dobbelaere et al. 2021; Schweidtmann et al. 2021; Shang and You 2019). The lack of understanding of the decision process and internal workings of data-driven models can impede users' trust towards these models, especially for control applications where safety is paramount. Thus, enhancing the transparency of ML models is a critical step towards gaining industrial approval of ML-based MPC.

Substantial reviews on the application of ML models to process system engineering have been discussed by Daoutidis et al. (2023), Everett (2021), Khan and Ammar Taqvi (2023), Lee et al. (2018), Mowbray et al. (2022), Pan et al. (2022), and Shang and You (2019). However, since the objective of these reviews was to provide an overview on the development of ML models and to analyze existing and potential applications of ML models in the industry, discussions on ML-based MPCs were limited. On the other hand, reviews specific to ML-based MPC focused on various aspects of ML-based MPC (Abdullah and Christofides 2023a; Berberich and Allgöwer 2024; Bonassi et al. 2022; Brunke et al. 2022; Dev et al. 2021; Gonzalez et al. 2023; Lu et al. 2019; Mesbah et al. 2022; Nian et al. 2020; Norouzi et al. 2023; Ren et al. 2022; Tang and Daoutidis 2022). For instance, Mesbah et al. (2022) and Nian et al. (2020) examine the applications of a particular type of ML-based MPC: reinforcement learning (RL)-based MPC, while Brunke et al. (2022) explores its safety aspect. While Ren et al. (2022) focuses on a broader category of ML models, neural networks (NNs), providing a tutorial review on their modeling approaches in MPC, there was limited mention of the challenges related to the implementation of ML-based MPC. Bonassi et al. (2022) consolidated recent efforts in the development of RNN-based MPC and discussed issues related to RNN-based MPC in terms of safe verification and interpretability of the RNN model, as well as stability and robustness of RNN-based MPC. Similarly, Berberich and Allgöwer (2024) focus on closed-loop stability and guarantee of ML-based MPC. However, to the best of the authors' knowledge, a comprehensive overview of the challenges faced in the implementation of ML-based MPC has yet to be established. Hence, this review aims to complement the existing literature by providing an extensive review of the

theoretical and practical challenges in ML-based MPC, specifically NN-based MPC, as well as a summary of the current efforts taken to address each of these challenges. The review also presents a dual perspective to approach some of the practical issues, namely from both modeling and control viewpoints.

This article is organized as follows: preliminary knowledge on the class of systems considered and an introduction to neural networks and ML-based MPC is provided in Section 2. In Section 3, the theoretical challenges of ML-based MPC and current advances in characterizing the generalization performance of ML models and analyzing the closed-loop stability of ML-based MPC are reviewed. In Section 4, practical challenges and potential solutions to resolve these issues are discussed. This includes topics such as data scarcity, data quality, the curse of dimensionality, model uncertainty, computational efficiency, and safety concerns of ML-based MPC. In Section 5, novel ML modeling and ML-MPC control methods mentioned in Section 4 are applied to a nonlinear chemical process to demonstrate their effectiveness. Finally, Section 6 concludes with an outlook on the future directions of ML-based MPC.

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. For a given matrix $A \in \mathbf{R}^{m\times n}$, its Frobenius norm is denoted by $\|A\|_F$. The notation $L_f V(x)$ denotes the standard Lie derivative where $L_f V(x) \coloneqq \frac{\partial V(x)}{\partial x^T} f(x)$. Set subtraction is denoted by "\" (i.e., $A \setminus B \coloneqq \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$). $\varnothing$ signifies the null set. The function $f(\cdot)$ is of class $\mathscr{C}^1$ if it is continuously differentiable in its domain. A function $f : \mathbf{R}^n \to \mathbf{R}^m$ is said to be L-Lipschitz, if for all $x, y \in \mathbf{R}^n$, if $|f(x) - f(y)| \le L|x - y|$ where $L > 0$. A continuous function $\alpha : [0, a) \to [0, \infty)$ is said to belong to class $\mathscr{K}$ if it is strictly increasing and is zero only when evaluated at zero. For a random variable X, its expected value is denoted as $\mathbb{E}[X]$. The notation $\mathbb{P}(A)$ represents the probability that an event A is occurring.

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 (ODEs):

$$\dot{x} = F(x, u, d), \; x(t_0) = x_0 \tag{1}$$

where $x \in \mathbf{R}^n$ is the state vector, $u \in \mathbf{R}^m$ is the manipulated input vector, and $d \in D$ is the disturbance vector with $D \coloneqq \{d \in \mathbf{R}^q \mid |d| \le d_m, \; d_m \ge 0\}$. The control input vector u is constrained by the following set $u \in U \coloneqq \{u_{i,\min} \le u_i \le u_{i,\max}, i = 1, \ldots, m\} \subset \mathbf{R}^m$. $F(\cdot, \cdot, \cdot)$ is a sufficiently smooth vector function. Throughout the manuscript, the initial time t_0 is taken to be zero ($t_0 = 0$), and it is assumed that $F(0, 0, 0) = 0$, and thus, the origin is a steady-state of the nominal system (i.e., $w(t) \equiv 0$) of Eq. (1) (i.e., $(x_s^*, u_s^*) = (0, 0)$, where u_s^* and x_s^* represent the steady-state input and state vectors, respectively).

2.3 Supervised learning – neural networks

Machine learning can be divided into four learning types: supervised, unsupervised, semi-supervised, and reinforcement learning. In supervised learning, the dataset S used for model construction is the collection of M labeled data, i.e., $S = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, M\}$. Each data sample $(\mathbf{x}_i, \mathbf{y}_i)$ consists of a feature input vector $\mathbf{x}_i \in \mathbf{R}^{d_x}$ and a labeled/target output $\mathbf{y}_i$, where d_x is the dimension of $\mathbf{x}_i$, i.e., the number of features. If the label $\mathbf{y}_i$ can only take on finitely many values, i.e., the labeled output $\mathbf{y}_i$ is discrete-valued, then the learning task is identified as a classification problem. On the other hand, if the labeled output $\mathbf{y}_i$ is continuous, then this is a regression task. In the context of MPC, where models are required to predict uncountably many values, this constitutes a regression problem. ML models such as linear and nonlinear regression, autoregressive models, state-space models, and neural networks have been widely adopted for MPC applications (Huang and Kadali 2008; Tang and Daoutidis 2022). Neural networks are a subset of machine learning models that consist of layers of interconnected neurons. Neurons are the most fundamental unit of NNs. Given a data sample $(\mathbf{x}_i, \mathbf{y}_i)$, each neuron in the NN acts as a processing unit that first computes the weighted sum of the inputs and the bias term associated with the neuron, that is, $\left(\sum_{j=1}^{d_x} w_j x_{i,j}\right) + b$, where $\mathbf{x_i} = [x_{i,1}, x_{i,2}, \ldots, x_{i,d_x}]$, b is the bias term, and w_j is the weight associated with the j-th feature. The neuron then applies an activation function $f(\cdot)$ to the weighted sum to produce its output $\mathbf{y}_i$. The different arrangements and connections of neurons within the network architecture determine the type of neural network. The formulations of two neural networks commonly used in MPC applications, feedforward neural networks (FNNs) and recurrent neural network (RNNs), will be provided in the following section. Discussions on data generation, model training, and model incorporation into MPC will not be

covered in this review as an in-depth review has been provided in (Ren et al. 2022).

2.3.1 FNN and RNN

The formulation of a one-hidden-layer FNN is provided below, with hidden states $\mathbf{h} \in \mathbf{R}^{d_h}$ computed as follows:

$$\mathbf{h} = \sigma_h(W\mathbf{x} + \mathbf{b}_h) \tag{2}$$

where σ_h is the element-wise nonlinear activation function (e.g., ReLU) and $\mathbf{b}_h \in \mathbf{R}^{d_h}$ is the bias vector of the hidden state. $W \in \mathbf{R}^{d_h \times d_x}$ is the weight matrix connected to the input vector $\mathbf{x}$. The output layer $\mathbf{y}$ is computed as follows:

$$\mathbf{y} = \sigma_y(V\mathbf{h} + \mathbf{b}_y) \tag{3}$$

where $V \in \mathbf{R}^{d_y \times d_h}$ is the weight matrix, $\mathbf{b}_y$ is the bias vector for the output, and σ_y is the element-wise activation function in the output layer (typically linear unit for regression problems). To develop an FNN model for the nonlinear system of Eq. (1) using a training dataset with data collected at every sampling time $t = t_k$, $k = 1, 2, \ldots$, where $t_{k+1} \coloneqq t_k + \Delta$, and Δ represents one sampling period, one can choose the current state $x(t_k)$ and the manipulated input $u(t_k)$ that is applied over $t \in [t_k, t_{k+1})$ as FNN inputs, i.e., $\mathbf{x} = \left[x(t_k)^T \; u(t_k)^T\right]^T$, to predict the output $\mathbf{y} = x(t_{k+1})$ at the next sampling time.

RNNs are a type of neural network that uses sequential data or time-series data. While the hidden layer configuration varies among different NNs, the output layer configuration remains consistent for FNNs and RNNs. The key difference between an FNN and an RNN lies in the direction of information flow. A figure showing the network structures of an FNN and an RNN is presented in Figure 1.

From Figure 1, the flow of information in an FNN is observed to be unidirectional, where information is passed sequentially in the forward fashion through the input layer, hidden layer, and finally to the output layer. On the other hand, RNNs are designed for sequential data where the order of inputs is important. RNNs represent an improvement over FNNs in the sense that RNNs have connections that create loops within the networks. The recurrent structure of the RNNs allows them to retain information from previous time steps which facilitates the capturing of patterns in sequential data. Therefore, RNNs are widely utilized for modeling nonlinear dynamic processes, particularly in applications that require time series predictions where RNNs have demonstrated their efficiency to capture nonlinear behaviors over some time period. To illustrate the difference, we consider a one-hidden-layer RNN. The computation of RNN hidden states $\mathbf{h}_t \in \mathbf{R}^{d_h}$ is slightly different from FNN, with the inclusion of a time factor t, as shown in Eq. (4) below:

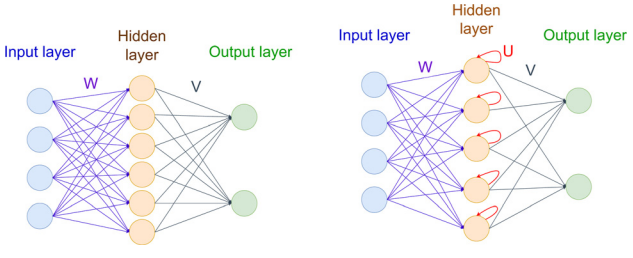

Figure 1: A feedforward neural network (left) and a recurrent neural network (right).

$$\mathbf{h}_t = \sigma_h(U\mathbf{h}_{t-1} + W\mathbf{x}_t + \mathbf{b}_h) \tag{4}$$

where σ_h is the element-wise nonlinear activation function of the hidden layer and $\mathbf{b}_h \in \mathbf{R}^{d_h}$ is the bias vector of the hidden state. $U \in \mathbf{R}^{d_h \times d_h}$ and $W \in \mathbf{R}^{d_h \times d_x}$ are weight matrices connected to the hidden states and the input vector, respectively. From Eq. (4), it can be seen that in addition to using the current input vector $\mathbf{x}_t$ for computation, RNNs also use the hidden state of the previous time step $\mathbf{h}_{t-1}$ in calculating the current hidden state $\mathbf{h}_t$.

The aforementioned configuration constitutes the standard and simplest RNN structure. Since the first introduction of RNNs in the 1980s, many variations of the conventional RNNs have emerged and demonstrated exceptional capabilities in processing time series data. Popular variants of RNNs include the long short-term memory (LSTM) network (Hochreiter and Schmidhuber 1997) and gated recurrent unit (GRU) (Cho et al. 2014). LSTMs and GRUs are gated variants of RNNs that modify the computation of RNN hidden states. A standard LSTM cell uses three gates, namely the forget, input, and output gates, to update its cell and hidden states. The update equations are listed below:

$$\mathbf{f}_t = \sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + \mathbf{b}_f) \tag{5a}$$

$$\mathbf{i}_t = \sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + \mathbf{b}_i) \tag{5b}$$

$$\mathbf{o}_t = \sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{5c}$$

$$\tilde{\mathbf{c}}_t = \tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{5d}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{5e}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{5f}$$

where $\mathbf{h}_t, \mathbf{c}_t \in \mathbf{R}^{d_h}$ represent the hidden state and cell state vectors, respectively, with initial values $\mathbf{h}_0 = \mathbf{c}_0 = 0$. $\mathbf{x}_t \in \mathbf{R}^{d_x}$ is the input feature vector and $\odot$ denotes the Hadamard product operator. Equations (5a)–(5c) define the gate functions. Specifically, $\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t \in \mathbf{R}^{d_h}$ represent the forget, input, and output gates at time t, respectively, with their associated

bias terms $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o \in \mathbf{R}^{d_h}$. The gates use element-wise nonlinear activation functions (e.g., sigmoid function $\sigma(\cdot)$ and $\tanh(\cdot)$). The weight matrices $W_f, W_i, W_o, W_c, \in \mathbf{R}^{d_h \times d_x}$ and $U_f, U_i, U_o, U_c \in \mathbf{R}^{d_h \times d_h}$ are used to connect the input vector and the hidden states to the different gates, respectively. The term $\tilde{\mathbf{c}}_t$ represents the candidate cell state in the LSTM cell. It serves as an intermediate state that reflects the potential information used to update the cell state $\mathbf{c}_t$, modulated by the input and forget gates in Eq. (5e).

Compared to LSTMs, GRUs have a more simplified structure with only two gates, the update $\mathbf{r}_t$ and reset gates $\mathbf{z}_t$. The update equations of GRU are listed as follows:

$$\mathbf{r}_t = \sigma(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1} + \mathbf{b}_r) \tag{6a}$$

$$\mathbf{z}_t = \sigma(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{6b}$$

$$\tilde{\mathbf{h}}_t = \tanh(W_h \mathbf{x}_t + U_h (\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h) \tag{6c}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{6d}$$

with weight matrices $W_r, W_r, W_h \in \mathbf{R}^{d_h \times d_x}$, $U_r, U_z, U_h \in \mathbf{R}^{d_h \times d_h}$ and bias parameters $\mathbf{b}_r, \mathbf{b}_z, \mathbf{b}_h \in \mathbf{R}^{d_h}$. The term $\tilde{\mathbf{h}}_t$ represents the candidate hidden state which is used to compute the GRU hidden state $\mathbf{h}_t$ in Eq. (6d). A summary of the differences in the network structures of a simple RNN, an LSTM, and a GRU is provided in Figure 2.

While RNNs, LSTMs, and GRUs are all types of neural networks used for processing sequential data, they have different structures and mechanisms to handle the vanishing gradient problem and maintain long-term dependencies. In general, RNNs are suitable for simpler tasks, LSTMs can be used for tasks that require long-term memory, and GRUs may achieve a balance of efficiency and performance. We will primarily focus on simple RNNs when addressing theoretical and practical challenges in the following sections. However, it should be noted that the methods discussed in this manuscript can also be readily applied to other types of RNNs, such as LSTMs and GRUs.

2.4 ML-based MPC

To simplify the notation for MPC using RNN models, we represent the RNN model in the following continuous-time form for the nominal system of Eq. (1) (i.e., $w(t) \equiv 0$):

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u) \tag{7}$$

where $\hat{x} \in \mathbf{R}^n$ and $u \in \mathbf{R}^m$ are the state vector predicted by the RNN model and the manipulated input vector, respectively. Note that neural networks are generally developed as discrete-time models with sampled training data. While there are some methods such as using differential equations to model the evolution of the hidden state, or using neural ordinary differential equations (neural ODEs) (Chen et al. 2018a), that directly model continuous dynamics in neural networks, the continuous-time representation of RNNs adopted in this work is primarily to simplify notation. In other words, when incorporated into MPC, the RNN model is used to predict states at discrete future time steps, rather than generating a continuous-time state trajectory. Consequently, the objective function and constraints in the MPC framework will be based solely on the states predicted by the RNN models.

A general tracking model predictive control design is given by the following optimization problem:

$$\mathscr{J} = \min_{u} \int_{t_k}^{t_{k+N}} L\Big(\tilde{x}(t), u(t)\Big) \mathrm{d}t \tag{8a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}\Big(\tilde{x}(t), u(t)\Big) \tag{8b}$$

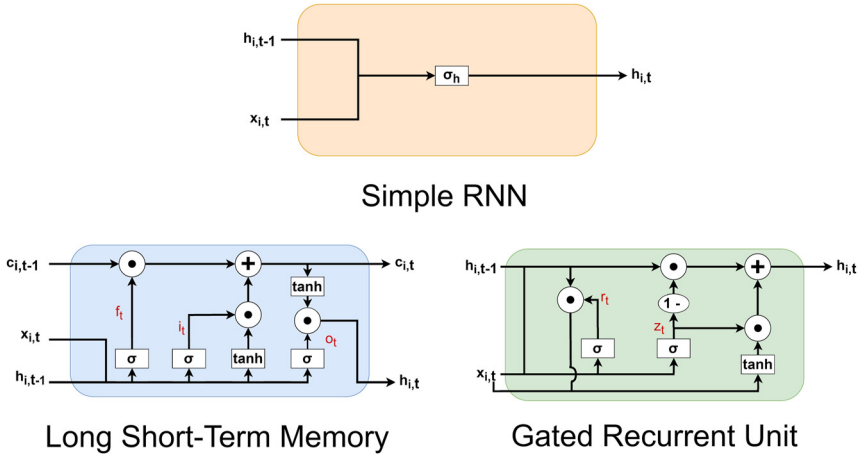

Figure 2: Network architecture of a simple recurrent neural network cell, a long short-term memory and a gated recurrent unit.

$$\tilde{x}(t_k) = x(t_k) \tag{8c}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{8d}$$

$$x(t) \in X, \ \forall \ t \in [t_k, t_{k+N}) \tag{8e}$$

where the objective function seeks to minimize the integral of the cost function $L(\tilde{x}(t), u(t))$ over the prediction horizon $t \in [t_k, t_{k+N})$ by optimizing the manipulated input u. The term N represents the number of sampling periods in the prediction horizon. $\tilde{x}(t)$ is the predicted state trajectory by the model, using the state measurement at t_k as its initial condition. Equations (8d) and (8e) are the constraints imposed on the input and state variables, respectively, where U and X represent the sets of feasible u and x.

As it is a well-known fact that the MPC formulation given by Eq. (8) is not always stabilizing, several approaches have been proposed in the literature to achieve closed-loop stability. One approach involves using infinite prediction horizons or carefully designed terminal penalty terms; for a comprehensive review of these methods, see Bitmead et al. (1990) and Mayne et al. (2000). Another approach is to enforce stability constraints directly within the MPC optimization problem (Chen and Allgöwer 1998; Mayne et al. 2000). Ensuring closed-loop stability under MPC with terminal constraints extends standard MPC by incorporating additional constraints on the system's state at the end of the prediction horizon. Proper design of terminal constraints and terminal regions is crucial to ensure that the system reaches a desired or safe operating point within a finite time horizon.

Another important approach in stabilizing MPC is Lyapunov-based MPC (LMPC). LMPC provides an explicit characterization of the stability region and guarantees controller feasibility and closed-loop stability. In the context of predictive control for the system of Eq. (1), the LMPC is designed based on an existing explicit control law $\Phi_{nn}(x)$ that is capable of stabilizing the closed-loop system. The formulation of ML-based LMPC is as follows (Wu et al. 2019c,d):

$$\mathscr{J} = \min_{u^* \in S(\Delta)} \int_{t_k}^{t_{k+N}} L\Big(\tilde{x}(t), u(t)\Big) \mathrm{dt} \tag{9a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{9b}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{9c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{9d}$$

$$\dot{\hat{V}}(x(t_k), u) \leq \dot{\hat{V}}(x(t_k), \Phi_{nn}(x(t_k))), \ \text{if} \ x(t_k) \in \Omega_\rho \setminus \Omega_{\rho_{nn}} \tag{9e}$$

$$\dot{\hat{V}}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+N}), \ \text{if} \ x(t_k) \in \Omega_{\rho_{nn}} \tag{9f}$$

where $\tilde{x}$ is the predicted state trajectory, $S(\Delta)$ is the set of piecewise constant functions with period Δ, and N is the number of sampling periods in the prediction horizon. $\dot{\hat{V}}(x, u)$ is used to represent $\frac{\partial \hat{V}(x)}{\partial x}(F_{nn}(x, u))$. The optimal input trajectory computed by the LMPC is denoted by $u^*(t)$, which is calculated over the entire prediction horizon $t \in [t_k, t_{k+N})$. The control action computed for the first sampling period of the prediction horizon $u^*(t_k)$ is sent by the LMPC to be applied over the first sampling period, and the LMPC is resolved at the next sampling time. The set Ω_ρ is the closed-loop stability region for the nonlinear system of Eq. (1) defined as a level set of the Lyapunov function $V(x)$, that is $\Omega_\rho := \{x \in \mathbf{R}^n \mid V(x) \leq \rho\}$. The term $\Omega_{\rho_{nn}}$ denotes a small level set of the Lyapunov function $V(x)$ around the origin within which the system is considered practically stable as long as the states x remain inside this region.

In the optimization problem of Eq. (9), the objective function of Eq. (9a) is the integral of $L(\tilde{x}(t), u(t))$ over the prediction horizon. The constraint of Eq. (9b) is the RNN model of Eq. (4) that is used to predict the states of the closed-loop system. Equation (9c) defines the input constraints applied throughout the prediction horizon. Equation (9d) defines the initial condition $\tilde{x}(t_k)$ of Eq. (9b), which is the state measurement at $t = t_k$. The constraint of Eq. (9e) forces the closed-loop state to move towards the origin if $x(t_k) \in \Omega_\rho \setminus \Omega_{\rho_{nn}}$. However, if $x(t_k)$ enters $\Omega_{\rho_{nn}}$, the states predicted by the RNN model of Eq. (9b) will be maintained in $\Omega_{\rho_{nn}}$ for the entire prediction horizon.

While ML-based MPC offers promising opportunities for addressing complex control problems by using data-driven models, several challenges need to be addressed to ensure the effectiveness and reliability of ML-based MPC approaches. In this review paper, we will discuss the state-of-the-art technologies that tackle the emerging challenges in ML-MPC.

Remark 1. In addition to the general set-point tracking MPC, machine learning models can also be used in zone-tracking MPC (Ferramosca et al. 2010; González and Odloak 2009). Unlike MPC that tracks the set-point, zone-tracking MPC allows the system to operate within a predefined region. It has been reported that a single NN model may not be sufficient to fully capture the system dynamics over the entire operating region consisting of multiple operating points, especially for complex nonlinear processes (Huang et al. 2023; Wu et al. 2019d). The recent work of Huang et al. (2023) divided the operating region into three overlapping sub-regions and trained separate LSTM models, which were combined in the first layer of a neural network. This two-layer NN, integrated into a zone-tracking MPC system, improved open-loop prediction and provided feedback control for irrigation.

3 Theoretical challenges in ML-MPC

The stability of MPC is a fundamental consideration in its application. ML models trained on historical data may struggle to generalize to new or unseen operating conditions, leading to poor performance and stability issues in MPC. Developing a better understanding and techniques for improving the generalization of ML models is a key challenge in ML-based MPC.

3.1 Generalization performance

Early work in ML-MPC often assumes that the model-plant mismatch is bounded, and therefore, the closed-loop stability of MPC holds through the robust design of MPC. However, this assumption may not be true for practical ML models. The generalization error in machine learning refers to the expected error of a model on unseen data drawn from the same distribution as the training data. In other words, it measures how well a trained model performs on new, unseen data points. Generalization error is a critical concept in machine learning because the ultimate goal is to build models that can make accurate predictions on data that they have not seen during training. Initial research on the generalizability of ML models was developed using Vapnik–Chervonenkis (VC) dimension, a method that characterizes the capacity and complexity of models (Sontag 1998b; Vapnik et al. 1994). However, due to the simplified assumptions underlying the VC dimension approach, the derived generalization error bounds can be overly conservative (Chen et al. 2019). Thus, alternatives such as probably approximately correct (PAC) – Bayesian method (Neyshabur et al. 2017) and empirical Rademacher complexity approach under the PAC Learning framework (Bartlett et al. 2017) were introduced and adopted in recent years. Numerous studies have analyzed the generalizability of RNNs, predominantly focusing on their performance in classification tasks (Chen et al. 2019; Sontag 1998a; Wei and Ma 2019; Zhang et al. 2018). Recent works by Akpinar et al. (2019) and Wu et al. (2021b), have proposed PAC analysis frameworks to derive generalization error bounds for RNNs in regression problems. For demonstration purposes, this review will follow the empirical Rademacher complexity approach presented in Wu et al. (2021b) to derive the generalization error bound of RNNs. Interested readers may refer to works by Koiran and Sontag (1998) and Sontag (1998a) for the VC dimension method, Zhang et al. (2018) for the PAC-Bayesian approach, and Akpinar et al. (2019) for the traditional PAC framework in the derivation of the generalization error bound for RNNs.

3.1.1 Generalization error

Given a single-layer RNN model trained with M data samples, where each sample has a time sequence length of T, its input and output are denoted as $\mathbf{x}_{i,t} \in \mathbf{R}^{d_x}$ and $\mathbf{y}_{i,t} \in \mathbf{R}^{d_y}$ respectively, where $i = 1, 2, \ldots, M$ and $t = 1, 2, \ldots, T$. This RNN model is designed to predict the states over the next T time steps, based on past or current state measurements and any manipulated inputs. This approach is analogous to solving a nonlinear ODE (e.g., Eq. (1)) given the initial condition of x and the manipulated input u that will be applied. In the derivation of the generalization error bound, we assume negligible bias terms in the RNN model. The revised equations of the hidden layer and the output layer are provided below:

$$\mathbf{h}_{i,t} = \sigma_h(U\mathbf{h}_{i,t-1} + W\mathbf{x}_{i,t}) \tag{10}$$

$$\mathbf{y}_{i,t} = \sigma_y(V\mathbf{h}_{i,t}) \tag{11}$$

The loss function is denoted as $L(\check{\mathbf{y}}, \mathbf{y})$, where $\check{\mathbf{y}}$ is the predicted RNN output and $\mathbf{y}$ is the true/labeled output. The following assumptions are made on the RNN model and dataset.

Assumption 1. All inputs into the RNN model are bounded, that is, for all $i = 1, \ldots, M$ and $t = 1, \ldots, T$, $|\mathbf{x}_{i,t}| \le A_X$.

Assumption 2. The Frobenius norms of all the weight matrices are bounded in the following manner:

$$\|U\|_F \le A_{U,F}, \|V\|_F \le A_{V,F}, \|W\|_F \le A_{W,F}$$

Assumption 3. The nonlinear activation σ_h is positive homogeneous and 1-Lipschitz continuous, i.e., for all $\gamma \ge 0$, $x \in \mathbf{R}$, we have $\sigma_h(\gamma x) = \gamma\sigma_h(x)$.

Assumption 4. Data samples used for training, validation, and testing purposes are drawn from the same distribution.

All the assumptions made adhere to common practice in machine learning theory. Specifically, the first two assumptions assume the boundedness of RNN inputs and weights, a condition typically satisfied in many modeling tasks where a finite class of neural network hypotheses is used to model nonlinear systems based on data collected from a finite set. The third assumption can be satisfied by activation functions such as ReLU and its variants. It is used for the derivation of generalization error in this section, and can be omitted when using other proof techniques, as demonstrated in Golowich et al. (2018). The last assumption is a fundamental and necessary condition for analyzing generalization performance,

which is adopted in many machine learning works that consider the application of machine learning models to the same process without disturbances or model uncertainties. However, in the presence of disturbances that cause variations in process dynamics over time, the generalization error can still be derived for machine learning models by accounting for the drift in distribution. We will discuss this further when introducing online machine learning in Section 4.4.1.

The following text entails the essential definitions and lemmas widely used within the theoretical framework of machine learning.

Definition 1. The **expected loss/error** or **generalization error** of a function f which predicts output values y for each given input x, with an underlying distribution Q, is given as:

$$L_Q(f) \triangleq \mathbb{E}[L(f(x),y)] = \int_{X\times Y} L(f(x),y)P(x,y)\mathrm{d}x\mathrm{d}y \tag{12}$$

where the vector spaces of all possible inputs and outputs are denoted by X and Y, respectively, and the term $P(x,y)$ represents the joint probability distribution for x and y.

However, since the joint probability distribution P is often unknown, the empirical error, calculated from the data samples, is used as an estimate of the expected loss.

Definition 2. The **empirical error** or **risk** of a dataset with M data samples $S = (s_1, \ldots, s_M)$, where $s_i = (x_i, y_i)$, is defined as:

$$\widehat{\mathbb{E}}_S[L(f(x),y)] = \frac{1}{M}\sum_{i=1}^{M} L(f(x_i),y_i) \tag{13}$$

In order to ensure that the RNN model can capture the nonlinear dynamics of the system of Eq. (1) and generalize well to unseen operating conditions, it is essential to show that the generalization error $\mathbb{E}[L(f(\mathbf{x}),\mathbf{y})]$ can be bounded. Since the empirical error is used as a proxy measure of the generalization error, it is necessary that the empirical error be sufficiently small and bounded such that the generalization error can be bounded. The empirical error can also be viewed as the loss associated with the training dataset. As the training process of ML models is designed to reduce the models' training loss, it is achievable to obtain a sufficiently small and bounded empirical error. The subsequent text will outline the steps taken to derive the upper bound of the generalization error.

In this study, the loss function used is the mean squared error (MSE). While it can be readily demonstrated that the MSE loss function $L(\mathbf{y},\bar{\mathbf{y}})$ is not Lipschitz continuous for every $\mathbf{y},\bar{\mathbf{y}} \in \mathbf{R}^{d_y}$, we can prove that the MSE loss function is locally Lipschitz continuous for $\mathbf{y},\bar{\mathbf{y}}$ within a compact set in $\mathbf{R}^{d_y}$. As a finite hypothesis class that fulfills Assumptions 1–4 was considered, the RNN output can be proven to be bounded. This aligns with the fact that the nonlinear system described in Eq. (1) operates in the stability region Ω_ρ, thus ensuring that the RNN outputs are bounded within a compact set. Thus, the upper bound of $\mathbf{y}_t$ is denoted by $r_t > 0$, that is, $|\mathbf{y}_t| \le r_t$, $t = 1, \ldots, T$. Without loss of generality, it is assumed that the true outputs are also constrained by r_t. Since the RNN outputs and the true outputs are bounded, i.e., for all $|\mathbf{y}_t|, |\bar{\mathbf{y}}_t| \le r_t$, we can show that the MSE loss function is a locally Lipschitz continuous function satisfying the following inequality.

$$|L(\mathbf{y}_1,\bar{\mathbf{y}}) - L(\mathbf{y}_2,\bar{\mathbf{y}})| \le L_r|\mathbf{y}_1 - \mathbf{y}_2| \tag{14}$$

where L_r is the Lipschitz constant. Since Lipschitz conditions occur several times throughout this article for different functions, to clarify, the definition of an L-Lipschitz function in Section 2.1 (Notation) refers to the global Lipschitz condition. However, when we assume that the mean-squared-error loss function is Lipschitz continuous, we are referring to the local Lipschitz condition. This is because the neural network's input and output data are drawn from a compact set in the state space.

Further analysis shows that the generalization error can be perceived as a combination of the approximation and the estimation error. The breakdown of the generalization error of a given neural network function f_S taken from a hypothesis class $\mathcal{F}$, trained with a specific learning algorithm using a training dataset S sampled from distribution Q, is as follows:

$$\begin{aligned} L_Q(f_S) - L_Q(f^*) = {} & (\min_{f\in\mathcal{F}} L_Q(f) - L_Q(f^*)) + (L_Q(f_S) \\ & - \min_{f\in\mathcal{F}} L_Q(f)) \end{aligned} \tag{15}$$

where $L_Q(f_S)$ is the generalization error of the function f_S on the underlying data distribution Q. The term f^* represents the global optimal hypothesis that yields the lowest generalization error, for the data distribution Q; this hypothesis could be outside of the finite hypothesis class $\mathcal{F}$. The term $\min_{f\in\mathcal{F}} L_Q(f)$ searches for the optimal hypothesis within $\mathcal{F}$ that minimizes loss functions over the distribution Q. Thus, $\min_{f\in\mathcal{F}} L_Q(f)$ can be seen as the generalization error of the local optimal hypothesis. The term $L_Q(f_S) - L_Q(f^*)$, measures the generalization gap between the selected hypothesis f_S and the global optimal hypothesis f^*. This difference can be decomposed into the **approximation error** and **estimation error**, represented by the first and second parenthesized terms, respectively.

The approximation error can be thought of as how far the local optimal $\min_{f\in\mathcal{F}} L_Q(f)$ deviates from the global optimal $L_Q(f^*)$. This provides insights into how close the hypothesis class $\mathcal{F}$ is to the global optimal hypothesis f^*. In

general, the likelihood of the optimal hypothesis f^* being in the hypothesis class $\mathcal{F}$ is greater for a larger hypothesis class. Hence, larger hypothesis classes tend to have a smaller approximation error. On the other hand, the estimation error compares the performance of the candidate hypothesis f_S, trained with training dataset S, with the best hypothesis within the hypothesis class, $\min_{f\in\mathcal{F}} L_Q(f)$. Thus, the estimation error is dependent on both hypothesis class and training data. In general, an increase in the size of the hypothesis class $\mathcal{F}$ could result in a higher estimation error, as it may be more challenging to search for the optimal hypothesis in a larger hypothesis class. The error decomposition analysis in Eq. (15) illustrates how the generalization error is influenced by the size of the training dataset and the complexity of the hypothesis class. Thus, the next segment will explore methods to quantify the effect of these factors on the generalization error.

3.1.2 Upper bound for generalization error

Generalization error is a measure of how well a model generalizes from the training data to unseen data. While it is typically estimated using a separate validation dataset or through techniques such as cross-validation, deriving a theoretical understanding is also important as it can help improve the architecture and training of ML models to achieve the desired generalization performance. The computer science community has made great efforts to derive an upper bound for the generalization error of various neural network models. Specifically, the following lemma characterizes the upper bound of the generalization error using the Rademacher complexity $\mathscr{R}_S(\cdot)$, which quantifies the richness of a class of functions, and is often used in machine learning theory to bound the generalization error.

Definition 3. (Empirical Rademacher Complexity) The empirical Rademacher complexity of a given hypothesis class $\mathscr{K}$ of real-valued functions, trained with a set of data samples $S = \{s_1, \ldots, s_M\}$, is defined as

$$\mathscr{R}_S(\mathscr{K}) = \mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{k\in\mathcal{K}} \frac{1}{M}\sum_{i=1}^{M} \epsilon_i k(s_i)\right] \tag{16}$$

where $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_M)^T$ with ϵ_i being independent and identically distributed (i.i.d.) Rademacher random variables satisfying $\mathbb{P}(\epsilon_i = 1) = \mathbb{P}(\epsilon_i = -1) = 0.5$.

Lemma 1 . (c.f. Theorem 3.3 in Mohri et al. (2018)) Consider a class of loss functions $\mathscr{G}_t$ ($\mathscr{G}_t = \{g_t : (\mathbf{x}, \mathbf{y}) \to L(f(\mathbf{x}), \mathbf{y})\}$) associated to the hypothesis class $\mathcal{F}_t$ of vector-valued functions $f(\cdot)$ that map the RNN inputs to the RNN output at t-th time step, and trained with M i.i.d. data samples; the following inequality holds for all $g_t \in \mathscr{G}_t$, with probability at least $1-\delta$ over samples $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$, $i = 1, \ldots, M$

$$\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] \le \frac{1}{M}\sum_{i=1}^{M} g_t(\mathbf{x}_i, \mathbf{y}_i) + 2\mathscr{R}_S(\mathscr{G}_t) + 3\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2M}} \tag{17}$$

It can be seen from Eq. (17) that the generalization error bound depends on the empirical error (the first term), Rademacher complexity (the second term), and an error function associated with confidence δ and the number of samples M (the last term). Since the first and last terms are known given a set of M training data, in order to characterize the upper bound for the generalization error $\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})]$, we need to determine the upper bound for the Rademacher complexity $\mathscr{R}_S(\mathscr{G}_t)$.

Intuitively, the Rademacher complexity measures the maximum correlation between functions in the hypothesis class $\mathcal{F}$ and random noise. A smaller Rademacher complexity indicates that the hypothesis class is less likely to fit random noise and therefore may generalize better to unseen data. The following error bound was derived for the generalization error of the RNN of Eqs. (10) and (11).

Theorem 1 . (c.f. Theorem 1 in Wu et al. (2021b)) Given an i.i.d training samples of size M, $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$, $i = 1, \ldots, M$ and an RNN model that satisfy Assumptions 1–4, the following inequality holds for $\mathscr{G}_t$, the family of loss function associated to the hypothesis class $\mathcal{F}_t$ of vector-valued functions that map the RNN inputs to the RNN output at t-th time step, with probability at least $1-\delta$ over S:

$$\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] \le \mathscr{O}\left(L_r d_y \frac{H(\sqrt{2\log(2)t}+1)A_X}{\sqrt{M}}\right) + \frac{1}{M}\sum_{i=1}^{M} g_t(\mathbf{x}_i, \mathbf{y}_i) + 3\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2M}} \tag{18}$$

where $H = A_{V,F}A_{W,F}\frac{(A_{U,F})^t - 1}{A_{U,F} - 1}$.

Remark 2. The generalization error bound of Eq. (18) implies that the following attempts can be taken to reduce the generalization error: (1) minimize the empirical loss $\frac{1}{M}\sum_{i=1}^{M} g_t(\mathbf{x}_i, \mathbf{y}_i)$ over the training data samples S through a careful design of neural network, and (2) increase the number of training samples M. Additionally, as discussed in the error decomposition of Eq. (15), increasing the complexity hypothesis class in terms of larger weight matrices bounds M could decrease the approximation error, but may also increase the estimation error, which corresponds to the term $\mathscr{O}(\cdot)$ in Eq. (18). This is consistent with the

analysis of the trade-off between approximation error and estimation error in Eq. (15). Therefore, in practice, we generally start with a simple neural network and gradually increase its complexity in terms of more neurons, layers and larger weight matrices bounds to improve the training and testing performance. The whole process stops when the testing error starts increasing, which indicates the occurrence of overfitting.

Note that for neural networks with different architectures (e.g., types of NNs, number of layers and neurons, activation functions, etc.), we will have different values for Rademacher complexity. For instance, Golowich et al. (2018) derived the Rademacher complexity upper bounds for a multi-layer FNN (see Eq. (19)).

$$\mathscr{O}\left(\frac{B_X\left(\sqrt{2\log(2)\eta}+1\right)\prod_{j=1}^{d}B_{j,F}}{\sqrt{M}}\right) \tag{19}$$

The FNN has η hidden layers in total, the Frobenius norm of the weight matrices of the j-th hidden layer is bounded by $B_{j,F}$. Similar to the RNN, the FNN is trained using a dataset of size M and its input $\mathbf{x}$ is bounded by B_X, i.e., $|\mathbf{x}| \le B_X$.

3.2 Closed-loop stability

Closed-loop stability in MPC is important for ensuring the safety and reliability of plant operations, and some recent efforts have been made to investigate the stability of ML-based MPC (Limon et al. 2017; Meng et al. 2022; Soloperto et al. 2022; Wu et al. 2019c, 2021b). This section will explore and analyze the closed-loop stability of ML-based MPC based on the LMPC formulation of Eq. (9). Specifically, in the LMPC of Eq. (9), the constraint of Eq. (9e) ensures that the time derivative of the Lyapunov function $V(x)$, at time t_k remains less than or equal to the value it would attain if the nonlinear control law $u = \Phi_{nn}(x)$ is applied in a sample-and-hold manner within the closed-loop system. This constraint allows us to demonstrate (given state measurements at synchronous sampling times) that LMPC inherits the stability and robustness properties of the nonlinear control law $\Phi_{nn}(x)$. The stability characteristic of LMPC is directly inherited from the stability of the nonlinear control law $\Phi_{nn}(x)$ when applied in a sample-and-hold manner.

Additionally, the feasibility of LMPC is inherently guaranteed by the nonlinear control law $\Phi_{nn}(x)$, since it is a feasible solution to the optimization problem of Eq. (9). Detailed results on this aspect for MPCs using first-principles model can be found in Mahmood and Mhaskar (2008) and Mhaskar et al. (2006) (note that the results in these papers can be readily applied to neural-network-based MPC provided that a stabilizing controller $\Phi_{nn}(x)$ can be found). One of the primary advantages of the LMPC approach over the nonlinear control law $\Phi_{nn}(x)$ is its ability to explicitly incorporate optimality considerations, as well as constraints on inputs and states within an online optimization framework. This approach improves the closed-loop performance of the system. Furthermore, since the closed-loop stability and feasibility of LMPC are guaranteed by $\Phi_{nn}(x)$, there is no need to introduce a terminal penalty term in the cost function. Additionally, while the horizon length N affects the closed-loop performance, it does not impact the stability of the closed-loop system.

A key step for closed-loop stability under MPC is to ensure that the discrepancy between NN predictions and the actual state evolution is bounded. If we consider a deterministic error bound, that is, the error between NN predictions and the true evolution of states is bounded for all times, the ML-MPC of Eq. (9) guarantees closed-loop stability by designing the nonlinear control law $\Phi_{nn}(x)$ appropriately to render the steady-state stable in the presence of the worst-case scenario where the prediction error reaches its bound at all times. However, since in reality, the generalization errors of any neural networks developed using supervised learning methods are bound in some probability (e.g., the error bound in Eq. (18)), closed-loop stability under ML-MPC is actually guaranteed in a probabilistic sense. This implies that closed-loop stability is guaranteed with a certain probability for each time step. In other words, there exists a small probability that stability may not hold if the prediction error exceeds the theoretical bound. Additionally, although stability cannot be guaranteed due to the probabilistic nature of its prediction error, it should be noted that the actual probability of closed-loop stability for each time step could be higher than the lower bound $1-\delta$ for many reasons. For instance, (1) if the RNN model is well trained and the modeling error remains significantly below its upper bound, and (2) if the next state remains within the stability region even when the modeling error surpasses its upper limit in a single sampling period, then the probability. Therefore, the theoretical error bound of Eq. (18) serves as a conservative estimate of the probability of maintaining closed-loop stability, and can be used to guide the construction of network architecture and selection of sample size.

4 Practical challenges in applications of ML-MPC

In addition to the theoretical understanding of the generalization performance of ML models and the resulting

closed-loop stability properties, there exist many other practical challenges for the implementation of ML-based MPC systems in real-world systems.

4.1 Data scarcity

The quantity and quality of the data used for model development are paramount for the performance and accuracy of ML models. In Section 3.1, we see that an increase in the number of training samples is helpful in reducing the generalization error of the model, thereby improving its accuracy. However, in practice, it can be difficult to gather a substantial amount of data samples to meet the requirements of developing an accurate ML model. This is especially true for complex systems with a large number of feature variables, where data collection can be costly and limited. Hence, this section presents an overview of some popular techniques to address data scarcity in machine learning.

4.1.1 Physics-informed machine learning

Physics-informed machine learning (PIML) is an emerging ML technique that seeks to improve the accuracy, robustness, interpretability, and physical consistency of ML models by integrating physics laws and domain knowledge into the learning process (Karniadakis et al. 2021). According to Karniadakis et al. (2021), physics can be incorporated into ML models in three ways: (1) through observational data that reflect the fundamental physics laws, (2) by implicitly integrating domain knowledge into ML models by customizing the model architecture, and (3) by explicitly embedding physics into the model algorithm, typically through additional loss functions and constraints. In particular, physics-informed neural networks (PINNs) are a prominent subclass of PIML that have gained traction in recent times due to their ability to learn effectively in small data regimes (Raissi et al. 2019; Zheng et al. 2023). Since their advent, PINNs have been extensively applied in engineering, ranging from the modeling of chemical/biochemical processes (Rogers et al. 2023; Subraveti et al. 2022) and reactors (Bangi et al. 2022; Patel et al. 2023; Wu et al. 2024) to process control (Antonelo et al. 2024; Arnold and King 2021; Wang and Wu 2024b; Zheng et al. 2023).

PINNs operate by integrating physical laws directly into the learning process, where physics, in the form of ordinary or partial differential equations (ODEs/PDEs), is embedded into the loss function. This can be illustrated using the formulation of a physics-informed RNN (PIRNN) model provided in Zheng et al. (2023). Given a PIRNN that uses the current state of a system $x(t_k)$ and manipulated input $u(t_k)$ to predict the evolution of the state of the system over a period of time, that is, $x(t) \ \forall\, t \in [t_k, t_k + \Delta]$, where Δ represents one sampling period. The state trajectory $x(t)$ consists of multiple internal states, i.e., the collection of states between t_k and $t_k + \Delta$ separated by a fixed time interval τ (τ can be considered as the smallest time interval for which sensor measurements are available). The loss function of PIRNN is defined as follows:

$$\text{Loss} = \alpha_{\mathscr{X}} \text{Loss}_{\mathscr{X}} + \alpha_{\mathscr{G}} \text{Loss}_{\mathscr{G}} \tag{20a}$$

where

$$\text{Loss}_{\mathscr{X}} = \frac{1}{N_{\mathscr{X}}} \sum_{n=1}^{N_{\mathscr{X}}} \frac{1}{N_T} \sum_{i=0}^{N_T} |x_n(t_i) - \tilde{x}_n(t_i)|^2 \tag{20b}$$

$$\begin{aligned} \text{Loss}_{\mathscr{G}} = \ & \gamma \frac{1}{N_{\mathscr{G}}} \sum_{n=1}^{N_{\mathscr{G}}} |x_n(t_0) - \tilde{x}_n(t_0)|^2 \\ & + \frac{1}{N_{\mathscr{G}}} \sum_{n=1}^{N_{\mathscr{G}}} \frac{1}{N_T} \sum_{i=0}^{N_T} |\mathscr{G}(\tilde{x}_n(t_i), u_n)|^2 \end{aligned} \tag{20c}$$

$N_{\mathscr{X}}$ is the number of training data, i.e., the number of dynamic state trajectories of Eq. (1) used for training. $N_{\mathscr{G}}$ is the number of collocation points (i.e., initial conditions for the RNN model introduced in this article) for each sampling time Δ, where the initial condition encompasses only the initial state and the manipulated inputs. N_T is the number of internal states of PIRNN within one sampling period Δ.

The loss term consists of two terms, where the subscripts $\mathscr{X}$ and $\mathscr{G}$ are used to represent the loss terms with respect to the standard supervised loss and the physics-driven loss, respectively. The first loss term $\text{Loss}_{\mathscr{X}}$ in Eq. (20b) measures the MSE between the predicted output $\tilde{x}_n(t_i)$ and the labeled output $x_n(t_i)$. The second loss term $\text{Loss}_{\mathscr{G}}$ represents the regularization term driven by physical laws. The first term of Eq. (20c) represents a supervised loss term at the initial conditions for each sampling period Δ and the second term represents an unsupervised loss term of the governing equations, where $\mathscr{G}$ is defined as $\mathscr{G}(\tilde{x}, u) \coloneqq \dot{\tilde{x}} - F(\tilde{x}, u)$. The terms $\alpha_{\mathscr{X}}$, $\alpha_{\mathscr{G}}$ and γ denote the weight coefficients that adjust the magnitude of different loss terms. Generally, these weights are user-defined to regulate the interaction between different contributing loss components and to assist the training of PIRNN.

The computation of the ODE residual $\mathscr{G}(x, u) = \dot{\tilde{x}} - F(\tilde{x}, u)$ is divided into three steps. Firstly, the finite difference method is used to approximate the value of $\dot{\tilde{x}}$ using the predicted state trajectory by PIRNN. Secondly, the ODE, i.e., the function $F(\tilde{x}, u)$, is calculated by substituting the RNN predicted

states $\tilde{x}$ and the manipulated inputs u into $F(\tilde{x}, u)$. Finally, the ODE residual can be calculated by subtracting $F(\tilde{x}, u)$ from $\dot{\tilde{x}}$. Specifically, given an initial condition, we do not need labeled output data (i.e., dynamic state trajectory from the training data starting at that initial condition) to evaluate $\text{Loss}_{\mathcal{G}}$. In particular, one could take a set of initial points (possibly random) in the domain of consideration where there is no labeled output (i.e., state trajectories), use the PIRNN to predict the trajectory forward with that starting point $(\tilde{x}_n(t, u))$, and evaluate the physics-driven loss $\text{Loss}_{\mathcal{G}}$ directly using the physics-informed knowledge of the first-principles model $F(x, u)$. It is also noted that in the extreme case with no labeled output data used as training data (hence $N_{\mathcal{X}} = 0$), the PIRNN learns the dynamics using only the loss function $\text{Loss}_{\mathcal{G}}$. A summary of the PIRNN training process is provided in Figure 3, interested reader may refer to Zheng et al. (2023) for details.

A forward problem is a problem that involves solving for the outputs of a system when the inputs and governing equations are known. On the other hand, an inverse problem seeks to infer unknown inputs or parameters from the observed outputs by working backward. While the discussion above introduced and explored the applications of PINNs to forward problems, PINNs have also been extensively applied to inverse problems. For example, PINNs demonstrated remarkable accuracy when used to derive velocity and pressure fields from fluid flow images, such as temperature gradient maps that depict the flow in an espresso cup (Cai et al. 2021; Raissi et al. 2020).

Following the study of a generalization error bound for purely data-driven RNN models, recent efforts have been made to analyze the generalization performance of PINNs. For example, in Mishra and Molinaro (2022, 2023), the generalization error analysis for a general class of PINNs approximating solutions of the forward and inverse problems for PDEs was performed, respectively. Furthermore, in Zheng et al. (2023), the results of generalization errors were developed specifically for PIRNNs. As PINNs were developed from domain knowledge, such as the first-principles model, the accuracy of these theoretical models may affect the generalization performance of PINNs. The limitations of the theoretical models can be addressed by applying the PINNs in an inverse manner, using observed data. For example, Zheng and Wu (2023) proposed an inverse problem of PIRNN to improve the first-principles model used in the loss function of PIRNN using real-time data.

In addition, different types of domain knowledge can be incorporated into the design of PINNs. For example, for systems with physical constraints on states and/or inputs, additional loss terms can be included in the loss function to enforce the constraints on the relevant states/inputs. The following loss term is an example that imposes non-negative constraints on the states (Wu et al. 2023a):

$$\text{Loss}_{\text{ReLU}} = \beta \frac{1}{N_{\mathcal{G}}} \sum_{n=1}^{N_{\mathcal{G}}} \frac{1}{N_T} \sum_{i=0}^{N_T} ReLU(-\tilde{x}_n(t_i)) \tag{21}$$

where β denotes the weight coefficient of the loss term. In Eq. (21), the non-negative constraints are implemented by the ReLU function. As the output of the ReLU function is always non-negative, the additional loss term guarantees physically reasonable predictions by ensuring that the model is penalized whenever the physical constraints are violated. It has to be noted that the physical constraints, such as non-negativity constraints on process states, are integrated into PIRNN models as soft constraints. Unlike using a ReLU activation function at the output layer, the soft constraint method offers greater flexibility by modifying the loss function without requiring adjustments to the network architecture.

In addition to incorporating physics-induced loss terms, domain knowledge such as process structure knowledge of a process network can be used to improve the design of the NN architecture to reflect the underlying physics (de Giuli et al. 2024; Wu et al. 2020). In many industrial chemical processes, operations in the upstream phase of production have a direct impact on those in the downstream phase, whereas

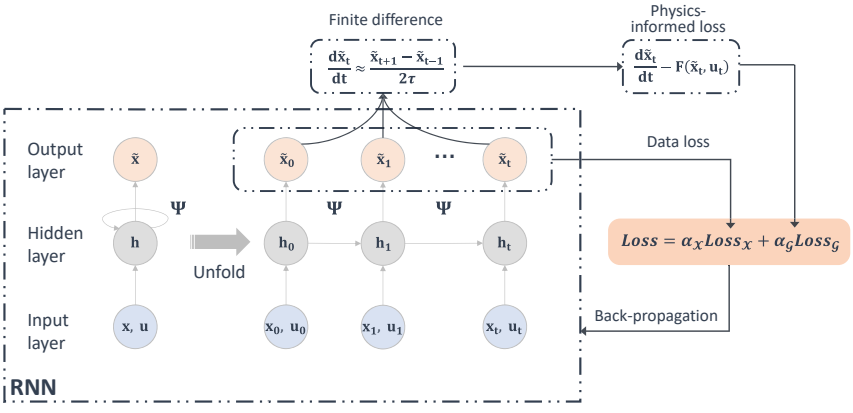

Figure 3: Structure of the PIRNN model encompassing data-driven and physics-informed regularization terms into the loss function.

the reverse influence is often negligible, unless recycling is involved. While theoretical models (if available) are able to capture the relationship between the upstream and downstream processes in their equations, this relational information is rarely utilized in data-driven models. In Wu et al. (2020), a partially-connected RNN structure that mirrors the process network of a two continuous stirred tank reactors (CSTR) system, was proposed. Figure 4 shows how a standard RNN model, with a fully connected structure, can be decoupled into a partially-connected structure. Unlike the fully connected structure where all inputs affect all output, in the partially connected structure, the output of the first RNN layer $x^1 = [C_{A1}\ T_1]$ is designed to be affected only by the input $u^1 = [C_{A10}\ Q_1]$, and the output of the second RNN layer $x^2 = [C_{A2}\ T_2]$ is affected by both inputs u^1 and $u^2 = [C_{A20}\ Q_2]$. C_{A1}, T_1, C_{A2}, T_2 denote the concentration of the reactant A and the temperature in the two reactors, respectively, and C_{A10}, Q_1, C_{A20}, Q_2 denote the inlet concentration of A and heat input rate for the outer cooling jacket, respectively. Thus, the partially-connected network resembles the connections in a two-CSTR system. Likewise, in de Giuli et al. (2024), relational information was used in developing a data-driven model for a district heating system, where individual RNN models were connected based on the physical system network structure. Both studies reported a significant improvement in the model's generalization performance and accuracy, highlighting the merits of PIMLs.

4.1.2 Transfer learning

Another perspective to address data scarcity in data-centric approaches would be to reuse models already developed for similar tasks. This is the key concept of transfer learning (TL), where knowledge learned from a task (source) can be transferred to a related task (target) to boost performance or reduce the data requirement for the new task (Alhajeri et al. 2024; Thebelt et al. 2022). Pan and Yang (2009) provided a comprehensive overview of the types of TL tasks. In summary, TL can be classified into transductive, inductive, and unsupervised TL, based on the availability of label information from source and target domains. If only the source domain labels are available, i.e., no label from target domain, then this is a transductive TL problem. On the other hand, if the target domain labels are available, then this is an inductive TL task. If both the source and target domain labels are unavailable, then this constitutes an unsupervised TL task. In this review, we will focus our discussion on the inductive TL problem, where labeled data from both the source and the target domains are available.

Consider an inductive TL task of transferring knowledge from a source task to a target task. The process first involves developing a pre-trained model (e.g., RNN model) on a large dataset from the source domain. Thereafter, the pre-trained RNN model is adapted to the target domain. Formally, we define Q and P to be the source and target distributions. The RNN input is denoted as $\mathbf{x_i} \in \mathbf{R}^{d_x}$ and the labeled output as $\mathbf{y_i} \in \mathbf{R}^{d_y}$. The set $S = (s_1, \ldots, s_M) = ((\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_M}, \mathbf{y_M})) \in (\mathbf{X} \times \mathbf{Y})^M$ denotes the collection of M labeled data sampled from a certain distribution, where $\mathbf{X}$ denotes the input space and $\mathbf{Y}$ denotes the output space. The labeling function or ground truth model that returns $\mathbf{y_i} = f(\mathbf{x_i})$, is denoted as $f : \mathbf{X} \to \mathbf{Y}$. It is noted that the labeling function for the target task f_P can be different from the source task f_Q. Furthermore, the loss function is defined as $L(\cdot, \cdot) : \mathbf{Y} \times \mathbf{Y} \to \mathbf{R}_+$, where $\mathbf{R}_+$ denotes the set of all positive real numbers. The expected loss for any two function $a(x), b(x)$ on the distribution Q, that maps the input space $\mathbf{X}$ to the output space $\mathbf{Y}$, is given by $\mathscr{L}_Q(a, b) \coloneqq \mathbb{E}_{x\sim Q}[L(a(\mathbf{x}), b(\mathbf{x}))]$.

In modeling nonlinear processes, it is assumed that both the source and target processes can be represented in the form of Eq. (1), but with different process dynamics. For instance, in the case of a chemical reactor, the source process with similar configurations might involve the same reactor type but under varying operating conditions and reactions, different types of reactors performing the same reactions, or even different reactors under different conditions (see Figure 5). Therefore, the distributions of Q and P for the source and target processes are different. However, in reality, because the source and target distributions are typically unknown, we use the corresponding empirical distributions $\hat{Q}$ and $\hat{P}$ for the source and target samples, respectively. These samples, S from the source distribution Q, and T from the target distribution P, are

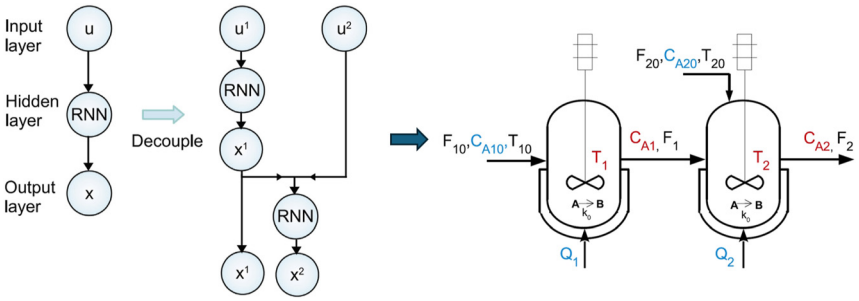

Figure 4: Structure of a partially-connected RNN for a two-CSTR system.

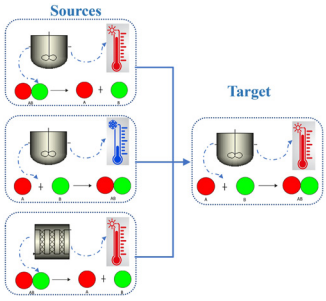

Figure 5: Transfer knowledge from source processes to a target process, where the source processes from bottom to top represent various scenarios: a different reaction in different types of reactors, the same reaction in the same type of reactor under different operating conditions, and different reactions in the same reactor, respectively.

drawn independently and identically distributed (i.i.d.) and used to train the model.

In general, TL approaches can be generalized to instance-based, feature-based, parameter-based, and relational-based approaches (Pan and Yang 2009; Zhuang et al. 2020). In the instance-based approach, each training sample from the source domain is assigned a weight in the calculation of the loss function. The weights will be optimized to minimize discrepancies between samples from the source and target domains (Huang et al. 2006). The feature-based method seeks to find representative features that describe both the source and target domains well. This involves strategies such as feature mapping, feature extraction, and feature encoding. The parameter transfer approach aims to transfer knowledge by sharing the model parameters of the source with the target domain. Finally, the relational-based approach focuses on the transfer of learned relationships between entities in the source domain to the target domain.

TL methods such as feature-based (Bi et al. 2020; Guo et al. 2020; Zhu et al. 2022) and parameter-based (Briceno-Mena et al. 2022; Lee et al. 2022; Xiao et al. 2023) approaches have been widely adopted in the field of engineering. Specifically, parameter sharing is a common TL strategy for NN models. In detail, during the fine-tuning process, a portion of the model's parameters will be adjusted to adapt the pre-trained model to the new task while the remaining parameters are kept unchanged, preserving the knowledge gained during the initial pre-training. For example, in Xiao et al. (2023), a single hidden layer RNN model was first trained with labeled source data from distribution $\hat{Q}$. Afterwards, a new hidden layer ('adaptation layer') was added to the developed RNN model. While keeping the weights in the first hidden layer frozen, the weights of the adaptation layer were optimized using target training samples T from the distribution $\hat{P}$. In the final step, the entire model was fine-tuned using the target samples T.

Although the implementation of transfer learning methods is straightforward, a critical challenge is that transferred knowledge can sometimes harm the performance of the target task. This phenomenon is also known as negative transfer (Wang et al. 2019). According to Wang et al. (2019), factors that lead to negative transfer include the choice of algorithm, differences in source and target distributions, and the size of the labeled target dataset. Therefore, how to quantify the effectiveness of knowledge transfers from the source to the target domain and select a relevant source domain remain important questions. To this end, the generalization error of transfer learning methods has been developed to show the performance of TL models in target learning tasks (Ben-David et al. 2006, 2010; Blitzer et al. 2007; David et al. 2010; Mansour et al. 2009; Xiao et al. 2023). Specifically, Xiao et al. (2023) established a quantitative analysis for the generalization error of TL models for regression tasks and shows its dependence on a number of factors, including the discrepancy distance between source and target distributions, and the complexity of networks.

As an extension to the classic single-source single-target TL problem, multi-source TL, where two or more sources are used for knowledge transfer, has been proposed to enhance the robustness of TL models (Mansour et al. 2008; Tian et al. 2022; Xiao et al. 2024; Yao and Doretto 2010). Furthermore, TL can be coupled with PIML to accelerate and improve prediction accuracy in the presence of data scarcity (Wu et al. 2023b; Xiao and Wu 2023).

4.1.3 Synthetic data generation

Data augmentation is a technique used to artificially increase the size and diversity of a dataset by applying various transformations or perturbations to existing data samples. This helps improve the generalization and robustness of machine learning models, especially when the original dataset is small or imbalanced. Generative AI methods have been used to enrich data sets for the design of new molecules/materials (Abbasi et al. 2022; Batra et al. 2020; Han et al. 2019; Kim et al. 2020; Nouira et al. 2018; Schilter et al. 2023). However, we observe that applications of generative AI to modeling of nonlinear dynamic systems in the chemical industry are still at an early stage of development. Many efforts in synthetic data generation focus on soft sensor development (Lee and Chen 2023; Xie et al. 2019; Zhu et al. 2021), fault diagnosis model development, and risk analysis model development (He et al. 2020; Qin and Zhao 2022) for process industries. For example, the sampling rates of quality variables and process variables are usually inconsistent in process industries due to the high cost of the acquisition of quality data. Additionally, missing data due to sensor failures may occur in process industries,

leading to insufficient training data. Therefore, synthetic data generation is applied as an augmentation of incomplete and unlabeled process signal data. Various generative models have been proposed for data augmentation such as generative adversarial networks (GAN), variational autoencoders (VAE), normalizing flow (NF) models, Gaussian mixture models, hidden Markov models, latent Dirichlet allocation, and Boltzmann machines (Bond-Taylor et al. 2021; Harshvardhan et al. 2020). Specifically, GAN, VAE, and NF have been applied in some recent works in chemical engineering for synthetic data generation, e.g., He et al. (2020), Lee and Chen (2023), Qin and Zhao (2022), Xie et al. (2019), Zhang et al. (2021b, 2024), and Zhu et al. (2021).

GANs are a class of generative models inspired by the concept of Nash equilibrium in game theory (Goodfellow et al. 2014). A typical GAN consists of two networks: a generator G and a discriminator D. The generator takes random variables z $(z{\sim}p_z(z))$ as input, and aims to generate samples $G(z){\sim}p_g$ that can fool the discriminator. The discriminator is used to evaluate whether a given sample x is from real data $(x{\sim}p_{\text{data}}(x))$ or from the generator $x{\sim}p_g(x)$. It returns a score $D(x)$, where $D(x) = 1$ if x is classified as real data, and $D(x) = 0$ if x is classified as generated data. Both the generator and discriminator continuously optimize themselves until they reach Nash equilibrium. Through this process, the difference between two distributions $p_g(x)$ and $p_{\text{data}}(x)$ is minimized such that the generator can capture the distribution of real data p_{data}.

VAE is another class of generative models that combines Bayesian inference with deep networks (Kingma and Welling 2013). Typically structured like an autoencoder, a VAE consists of an encoder and a decoder. Given real data x following the distribution $x{\sim}p_{\text{data}}(x)$, the encoder $q_\phi(z|x)$ maps the real data to a latent space, and the decoder $p_\theta(x|z)$ reconstructs the latent variable to original data, where the latent variables z are designed to follow a continuous distribution $z{\sim}p(z)$. The objective of VAE training is to balance the reconstruction accuracy and the divergence between the latent distribution $q_\phi(z|x)$ and the prior distribution $p(z)$. Therefore, by sampling from the prior distribution $p(z)$ and decoding these samples, VAEs are effective in generating new samples that closely mimic the statistical properties of real data.

Remark 3. In addition to GANs and VAEs, traditional methods, such as bootstrapping in statistics and interpolation techniques (e.g., linear or quadratic interpolation between nearby points), are also commonly used for data augmentation. Bootstrapping generates new datasets by resampling with replacement from the original data, while interpolation creates synthetic data points by estimating values between existing data. These approaches, though simpler than modern generative models, can be effective for certain applications where data patterns are relatively well understood and straightforward.

4.1.4 Active learning

Active learning is another approach that intelligently selects data points that are associated with high uncertainty or low confidence predictions by the current model for labeling and inclusion in training. Unlike synthetic data generation, the goal of active learning is to choose as few labeled samples as possible to minimize the cost of obtaining real data. Active learning methods can generally be classified into three categories: pool-based sampling, stream-based selective sampling, and membership query synthesis (Settles 2009). In pool-based sampling, a large set of unlabeled data is available, from which samples are drawn iteratively at no cost. On the other hand, stream-based selective sampling involves drawing unlabeled samples one at a time. In membership query synthesis, the active learner generates synthetic samples and requests labels for them. In Zhao et al. (2022b), pool-based active learning is used to enrich the training set for modeling a nonlinear chemical process by iteratively identifying the training data that improve model performance most efficiently. Additionally, active learning has the potential to be intergrated with machine-learning-aided optimal experiment design strategies aimed at minimizing time and costs associated with experiments in chemical engineering problems, such as identifying the proper kinetic model structure (Sangoi et al. 2022, 2024).

4.2 Data quality

Another common issue faced in the development and implementation of ML models for real-world applications is the quality of the data. The presence of noise in sensor data is almost inevitable due to factors such as sensor limitation, environmental conditions, measurement error, etc. Since noisy data can impede the learning process of ML models, this section will explore popular and innovative solutions to handle and mitigate the issue of noise-corrupted data.

4.2.1 Conventional approaches

Dropout method is a popular regularization technique used in the training of neural networks to prevent overfitting (Abdullah et al. 2022a; Hinton et al. 2012; Srivastava et al. 2014). Overfitting occurs when a model learns the training data too

well, including its noise and outliers, which results in poor generalization to new, unseen data. Introduced by Hinton et al. (2012), dropout involves randomly "dropping out" a fraction of the neurons in the network during each training iteration. This means that for each forward and backward pass, certain neurons are temporarily removed from the network, along with all their incoming and outgoing connections. The neurons to be dropped out are chosen at random with a probability p, known as the dropout rate. By training the network with dropout, the model becomes more robust and less likely to rely on specific neurons, encouraging it to learn more distributed and generalizable representations. During testing or inference, dropout is turned off, and all neurons are used, but their outputs are scaled by the dropout rate to maintain consistency with the training phase.

Furthermore, Monte Carlo (MC) dropout, a technique used to estimate uncertainty in deep neural networks, can be used to develop stochastic neural networks that characterize the uncertainties of prediction (Gal and Ghahramani 2016a,b). In contrast to the standard dropout, which is only applied during the training phase, the MC dropout applies dropout during both the training and inference phases. Hence, predictions made by NN models using the MC dropout are not deterministic. While the standard dropout helps mitigate the impact of data overfitting by learning a deterministic model, the MC dropout learns a stochastic model that can quantify the system's uncertainty. This ability to estimate uncertainty is particularly valuable for improving controller design under uncertainty. Therefore, MC dropout has been adopted in many chemical process modeling works when training datasets (e.g., sensor measurements) is corrupted with noise, and there is a need to estimate prediction uncertainty (Alhajeri et al. 2022; Wu et al. 2021a).

Specifically, the MC dropout method aims to find the posterior distribution of the model weights $p(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})$, where $\mathbf{X}$ and $\mathbf{Y}$ respectively denote the input and output matrices of the NN, and $\mathbf{W}$ denotes the weight matrix of the NN. However, since it is intractable to obtain the posterior distribution in practice, an estimation of the predictive distribution of the NN output is used and its calculation is provided as follows (Wu et al. 2021c):

$$p(\mathbf{y}^* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{N_t} \sum_{k=1}^{N_t} p(\mathbf{y}^* \mid \mathbf{x}^*, \mathbf{W}_k) \tag{22}$$

where N_t represents the total number of times the model is executed with different dropout masks (i.e., the number of realizations). Since the NN model with MC dropout is stochastic, we are able to generate random predictions by running the MC dropout-NN model multiple times using the same input. By performing multiple forward passes through the network with different dropout masks and averaging the results, we can gather an approximate probabilistic distribution of NN output. Hence, by allowing us to quantify the uncertainty in predictions and reducing the risk of overfitting, MC dropout is a powerful method for learning the ground truth from data corrupted by noise.

4.2.2 Co-teaching method

Co-teaching is an innovative way to address noise in labeled data. The idea behind co-teaching stems from the observations that deep learning models tend to fit simple patterns at the early stage of the training process and progressively learn more complex nuances as training continues (Han et al. 2018). Based on the belief that the training loss is related to the level of noise in the data sample, i.e., noise-free or 'clean' data samples are more likely to have small training loss and vice versa, co-teaching is designed to have two simultaneously trained NNs. A schematic of the co-teaching method with two NNs, A and B, is shown in Figure 6. For every mini batch training iteration, the models identify and collect a small portion of the data samples with small training losses. Subsequently, the models exchange the identified dataset with 'clean' samples and update their weights based on the exchanged dataset. The process is repeated until all training epochs have been completed.

Although the co-teaching method was initially proposed for classification problems with noisy labels, co-teaching has been successfully adapted for regression problems, such as modeling of nonlinear processes in the presence of noise (Abdullah et al. 2022b; Wu et al. 2021c). In addition to the standard co-teaching algorithm highlighted in this section, variants such as asymmetric co-teaching (Yang et al. 2020), stochastic co-teaching (de Vos et al. 2023; Robinet et al. 2022), and co-teaching+ (Yu et al. 2019) have been proposed to improve the accuracy of the model. In essence, by leveraging

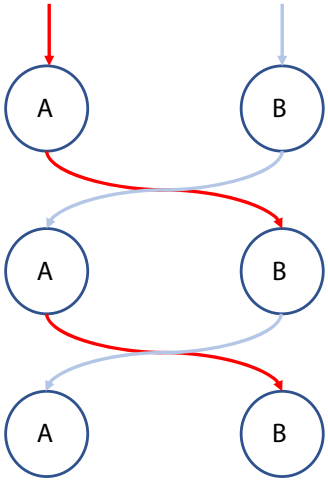

Figure 6: The symmetric co-teaching framework that trains two networks (A and B) simultaneously.

on the peer network's perspective, co-teaching is an effective method to reduce the influence of noisy data and improves the overall generalization capability of the NN model.

4.2.3 Lipschitz-constrained NN

To reduce the effect of data noise on the generalization performance of the model, another approach is to design an inherently robust NN. In particular, Lipschitz-based NN have demonstrated robustness and trustworthiness, especially in handling adversarial attacks. Since many real-world systems (e.g., chemical processes) are Lipschitz continuous, developing a Lipschitz-constrained NN provides a promising solution to addressing data noise in the training set. In mathematical terms, given a function f that maps from set $X \in \mathbf{R}^n$ to set $Y \in \mathbf{R}^m$, i.e., $f : X \to Y$, f is said to be **Lipschitz continuous** if there exists a constant $C_L \geq 0$ such that for all $x, y \in X$, the inequality $|f(x) - f(y)| \leq C_L \cdot |x - y|$ holds. The term C_L is known as the Lipschitz constant, and the function f can also be referred to as C_L-Lipschitz. The Lipschitz constant C_L allows one to quantify the maximum change in the output of a Lipschitz continuous function f with respect to changes in its input. Hence, if the Lipschitz constant C_L is small, the function f will be less sensitive to perturbations in its input, making it more robust. Various methods have been proposed to constrain the Lipschitz constant of neural network models, with most focusing on weight matrices (Arjovsky et al. 2017; Cisse et al. 2017; Gouk et al. 2021), gradients (Anil et al. 2019; Gulrajani et al. 2017; Hein and Andriushchenko 2017), or network architectures (Tang 2023; Wang and Manchester 2023).

Here, we give an example of using the SpectralDense layer that was proposed by (Serrurier et al. 2021). Specifically, **SpectralDense layers** are dense layers such that (1) the activation function σ is a GroupSort function and (2) the largest singular value of the weight matrix W is 1. The following equations describe the **GroupSort function** (of group size 2) i.e., $\sigma : \mathbf{R}^m \to \mathbf{R}^m$:

$$\sigma\left([x_1, x_2, \ldots, x_{m-1}, x_m]^T\right) = [\max(x_1, x_2), \min(x_1, x_2), \ldots, \max(x_{m-1}, x_m), \min(x_{m-1}, x_m)]^T \tag{23a}$$

$$\sigma\left([x_1, x_2, \ldots, x_{m-2}, x_{m-1}, x_m]^T\right) = [\max(x_1, x_2), \min(x_1, x_2), \ldots, \min(x_{m-2}, x_{m-1}), x_m]^T \tag{23b}$$

where Eq. (23a) applies when m is an even number and Eq. (23b) when m is a odd number. Aside from the activation function that does not operate component-wise and the weight matrices having a spectral norm of 1, the SpectralDense layers are structurally similar to the conventional dense layers. The spectral norm of the weight matrix w, denoted as $\|W\|_2$ is 1, as the spectral norm of a matrix is defined to be equal to the largest singular value in its singular value decomposition (SVD). Since the Jacobian matrix of the activation function $\sigma : \mathbf{R}^m \to \mathbf{R}^m$ has a spectral norm of 1 almost everywhere (except for a set of measure 0), then based on Theorem 3.1.6 in (Federer 2014), the function σ is 1-Lipschitz continuous with respect to the Euclidean norm. Hence, it can be concluded that every SpectralDense layer is 1-Lipschitz continuous. Following this, the definition of the class of Lipschitz-constrained neural networks (LCNNs) is given as follows.

Definition 4. Let $\mathscr{LN}_n^m$ be the class of Lipschitz-constrained neural networks (LCNNs) defined as follows:

$$\begin{aligned}\mathscr{LN}_n^m := \{\, f \mid f : \mathbf{R}^n \to \mathbf{R}^m,\ \exists j \in \mathbb{N} \\ \text{such that } f = W_{j+1} f_j \circ f_{j-1} \circ \ldots \circ f_2 \circ f_1, \\ \text{where } f_i = \sigma(W_i \mathbf{x} + b),\ \text{ and }\ \|W_i\|_2 = 1, i = 1, \ldots, j \,\}\end{aligned} \tag{24}$$

where σ denotes the GroupSort activation function with group size 2. Every LCNN in $\mathscr{LN}_n^m$ is a composition of many SpectralDense layers (i.e., W_i, $i = 1, \ldots, j$), with a final weight matrix W_{j+1} at the end. The spectral norm constraint is applied to all weight matrices except the final weight matrix W_{j+1}.

Since every SpectralDense layer f_i, $i = 1, \ldots, j$, is 1-Lipschitz continuous, i.e., the Lipschitz constant for each SpectralDense layer is bounded by 1, it can be easily shown that for every LCNN in class $\mathscr{LN}_n^m$, its Lipschitz constant is bounded by the spectral norm of the final weight matrix W_{j+1}.

The SpectralDense LCNN approach was adopted in Tan and Wu (2024) and Tan et al. (2024b) to handle noisy data when modeling chemical processes. The proposed LCNN demonstrated higher accuracy and generalization performance compared to the conventional Dense NN trained on the same set of noisy data. This highlights the effectiveness of enforcing Lipschitz continuity in NN designs in handling noisy data.

4.3 Curse of dimensionality: ML-MPC of large-scale systems

The curse of dimensionality is a practical challenge for modeling of large-scale systems. It refers to the phenomena that arise when working with high-dimensional systems, the requirement for data amount grows exponentially, leading to more complex network structures, longer training time, and poorer model performance, as the number of dimensions

(features or variables) increases. This issue can be addressed from the modeling and control perspectives, respectively.

4.3.1 Model perspective: reduced-order modeling

Reduced-order modeling (ROM) is a powerful technique to address the curse of dimensionality in high-dimensional systems by reducing the complexity of the system while preserving its essential behavior. In the context of machine learning and data analysis, reduced-order modeling aims to capture the most significant features or dynamics of the data while reducing the dimensionality of the problem. Common dimensionality reduction techniques include methods such as principal component analysis (PCA) (e.g., Hassanpour et al. 2020 integrates PCA with neural networks), which identifies a linear transformation that maps data from a higher-dimensional space to a lower-dimensional space with minimal information loss by minimizing the squared sum of the orthogonal distances between the measured data points and a straight line, and, more recently, autoencoders. Dimensionality reduction is particularly advantageous in process systems engineering where time-scale separation is a common phenomenon in unit models such as distillation columns and catalytic reactors (Chang and Aluko 1984), which can justify the use of reduced-order models. Specifically, if such a timescale separation is not factored into the design of a standard nonlinear feedback controller, the controller may become ill-conditioned due to the stiff ordinary differential equations that arise, resulting in performance deterioration and possibly even unstable closed-loop dynamics (Kokotović et al. 1999).

Reduced-order modeling for two-time-scale systems using sparse identification of nonlinear dynamics (SINDy) was studied in Abdullah et al. (2021a,b). In Abdullah et al. (2021a), the mathematical framework of singular perturbations was utilized to decompose the original two-time-scale system into two lower-order subsystems, each separately modeling the slow and fast dynamics of the original multiscale system. Specifically, after a brief transient period, the fast states converge to a slow manifold and can be algebraically related to the slow states using nonlinear functional representations. To capture the nonlinear relationship between the slow and fast states, nonlinear principal component analysis (NLPCA), developed by Dong and McAvoy (1996), was applied in Abdullah et al. (2021a), following which SINDy was used to derive well-conditioned, reduced-order ODE models for the slow states. The reduced-order SINDy models, owing to their numerical stability, allowed for integration with much larger time steps. Once the slow states were predicted with the SINDy ODE model, NLPCA was used to algebraically predict the fast states without any integration. NLPCA is one of the manifestations or interpretations of a nonlinear extension of the aforementioned linear dimensionality reduction technique PCA and is fundamentally an autoencoder with a nonlinear activation function. The use of a feedforward neural network in NLPCA renders it a static model at the cost of reduced complexity.

The aforementioned SINDy modeling approach for multiscale systems was later used in Abdullah et al. (2021b) to develop a controller based on the slow dynamics. The reduced-order model-based controller, due to its lower complexity and computational cost, was able to outperform a full-order first-principles model-based controller as the former could use a longer prediction horizon in the model predictive control scheme, which is impacted significantly by the prediction horizon length. While there is an inevitable loss of accuracy in a reduced-order model, for tasks involving optimization such as process intensification and optimal control, the computational tractability of solving the mathematical optimization problem is of greater priority, justifying the construction and deployment of such reduced-order models in process systems engineering. SINDy was further developed to handle noisy data and real-time changes in process dynamics using subsampling, co-teaching, error-triggered model update mechanisms, and partial model update algorithms (Abdullah and Christofides 2023b; Abdullah et al. 2022a,b), all of which are techniques that can be extended to the reduced-order modeling framework of Abdullah et al. (2021a,b) as well.

In addition to PCA and SINDy, the autoencoder (AE) is an unsupervised learning model that adopts an FNN architecture to perform tasks such as dimensionality reduction (Kramer 1991). A typical AE comprises two components, the encoder and the decoder. The encoder, parameterized by $\theta = \{W_e, b\}$, compresses the input data $\mathbf{x} \in \mathbf{R}^{d_x}$ into a lower-dimensional representation $\mathbf{x}_r \in \mathbf{R}^{d_h}$ by a function $f_e(\cdot)$, described by Eq. (25).

$$\mathbf{x}_r = f_e(\mathbf{x}) = \sigma_e(W_e\mathbf{x} + b) \tag{25}$$

where $W_e \in \mathbf{R}^{d_h \times d_x}$ is the weight matrix and $b \in \mathbf{R}^{d_h}$ is the bias. $\sigma_e(\cdot)$ is the nonlinear activation function (e.g., hyperbolic tangent function). The encoded representation $\mathbf{x}_r$ consists of lower-dimensional, complex hierarchical nonlinear features, which are often regarded as more efficient representations of the original data (Bank et al. 2023). On the other hand, the decoder, parameterized by $\theta' = \{W_d, b'\}$, aims to reconstruct the input data $\mathbf{x} \in \mathbf{R}^{d_x}$ from the encoded lower-dimensional representation $\mathbf{x}_r \in \mathbf{R}^{d_h}$ via function $f_d(\cdot)$:

$$\mathbf{x}' = f_d(\mathbf{x}_r) = \sigma_d\left(W_d\mathbf{h} + b'\right) \tag{26}$$

where $W_d \in \mathbf{R}^{d_x \times d_h}$, $b' \in \mathbf{R}^{d_x}$ are another weight matrix and bias, respectively. $\sigma_d(\cdot)$ is the linear activation function. The objective of an AE is to encode an input data efficiently,

i.e., learning the optimal weight matrices W_e, W_d and biases $b, b^{'}$, by minimizing its reconstruction error, described by the following MSE loss function:

$$\theta, \theta^{'} = \min_{\theta,\theta^{'}} \frac{1}{M} \sum_{i=1}^{M} L(\mathbf{x}_i, \mathbf{x}_i^{'}) \tag{27}$$

where L is the loss function or the reconstruction error, represented by $L(\mathbf{x}_i, \mathbf{x}_i^{'}) = \|\mathbf{x}_i - \mathbf{x}_i^{'}\|^2$, and M is the number of training data. By using the input data $\mathbf{x}$ as both the input $\mathbf{x}$ and target $\mathbf{x}^{'}$, AE is inherently self-supervised as the model learns from the input data itself without needing external labels. AE can be trained using similar algorithms as the traditional FNN, such as the classic minibatch gradient descent with gradients computed from back-propagation.

The close resemblance between AE and the commonly used linear dimensionality reduction method, PCA, is noted (Xiu et al. 2020). In a case where only the linear activation function is used in the AE calculation, the encoder output will correspond directly to the principal components in PCA. However, due to its flexibility in adopting a broad range of activation functions and producing complex nonlinear representations of the input data, AE is generally preferred over PCA for more effective dimensionality reduction. The benefits of incorporating AEs into machine learning tasks include reduced training duration, enhanced robustness against overfitting, and improved convenience in data visualization. Due to its advantages, NN models coupled with AE have been widely applied in the field of engineering, in areas such as process monitoring (Cheng et al. 2019; Lee et al. 2019), fault diagnosis (Zhang and Qiu 2022; Zheng and Zhao 2020), and process modeling (Na et al. 2018; Saraswathi K et al. 2020).

Since recursive prediction of RNNs could be time-consuming for high-dimensional systems, we can integrate RNNs with AE by developing RNNs that predict future states in the latent space. Figure 7 illustrates the structure of the integrated AE and RNN (termed AERNN model) (Zhao et al. 2022a; Zheng et al. 2022a). Specifically, an AE was developed to encode and decode state variables $\mathbf{x}$. The addition of an AE helps to accelerate the RNN model training and computation time by projecting the state variables in a lower dimension space. In other words, as a result of the reduced representation of the RNN input, the number of neurons at the input and output layers of the RNN model can be significantly reduced, and hence fewer computations are required. After the RNN computations, the output from the RNN model is reverted back to its original space for interpretation. Additionally, in some recent works, autoencoders have been integrated with Koopman operators for model order reduction in MPC to effectively handle high-dimensional data (Chandrasekar et al. 2024; Wang et al. in press). Moreover, as part of an effort to improve the robustness of AERNN to noisy

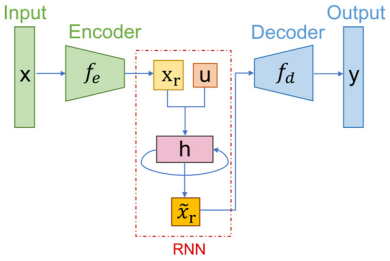

Figure 7: Structure of autoencoder RNN model.

data, the Lipschitz-constrained network structure proposed in Section 4.2.3 can be used as the backbone for the development of both AE and RNN (Tan et al. 2024b).

Although AE and other similar feature extraction techniques (e.g., PCA) are able to transform high-dimensional feature datasets into a lower-dimensional space whilst retaining most of the crucial information from the dataset, these reduced-order representations often do not carry any physical meaning. The lack of interoperability of the encoded features makes these approaches less appealing, particularly in safety–critical chemical processes, where a physical interpretation of the features is necessary, especially those features identified to be "important". Hence, another perspective on reduced-order modeling arises from the idea of selecting a subset of relevant features for model construction. In feature selection, the chosen features should show a high correlation with the system output and exhibit strong interpretability to support further analysis. In general, feature selection methods can be categorized into three classes based on their selection and learning processes, namely, wrapper, filter, and embedded methods (Chandrashekar and Sahin 2014). In summary, the wrapper methods scan for the best performing subset of features among potential subsets of features, using a specific search algorithm and a pre-defined ML algorithm (Karagiannopoulos et al. 2007). However, as the input dimension increases, the search space grows exponentially, making the wrapper methods increasingly computationally intensive. In contrast, filter methods do not require the use of ML algorithms in the selection process. Filter methods select relevant features by ranking and ordering the features using a suitable ranking criterion that measures the correlation between the input features and the target output, e.g., the Pearson correlation coefficient (Rendall et al. 2019). Features with scores below a predefined threshold are removed from the feature set, and the remaining features are used to substitute the original high-dimensional feature set in the modeling process (Degeest et al. 2019). The embedded methods, as the name suggests, directly integrate the feature selection process into the model training process to save computational time. In

particular, regularization models are the most commonly used embedded methods, where the loss function is modified such that the model learns the important features while minimizing the fitting error (Li et al. 2017). Interested reader may refer to Karagiannopoulos et al. (2007) and Li et al. (2017) for detailed reviews on the various feature selection methods and Zhao et al. (2023) on their applications in reduced-order RNN models.

4.3.2 Control perspective: distributed MPC

From a control point of view, another way to mitigate the curse of dimensionality in large-scale systems is by applying distributed control techniques. In distributed control systems, the system is divided into smaller subsystems, where individual controllers are designed for each subsystem. Although control calculations are performed on separate processors, controllers in distributed control systems are able to communicate and cooperate with each other to achieve the objectives of the closed-loop plant. By decomposing the system into smaller subunits, the complexity and computational demands to model and control the process network can be significantly reduced. In this regard, distributed MPCs (DMPC) and decentralized MPCs using ML models have been developed in Chen et al. (2020a,b). A schematic of two DMPC architectures is shown in Figure 8. The key difference between sequential and iterative DMPCs is that the communication between two MPCs in a sequential DMPC framework is one-way only, while the controllers in iterative DMPCs communicate with each other to cooperatively optimize the control actions. Since the designs of ML-based DMPCs closely follow those using first-principles models, the formulations of ML-DMPCs are omitted here. Various alternative configurations of DMPC systems have been proposed in literature, each varies in terms of the coordination and communication schemes between the subsystems' MPC. Readers are directed to Christofides et al. (2013), Scattolini (2009), and Stewart et al. (2010) for comprehensive reviews of DMPC.

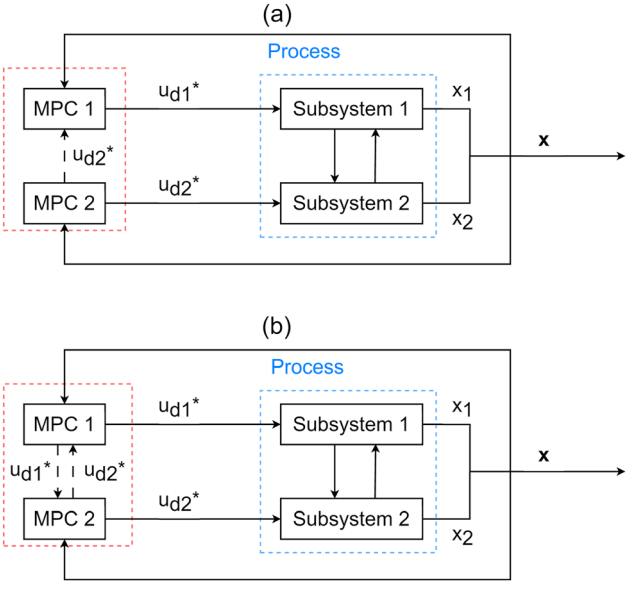

Figure 8: Schematic diagrams of (a) sequential distributed MPC and (b) iterative distributed MPC systems.

4.4 Model uncertainty and process disturbances

Machine learning models are generally developed using historical data, and cannot take model uncertainty and process disturbances into account. To build more robust and reliable models that adapt to the variations in system dynamics, online learning and robust control can be adopted to improve the performance of models and controllers, respectively.

4.4.1 Model perspective: online update of ML models

Online machine learning refers to a paradigm of machine learning where models are continuously updated as new data becomes available, often in a streaming fashion. Unlike traditional batch learning, where models are trained on fixed datasets, online learning enables models to adapt and evolve over time as they receive new data points. This approach is particularly useful for machine learning modeling of systems with time-varying dynamics due to disturbances. In-depth discussions on online learning and its theoretical analysis can be found in Hoi et al. (2021), Rakhlin et al. (2010), and Shalev-Shwartz (2012), respectively. An early attempt to implement online learning in the predictive models of MPC was recorded in Murray-Smith et al. (2003), where the authors added new process information to the training set at every timestep and adjust the model's hyperparamters accordingly. More recent work on integrating online learning into MPC applications can be found in Bhadriraju et al. (2019), Bradford et al. (2020), Limon et al. (2017), and Ning and You (2021).

Updating machine learning models online can be handled using various strategies. An effective method is error-triggered updates, where the model is updated only when the prediction error exceeds a certain threshold. This helps in making efficient use of computational resources and ensures that the model remains accurate and up-to-date with minimal overhead. Error-triggered online learning typically involves the following steps recorded in Abdullah and Christofides (2023b), Wu et al. (2019b), and Zheng et al. (2022c): (1) start with an initial model pre-trained on

historical data, (2) monitor incoming data and compute the prediction error using new data, and (3) if the current prediction error or accumulated prediction error in a sliding time window exceeds the predefined threshold, update the model using the new data. Furthermore, when incorporating online machine learning models into MPC, event-triggered mechanism designed based on stability criteria can be adopted to update models (Wu et al. 2019b). While online learning helps maintain the accuracy of machine learning models in dynamic environments, potential drift in the underlying data distribution over time due to the change of process dynamics under disturbances can pose a significant challenge to the performance of updated models. To this end, some recent works have investigated the generalization performance of online learning models that take independent and identically distributed (i.i.d.) real-time data and non-i.i.d. real-time data for online learning, respectively (Hu and Wu 2024; Hu et al. 2023a,b). These two cases represent the scenarios where system dynamics remain unchanged and where they change over time, respectively. Specifically, it is shown in Hu and Wu (2024) that the generalization performance of online learning models depends on several factors, including the divergence between historical data and real-time data distributions, network complexity, and sample size.

4.4.2 Control perspective: robust MPC and tube-based MPC

From the control perspective, we can design robust MPC and tube-based MPC to account for plant-model mismatch in uncertain systems. Specifically, tube-based MPC addresses the uncertainty in system dynamics by considering a range of possible future trajectories rather than a single trajectory. It creates a "tube" around the nominal trajectory, within which the actual trajectory is expected to lie. Tube-based MPC uses techniques like robust optimization or stochastic optimization to compute the tube around the nominal trajectory. Tube-based MPC often involves solving optimization problems with constraints that ensure that the system remains within the defined tube despite uncertainties. Machine learning techniques have been incorporated into tube-based MPC to further improve the characterization of uncertainties and robustness. Recent developments in tube-based MPC using machine learning include work by Gao et al. (2024), Zhang et al. (2022), and Zheng et al. (2022b).

Robust MPC directly incorporates uncertainty into the control law formulation. It aims to optimize control inputs such that the system remains stable and satisfies performance criteria under the worst-case scenario of uncertainty. Robust MPC typically involves solving optimization problems with robust constraints or using techniques like min-max optimization to find control inputs that perform well under uncertainty. In recent works by Berberich et al. (2020), Chen and You (2021), Hu and You (2023), Mahmood et al. (2023), and Manzano et al. (2020), robust data-driven MPCs and robust learning-based MPCs have been developed to enhance the robustness of controllers against uncertainties such as prediction errors from machine learning models and process disturbances. For example, Chen and You (2021) used ML to learn uncertainties, and designed a robust MPC for greenhouse in-door climate control problems. Mahmood et al. (2023) developed a robust data-driven-based MPC based on the minimax approach for temperature control and optimization of energy consumption.

4.5 Computational efficiency

ML-based MPC is generally solved slowly due to the complexity of ML models (the ML model is often required to be evaluated multiple times during optimization, which can significantly increase computational burden), and the nonconvexity of optimization problem.

4.5.1 Model perspective: optimization and convexification

One approach to improve the computational efficiency of ML-MPC is to simplify the model architecture, such as by reducing the number of neurons or layers. While manually doing this can be challenging, automated tools and techniques can assist in finding an optimal configuration, thereby reducing computational overhead. Reduced-order modeling that has been introduced in the previous section could be a solution to large-scale nonlinear systems. Additionally, hyperparameter optimization could be one solution to finding the optimal hyperparameters for ML models. Some common techniques for hyperparameter optimization of ML models include grid search (Bergstra et al. 2011) and Bayesian optimization (e.g., tools such as *Optuna* (Akiba et al. 2019) and *Hyperopt* (Bergstra et al. 2013)). An analysis and comparison of common hyperparameter optimization approaches for developing an LSTM forecast model for a cyber-physical production system can be found in Pravin et al. (2022).

Another approach is to build input-convex ML models (Amos et al. 2017; Chen et al. 2018c; Wang et al. 2025; Yang and Bequette 2021). An input-convex model in the context of machine learning refers to a model whose loss function is convex with respect to its input. This property can be highly beneficial for optimization because convex functions have a single global minimum, making the optimization process more straightforward and ensuring that gradient-based methods

converge reliably. Certain linear models, such as linear regression and logistic regression, are inherently convex because their loss functions are convex with respect to the model parameters. However, for a more general class of nonlinear ML models, it is possible to design neural network architectures and loss functions to be input convex under certain conditions. This design can significantly improve training stability and convergence. We provide an example of enforcing input convexity in FNNs. Following the same idea, input-convex RNNs and LSTMs have been designed in some recent works (Chen et al. 2018c; Wang et al. 2025). The output of each layer of input-convex FNN follows:

$$\mathbf{z}_{l+1} = g_l\left(W_l^{\mathbf{z}}\mathbf{z}_l + W_l^{\mathbf{x}}\mathbf{x} + \mathbf{b}_l\right), \quad l = 0, 1, \ldots, L-1, \tag{28}$$

and with $\mathbf{z}_0, W_0^{\mathbf{z}} = 0$. The output $\mathbf{z}_{l+1}$ is input-convex for single-step prediction if all weights $W_l^{\mathbf{z}}$ are non-negative and all activation functions g_l are convex and non-decreasing (Amos et al. 2017), while the output $\mathbf{z}_{l+1}$ is input-convex for multi-step ahead predictions if all weights $W_l^{\mathbf{z}}$ and $W_l^{\mathbf{x}}$ are non-negative and all activation functions g_l are convex and non-decreasing (Bünning et al. 2021). Therefore, under certain conditions (e.g., convex objective functions and convex constraints), the MPC using input-convex NNs (ICNNs) becomes a convex optimization problem, which is computationally less expensive to solve.

Remark 4. It is important to note that while ICNN models offer benefits such as global optimality and stability, they may lose accuracy when applied to highly non-convex functions due to their inherent convexity. However, for many practical systems that are not highly non-convex, ICNN models can provide a computationally efficient alternative for ML model-based optimization problems while maintaining the desired accuracy. In practical applications, comparing the testing losses of ICNN models with traditional FNN models can be an effective way to evaluate the performance of ICNN models. If they yield similar accuracy, ICNN models can be considered a good approximation for nonlinear systems. Additionally, partially input convex architecture of ICNN (PICNN) can be utilized to further restore the representation ability of ICNN models by making the output a convex function to some elements of the input (Amos et al. 2017). Therefore, developing ICNN models requires a delicate balance between convexity and representation power to ensure optimal performance for various applications.

4.5.2 Control perspective: explicit ML-MPC

Explicit MPC provides another solution from the control perspective to improve computational efficiency. In explicit MPC, the control law is precomputed and stored as a piecewise function of the system state. This precomputation allows for real-time implementation with constant-time complexity, regardless of the system's complexity or prediction horizon. By eliminating the need for online optimization during operation, explicit MPC can achieve faster control loop execution times, making it suitable for applications with stringent real-time requirements.

The explicit control law is derived using multi-parametric programming algorithms, which include multi-parametric linear/quadratic programming (mpLP/mpQP) and multi-parametric mixed-integer linear/quadratic programming (mpMILP/mpMIQP) (Pistikopoulos et al. 2020). Unlike typical optimization problem where the parameters, e.g., system state x, are fixed and known, in multi-parametric programming, the parameters are unknown at the point of computation. Multi-parametric programming addresses this uncertainty by generating an optimal solution map for all possible values of the uncertain parameters, e.g., finding the optimal control action u^* for all possible states x (Ali et al. 2023; Tian et al. 2021). By obtaining precomputed solutions offline, the online computational load of the MPC is significantly reduced. By transforming MPC problems of discrete-time, linear time-invariant state-space models with linear/quadratic cost functions into mpLP/mpQP problems, these MPC problems can be solved explicitly, using solvers such as Python Parametric OPtimization Toolbox (PPOPT), Parametric OPtimization Toolbox (POP), and Multi-Parametric Toolbox (MPT) (Kenefake and Pistikopoulos 2022; Kvasnica et al. 2004; Oberdieck et al. 2016).

As it can be time-consuming to solve ML-based MPC (Wu et al. 2019d), there has been a growing interest in converting ML-MPC into an explicit ML-MPC for faster computation. However, the black-box nature of ML models creates obstacles in the path towards explicit ML-MPC. As ML models can be difficult to express explicitly, i.e., do not have explicit expressions, it is a challenge to adopt existing explicit MPC algorithms for ML-MPC. An approach to bypass this problem is to utilize the unique property of the ReLU activation function and represent the ML model as a mixed-integer linear programming (MILP) problem. The MILP problem is then incorporated into the formulation of an explicit ML-MPC and solved using mpMILP (Chen et al. 2018b; Grimstad and Andersson 2019; Katz et al. 2020). An althernative approach to solve the explicit ML-MPC using multi-parametric nonlinear programming (mpNLP) methods. As deriving the exact solutions to mpNLP is still an unsolvable problem, existing mpNLP algorithms generally use either piecewise linearization or quadratic constraints to approximate the strong nonlinear terms (Kassa and Kassa 2016; Pappas et al. 2021). In Wang et al. (2024a,b), the authors developed explicit ML-MPC for ML models with a general class of nonlinear activation functions by first approximating ML models by piecewise linear

functions. The corresponding mpNLP problems are then approximated into mpLP/mpQP problems which can be solved efficiently by existing algorithms.

In addition, compared to some works that develop an ML model to learn state-input relationship that can be used to replace the controller in a closed-loop system, explicit MPC offers transparency and interpretability since the control law in explicit MPC is represented as a piecewise function of the system state, for which engineers can easily analyze and understand how control actions are determined based on the current state of the system. This interpretability is valuable for troubleshooting, tuning, and verifying the controller's behavior in practical applications.

Remark 5. The main challenge with explicit ML-MPC is that ML models are typically black-box models without an explicit functional form (or they are too complex to be directly incorporated into explicit MPC solvers). Nonlinearity adds another layer of difficulty, as a nonlinear ML model leads to mpNLP, which is generally hard to solve. Potential solutions include using the unique properties of ReLU activation functions for ML models to formulate a mixed-integer linear programming (MILP) problem, or approximating nonlinear ML models with piecewise linear functions, allowing for the formulation of mpLP or mpQP problems.

4.6 Safe and secure ML-MPC

While most existing research of ML-MPC in engineering disciplines has focused on improving its prediction accuracy and performance, safety and security are emerging research areas of significant importance. The misuse of ML-MPC technologies could lead to unsafe, and potentially catastrophic, consequences in safety-critical systems, causing environmental damage, capital loss, and human injuries.

4.6.1 Safety

Ensuring safety of ML-MPC includes safe learning (data collection), safe modeling, and safe implementation. **Safe data collection** is often not the most critical issue in supervised learning because datasets are provided for offline learning. The data used in supervised learning is usually pre-collected, cleaned, and labeled, which reduces the risks associated with data collection. However, the importance of safe data collection becomes much more pronounced in other machine learning techniques, such as RL. Specifically, in RL, an agent interacts with an environment to learn optimal actions through trial and error. This interaction can involve significant risks, especially in real-world applications like autonomous driving, robotics, and chemical plants, where unsafe actions can lead to accidents, injuries, or other severe consequences. To ensure safe exploration, safe RL has recently been studied, where various techniques such as reward shaping and safety constraints through barrier functions have been developed to limit the action and state space (Garcıa and Fernández 2015; Kim and Kim 2022; Wang and Wu 2024a).

Safe modeling in supervised learning often refers to ensuring the robustness and reliability of predicted outputs. This involves several strategies and techniques with the goal of ensuring that the model predictions are consistent, reliable, and conform to necessary constraints in real-world systems. To achieve safe modeling in terms of reliable predictions, we can impose hard constraints on NN outputs through the design of activation functions, or incorporate the constraints as a regularization term (similarly to physics-informed ML introduced in Section 4.1.1). Additionally, robustness requires that the prediction of ML models is robust to small perturbations in input data. Some common techniques include adversarial training that intentionally introduces adversarial examples in the training process to improve its robustness, and novel design of NN architectures with inherent robustness such as Lipschitz-constrained NNs that have been introduced in Section 4.2.3.

Lastly, from the control perspective, **safe implementation** of ML models in MPC requires the improvement of existing controllers to account for the impact of safety as the last line of defense. Due to the approximation of ML models, ML-based MPC may lead to suboptimal or even unreasonable control actions that may cause unsafe operations. To mitigate these risks, safety constraints have been incorporated into the design of MPCs, ensuring that control actions and the resulting state evolution remain within safe bounds. For example, barrier functions can be used to design MPCs to effectively prevent the system from violating safety constraints by heavily penalizing states near the constraint boundaries. While there are various types of barrier functions, we provide an example of the control barrier function (CBF) for the nonlinear affine control system $\dot{x} = f(x) + g(x)u$ proposed in Wieland and Allgöwer (2007).

Definition 5. Given a set of unsafe states in state-space $\mathscr{D}$, a $\mathscr{C}^1$ function $B(x) : \mathbf{R}^n \to \mathbf{R}$ is a CBF if the following properties are satisfied:

$$B(x) > 0, \quad \forall\, x \in \mathscr{D} \tag{29a}$$

$$L_f B(x) \le 0, \ \forall\, x \in \{z \in \mathbf{R}^n \setminus \mathscr{D} \mid L_g B(z) = 0\} \tag{29b}$$

$$\mathscr{U} := \{x \in \mathbf{R}^n \mid B(x) \le 0\} \neq \varnothing \tag{29c}$$

To further reinforce closed-loop stability while ensuring safety simultaneously, CBFs can be integrated with control Lyapunov functions via weighted sums. As a result, control

Lyapunov-barrier functions (CLBFs) that was proposed in Romdlony and Jayawardhana (2016) has been used to design safe MPC in Wu et al. (2018, 2019a). The definition of CLBFs is given as follows:

Definition 6. Consider the nonlinear system $\dot{x} = f(x) + g(x)u$ with a set of unsafe states in state-space (i.e., $\mathscr{D}$), a proper, lower-bounded and $\mathscr{C}^1$ function $W_c(x): \mathbf{R}^n \to \mathbf{R}$ is a CLBF if $W_c(x)$ has a minimum at the origin and also satisfies the following properties:

$$W_c(x) > \rho_c, \quad \forall\, x \in \mathscr{D} \subset \phi_{uc} \tag{30a}$$

$$L_f W_c(x) < 0,$$
$$\forall\, x \in \{z \in \phi_{uc} \backslash (\mathscr{D} \cup \{0\} \cup \mathbb{X}_e) \mid L_g W_c(z) = 0\} \tag{30b}$$

$$\mathscr{U}_{\rho_c} \coloneqq \{x \in \phi_{uc} \mid W_c(x) \le \rho_c\} \neq \varnothing \tag{30c}$$

$$\overline{\phi_{uc} \backslash (\mathcal{D} \cup \mathcal{U}_{\rho_c})} \cap \overline{\mathcal{D}} = \varnothing \tag{30d}$$

where $\rho_c \in \mathbf{R}$, ϕ_{uc} is a neighborhood around the origin, and $\mathbb{X}_e \coloneqq \{x \in \phi_{uc} \setminus (\mathscr{D} \cup \{0\}) \mid \frac{\partial W_c(x)}{\partial x} = 0\}$ is a set of states where $L_f W_c(x) = 0$ (for $x \neq 0$) due to $\partial W_c(x)/\partial x = 0$. The formulation of CLBF-MPC can be found in Wu et al. (2019a). Additionally, in Wu and Christofides (2020), CLBF-MPC using ML models were developed to control chemical processes with unknown process models. Chen et al. (2022a) discussed the use of ML methods for the construction of barrier functions when safe and unsafe regions cannot be represented in functional forms. Furthermore, in Chen et al. (2022b), the generalization performance was analyzed for ML-based construction of barrier functions and the resulting safe MPC. In addition to control Lyapunov and control barrier functions that can be used to ensure stability and safety, respectively, in MPC, control invariant sets can been incorporated into machine-learning-based controllers to improve stability (e.g., reinforcement learning-based controllers in Bo et al. 2023).

4.6.2 Data security

Data security is also an emerging challenge in the design and implementation of ML-MPC. Data risks can arise during both the **offline modeling stage** and the **online implementation** of ML-MPC. Specifically, since model training often involves collecting and processing data in a centralized manner (e.g., on a central server), communication channels can be vulnerable to attacks during data collection process, making the data susceptible to breaches, tampering, and unauthorized access. Ensuring data security is essential to protect sensitive information and maintain the integrity of the training process. As discussed in Parker et al. (2023), cybersecurity of industrial control systems can be improved through a variety of fundamental operation and control methods that address the following aspects: security by design, advanced recovery, advanced threat detection, secure remote access, and combined safety. Specifically, we can improve data security in both the learning and implementation stages of ML-MPC. For example, unlike the conventional ML approaches for modeling a nonlinear process network with multiple subsystems, where the training process is performed on a central server with training data collected from all subsystems, federated learning (FL), an emerging distributed ML framework to preserve data privacy, distributes the training data across multiple local subsystems, and subsequently, aggregate the submodels trained locally for each subsystem to create a global FL model (Zhang et al. 2021a; Zhao et al. 2018). Since FL only exchanges the NN weight information, and maintains local data in local systems without sharing with each other (see Figure 9), data security is significantly improved under the FL framework. In Xu and Wu (2024), FL was applied to model the distributed nonlinear systems with guaranteed data privacy for ML methods, and then incorporated into the design of MPC.

In addition to data security in the learning stage, the smooth operation of ML-MPC in real-time heavily depends on the accuracy of recorded data and the reliability of networked communication channels. Any compromise in the integrity or confidentiality of this data due to unauthorized access or manipulation by malicious entities can lead to serious consequences, impacting operational safety and economic performance. As sophisticated cyber-attacks pose risks to system information, there is a need to develop ML-MPC that ensures the confidentiality of industrial data. A promising solution to tackle this challenge is the adoption of an encrypted control system (Farokhi et al. 2017; Kim et al.

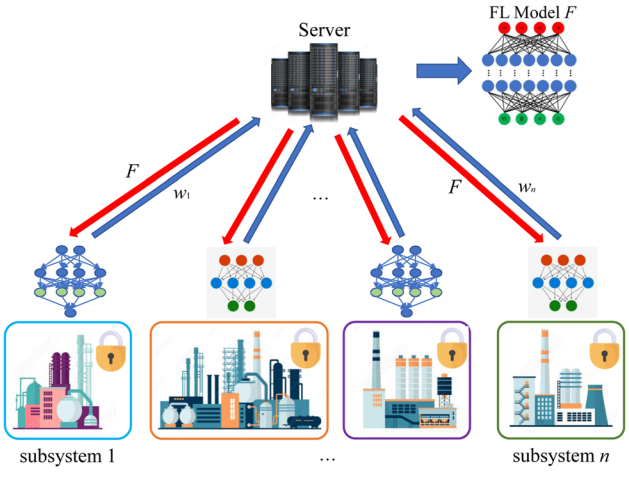

Figure 9: A schematic of federated learning, where F denotes the ML model, and w denotes the model weights.

2016), offering a versatile and effective means to improve data security and confidentiality. It can be seamlessly implemented across various systems without requiring system-specific modifications, thus addressing the core challenge of secure data transmission in networked systems.

The work of Suryavanshi et al. (2023) presents the closed-loop architecture of an encrypted MPC. As depicted in Figure 10, the sensor signals $x(t)$ are subjected to encryption using the public key before being sent to the model predictive controller (MPC). Regarding encryption techniques, Schlüter et al. (2023) discussed various potential methods that can be used to ensure the confidentiality of transmitted data. These methods include homomorphic encryption (HE), secure multi-party computation (SMPC), differential privacy (DP), and random affine transformations (RAT). Other encryption methods include symmetric encryption and partially homomorphic encryption (PHE) to secure data. Symmetric encryption, like advanced encryption standard (AES), is a non-homomorphic technique that prohibits mathematical operations within encrypted data (Rijmen and Daemen 2001). It is noted that fully homomorphic encryption, as seen in schemes like Brakerski–Gentry–Vaikuntanathan (BGV), allows both addition and multiplication operations within encrypted data (Gentry et al. 2012), while partially homomorphic encryption enables either addition or multiplication operations within encrypted data. For example, the Paillier cryptosystem supports addition operations in an encrypted environment (Paillier 1999). Paillier encryption, one of the semi-homomorphic cryptosystems with additive homomorphism, has been widely used for its computational efficiency compared to other semi-homomorphic encryption schemes like El-Gamal (Elgamal 1985), and its ability to perform additive operations in an encrypted space, without decryption. Its security guarantees rely on a standard cryptographic assumption called decisional composity residuosity (DCR). Thus, homomorphic (partially and fully) encryption must be used when the transmitted ciphertexts need to be utilized for performing linear mathematical operations without decryption.

After obtaining the encrypted data, it undergoes decryption, resulting in quantized states $\hat{x}(t)$. These quantized states serve as the initial values for the plant model within the MPC at time t. The MPC subsequently computes optimized inputs $u(t)$, which are encrypted prior to transmission to the actuator. After the actuator receives the encrypted signals as input, the encrypted input is decrypted, leading to a quantized input, $\hat{u}(t)$ that is applied to the process. Additionally, Kadakia et al. (2024a) developed an encrypted distributed MPC for networked systems, and Kadakia et al. (2024d) further integrated cyber-attack detection with encrypted control systems. Moreover, to control nonlinear processes with the objective of maximizing economic performance or achieving desired tracking performance, a two-layer framework was proposed in Kadakia et al. (2024b) and Kadakia et al. (2024c) to integrate encrypted feedback control with dynamic process economics optimization through economic MPC, and tracking MPC, respectively.

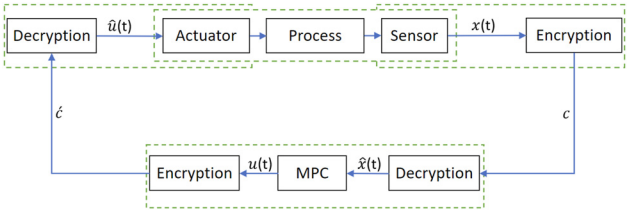

Figure 10: Illustration of the data transfer process in an encrypted MPC system (Suryavanshi et al. 2023).

5 Applications of ML-MPC to a chemical process example

In this section, we use a nonlinear chemical process to demonstrate the performance of various ML modeling and ML-MPC control methods, addressing different practical challenges discussed in previous sections. We begin with a brief introduction to developing conventional RNN models for nonlinear dynamic systems. We then explore advanced RNN models incorporating physics-informed ML, transfer learning, dropout, co-teaching, Lipschitz-constrained architecture, input-convex structure, online learning, and federated learning. These advanced methods are designed to tackle practical issues such as data scarcity, noise, robustness, convexity, model uncertainties, and data security. Following this, we show several novel designs of ML-MPCs that enhance computational efficiency, process operational safety, and cybersecurity. Additionally, the Python codes for some of the aforementioned ML and ML-MPC methods are provided for reference.

5.1 Process description

Consider a well-mixed, non-isothermal CSTR where an irreversible second-order exothermic reaction is taking place. The reaction involves the conversion of the reactant A to the product B $(A \rightarrow B)$. The concentration of A at the reactor inlet is denoted as C_{A0}. The temperature and volumetric flow rate of the reactor inlet are represented as T_0 and F, respectively. The CSTR is equipped with a heating jacket that supplies/removes heat at a rate Q. The following

Table 1: Parameter values of the CSTR.

$T_0 = 300$ K	$F = 5$ m^3/h
$V = 1$ m^3	$E = 5 \times 10^4$ kJ/kmol
$k_0 = 8.46 \times 10^6$ m^3/kmol h	$\Delta H = -1.15 \times 10^4$ kJ/kmol
$C_p = 0.231$ kJ/kg K	$R = 8.314$ kJ/kmol K
$\rho_L = 1,000$ kg/m^3	$C_{A0_s} = 4$ kmol/m^3
$Q_s = 0.0$ kJ/h	$C_{A_s} = 1.95$ kmol/m^3
$T_s = 402$ K	

material and energy balance equations describe the CSTR dynamic behavior:

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT}} C_A^2 \tag{31a}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} \tag{31b}$$

where C_A is the concentration of reactant A in the reactor, T is the temperature of the reactor, V is the volume of the reacting liquid in the reactor, and Q denotes the heat input rate. The reacting liquid has a constant density of ρ_L and a heat capacity of C_p. $\Delta H, k_0, E$, and R represent the enthalpy of reaction, the pre-exponential constant, the activation energy, and the ideal gas constant, respectively. Process parameter values are listed in Table 1.

The CSTR has an unstable steady-state $(C_{As}, T_s) = (1.95 \text{ kmol/m}^3, 402 \text{ K})$, and $(C_{A0_s} Q_s) = (4 \text{ kmol/m}^3, 0 \text{ kJ/h})$. The manipulated inputs are the inlet concentration of reactant 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/h. Therefore, the states and the inputs of the closed-loop system are $x^T = [\Delta C_A \ \Delta T]$ ($\Delta C_A = C_A - C_{As}$ and $\Delta T = T - T_s$) 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 control objective is to operate the CSTR at the unstable equilibrium point (C_{As}, T_s) by manipulating the inlet concentration ΔC_{A0} and the heat input rate ΔQ under the MPC using RNN models. The dynamic model of Eq. (31) is numerically simulated using the explicit Euler method with an integration time step of $h_c = 10^{-4}$ h. The nonlinear optimization problem of the LMPC of Eq. (9) is solved using the Python module of the IPOPT software package, termed PyIpopt (Wächter and Biegler 2006), with the sampling period $\Delta = 10^{-2}$ h. It is important to note that the CSTR equations of Eq. (31) are assumed to be unknown to MPCs. They are used solely for data generation and represent the real-world system to which MPC control actions will be applied.

5.2 Development of RNNs

The RNN models in this example are developed to predict the states for the next sampling period $x(t), \forall t \in [t_k, t_{k+1})$ based on the current state $x(t_k)$ and the manipulated input $u(t_k)$ that will be applied for $t \in [t_k, t_{k+1})$, where $t_{k+1} = t_k + \Delta$. Note that the RNN predictions include the states for intermediate time steps within one sampling period, since one sampling period $\Delta = 10^{-2}$ h includes 100 integration time steps $h_c = 10^{-4}$ h. The development of NN models in the context of supervised learning involves the following steps: (1) data generation and processing, (2) network construction including the design of architecture, hyperparameters, loss function, and optimizers, and (3) testing its performance and fine-tuning the model. In the review by Ren et al. (2022), the authors have provided a step-by-step guide on generating data using computer simulations, and analysis of the performance of three popular NN models: FNN, RNN, and encoder-decoder, for dynamic systems. Therefore, in this case study, we focus more on the construction and training of novel RNN models.

5.2.1 Physics-informed RNNs

In this section, we will introduce the development and construction of PIRNN using the results from Zheng et al. (2023). In Zheng et al. (2023), three RNN models were developed, namely, a standard RNN, PIRNN, and a purely physics driven RNN model (termed PIRNN without $\text{MSE}_{\mathscr{x}}$, where $\text{MSE}_{\mathscr{x}}$ corresponds to the regular supervised loss term $\text{Loss}_{\mathscr{x}}$ in Eq. (20)). To demonstrate the effectiveness of PIRNNs, the models were trained with limited process data, concentrated in a small region around the unstable steady state (C_{A_s}, T_s). Specifically, the initial states $(\Delta C_A, \Delta T)$ and the manipulated inputs $(\Delta C_{A0}, \Delta Q)$ are uniformly and randomly sampled from a small region around their respective steady states (i.e., 8 data points each from ΔC_A (kmol/m^3)~$U(-0.2, 0.2)$ and ΔT (K)~$U(-20, 20)$), Similarly, the manipulated inputs $(\Delta C_{A0}, \Delta Q)$ are randomly and uniformly selected from a small neighborhood around their respective steady states C_{A0_s} and Q_s (i.e., 10 and 15 data points from ΔC_{A0} (kmol/m^3)~$U(-1.5, 1.5)$, and ΔQ (kJ/h)~$U(-5 \times 10^3, 5 \times 10^3)$ respectively). In total, a total of 9,600 state trajectories were generated.

As mentioned in Section 4.1.1, the introduction of initial conditions/collocation points into the loss function help to incorporate physics into the NN model. In this example, the collocation points comprise the initial system states ΔC_A and ΔT, and the manipulated inputs ΔC_{A0} and ΔQ. These collocation points are obtained by uniform sampling across the closed-loop stability region (this includes the points within

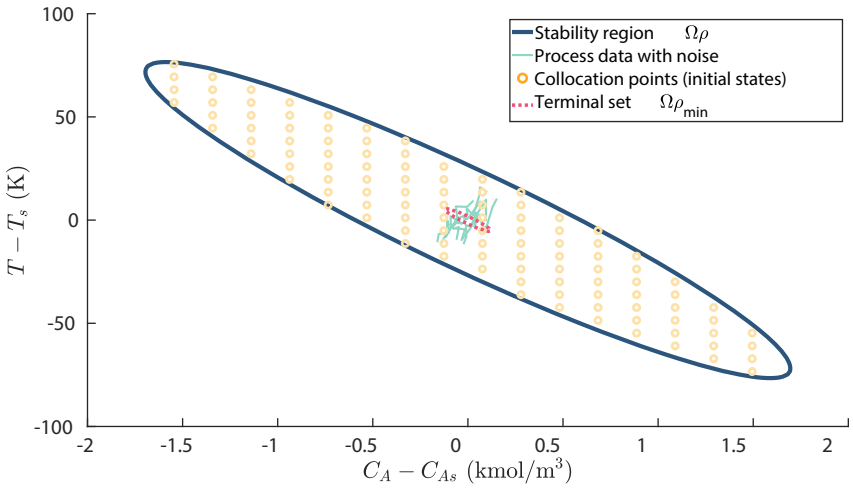

Figure 11: Collocation points sampled uniformly across the stability region Ω_ρ, together with a few examples of the noisy process state trajectories captured within a small neighborhood around the origin (i.e., steady-state C_{A_s} and T_s).

the region captured in the process data, as well as the area beyond). 110 pairs of initial states were sampled across the closed-loop stability region Ω_ρ, as shown in Figure 11. Each initial state was subjected to 500 uniformly sampled manipulated inputs in a sample-and-hold fashion. Hence, a total of 55,000 collocation points (i.e., the total number of combinations with 110 pairs of initial states and 500 pairs of manipulated inputs) were obtained. During the model training process, 80 % of the process data and collocation points were used for training, and the remaining 20 % were saved for validation.

The standard RNN model was trained solely with the process data. It is a purely data-driven model that serves as a baseline to evaluate the predictive capabilities of the PIRNN models in regions beyond the range provided by the training data. Additionally, a purely physics-based RNN model, i.e., PIRNN without $\text{MSE}_{\mathscr{X}}$, was developed. The PIRNN without MSE_χ model was trained only using the collocation points, without utilizing the observed process data. It serves as another baseline for comparison with the standard PIRNN. Finally, the standard PIRNN was created using both collocation points and process data, with its loss function described in Eq. (20). All three RNN models were developed in PyTorch and share the same network architecture, which consists of three hidden recurrent layers with 128, 256, and 64 recurrent units, respectively. The models also had the same parameter settings: the number of training epochs was set to 300, with Adam as the optimizer (learning rate of 0.001) and tanh as the activation function. The models had 4 input features and 2 output features. Specifically, given the initial state measurements (ΔC_A and ΔT at the current time step), and the manipulated inputs (ΔC_{A0} and ΔQ), the models are required to predict the future system states (i.e., ΔC_A and ΔT) over a sampling period $\Delta = 1 \times 10^{-2}$ h.

The open-loop state profiles predicted by the three models are presented in Figure 12. It can be seen from Figure 12 that the prediction performance of the standard RNN model starts to deviates from ground truth from $t = 0.25$ h onward, the time when the states begin to deviate from their respectively steady-state values. The unsatisfactory performance of RNN after $t = 0.25$ h highlights the poor generalizability of data-driven models when provided with unseen data beyond its training set. On the other hand, the standard PIRNN model that was trained with both process data and collocation points, demonstrated remarkable generalization performance, in the sense that its prediction matches the ground truth closely. Moreover, the PIRNN without $\text{MSE}_{\mathscr{X}}$ model was able to provide satisfactory prediction performance despite being a purely physics-driven model (i.e., no observed data was used for training). Hence, the strength of physics-informed ML can be corroborated from this example, where the predictive performance of the RNN models was significantly enhanced with the incorporation of physical knowledge into the models. The exceptional generalizability of physics-informed ML can be especially beneficial for controlling dynamic systems in the chemical industry, where the process data collected are often ill-sampled (e.g., concentrated within a small region around the steady-state set points). The code for PIRNN is available in our GitHub repository.[1]

5.2.2 Transfer learning RNNs

In the presence of data scarcity, transfer learning can be used to accelerate the training process and improve the

1 GitHub link to the PIRNN code: https://github.com/Keerthana-Vellayappan/Demonstration-of-Physics-Informed-Machine-Learning-Model.

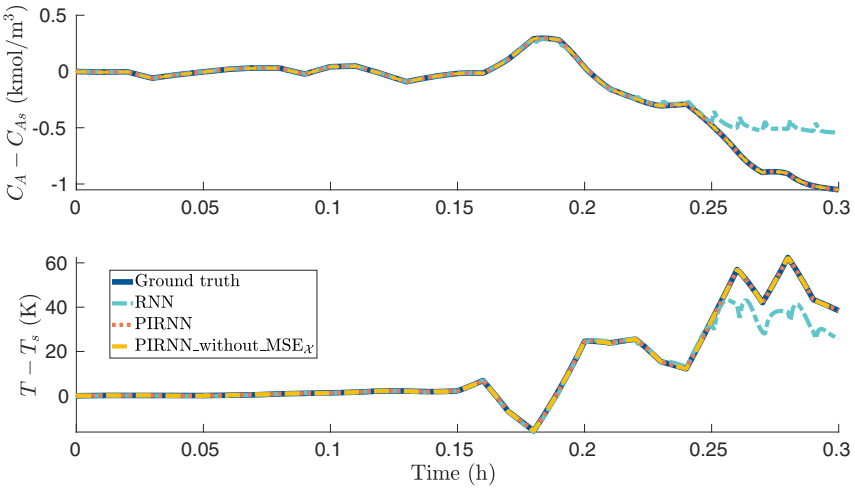

Figure 12: Comparison of open-loop state profiles (i.e., ΔC_A (top figure), and ΔT (bottom figure)) predicted by RNN (green dotted-dashed line), PIRNN (red dotted line), and purely physics-driven RNN (orange dashed line) with the ground truth (blue solid line). Noise-free state measurements were used to train the three RNN models.

generalization performance of RNNs for a target process with limited data using the pre-trained model for a similar source process with sufficient data. The model development framework and results of transfer learning based RNN models presented in this section are taken from Xiao et al. (2023). In Xiao et al. (2023), one source CSTR was selected for the construction of a TL-based model of a target CSTR process. Except for the ideal gas constant R, the two CSTRs had different parameter values. Specifically, the parameters of the source CSTR were 1.1 times its counterparts in the target CSTR. The model construction framework has been provided in Section 4.1.2. In essence, a single hidden layer RNN is developed using data from the source CSTR. Subsequently, the target model is obtained by adding a new RNN hidden layer to the pre-trained RNN source model and fine-tuned using the data from the target CSTR. To obtain the source model in Keras, we trained an RNN model with one hidden layer of 32 neurons using 42,840 training samples and 7,560 testing samples, all collected from the source CSTR. The training time for 150 training epochs was 112 s. The testing error of the source model is reported to be 1.834×10^{-5}, which is a sufficiently small modeling error using normalized data from the CSTR described in Eq. (31). Afterward, a TL-RNN model was built for the target CSTR by adding a hidden layer of size 32 to the pre-trained source model. Thus, the target model has two RNN hidden layers of 32 neurons. The training process for the TL-RNN is divided into two steps. In the first step, the parameters in the first hidden layer, i.e., the pre-trained source model, are set to be 'untrainable' by using the function 'model.layers[0].trainable = False' in Keras. In this stage, only the second hidden layer is trained. The second hidden layer was trained for 150 epochs. In the second step, all the hidden layers in the RNN model are set as 'trainable', and the entire model is further trained for 150 epochs. As a benchmark for comparison, a standard RNN model with two hidden layers each containing 32 neurons was developed and trained for 300 epochs, using solely the target data set.

Table 2 presents the training time and testing errors of TL-RNN and standard RNN trained different sizes of the target data set. Given sufficient target data (i.e., 16,800 training samples and 7,200 testing samples), it can be observed from Table 2 that the prediction performance of TL-RNN and RNN models are comparable (i.e., TL-RNN testing error = 3.144×10^{-5}, RNN testing error = 2.635×10^{-5}). This shows that the TL-RNN model can achieve similar results as the best performing RNN equivalent under standard training process. Moreover, it can be seen from Table 2 that the TL-RNN model used up less training time than the standard RNN model when trained with 16,800 data samples. This could be attributed to the fact that the TL-RNN model had less trainable parameters than the standard RNN during the first part of its training process, where the parameters in the first hidden layer of the TL-RNN were set to be untrainable. This reduction in trainable parameters could have accelerated the training process of TL-RNN. The results under 24,000 samples suggest that, when sufficient training samples are provided for the target process, transfer learning can achieve a similar performance as the standard RNN model while requiring less training time. Additionally, we consider the scenario where the target dataset contains fewer samples (i.e., 1,920 training samples and 1,280 testing samples). It is observed from Table 2 that in the case of a small dataset (i.e., 3,200 samples), the transfer learning RNN model can achieve better prediction performance while reducing the training time compared to the standard RNN model. The code for transfer learning is available in our GitHub repository.[2]

2 GitHub link to the transfer learning-based RNN code: https://github.com/MingXiaop/Transfer-Learning-for-nonlinear-chemical-process.

Table 2: Testing errors of standard and TL-RNNs.

	Data set size	Training time (s)	Testing error
TL-RNN	24,000	153.24	3.144×10^{-5}
Standard RNN	24,000	187.80	2.635×10^{-5}
TL-RNN	3,200	43.91	2.141×10^{-4}
Standard RNN	3,200	48.70	4.090×10^{-4}

5.2.3 Dropout and co-teaching RNNs with noisy data

Since neural networks are demonstrated to be able to mitigate the impact of Gaussian noise to some extent. In this section, we will use the findings in (Wu et al. 2021c) to understand the capability of the LSTM model to handle non-Gaussian noise, as well as how approaches such as dropout and co-teaching can help to improve the learning performances of the LSTM models developed with noisy data.

We will first explore the effect of implementing the MC dropout, proposed in Gal and Ghahramani (2016a,b), in an LSTM model. By treating the LSTM weights as random variables and finding the posterior distribution of the weights by sampling the network with randomly dropped out weights during testing, the MC dropout method helps quantify the uncertainty in the prediction and uses the information to update the weights. The open-loop prediction results of the standard LSTM and the dropout LSTM are presented in Figure 13 for comparison. As the predictions made by the LSTM model using MC dropout are stochastic in nature, the LSTM predictions were executed repeatedly 300 times to generate the predicted state trajectories distribution. The mean state trajectory is represented as a red line, and the 95 % standard deviation interval is marked by the gray region in Figure 13. It is observed that the prediction made by the standard LSTM model (yellow line), trained with non-Gaussian noise, deviates significantly from the ground truth (i.e., the nominal state trajectory in black), especially at the start. Conversely, the mean state trajectory predicted by the dropout LSTM shows a closer match to the ground truth, showing the capability of MC dropout in handling noisy data.

The effect of incorporating co-teaching into LSTM was also studied in Wu et al. (2021c). As mentioned in Section 4.2.2, co-teaching involves training two NN models. In Wu et al. (2021c), the co-teaching process starts by training the two LSTM models with a noisy dataset. Subsequently, the models iteratively identify and exchange clean data sequences and update their weights accordingly. This allows the co-teaching models to capture a balanced pattern that accounts for both noisy and clean data. To understand the effectiveness of the co-teaching method, the testing performances of the standard LSTM, dropout LSTM and co-teaching LSTM models were compared and are listed in Table 3. For fair comparison, all LSTM models were trained and tested on the same noisy dataset. The LSTM models also shared the same network structure and hyperparameters, i.e., the same number of neurons, layers, epochs, and activation functions. The difference between the predicted state trajectories and the underlying (noise-free) state trajectories, i.e., MSE, was chosen as the criterion for performance evaluation, where a smaller MSE value signifies better model performance.

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

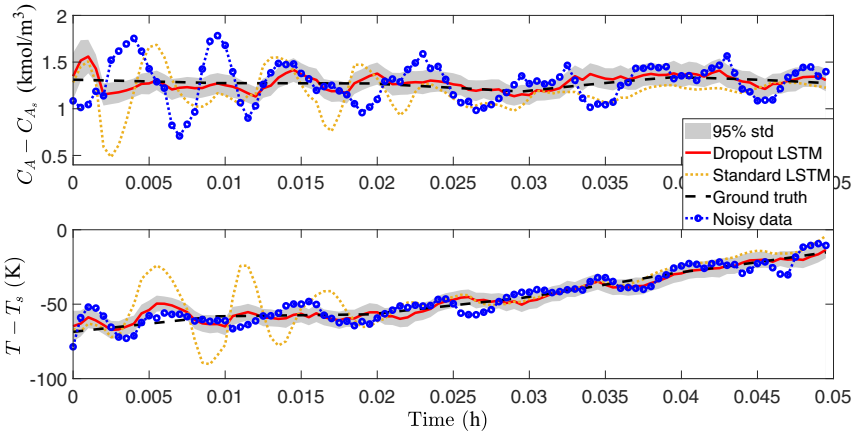

Figure 13: State profiles predicted by the dropout LSTM and the standard LSTM, where the red line is dropout LSTM, the black, dashed line is the ground truth, the yellow line is the standard LSTM, and the blue, dotted line is the noisy state measurement.

As shown in Table 3, the standard LSTM models had the highest MSE out of the three methods. Furthermore, when comparing standard LSTM trained noisy data (i.e., 1c in Table 3) with mixed LSTM trained data (i.e., 1b in Table 3), a slight improvement in model prediction was observed, highlighting the adverse impact noisy data have on model accuracy. These observations imply that the standard modeling approach cannot achieve the desired model accuracy without a high-quality dataset. However, co-teaching LSTM and dropout models developed with the same noisy dataset outperformed the standard LSTM models, with co-teaching LSTM achieving the best performance among all models, demonstrating the effectiveness of the co-teaching model in mitigating the impact of noisy labels.

5.2.4 Lipschitz-constrained NNs

By improving the model's robustness to noisy data, LCNN is an alternative approach to address noise in datasets. Specifically, LCNNs are designed to control and bound the Lipschitz constant of the neural network such that the models will be less sensitive and more robust to changes in the input. As there are various ways to construct an LCNN, we will use the SpectralDense layer method mentioned in Section 4.2.3 for demonstration and share the simulation results from Tan et al. (2024b). To assess the model's performance, we compared the LCNN models to the standard FNNs. Specifically, LCNNs were developed with SpectralDense hidden layers, while the conventional Dense FNNs were developed using the dense layers from Tensorflow with ReLU activation functions. Both SpectralDense LCNNs and Dense FNNs were trained on the same datasets, corrupted with Gaussian noise of a standard deviation of 0.1 or 0.2. Moreover, both models shared the same network structure of two hidden layers of the same size, either 640 or 1,280 neurons. The models were trained using the optimizer Adam and the training hyperparameters such as the number of epochs, batch size, early stopping callback configurations were kept the same for all models.

The testing errors of the models are provided in Table 4. As seen in Table 4, the testing errors of Dense FNNs, with orders of magnitude ranging between 10^{-3} and 10^{-2}, are significantly greater than those of the SpectralDense LCNNs, with order of magnitude of around 10^{-5}. The poor performance in Dense FNNs is likely to due to over-fitting as the testing errors were observed to share a similar order of magnitude as the variances of the Gaussian noise used, which are around 10^{-2}. To visualize how the magnitude of the network's Lipschitz constant affects its performance, the authors estimated the Lipschitz constants of the LCNNs and the Dense FNNs developed for the CSTR of Eq. (31). In particular, the Lipschitz Branch and Bound (LipBaB) algorithm developed by Bhowmick et al. (2021) was used to compute the Lipschitz constant for the Dense FNNs. For SpectralDense LCNNs, SVD of the final layer weight matrix was performed, and the spectral norm of this weight matrix was used as the upper bound of the Lipschitz constant. The results can also be found in Table 4. From Table 4, we see that the standard Dense FNNs have a much larger Lipschitz constant than its SpectralDense LCNN counterparts. This indicates that the LCNNs are theoretically less sensitive to input perturbations as compared to the Dense FNNs. This was verified in practice, where the SpectralDense LCNNs demonstrated superior performance over the Dense FNNs, when trained with noisy data. The Python code for developing LCNNs can be found in our GitHub repository.[3]

Table 4: Comparison of the testing errors (TEs) and Lipschitz constants (LCs) for various hidden layer architectures and standard deviation (SD) of noise introduced into the training dataset.

Hidden layers	Noise SD	LCNN TE ($\times 10^{-5}$)	Dense TE ($\times 10^{-2}$)	LCNN LC	Dense LC ($\times 10^{2}$)
(640, 640)	0.1	3.617	0.4573	1.119	2.447
(1,280, 1,280)	0.1	3.850	0.6928	1.116	2.682
(640, 640)	0.2	2.088	1.692	1.114	2.511
(1,280, 1,280)	0.2	1.393	3.095	1.107	9.285

5.2.5 Error-triggered online learning

Neural networks are generally trained offline using historical data, and cannot capture the real-time dynamics subject to process disturbances. Hence, online learning and updating of ML models can be a viable solution for systems with time-varying dynamics. To illustrate how online learning can help address process disturbances, we consider the CSTR of Eq. (31) with model variations caused by the following disturbances: (1) As a result of an upstream disturbance, the feed flow rate F becomes time-varying with the constraint: $0 \le F \le 12$ m^3/h. (2) Additionally, catalyst activation is taken into account during the operation of the CSTR of Eq. (31), resulting in a reduction in the reaction pre-exponential factor k_0 with the constraint: $0 < k_0 < 8.46 \times 10^6$ m^3/kmol h. In particular, the feed flow rate F increases to 12 m^3/h at $t = 0.05$ h, and k_0 gradually decreases to $0.8k_0$, $0.6k_0$ and $0.4k_0$ at $t = 0.1$ h, 0.2 h and 0.4 h, respectively, and remains unchanged afterward. In Wu et al. (2019b), the authors

3 GitHub link to LCNN code: https://github.com/killingbear999/lipschitz-constrained-neural-networks.

developed two RNN models for the CSTR of Eq. (31) subjected to the aforementioned disturbances, one being a standard RNN model trained offline using historical data, and the other one being an RNN model updated online using real-time data. The closed-loop state trajectories under Lyapunov MPC (LMPC) controllers designed with the two models are shown in Figure 14a. As shown in Figure 14a, the closed-loop state trajectory under LMPC using the standard RNN models (that is, without online update of the RNN model) exhibited oscillatory behavior around the origin due to disturbances. Whereas, the trajectory under the LMPC with online update of the RNN model was able to drive the closed-loop state into a small neighborhood around the origin successfully.

Figure 14b shows the evolution of the moving-horizon error detector $E_{rnn}(t)$ that is designed as the accumulated prediction error for the closed-loop system of Eq. (31) under the LMPC of Eq. (9) with online update of RNN models triggered by errors. It is shown that the update of RNN models is triggered two times throughout the operation.

Remark 6. Due to space constraints, we are unable to present all the NN modeling approaches that address each practical issue discussed in this article. Readers who are interested in reduced-order modeling, ML-based distributed MPC, and federated learning methods, which often require more complex process networks for demonstration, can refer to the references provided in the corresponding sections.

5.3 NN-based MPC

After we obtain the NN models that learn the dynamics of the CSTR of Eq. (31), NN-based MPC can be developed to control the

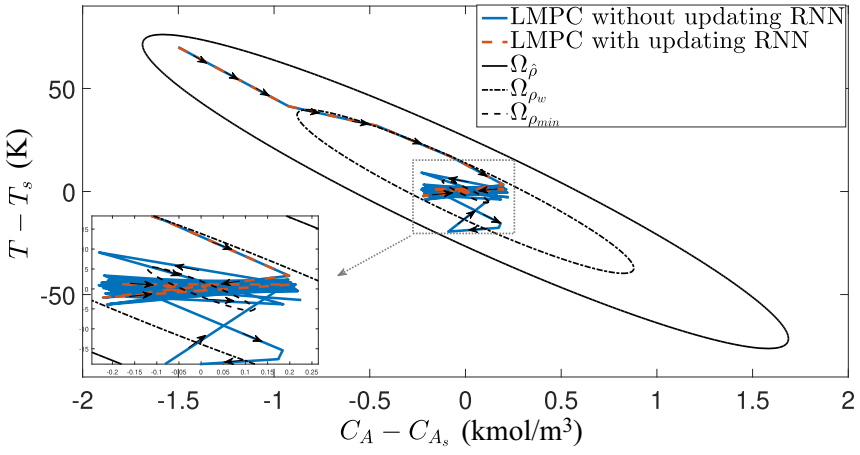

Figure 14: Closed-loop simulation results under LMPC using online update of RNN models. (a) The state-space profiles for the closed-loop CSTR under the LMPC of Eq. (9) with and without online update of RNN model for the initial condition (−1.5, 70). (b) Value of the prediction error $E_{rnn}(t)$ for the closed-loop system of Eq. (31) under the LMPC of Eq. (9) with error-triggered online update of RNN models.

system by manipulating C_{A0} and Q. A conventional LMPC scheme using standard RNN models has been developed in Ren et al. (2022) and has been shown to achieve the desired closed-loop performance by stabilizing the states in the steady state. The Python code for MPC using conventional RNN models can be found in our GitHub repository.[4] In this subsection, we again will focus more on novel designs of ML-MPCs that address the practical issues such as computational efficiency, safety and data security in real-world applications.

5.3.1 Convex MPC using input-convex NNs

While neural networks offer advantages in process modeling, ensuring computational efficiency is crucial for real-time optimization and control tasks. In a chemical plant, numerous operations require real-time or near-real-time control to maintain product quality, safety, and operational efficiency. Swift decision-making is pivotal for safety in chemical processes, as delays in addressing reactant changes can result in undesired reactions or unsafe conditions. Inspired by the fact that convex optimization is easier to solve than non-convex optimization, in this subsection, our goal is to preserve the convexity in neural-network-based predictive control that will be discussed later by developing input-convex NNs where the neural network outputs remain convex with respect to the input. Specifically, in addition to the input-convex feedforward neural network introduced in Section 4.5.1, there are a variety of input-convex NNs in the family of RNNs such as input-convex RNNs and input-convex LSTMs. Specifically, we develop input-convex LSTM (ICLSTM), following the formulation in Wang et al. (2025), and compare its performance in closed-loop control with the MPC using plain LSTM model. Subsequently, we consider a simple MPC scheme using a neural network model as the prediction model given by the following optimization problem:

$$\mathscr{L} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} J(\widetilde{x}(t), u(t))\mathrm{dt} \tag{32a}$$

$$\text{s.t. } \dot{\widetilde{x}}(t) = F_{nn}(\widetilde{x}(t), u(t)) \tag{32b}$$

$$u(t) \in U, \ \forall t \in [t_k, t_{k+N}) \tag{32c}$$

$$\widetilde{x}(t_k) = x(t_k) \tag{32d}$$

where $\widetilde{x}$ is the predicted state trajectory, $S(\Delta)$ is the set of piecewise constant functions with period Δ, and N is the number of sampling periods in the prediction horizon.

Table 5: Convergence runtime of MPCs using LSTM and ICLSTM.

$[C_{A_i}, T_i]$	Plain LSTM		ICLSTM
	Time (s)	% Decrease	Time (s)
$[-1.5, 70]$	1,688.68 ± 3.40	78.08 %	370.17 ± 11.22
$[-1.3, 60]$	1,632.31 ± 7.05	65.46 %	563.72 ± 17.80
$[-1, 55]$	1,391.79 ± 3.36	68.73 %	435.26 ± 4.08
$[-1.25, 50]$	1,453.57 ± 27.28	68.69 %	455.10 ± 3.58
$[-0.75, 40]$	1,079.38 ± 15.65	28.27 %	774.26 ± 4.90
$[-0.5, 30]$	764.51 ± 14.80	44.11 %	427.26 ± 2.87
$[-0.45, 15]$	757.02 ± 13.81	41.89 %	439.91 ± 5.89
$[1.5, -70]$	1,556.54 ± 39.61	11.16 %	1,382.98 ± 1.48
$[1.35, -55]$	1,472.01 ± 11.43	−12.71 %	1,659.11 ± 35.60
$[1.1, -45]$	1,099.92 ± 17.66	4.42 %	1,051.35 ± 36.47
$[0.9, -30]$	1,453.77 ± 22.88	59.61 %	587.12 ± 17.25
$[0.75, -40]$	1,030.41 ± 9.11	9.03 %	937.35 ± 28.40
$[0.6, -25]$	927.37 ± 33.98	18.96 %	751.56 ± 8.47
$[0.4, -35]$	765.72 ± 20.85	39.08 %	466.47 ± 8.45
$[0.2, -15]$	725.71 ± 19.64	46.74 %	386.48 ± 5.55
Average	1,186.6	40.0 %	712.5

The objective function $\mathscr{L}$ in Eq. (32a) incorporates a cost function J in terms of the system states x and the control actions u. The dynamic function $F_{nn}(\widetilde{x}(t), u(t))$ in Eq. (32b) is parameterized as recurrent neural networks (i.e., plain LSTM and ICLSTM). In this experiment, the PyIpopt library was executed on an Intel Core i7-12700 processor with 64 GB of RAM, using 15 different initial conditions within the stability region (i.e., covering the whole stability region). Table 5 presents the average runtime across 3 runs for each case and their corresponding percentage decrease with respect to ICLSTM, showing that ICLSTM-based MPC yields an improvement in convergence runtime. Specifically, it attains an average percentage decrease of 40.0 % compared to plain LSTM. The Python code for MPC using ICLSTM models can be found in our GitHub repository.[5]

5.3.2 Safe ML-based MPC

Safe MPCs should be developed to ensure that process operations remain within the safe operating region, particularly when there are potential unsafe operating conditions in chemical processes. CLBF functions can be incorporated into the MPC scheme (termed CLBF-MPC) to regulate the CSTR of Eq. (31) to the steady-state while avoiding the unsafe operation at the same time. The CLBF-MPC scheme is formulated by the following optimization problem (Wu et al. 2019a):

4 GitHub link to RNN-based MPC code: https://github.com/GuoQWu/Machine-learning-based-model-predictive-control.

5 GitHub link to ICLSTM-based MPC code: https://github.com/killingbear999/ICLSTM.

$$\min_{u\in S(\Delta)} \int_{t_k}^{t_{k+N}} l_t(\tilde{x}(t), u(t))\mathrm{d}t \tag{33a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{33b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{33c}$$

$$u(t) \in U, \ \forall\ t \in [t_k, t_{k+N}) \tag{33d}$$

$$\dot{W}_c\ (x(t_k), u(t_k)) \le \dot{W}_c\ (x(t_k), \Phi(x(t_k))), \tag{33e}$$

$$\text{if } W_c(x(t_k)) > \rho'_{\min} \text{ and } x(t_k) \notin \mathscr{B}_\delta(x_e)$$

$$W_c(\tilde{x}(t)) \le \rho'_{\min}, \ \forall\ t \in [t_k, t_{k+N}), \quad \text{if } W_c(x(t_k)) \le \rho'_{\min} \tag{33f}$$

$$W_c(\tilde{x}(t)) < W_c(x(t_k)), \ \forall\ t \in (t_k, t_{k+N}), \quad \text{if } x(t_k) \in \mathscr{B}_\delta(x_e) \tag{33g}$$

where Δ is the sampling period, $S(\Delta)$ is the set of piecewise constant functions with time interval Δ, $\tilde{x}(t)$ is the predicted state trajectory, and N is the number of sampling steps in the prediction horizon. We use $\dot{W}_c(x,u)$ to represent $\frac{\partial W_c(x)}{\partial x}(F_{nn}(x,u))$. The optimization problem of Eq. (33) is to minimize the object function of Eq. (33a) subject to the constraints of Eqs. (33b)–(33g). The NN model that captures the dynamics of Eq. (31) can be used as the prediction model in Eq. (33b). The initial condition for this prediction model is determined by the current state measurement, as shown in Eq. (33c). The constraints outlined in Eqs. (33e)–(33g) ensure that the closed-loop state remains bounded within a small neighborhood around the origin (i.e., $\mathscr{U}_{\rho'_{\min}}$) and does not enter the unsafe region for all times. Specifically, when $x(t_k)$ is outside of $\mathscr{U}_{\rho'_{\min}}$ and $x(t_k) \notin \mathscr{B}_\delta(x_e)$, the constraint of Eq. (33e) drives the closed-loop state into a smaller level set of $W_c(x)$ by decreasing the value of $W_c(\tilde{x})$ along the predicted state trajectory at least at the rate under the CLBF-based controller $u = \Phi(x)$. When $x(t_k)$ enters $\mathscr{U}_{\rho'_{\min}}$ (i.e., $x(t_k)$ is also bounded in a small ball around the origin $\mathscr{B}_d(0) \coloneqq \{x \in \mathbf{R}^n \mid |x| \le d\}$), the constraint of Eq. (33f) maintains the closed-loop state inside $\mathscr{B}_d(0)$ afterwards. However, if the state is trapped in other stationary points during the path towards the origin, i.e., $x(t_k) \in \mathscr{B}_\delta(x_e)$, we activate the constraint of Eq. (33g) to drive the state away from x_e in the direction of decreasing $W_c(x)$.

We consider a bounded unsafe region $\mathscr{D}_b$ in state-space, and demonstrate that the CLBF-MPC of Eq. (33) can drive the state to a small neighborhood around the origin while not entering the unsafe region. Specifically, the unsafe region is defined as an ellipse: $\mathscr{D}_b \coloneqq \left\{x \in \mathbf{R}^2 \mid F(x) = \frac{(x_1+0.92)^2}{1} + \frac{(x_2-42)^2}{500} < 0.06\right\}$. $\mathscr{H}$ is defined as $\mathscr{H} \coloneqq \{x \in \mathbf{R}^2 \mid F(x) < 0.07\}$. The CLBF is designed as the weighted sum of the following control barrier function $B(x)$ and control Lyapunov function $V(x)$:

$$B(x) = \begin{cases} e^{\frac{F(x)}{F(x)-0.07}} - e^{-6}, & \text{if } \ x \in \mathscr{H} \\ -e^{-6}, & \text{if } \ x \notin \mathscr{H} \end{cases} \tag{34}$$

and $V(x) = x^T P x$ with the following positive definite P matrix:

$$P = \begin{bmatrix} 1,060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{35}$$

In Figure 15, it is demonstrated that for all initial states x_0 in $\mathscr{U}_{\hat{\rho}}$ (marked by circles), the closed-loop trajectories avoid the bounded unsafe region $\mathscr{D}_b$ that is embedded within $\mathscr{U}_{\hat{\rho}}$ (a subset of the safe operating region $\mathscr{U}_\rho$), and ultimately converges to $\mathscr{U}_{\rho_{\min}}$ under the CLBF-MPC of Eq. (33).

Remark 7. Although a CSTR example was used to illustrate the applications of various machine learning modeling and ML-based MPC methods, it is important to note that ML-based MPC can be applied to a variety of complex chemical engineering problems. Due to space constraints, we will not provide a detailed discussion in the review; however, we have provided some examples on the application of ML models, as well as, ML-based MPC methods to model and control complex systems, for interested readers seeking more examples in this area. For example, neural network models have been applied to model an industrial ethylene splitter in Jalanko et al. (2021) and experimental electrochemical reactors in Çıtmacı et al. (2022) and Luo et al. (2022). Moreover, an LSTM-based MPC method has been developed in Luo et al. (2023) for the same electrochemical reactor of Çıtmacı et al. (2022). Other notable works on ML-based MPC include: using an LSTM-based

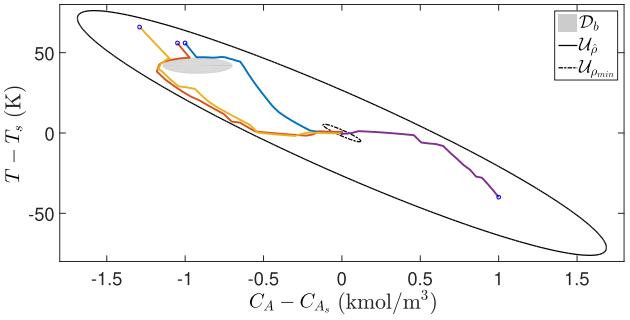

Figure 15: Closed-loop state trajectories for the system of Eq. (31) under the CLBF-MPC using an RNN model. The initial conditions are marked by circles, and the set of bounded unsafe states $\mathscr{D}_b$ is the gray area embedded within $\mathscr{U}_{\hat{\rho}}$.

economic MPC to control the heating, ventilation, and air conditioning (HVAC) system of a building in Ellis and Chinde (2020), and using an ANN-based MPC to control the film properties in the thin film chemical deposition of quantum dots in Sitapure and Kwon (2022).

6 Conclusion and outlook

The tutorial provided an overview of machine learning-based model predictive control methods, highlighting both theoretical insights and practical challenges associated with the development of NNs and the incorporation of NNs into MPC. Closed-loop stability of ML-based MPC was first established based on the generalization error analysis for NNs. Various ML methods such as physics-informed ML, transfer learning, and novel designs of NN architectures were discussed alongside advanced control methods to address the practical challenges including data scarcity, data quality, the curse of dimensionality, model uncertainty, computational efficiency, and safety in ML-MPC. Finally, a chemical process example was studied to demonstrate the effectiveness of various ML-MPC methods to address the aforementioned practical issues.

In addition to the topics covered in this paper, several emerging areas in ML-based MPC require significant attention for future research. For example, explainable AI (XAI) is critical to improving the transparency, trustworthiness, and usability of ML models in MPC. By understanding how a neural network arrives at its predictions, users can trust more on the model, and identify the errors more effectively in real-world applications. Although neural networks are powerful tools for learning complex patterns and making predictions across various domains, they are typically developed as black-box models with inherent complexity, which makes it challenging to understand the reasoning behind their outputs. Physics-informed ML provides one solution to incorporate domain knowledge into NN models, yet it does not completely address the challenge of model explainability. One common approach for XAI is SHapley Additive exPlanations (SHAP). SHAP is a method based on cooperative game theory that assigns each feature an importance value for a particular prediction. It provides a unified framework to explain the output of any machine learning model by attributing the prediction outcome to different input features. However, developing suitable XAI methods to explain predictions, limitations, and resulting behaviors of neural network models in MPC remains an ongoing challenge.

Regarding physics-informed machine learning, while this review paper discusses several approaches integrating physics knowledge (e.g., first-principles models and structural process knowledge) into NN development, there are numerous types of knowledge that can improve model performance. In many ML-MPC applications, NN models are initially trained offline until achieving sufficiently small errors before incorporation into MPC. However, this process involves extensive data collection and training, potentially consuming time and resources. Therefore, a future direction is to integrate stability requirements into NN model development, ensuring that NN models naturally meet the MPC stability criteria and can be easily implemented within MPC frameworks (Tan et al. 2024a). Additionally, for modeling distributed systems, knowledge of network structure (i.e., units [nodes] and their relationships [edges]) can be integrated into the development of graph neural networks (GNNs) to improve the modeling accuracy. Overall, there are various types of domain knowledge that can be integrated into neural networks tailored to specific ML-MPC applications in different ways (e.g., loss function, network architecture, weight constraints, learning algorithms, etc.).

To successfully implement ML-MPC in real-world large-scale systems, addressing adaptability and scalability is important to ensure computational efficiency and maintaining performance across diverse applications. Transfer learning offers a promising approach by leveraging knowledge from one process to another in modeling and control tasks for process scale-up. However, finding a suitable source process that closely matches the target process can be challenging in practice. Inspired by the success of large language models in many recent studies and applications, a compelling future direction is to develop a single, universal neural network (referred to as a foundation model) capable of rapidly adapting to model any new chemical process (Wang and Wu 2024c). Foundation models have shown success in fields such as computer science, chemistry, and material sciences. In the field of chemical engineering, large language models have been applied in Hirtreiter et al. (2024) to generate control structures for process flow diagrams (PFDs) from PFDs without control structures, as part of an effort to automate the generation of piping and instrumentation diagrams (P&IDs). However, the application of foundation models to chemical process modeling and control is still in its infancy (Decardi-Nelson et al. 2024). This is partly due to the complexity of chemical engineering, which involves large-scale industrial processes characterized by proprietary complex data that is rarely shared publicly by industries. Additionally, adapting ML-based MPC from a small-scale to a large-scale system involves several key considerations such as real-time computation requirements, availability of sensor data and sensor-related issues (e.g., missing, delayed, and asynchronous measurements), and optimization of MPC hyper-parameters across different scales. Addressing these challenges not only enables more efficient utilization of data but also improves the

applicability of ML-MPC in various chemical engineering applications.

Research ethics: Not applicable.
Informed consent: Not applicable.
Author contributions: Z.W. and P.D.C. conceptualized the review and oversaw all aspects of the project. Z.W. and W.W. were responsible for the initial literature review. Z.W. took the lead in writing the original manuscript, with significant inputs from W.W. and F.A., Y.W., F.A., A.A. and Y.K. contributed significantly to the review and editing process. All authors have accepted responsibility for the entire content of this manuscript and approved its submission.
Use of Large Language Models, AI and Machine Learning Tools: None declared.
Conflict of interest: The authors state no conflict of interest.
Research funding: Financial support from the National Science Foundation, the Department of Energy, NRF-CRP (27-2021-0001), MOE AcRF Tier 1 FRC Grant (22-5367-A0001), Singapore, and A*STAR MTC YIRG 2022 (M22K3c0093), Singapore is gratefully acknowledged.
Data availability: The raw data can be obtained on request from the corresponding authors.

References

Abbasi, M., Santos, B.P., Pereira, T.C., Sofia, R., Monteiro, N.R., Simões, C.J., Brito, R.M., Ribeiro, B., Oliveira, J.L., and Arrais, J.P. (2022). Designing optimized drug candidates with generative adversarial network. *J. Cheminf.* 14: 40.

Abdullah, F. and Christofides, P.D. (2023a). Data-based modeling and control of nonlinear process systems using sparse identification: an overview of recent results. *Comput. Chem. Eng.* 174: 108247.

Abdullah, F. and Christofides, P.D. (2023b). Real-time adaptive sparse-identification-based predictive control of nonlinear processes. *Chem. Eng. Res. Des.* 196: 750–769.

Abdullah, F., Wu, Z., and Christofides, P.D. (2021a). Data-based reduced-order modeling of nonlinear two-time-scale processes. *Chem. Eng. Res. Des.* 166: 1–9.

Abdullah, F., Wu, Z., and Christofides, P.D. (2021b). Sparse-identification-based model predictive control of nonlinear two-time-scale processes. *Comput. Chem. Eng.* 153: 107411.

Abdullah, F., Alhajeri, M.S., and Christofides, P.D. (2022a). Modeling and control of nonlinear processes using sparse identification: using dropout to handle noisy data. *Ind. Eng. Chem. Res.* 61: 17976–17992.

Abdullah, F., Wu, Z., and Christofides, P.D. (2022b). Handling noisy data in sparse model identification using subsampling and co-teaching. *Comput. Chem. Eng.* 157: 107628.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, August 4–8, 2019: optuna: a next-generation hyperparameter optimization framework*. Association for Computing Machinery, Anchorage, AK, USA, pp. 2623–2631.

Akpinar, N.-J., Kratzwald, B., and Feuerriegel, S. (2019). Sample complexity bounds for recurrent neural networks with application to combinatorial graph problems. *arXiv preprint arXiv:1901.10289*.

Alhajeri, M.S., Abdullah, F., Wu, Z., and Christofides, P.D. (2022). Physics-informed machine learning modeling for predictive control using noisy data. *Chem. Eng. Res. Des.* 186: 34–49.

Alhajeri, M.S., Ren, Y.M., Ou, F., Abdullah, F., and Christofides, P.D. (2024). Model predictive control of nonlinear processes using transfer learning-based recurrent neural networks. *Chem. Eng. Res. Des.* 205: 1–12.

Ali, M., Cai, X., Khan, F.I., Pistikopoulos, E.N., and Tian, Y. (2023). Dynamic risk-based process design and operational optimization via multi-parametric programming. *Digit. Chem. Eng.* 7: 100096.

Amos, B., Xu, L., and Kolter, J.Z. (2017). *Proceedings of the 34th international conference on machine learning, August 6–11, 2017: input convex neural networks*. PMLR, Sydney, Australia, pp. 146–155.

Anil, C., Lucas, J., and Grosse, R. (2019). *Proceedings of the 36th international conference on machine learning, June 9–15, 2019: sorting out Lipschitz function approximation*. PMLR, California, USA, pp. 291–301.

Antonelo, E.A., Camponogara, E., Seman, L.O., Jordanou, J.P., de Souza, E.R., and Hübner, J.F. (2024). Physics-informed neural nets for control of dynamical systems. *Neurocomputing* 579: 127419.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). *Proceedings of the 34th international conference on machine learning, August 6–11, 2017: Wasserstein generative adversarial networks*. PMLR, Sydney, Australia, pp. 214–223.

Arnold, F. and King, R. (2021). State-space modeling for control based on physics-informed neural networks. *Eng. Appl. Artif. Intell.* 101: 104195.

Bangi, M.S.F., Kao, K., and Kwon, J.S.-I. (2022). Physics-informed neural networks for hybrid modeling of lab-scale batch fermentation for β-carotene production using Saccharomyces cerevisiae. *Chem. Eng. Res. Des.* 179: 415–423.

Bank, D., Koenigstein, N., and Giryes, R. (2023) Autoencoders. In: *Machine learning for data science handbook: data mining and knowledge discovery handbook*. Springer, Cham, pp. 353–374.

Bartlett, P.L., Foster, D.J., and Telgarsky, M.J. (2017). Spectrally-normalized margin bounds for neural networks. In: *Advances in neural information processing systems*, Vol. 30. Curran Associates, Inc, Red Hook, NY.

Batra, R., Dai, H., Huan, T.D., Chen, L., Kim, C., Gutekunst, W.R., Song, L., and Ramprasad, R. (2020). Polymers for extreme conditions designed using syntax-directed variational autoencoders. *Chem. Mater.* 32: 10489–10500.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2006). Analysis of representations for domain adaptation. In: *Advances in neural information processing systems*, Vol. 19. MIT Press, Cambridge, MA.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J.W. (2010). A theory of learning from different domains. *Mach. Learn.* 79: 151–175.

Berberich, J. and Allgöwer, F. (2024). An overview of systems-theoretic guarantees in data-driven model predictive control. *arXiv preprint arXiv:2406.04130*.

Berberich, J., Köhler, J., Müller, M.A., and Allgöwer, F. (2020). Data-driven model predictive control with stability and robustness guarantees. *IEEE Trans. Automat. Control* 66: 1702–1717.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In: *Advances in neural information processing systems*, Vol. 24. Curran Associates, Inc, Red Hook, NY.

Bergstra, J., Yamins, D., and Cox, D. (2013). *Proceedings of the 30th international conference on machine learning, June 16–21, 2013: making a science of model search: hyperparameter optimization in hundreds of*

dimensions for vision architectures. PMLR, Atlanta, GA, USA, pp. 115–123.

Bhadriraju, B., Narasingam, A., and Kwon, J.S.-I. (2019). Machine learning-based adaptive model identification of systems: application to a chemical process. *Chem. Eng. Res. Des.* 152: 372–383.

Bhowmick, A., D'Souza, M., and Raghavan, G.S. (2021) LipBaB: computing exact Lipschitz constant of ReLU networks. In: *Artificial neural networks and machine learning – ICANN 2021*. Springer, Cham, pp. 151–162.

Bi, K., Beykal, B., Avraamidou, S., Pappas, I., Pistikopoulos, E.N., and Qiu, T. (2020). Integrated modeling of transfer learning and intelligent heuristic optimization for a steam cracking process. *Ind. Eng. Chem. Res.* 59: 16357–16367.

Bitmead, R.R., Gevers, M., and Wertz, V. (1990). *Adaptive optimal control the thinking man's GPC*. Prentice Hall, Victoria, Australia.

Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. (2007). Learning bounds for domain adaptation. In: *Advances in neural information processing systems*, Vol. 20. Curran Associates, Inc, Red Hook, NY.

Bo, S., Agyeman, B.T., Yin, X., and Liu, J. (2023). Control invariant set enhanced safe reinforcement learning: improved sampling efficiency, guaranteed stability and robustness. *Comput. Chem. Eng.* 179: 108413.

Bonassi, F., Farina, M., Xie, J., and Scattolini, R. (2022). On recurrent neural networks for learning-based control: recent results and ideas for future developments. *J. Process Control* 114: 92–104.

Bond-Taylor, S., Leach, A., Long, Y., and Willcocks, C.G. (2021). Deep generative modelling: a comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. *IEEE Trans. Pattern Anal. Mach. Intell.* 44: 7327–7347.

Bradford, E., Imsland, L., Zhang, D., and del Rio Chanona, E.A. (2020). Stochastic data-driven model predictive control using Gaussian processes. *Comput. Chem. Eng.* 139: 106844.

Briceno-Mena, L.A., Romagnoli, J.A., and Arges, C.G. (2022). PemNet: a transfer learning-based modeling approach of high-temperature polymer electrolyte membrane electrochemical systems. *Ind. Eng. Chem. Res.* 61: 3350–3357.

Brunke, L., Greeff, M., Hall, A.W., Yuan, Z., Zhou, S., Panerati, J., and Schoellig, A.P. (2022). Safe learning in robotics: from learning-based control to safe reinforcement learning. *Annu. Rev. Control Robot. Auton. Syst.* 5: 411–444.

Bünning, F., Schalbetter, A., Aboudonia, A., de Badyn, M.H., Heer, P., and Lygeros, J. (2021). *Proceedings of the 3rd conference on learning for dynamics and control, June 7–8, 2021: input convex neural networks for building MPC*. PMLR, pp. 251–262.

Cai, S., Wang, Z., Fuest, F., Jeon, Y.J., Gray, C., and Karniadakis, G.E. (2021). Flow over an espresso cup: inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks. *J. Fluid Mech.* 915: A102.

Chandrasekar, A., Abdulhussain, H., Thompson, M.R., and Mhaskar, P. (2024). Utilizing neural networks for image-based model predictive controller of a batch rotational molding process. *IFAC-PapersOnLine* 58: 470–475.

Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Comput. Electr. Eng.* 40: 16–28.

Chang, H.-C. and Aluko, M. (1984). Multi-scale analysis of exotic dynamics in surface catalyzed reactions–I: justification and preliminary model discriminations. *Chem. Eng. Sci.* 39: 37–50.

Chen, H. and Allgöwer, F. (1998). A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica* 34: 1205–1217.

Chen, W.-H. and You, F. (2021). Semiclosed greenhouse climate control under uncertainty via machine learning and data-driven robust model predictive control. *IEEE Trans. Control Syst. Technol.* 30: 1186–1197.

Chen, R.T., Rubanova, Y., Bettencourt, J., and Duvenaud, D.K. (2018a) Neural ordinary differential equations. In: *Advances in neural information processing systems*, Vol. 31. Curran Associates, Inc, Red Hook, NY.

Chen, S., Saulnier, K., Atanasov, N., Lee, D.D., Kumar, V., Pappas, G.J., and Morari, M. (2018b). *Proceedings of the 2018 annual American control conference, June 27–29, 2018: approximating explicit model predictive control using constrained neural networks*. Milwaukee, Wisconsin, USA, pp. 1520–1527.

Chen, Y., Shi, Y., and Zhang, B. (2018c). Optimal control via neural networks: a convex approach. *arXiv preprint arXiv:1805.11835*.

Chen, M., Li, X., and Zhao, T. (2019). On generalization bounds of a family of recurrent neural networks. *arXiv preprint arXiv:1910.12947*.

Chen, S., Wu, Z., and Christofides, P.D. (2020a). Decentralized machine-learning-based predictive control of nonlinear processes. *Chem. Eng. Res. Des.* 162: 45–60.

Chen, S., Wu, Z., Rincon, D., and Christofides, P.D. (2020b). Machine learning-based distributed model predictive control of nonlinear processes. *AIChE J.* 66: e17013.

Chen, S., Wu, Z., and Christofides, P.D. (2022a). Machine-learning-based construction of barrier functions and models for safe model predictive control. *AIChE J.* 68: e17456.

Chen, S., Wu, Z., and Christofides, P.D. (2022b). Statistical machine-learning-based predictive control using barrier functions for process operational safety. *Comput. Chem. Eng.* 163: 107860.

Cheng, F., He, Q.P., and Zhao, J. (2019). A novel process monitoring approach based on variational recurrent autoencoder. *Comput. Chem. Eng.* 129: 106515.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), October 25–29, 2014: learning phrase representations using RNN encoder–decoder for statistical machine translation*. Doha, Qatar, pp. 1724–1734.

Christofides, P.D., Scattolini, R., De La Pena, D.M., and Liu, J. (2013). Distributed model predictive control: a tutorial review and future research directions. *Comput. Chem. Eng.* 51: 21–41.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. (2017). *Proceedings of the 34th international conference on machine learning, August 6–11, 2017: parseval networks: improving robustness to adversarial examples*. PMLR, Sydney, Australia, pp. 854–863.

Çıtmacı, B., Luo, J., Jang, J.B., Canuso, V., Richard, D., Ren, Y.M., Morales-Guio, C.G., and Christofides, P.D. (2022). Machine learning-based ethylene concentration estimation, real-time optimization and feedback control of an experimental electrochemical reactor. *Chem. Eng. Res. Des.* 185: 87–107.

Daoutidis, P., Lee, J.H., Rangarajan, S., Chiang, L., Gopaluni, B., Schweidtmann, A.M., Harjunkoski, I., Mercangöz, M., Mesbah, A., Boukouvala, F., et al. (2023). Machine learning in process systems engineering: challenges and opportunities. *Comput. Chem. Eng.* 181: 108523.

David, S.B., Lu, T., Luu, T., and Pál, D. (2010). *Proceedings of the 13th international conference on artificial intelligence and statistics, May 13–15, 2010: impossibility theorems for domain adaptation*. JMLR Workshop and Conference Proceedings, Sardinia, Italy, pp. 129–136.

de Giuli, L.B., La Bella, A., and Scattolini, R. (2024). Physics-informed neural network modeling and predictive control of district heating systems. *IEEE Trans. Control Syst. Technol.* 32: 1182–1195.

de Vos, B.D., Jansen, G.E., and Išgum, I. (2023). Stochastic co-teaching for training neural networks with unknown levels of label noise. *Sci. Rep.* 13: 16875.

Decardi-Nelson, B., Alshehri, A.S., Ajagekar, A., and You, F. (2024). Generative AI and process systems engineering: the next Frontier. *Comput. Chem. Eng.* 187: 108723.

Degeest, A., Verleysen, M., and Frénay, B. (2019) About filter criteria for feature selection in regression. In: *Advances in computational intelligence*. Springer, Cham, pp. 579–590.

Dev, P., Jain, S., Arora, P.K., and Kumar, H. (2021). Machine learning and its impact on control systems: a review. *Mater. Today: Proc.* 47: 3744–3749.

Dobbelaere, M.R., Plehiers, P.P., Van de Vijver, R., Stevens, C.V., and Van Geem, K.M. (2021). Machine learning in chemical engineering: strengths, weaknesses, opportunities, and threats. *Engineering* 7: 1201–1211.

Dong, D. and McAvoy, T. (1996). Nonlinear principal component analysis–based on principal curves and neural networks. *Comput. Chem. Eng.* 20: 65–78.

Elgamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theor.* 31: 469–472.

Ellis, M.J. and Chinde, V. (2020). An encoder–decoder LSTM-based EMPC framework applied to a building HVAC system. *Chem. Eng. Res. Des.* 160: 508–520.

Everett, M. (2021). *Proceedings of the 60th IEEE conference on decision and control (CDC), December 14–17, 2021: neural network verification in control*. IEEE, Austin, TX, USA, pp. 6326–6340.

Farokhi, F., Shames, I., and Batterham, N. (2017). Secure and private control using semi-homomorphic encryption. *Control Eng. Pract.* 67: 13–20.

Federer, H. (2014). *Geometric measure theory*. Springer Berlin Heidelberg, Heidelberg.

Ferramosca, A., Limon, D., González, A.H., Odloak, D., and Camacho, E.F. (2010). MPC for tracking zone regions. *J. Process Control* 20: 506–516.

Gal, Y. and Ghahramani, Z. (2016a). *Proceedings of the 33rd international conference on machine learning, June 19–24, 2016: dropout as a Bayesian approximation: representing model uncertainty in deep learning*. PMLR, New York, USA, pp. 1050–1059.

Gal, Y. and Ghahramani, Z. (2016b) A theoretically grounded application of dropout in recurrent neural networks. In: *Advances in neural information processing systems*, Vol. 29. Curran Associates, Inc, Red Hook, NY.

Gao, Y., Yan, S., Zhou, J., Cannon, M., Abate, A., and Johansson, K.H. (2024). *Proceedings of the 6th annual learning for dynamics & control conference, July 15–17, 2024: learning-based rigid tube model predictive control*. PMLR, Oxford, UK, pp. 492–503.

Garcıa, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* 16: 1437–1480.

Gentry, C., Halevi, S., and Smart, N.P. (2012). *Proceedings of the annual cryptology conference– CRYPTO 2012, August 19–23, 2012: homomorphic evaluation of the AES circuit*. Springer Berlin Heidelberg, Santa Barbara, CA, USA, pp. 850–867.

Golowich, N., Rakhlin, A., and Shamir, O. (2018). *Proceedings of the 31st conference on learning theory, July 6–9, 2018: size-independent sample complexity of neural networks*. PMLR, Stockholm, Sweden, pp. 297–299.

González, A.H. and Odloak, D. (2009). A stable MPC with zone control. *J. Process Control* 19: 110–122.

Gonzalez, C., Asadi, H., Kooijman, L., and Lim, C.P. (2023). Neural networks for fast optimisation in model predictive control: a review. *arXiv preprint arXiv:2309.02668*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In: *Advances in neural information processing systems*, Vol. 27. Curran Associates, Inc, Red Hook, NY.

Gouk, H., Frank, E., Pfahringer, B., and Cree, M.J. (2021). Regularisation of neural networks by enforcing Lipschitz continuity. *Mach. Learn.* 110: 393–416.

Grimstad, B. and Andersson, H. (2019). ReLU networks as surrogate models in mixed-integer linear programs. *Comput. Chem. Eng.* 131: 106580.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A.C. (2017). Improved training of Wasserstein GANs. In: *Advances in neural information processing systems*, Vol. 30. Curran Associates, Inc, Red Hook, NY.

Guo, J., Du, W., and Nascu, I. (2020). Adaptive modeling of fixed-bed reactors with multicycle and multimode characteristics based on transfer learning and just-in-time learning. *Ind. Eng. Chem. Res.* 59: 6629–6637.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. (2018). Co-teaching: robust training of deep neural networks with extremely noisy labels. In: *Advances in neural information processing systems*, Vol. 31. Curran Associates, Inc, Red Hook, NY.

Han, X., Zhang, L., Zhou, K., and Wang, X. (2019). ProGAN: protein solubility generative adversarial nets for data augmentation in DNN framework. *Comput. Chem. Eng.* 131: 106533.

Harshvardhan, G., Gourisaria, M.K., Pandey, M., and Rautaray, S.S. (2020). A comprehensive survey and analysis of generative models in machine learning. *Comput. Sci. Rev.* 38: 100285.

Hassanpour, H., Corbett, B., and Mhaskar, P. (2020). Integrating dynamic neural network models with principal component analysis for adaptive model predictive control. *Chem. Eng. Res. Des.* 161: 26–37.

He, R., Li, X., Chen, G., Chen, G., and Liu, Y. (2020). Generative adversarial network-based semi-supervised learning for real-time risk warning of process industries. *Expert Syst. Appl.* 150: 113244.

Hein, M. and Andriushchenko, M. (2017). Formal guarantees on the robustness of a classifier against adversarial manipulation. In: *Advances in neural information processing systems*, Vol. 30. Curran Associates, Inc, Red Hook, NY.

Hewing, L., Wabersich, K.P., Menner, M., and Zeilinger, M.N. (2020). Learning-based model predictive control: toward safe learning in control. *Annu. Rev. Control Robot. Auton. Syst.* 3: 269–296.

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

Hirtreiter, E., Schulze Balhorn, L., and Schweidtmann, A.M. (2024). Toward automatic generation of control structures for process flow diagrams with large language models. *AIChE J.* 70: e18259.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.* 9: 1735–1780.

Hoi, S.C., Sahoo, D., Lu, J., and Zhao, P. (2021). Online learning: a comprehensive survey. *Neurocomputing* 459: 249–289.

Hoskins, J.C. and Himmelblau, D.M. (1988). Artificial neural network models of knowledge representation in chemical engineering. *Comput. Chem. Eng.* 12: 881–890.

Hu, C. and Wu, Z. (2024). Model predictive control of switched nonlinear systems using online machine learning. *Chem. Eng. Res. Des.* 209: 221–236.

Hu, G. and You, F. (2023). Multi-zone building control with thermal comfort constraints under disjunctive uncertainty using data-driven robust model predictive control. *Adv. Appl. Energy* 9: 100124.

Hu, C., Cao, Y., and Wu, Z. (2023a). Online machine learning modeling and predictive control of nonlinear systems with scheduled mode transitions. *AIChE J.* 69: e17882.

Hu, C., Chen, S., and Wu, Z. (2023b). Economic model predictive control of nonlinear systems using online learning of neural networks. *Processes* 11: 342.

Huang, B. and Kadali, R. (2008) System identification: conventional approach. In: *Dynamic modeling, predictive control and performance monitoring: a data-driven subspace approach*. Springer London, London, pp. 9–29.

Huang, J., Gretton, A., Borgwardt, K., Schölkopf, B., and Smola, A. (2006). Correcting sample selection bias by unlabeled data. In: *Advances in neural information processing systems*, Vol. 19. MIT Press, Cambridge, MA.

Huang, Z., Liu, J., and Huang, B. (2023). Model predictive control of agro-hydrological systems based on a two-layer neural network modeling framework. *Int. J. Adapt. Control Signal Process.* 37: 1536–1558.

Jalanko, M., Sanchez, Y., Mahalec, V., and Mhaskar, P. (2021). Adaptive system identification of industrial ethylene splitter: a comparison of subspace identification and artificial neural networks. *Comput. Chem. Eng.* 147: 107240.

Kadakia, Y.A., Abdullah, F., Alnajdi, A., and Christofides, P.D. (2024a). Encrypted distributed model predictive control of nonlinear processes. *Control Eng. Pract.* 145: 105874.

Kadakia, Y.A., Abdullah, F., Alnajdi, A., and Christofides, P.D. (2024b). Integrating dynamic economic optimization and encrypted control for cyber-resilient operation of nonlinear processes. *AIChE J.* 70: e18509.

Kadakia, Y.A., Suryavanshi, A., Alnajdi, A., Abdullah, F., and Christofides, P.D. (2024c). *Proceedings of the 2024 American control conference, July 10–12, 2024: a two-tier encrypted control architecture for enhanced cybersecurity of nonlinear processes*. Toronto, Canada, pp. 4452–4459.

Kadakia, Y.A., Suryavanshi, A., Alnajdi, A., Abdullah, F., and Christofides, P.D. (2024d). Integrating machine learning detection and encrypted control for enhanced cybersecurity of nonlinear processes. *Comput. Chem. Eng.* 180: 108498.

Karagiannopoulos, M., Anyfantis, D., Kotsiantis, S.B., and Pintelas, P.E. (2007). *Proceedings of the 8th hellenic European research on computer mathematics & its applications, September 20–22, 2007: feature selection for regression problems*. Athens, Greece.

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning. *Nat. Rev. Phys.* 3: 422–440.

Kassa, A.M. and Kassa, S.M. (2016). A branch-and-bound multi-parametric programming approach for non-convex multilevel optimization with polyhedral constraints. *J. Global Optim.* 64: 745–764.

Katz, J., Pappas, I., Avraamidou, S., and Pistikopoulos, E.N. (2020). Integrating deep learning models and multiparametric programming. *Comput. Chem. Eng.* 136: 106801.

Kenefake, D. and Pistikopoulos, E.N. (2022). *Proceedings of the 32nd European aymposium on computer-aided process engineering, June 12–15, 2022: PPOPT-multiparametric solver for explicit MPC*. Toulouse, France, pp. 1273–1278.

Khan, N. and Ammar Taqvi, S.A. (2023). Machine learning an intelligent approach in process industries: a perspective and overview. *ChemBioEng Rev.* 10: 195–221.

Kim, Y. and Kim, J.W. (2022). Safe model-based reinforcement learning for nonlinear optimal control with state and input constraints. *AIChE J.* 68: e17601.

Kim, J., Lee, C., Shim, H., Cheon, J.H., Kim, A., Kim, M., and Song, Y. (2016). Encrypting controller using fully homomorphic encryption for security of cyber-physical systems. *IFAC-PapersOnLine* 49: 175–180.

Kim, S., Noh, J., Gu, G.H., Aspuru-Guzik, A., and Jung, Y. (2020). Generative adversarial networks for crystal structure prediction. *ACS Cent. Sci.* 6: 1412–1420.

Kingma, D.P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Koiran, P. and Sontag, E.D. (1998). Vapnik-Chervonenkis dimension of recurrent neural networks. *Discrete Appl. Math.* 86: 63–79.

Kokotović, P., Khalil, H.K., and O'Reilly, J. (1999). *Singular perturbation methods in control: analysis and design*. Society for Industrial and Applied Mathematics, Chap. 3, pp. 93–156.

Kramer, M.A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE J.* 37: 233–243.

Kvasnica, M., Grieder, P., Baotić, M., and Morari, M. (2004). *Proceedings of the 7th international workshop on hybrid systems: computation and control (HSCC 2004), March 25–27, 2004: multi-parametric toolbox (MPT)*. Philadelphia, PA, USA, pp. 448–462.

Lanzetti, N., Lian, Y.Z., Cortinovis, A., Dominguez, L., Mercangöz, M., and Jones, C. (2019). *Proceedings of the 18th European control conference (ECC), June 25–28, 2019: recurrent neural network based MPC for process industries*. IEEE, Naples, Italy, pp. 1005–1010.

Lee, Y.S. and Chen, J. (2023). Developing semi-supervised latent dynamic variational autoencoders to enhance prediction performance of product quality. *Chem. Eng. Sci.* 265: 118192.

Lee, J.H., Shin, J., and Realff, M.J. (2018). Machine learning: overview of the recent progresses and implications for the process systems engineering field. *Comput. Chem. Eng.* 114: 111–121.

Lee, S., Kwak, M., Tsui, K.-L., and Kim, S.B. (2019). Process monitoring using variational autoencoder for high-dimensional nonlinear processes. *Eng. Appl. Artif. Intell.* 83: 13–27.

Lee, N., Kim, H., Jung, J., Park, K.-H., Linga, P., and Seo, Y. (2022). Time series prediction of hydrate dynamics on flow assurance using PCA and recurrent neural networks with iterative transfer learning. *Chem. Eng. Sci.* 263: 118111.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R.P., Tang, J., and Liu, H. (2017). Feature selection: a data perspective. *ACM Comput. Surv.* 50: 1–45.

Limon, D., Calliess, J., and Maciejowski, J.M. (2017). Learning-based nonlinear model predictive control. *IFAC-PapersOnLine* 50: 7769–7776.

Lu, J., Cao, Z., Zhao, C., and Gao, F. (2019). 110th anniversary: an overview on learning-based model predictive control for batch processes. *Ind. Eng. Chem. Res.* 58: 17164–17173.

Luo, J., Canuso, V., Jang, J.B., Wu, Z., Morales-Guio, C.G., and Christofides, P.D. (2022). Machine learning-based operational modeling of an electrochemical reactor: handling data variability and improving empirical models. *Ind. Eng. Chem. Res.* 61: 8399–8410.

Luo, J., Çıtmacı, B., Jang, J.B., Abdullah, F., Morales-Guio, C.G., and Christofides, P.D. (2023). Machine learning-based predictive control using on-line model linearization: application to an experimental electrochemical reactor. *Chem. Eng. Res. Des.* 197: 721–737.

Mahmood, M. and Mhaskar, P. (2008). Enhanced stability regions for model predictive control of nonlinear process systems. *AIChE J.* 54: 1487–1498.

Mahmood, F., Govindan, R., Bermak, A., Yang, D., and Al-Ansari, T. (2023). Data-driven robust model predictive control for greenhouse temperature control and energy utilisation assessment. *Appl. Energy* 343: 121190.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2008). Domain adaptation with multiple sources. In: *Advances in neural information processing systems*, Vol. 21. Curran Associates, Inc, Red Hook, NY.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009). Domain adaptation: learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*.

Manzano, J.M., Limon, D., de la Peña, D.M., and Calliess, J.-P. (2020). Robust learning-based MPC for nonlinear constrained systems. *Automatica* 117: 108948.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., and Scokaert, P.O. (2000). Constrained model predictive control: stability and optimality. *Automatica* 36: 789–814.

Meng, F., Shen, X., and Karimi, H.R. (2022). Emerging methodologies in stability and optimization problems of learning-based nonlinear model predictive control: a survey. *Int. J. Circ. Theor. Appl.* 50: 4146–4170.

Mesbah, A., Wabersich, K.P., Schoellig, A.P., Zeilinger, M.N., Lucia, S., Badgwell, T.A., and Paulson, J.A. (2022). *Proceedings of the 2022 American control conference, June 8–10, 2022: fusion of machine learning and MPC under uncertainty: what advances are on the horizon?* IEEE, Atlanta, GA, USA, pp. 342–357.

Mhaskar, P., El-Farra, N.H., and Christofides, P.D. (2006). Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control. *Syst. Control Lett.* 55: 650–659.

Mishra, S. and Molinaro, R. (2022). Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs. *IMA J. Numer. Anal.* 42: 981–1022.

Mishra, S. and Molinaro, R. (2023). Estimates on the generalization error of physics-informed neural networks for approximating PDEs. *IMA J. Numer. Anal.* 43: 1–43.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of machine learning*. MIT press, Cambridge, MA.

Mowbray, M., Vallerio, M., Perez-Galvan, C., Zhang, D., Chanona, A.D.R., and Navarro-Brull, F.J. (2022). Industrial data science–a review of machine learning applications for chemical and process industries. *React. Chem. Eng.* 7: 1471–1509.

Murray-Smith, R., Sbarbaro, D., Rasmussen, C.E., and Girard, A. (2003). Adaptive, cautious, predictive control with Gaussian process priors. *IFAC Proc. Vol.* 36: 1155–1160.

Na, J., Jeon, K., and Lee, W.B. (2018). Toxic gas release modeling for real-time analysis using variational autoencoder with convolutional neural networks. *Chem. Eng. Sci.* 181: 68–78.

Nagy, Z.K. (2007). Model based control of a yeast fermentation bioreactor using optimally designed artificial neural networks. *Chem. Eng. J.* 127: 95–109.

Nascimento, C.A.O., Giudici, R., and Guardani, R. (2000). Neural network based approach for optimization of industrial chemical processes. *Comput. Chem. Eng.* 24: 2303–2314.

Neyshabur, B., Bhojanapalli, S., and Srebro, N. (2017). A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*.

Nian, R., Liu, J., and Huang, B. (2020). A review on reinforcement learning: introduction and applications in industrial process control. *Comput. Chem. Eng.* 139: 106886.

Ning, C. and You, F. (2021). Online learning based risk-averse stochastic MPC of constrained linear uncertain systems. *Automatica* 125: 109402.

Norouzi, A., Heidarifar, H., Borhan, H., Shahbakhti, M., and Koch, C.R. (2023). Integrating machine learning and model predictive control for automotive applications: a review and future directions. *Eng. Appl. Artif. Intell.* 120: 105878.

Nouira, A., Sokolovska, N., and Crivello, J.-C. (2018). CrystalGAN: learning to discover crystallographic structures with generative adversarial networks. *arXiv preprint arXiv:1810.11203*.

Oberdieck, R., Diangelakis, N.A., Papathanasiou, M.M., Nascu, I., and Pistikopoulos, E.N. (2016). POP–parametric optimization toolbox. *Ind. Eng. Chem. Res.* 55: 8979–8991.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In: *Proceedings of the international conference on the theory and applications of cryptographic techniques, May 2–6, 1999: public-key cryptosystems based on composite degree residuosity classes*. Springer, Prague, Czech Republic, pp. 223–238.

Pan, S.J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* 22: 1345–1359.

Pan, I., Mason, L.R., and Matar, O.K. (2022). Data-centric engineering: integrating simulation, machine learning and statistics. Challenges and opportunities. *Chem. Eng. Sci.* 249: 117271.

Pappas, I., Diangelakis, N.A., and Pistikopoulos, E.N. (2021). The exact solution of multiparametric quadratically constrained quadratic programming problems. *J. Global Optim.* 79: 59–85.

Parker, S., Wu, Z., and Christofides, P.D. (2023). Cybersecurity in process control, operations, and supply chain. *Comput. Chem. Eng.* 171: 108169.

Patel, R., Bhartiya, S., and Gudi, R. (2023). Optimal temperature trajectory for tubular reactor using physics informed neural networks. *J. Process Control* 128: 103003.

Pistikopoulos, E.N., Diangelakis, N.A., and Oberdieck, R. (2020). *Multi-parametric optimization and control*. John Wiley & Sons, London.

Pravin, P., Tan, J.Z.M., Yap, K.S., and Wu, Z. (2022). Hyperparameter optimization strategies for machine learning-based stochastic energy efficient scheduling in cyber-physical production systems. *Digit. Chem. Eng.* 4: 100047.

Qin, R. and Zhao, J. (2022). High-efficiency generative adversarial network model for chemical process fault diagnosis. *IFAC-PapersOnLine* 55: 732–737.

Raissi, M., Perdikaris, P., and Karniadakis, G.E. (2019). Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 378: 686–707.

Raissi, M., Yazdani, A., and Karniadakis, G.E. (2020). Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations. *Science* 367: 1026–1030.

Rakhlin, A., Sridharan, K., and Tewari, A. (2010). Online learning: random averages, combinatorial parameters, and learnability. In: *Advances in neural information processing systems*, Vol. 23. Curran Associates, Inc, Red Hook, NY.

Ren, Y., Alhajeri, M.S., Luo, J., Chen, S., Abdullah, F., Wu, Z., and Christofides, P.D. (2022). A tutorial review of neural network modeling approaches for model predictive control. *Comput. Chem. Eng.*: 107956, https://doi.org/10.1016/j.compchemeng.2022.107956.

Rendall, R., Castillo, I., Schmidt, A., Chin, S.-T., Chiang, L.H., and Reis, M. (2019). Wide spectrum feature selection (WiSe) for regression model building. *Comput. Chem. Eng.* 121: 99–110.

Rijmen, V. and Daemen, J. (2001). Advanced encryption standard. In: Proceedin*gs of federal information processing standards publications*, Vol. 19. National Institute of Standards and Technology, p. 22.

Robinet, F., Parera, C., Hundt, C., and Frank, R. (2022). *Proceedings of the 2022 IEEE/CVF winter conference on applications of computer vision, January 4–8, 2022: weakly-supervised free space estimation through stochastic co-teaching*. Waikoloa, HI, USA, pp. 618–627.

Rogers, A.W., Cardenas, I.O.S., Del Rio-Chanona, E.A., and Zhang, D. (2023). Investigating physics-informed neural networks for bioprocess hybrid model construction. In: *Computer aided chemical engineering*, Vol. 52. Elsevier, Amsterdam, pp. 83–88.

Romdlony, M.Z. and Jayawardhana, B. (2016). Stabilization with guaranteed safety using control Lyapunov–barrier function. *Automatica* 66: 39–47.

Sangoi, E., Quaglio, M., Bezzo, F., and Galvanin, F. (2022). Optimal design of experiments based on artificial neural network classifiers for fast

kinetic model recognition. In: *Computer aided chemical engineering*, Vol. 49. Elsevier, Amsterdam, pp. 817–822.

Sangoi, E., Quaglio, M., Bezzo, F., and Galvanin, F. (2024). An optimal experimental design framework for fast kinetic model identification based on artificial neural networks. *Comput. Chem. Eng.* 187: 108752.

Saraswathi K, S., Bhosale, H., Ovhal, P., Parlikkad Rajan, N., and Valadi, J.K. (2020). Random forest and autoencoder data-driven models for prediction of dispersed-phase holdup and drop size in rotating disc contactors. *Ind. Eng. Chem. Res.* 60: 425–435.

Scattolini, R. (2009). Architectures for distributed and hierarchical model predictive control–a review. *J. Process Control* 19: 723–731.

Schilter, O., Vaucher, A., Schwaller, P., and Laino, T. (2023). Designing catalysts with deep generative models and computational data. A case study for Suzuki cross coupling reactions. *Digit. Discov.* 2: 728–735.

Schlüter, N., Binfet, P., and Darup, M.S. (2023). A brief survey on encrypted control: from the first to the second generation and beyond. *Annu. Rev. Control* 56: 100913.

Schweidtmann, A.M., Esche, E., Fischer, A., Kloft, M., Repke, J.-U., Sager, S., and Mitsos, A. (2021). Machine learning in chemical engineering: a perspective. *Chem. Ing. Tech.* 93: 2029–2039.

Serrurier, M., Mamalet, F., González-Sanz, A., Boissin, T., Loubes, J.-M., and Del Barrio, E. (2021). *Proceedings of the 2021 IEEE/CVF conference on computer vision and pattern recognition, June 20–25, 2021: achieving robustness in classification using optimal transport with hinge regularization*. Nashville, TN, USA, pp. 505–514.

Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648. University of Wisconsin–Madison.

Shalev-Shwartz, S. (2012). Online learning and online convex optimization. *Found. Trends Mach. Learn.* 4: 107–194.

Shang, C. and You, F. (2019). Data analytics and machine learning for smart process manufacturing: recent advances and perspectives in the big data era. *Engineering* 5: 1010–1016.

Sitapure, N. and Kwon, J.S.-I. (2022). Neural network-based model predictive control for thin-film chemical deposition of quantum dots using data from a multiscale simulation. *Chem. Eng. Res. Des.* 183: 595–607.

Soloperto, R., Müller, M.A., and Allgöwer, F. (2022). Guaranteed closed-loop learning in model predictive control. *IEEE Trans. Automat. Control* 68: 991–1006.

Sontag, E.D. (1998a). A learning result for continuous-time recurrent neural networks. *Syst. Control Lett.* 34: 151–158.

Sontag, E.D. (1998b). VC dimension of neural networks. *NATO ASI Ser. F Comput. Syst. Sci.* 168: 69–96.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15: 1929–1958.

Stewart, B.T., Venkat, A.N., Rawlings, J.B., Wright, S.J., and Pannocchia, G. (2010). Cooperative distributed model predictive control. *Syst. Control Lett.* 59: 460–469.

Su, H.T., McAvoy, T.J., and Werbos, P. (1992). Long-term predictions of chemical processes using recurrent neural networks: a parallel training approach. *Ind. Eng. Chem. Res.* 31: 1338–1352.

Subraveti, S.G., Li, Z., Prasad, V., and Rajendran, A. (2022). Physics-based neural networks for simulation and synthesis of cyclic adsorption processes. *Ind. Eng. Chem. Res.* 61: 4095–4113.

Suryavanshi, A., Alnajdi, A., Alhajeri, M., Abdullah, F., and Christofides, P.D. (2023). Encrypted model predictive control design for security to cyberattacks. *AIChE J.* 69: e18104.

Tan, W.G.Y. and Wu, Z. (2024). Robust machine learning modeling for predictive control using Lipschitz-constrained neural networks. *Comput. Chem. Eng.* 180: 108466.

Tan, G.Y., Xiao, M., Wu, G., and Wu, Z. (2024a). *Proceedings of the 2024 American control conference, July 10–12, 2024: machine learning modeling of nonlinear processes with Lyapunov stability guarantees*. Toronto, Canada, pp. 528–535.

Tan, W.G.Y., Xiao, M., and Wu, Z. (2024b). Robust reduced-order machine learning modeling of high-dimensional nonlinear processes using noisy data. *Digit. Chem. Eng.* 11: 100145.

Tang, W. (2023). Synthesis of data-driven nonlinear state observers using lipschitz-bounded neural networks. *arXiv preprint arXiv:2310.03187*.

Tang, W. and Daoutidis, P. (2022). *Proceedings of the 2022 American control conference, June 8–10, 2022: data-driven control: overview and perspectives*. IEEE, Atlanta, Georgia, USA, pp. 1048–1064.

Terzi, E., Bonassi, F., Farina, M., and Scattolini, R. (2021). Learning model predictive control with long short-term memory networks. *Int. J. Robust Nonlinear Control* 31: 8877–8896.

Thebelt, A., Wiebe, J., Kronqvist, J., Tsay, C., and Misener, R. (2022). Maximizing information from chemical engineering data sets: applications to machine learning. *Chem. Eng. Sci.* 252: 117469.

Tian, Y., Pappas, I., Burnak, B., Katz, J., and Pistikopoulos, E.N. (2021). Simultaneous design & control of a reactive distillation system–a parametric optimization & control approach. *Chem. Eng. Sci.* 230: 116232.

Tian, J., Han, D., Li, M., and Shi, P. (2022). A multi-source information transfer learning method with subdomain adaptation for cross-domain fault diagnosis. *Knowl. Base Syst.* 243: 108466.

Vapnik, V., Levin, E., and Le Cun, Y. (1994). Measuring the VC-dimension of a learning machine. *Neural Comput.* 6: 851–876.

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

Wang, R. and Manchester, I. (2023). *Proceedings of the 40th international conference on machine learning, July 23–29, 2023: direct parameterization of lipschitz-bounded deep networks*. PMLR, Hawaii, USA, pp. 36093–36110.

Wang, Y. and Wu, Z. (2024a). Control Lyapunov-barrier function-based safe reinforcement learning for nonlinear optimal control. *AIChE J.* 70: e18306.

Wang, Y. and Wu, Z. (2024b). Physics-informed reinforcement learning for optimal control of nonlinear systems. *AIChE J.* 70: e18542.

Wang, Z. and Wu, Z. (2024c). Foundation model for chemical process modeling: meta-learning with physics-informed adaptation. *arXiv preprint arXiv:2405.11752*.

Wang, X., Ayachi, S., Corbett, B., and Mhaskar, P. (in press). Integrating autoencoder with Koopman operator to design a linear data-driven model predictive controller. *Can. J. Chem. Eng*.

Wang, Z., Dai, Z., Póczos, B., and Carbonell, J. (2019). *Proceedings of the 2019 IEEE/CVF conference on computer vision and pattern recognition, June 15–20, 2019: characterizing and avoiding negative transfer*. Long Beach, CA, USA, pp. 11293–11302.

Wang, W., Wang, Y., Tian, Y., and Wu, Z. (2024a). Explicit machine learning-based model predictive control of nonlinear processes via multi-parametric programming. *Comput. Chem. Eng.* 186: 108689.

Wang, Z., Yu, D., and Wu, Z. (2025). Real-time machine-learning-based optimization using input convex long short-term memory network. *Appl. Energy* 377: 124472.

Wang, W., Zhang, H., Wang, Y., Tian, Y., and Wu, Z. (2024b). Fast explicit machine learning-based model predictive control using input convex neural networks. *Ind. Eng. Chem. Res.* 63: 17279–17293.

Wei, C. and Ma, T. (2019). Data-dependent sample complexity of deep neural networks via Lipschitz augmentation. In: *Advances in neural*

information processing systems, Vol. 32. Curran Associates, Inc, Red Hook, NY.

Wieland, P. and Allgöwer, F. (2007). Constructive safety using control barrier functions. *IFAC Proc. Vol.* 40: 462–467.

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

Wu, Z. and Christofides, P.D. (2020). Control Lyapunov-barrier function-based predictive control of nonlinear processes using machine learning modeling. *Comput. Chem. Eng.* 134: 106706.

Wu, Z., Durand, H., and Christofides, P.D. (2018). Safe economic model predictive control of nonlinear systems. *Syst. Control Lett.* 118: 69–76.

Wu, Z., Albalawi, F., Zhang, Z., Zhang, J., Durand, H., and Christofides, P.D. (2019a). Control Lyapunov-barrier function-based model predictive control of nonlinear systems. *Automatica* 109: 108508.

Wu, Z., Rincon, D., and Christofides, P.D. (2019b). Real-time adaptive machine-learning-based predictive control of nonlinear processes. *Ind. Eng. Chem. Res.* 59: 2275–2290.

Wu, Z., Tran, A., Rincon, D., and Christofides, P.D. (2019c). Machine learning-based predictive control of nonlinear processes. Part I: theory. *AIChE J.* 65: e16729.

Wu, Z., Tran, A., Rincon, D., and Christofides, P.D. (2019d). Machine-learning-based predictive control of nonlinear processes. Part II: computational implementation. *AIChE J.* 65: e16734.

Wu, Z., Rincon, D., and 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., Luo, J., Rincon, D., and Christofides, P.D. (2021a). Machine learning-based predictive control using noisy data: evaluating performance and robustness via a large-scale process simulator. *Chem. Eng. Res. Des.* 168: 275–287.

Wu, Z., Rincon, D., Gu, Q., and Christofides, P.D. (2021b). Statistical machine learning in model predictive control of nonlinear processes. *Mathematics* 9: 1912.

Wu, Z., Rincon, D., Luo, J., and Christofides, P.D. (2021c). Machine learning modeling and predictive control of nonlinear processes using noisy data. *AIChE J.* 67: e17164.

Wu, G., Yion, W.T.G., Dang, K.L.N.Q., and Wu, Z. (2023a). Physics-informed machine learning for MPC: application to a batch crystallization process. *Chem. Eng. Res. Des.* 192: 556–569.

Wu, Z., Zhang, B., Yu, H., Ren, J., Pan, M., He, C., and Chen, Q. (2023b). Accelerating heat exchanger design by combining physics-informed deep learning and transfer learning. *Chem. Eng. Sci.* 282: 119285.

Wu, Z., Li, M., He, C., Zhang, B., Ren, J., Yu, H., and Chen, Q. (2024). Physics-informed learning of chemical reactor systems using decoupling–coupling training framework. *AIChE J.*: e18436, https://doi.org/10.1002/aic.18436.

Xiao, M. and Wu, Z. (2023). Modeling and control of a chemical process network using physics-informed transfer learning. *Ind. Eng. Chem. Res.* 62: 17216–17227.

Xiao, M., Hu, C., and Wu, Z. (2023). Modeling and predictive control of nonlinear processes using transfer learning method. *AIChE J.* 69: e18076.

Xiao, M., Vellayappan, K., Pravin, P., Gudena, K., and Wu, Z. (2024). Optimization-based multi-source transfer learning for modeling of nonlinear processes. *Chem. Eng. Sci.* 295: 120117.

Xie, R., Jan, N.M., Hao, K., Chen, L., and Huang, B. (2019). Supervised variational autoencoders for soft sensor modeling with missing data. *IEEE Trans. Ind. Inf.* 16: 2820–2828.

Xiu, X., Yang, Y., Kong, L., and Liu, W. (2020). Laplacian regularized robust principal component analysis for process monitoring. *J. Process Control* 92: 212–219.

Xu, Z. and Wu, Z. (2024). Privacy-preserving federated machine learning modeling and predictive control of heterogeneous nonlinear systems. *Comput. Chem. Eng.* 187: 108749.

Yang, S. and Bequette, B.W. (2021). Optimization-based control using input convex neural networks. *Comput. Chem. Eng.* 144: 107143.

Yang, F., Li, K., Zhong, Z., Luo, Z., Sun, X., Cheng, H., Guo, X., Huang, F., Ji, R., and Li, S. (2020). Asymmetric co-teaching for unsupervised cross-domain person re-identification. In: *Proceedings of the thirty-fourth AAAI conference on artificial intelligence, February 7–12, 2020: asymmetric co-teaching for unsupervised cross-domain person re-identification*, Vol. 34. New York, USA, pp. 12597–12604.

Yao, Y. and Doretto, G. (2010). *Proceedings of the 2010 IEEE computer society conference on computer vision and pattern recognition, June 13–18, 2010: boosting for transfer learning with multiple sources*. San Francisco, CA, USA, pp. 1855–1862.

You, Y. and Nikolaou, M. (1993). Dynamic process modeling with recurrent neural networks. *AIChE J.* 39: 1654–1667.

Yu, X., Han, B., Yao, J., Niu, G., Tsang, I., and Sugiyama, M. (2019). *Proceedings of the 36th international conference on machine learning, June 9–15, 2019: how does disagreement help generalization against label corruption?* PMLR, California, USA, pp. 7164–7173.

Zhang, S. and Qiu, T. (2022). Semi-supervised LSTM ladder autoencoder for chemical process fault diagnosis and localization. *Chem. Eng. Sci.* 251: 117467.

Zhang, J., Lei, Q., and Dhillon, I. (2018) Stabilizing gradients for deep neural networks via efficient svd parameterization. In: *Proceedings of the 35th international conference on machine learning, July 10–15, 2018: stabilizing gradients for deep neural networks via efficient svd parameterization*. PMLR, Stockholm, Sweden, pp. 5806–5814.

Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., and Gao, Y. (2021a). A survey on federated learning. *Knowl. Base Syst.* 216: 106775.

Zhang, X., Zou, Y., and Li, S. (2021b). Semi-supervised generative adversarial network with guaranteed safeness for industrial quality prediction. *Comput. Chem. Eng.* 153: 107418.

Zhang, X., Pan, W., Scattolini, R., Yu, S., and Xu, X. (2022). Robust tube-based model predictive control with Koopman operators. *Automatica* 137: 110114.

Zhang, Z., Wang, X., Wang, G., Jiang, Q., Yan, X., and Zhuang, Y. (2024). A data enhancement method based on generative adversarial network for small sample-size with soft sensor application. *Comput. Chem. Eng.* 186: 108707.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. (2018). Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.

Zhao, T., Zheng, Y., Gong, J., and Wu, Z. (2022a). Machine learning-based reduced-order modeling and predictive control of nonlinear processes. *Chem. Eng. Res. Des.* 179: 435–451.

Zhao, T., Zheng, Y., and Wu, Z. (2022b). Improving computational efficiency of machine learning modeling of nonlinear processes using sensitivity analysis and active learning. *Digit. Chem. Eng.* 3: 100027.

Zhao, T., Zheng, Y., and Wu, Z. (2023). Feature selection-based machine learning modeling for distributed model predictive control of nonlinear processes. *Comput. Chem. Eng.* 169: 108074.

Zheng, Y. and Wu, Z. (2023). Physics-informed online machine learning and predictive control of nonlinear processes with parameter uncertainty. *Ind. Eng. Chem. Res.* 62: 2804–2818.

Zheng, S. and Zhao, J. (2020). A new unsupervised data mining method based on the stacked autoencoder for chemical process fault diagnosis. *Comput. Chem. Eng.* 135: 106755.

Zheng, Y., Wang, X., and Wu, Z. (2022a). Machine learning modeling and predictive control of the batch crystallization process. *Ind. Eng. Chem. Res.* 61: 5578–5592.

Zheng, Y., Zhang, T., Li, S., Zhang, G., and Wang, Y. (2022b). Gp-based MPC with updating tube for safety control of unknown system. *Digit. Chem. Eng.* 4: 100041.

Zheng, Y., Zhao, T., Wang, X., and Wu, Z. (2022c). Online learning-based predictive control of crystallization processes under batch-to-batch parametric drift. *AIChE J.* 68: e17815.

Zheng, Y., Hu, C., Wang, X., and Wu, Z. (2023). Physics-informed recurrent neural network modeling for predictive control of nonlinear processes. *J. Process Control* 128: 103005.

Zhu, Q.X., Xu, T.X., Xu, Y., and He, Y.L. (2021). Improved virtual sample generation method using enhanced conditional generative adversarial networks with cycle structures for soft sensors with limited data. *Ind. Eng. Chem. Res.* 61: 530–540.

Zhu, W., Zhang, J., and Romagnoli, J. (2022). General feature extraction for process monitoring using transfer learning approaches. *Ind. Eng. Chem. Res.* 61: 5202–5214.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive survey on transfer learning. *Proc. IEEE* 109: 43–76.