mathematics

MDPI

Article

Statistical Machine Learning in Model Predictive Control of Nonlinear Processes

Zhe Wu [1], David Rincon [1], Quanquan Gu [2] and Panagiotis D. Christofides [1,3,*]

[1] Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095-1592, USA; wuzhe@g.ucla.edu (Z.W.); fdrinconc@gmail.com (D.R.)
[2] Department of Computer Science, University of California, Los Angeles, CA 90095-1592, USA; qgu@cs.ucla.edu
[3] Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095-1592, USA
* Correspondence: pdc@seas.ucla.edu

Abstract: Recurrent neural networks (RNNs) have been widely used to model nonlinear dynamic systems using time-series data. While the training error of neural networks can be rendered sufficiently small in many cases, there is a lack of a general framework to guide construction and determine the generalization accuracy of RNN models to be used in model predictive control systems. In this work, we employ statistical machine learning theory to develop a methodological framework of generalization error bounds for RNNs. The RNN models are then utilized to predict state evolution in model predictive controllers (MPC), under which closed-loop stability is established in a probabilistic manner. A nonlinear chemical process example is used to investigate the impact of training sample size, RNN depth, width, and input time length on the generalization error, along with the analyses of probabilistic closed-loop stability through the closed-loop simulations under Lyapunov-based MPC.

Keywords: generalization error; recurrent neural networks; machine learning; model predictive control; nonlinear systems

check for updates

Citation: Wu, Z.; Rincon, D.; Gu, Q.; Christofides, P.D. Statistical Machine Learning in Model Predictive Control of Nonlinear Processes. *Mathematics* **2021**, *9*, 1912. https://doi.org/10.3390/math9161912

Academic Editor: Freddy Gabbay

Received: 29 July 2021
Accepted: 9 August 2021
Published: 11 August 2021

1. Introduction

Modeling large-scale, complex nonlinear processes has been a long-standing research problem in process systems engineering. The traditional approaches to modeling nonlinear processes include data-driven modeling approach with parameters identified from industrial/simulation data [1,2], and first-principles modeling approach based on a fundamental understanding of the underlying physico-chemical phenomena. While traditional first-principles modeling approach has been used extensively in monitoring, control and optimization of chemical processes, it can be time-demanding and inaccurate to model complex nonlinear processes using first-principle modeling tools. Machine learning methods have been increasingly adopted to model complex nonlinear systems due to their ability to model a rich set of nonlinear functions and handle efficiently with big datasets from processes [3–10]. Among many machine learning modeling techniques, recurrent neural network (RNN) is widely used to model nonlinear dynamic systems using time-series data [11–13]. While the history of machine learning methods in chemical process control can be traced back to 1990s [14–18], machine learning has become popular again this decade due to a number of reasons such as cheaper computation (mature and efficient libraries/hardware), availability of large datasets, and advanced learning algorithms. Designing MPC systems that utilize machine learning models with well-characterized accuracy is a new frontier in control systems that will impact the next generation of industrial control systems.

Despite the success of machine learning methods in modeling nonlinear chemical processes in the context of MPC, there remain fundamental challenges that limit the implementation of machine-learning-based MPC to real chemical processes. One important challenge is to characterize the generalization ability on unseen data for machine learning models trained using finite training samples. Furthermore, a theoretical analysis of closed-loop stability for MPC using machine learning models needs to be developed via machine learning and control theory. Typically, theoretical developments on machine-learning-based MPC derived closed-loop stability properties based on the assumption of bounded modeling errors. For example, in [9], a Lyapunov-based MPC scheme using RNN models as the prediction model has been developed with guaranteed closed-loop stability by assuming that the RNN models are able to obtain a sufficiently small and bounded testing error. Similarly, a neural Lyapunov MPC that trains a stabilizing nonlinear MPC based on surrogate model and neural-network-based terminal cost was proposed in [19] with stability properties derived by assuming the boundedness of modeling error. Additionally, in [20], a nonparametric machine learning model is implemented together with MPC in which input-to-state stability is evaluated. In [21], a learning-based MPC targeting deterministic linear models is proposed in which safety, stability, and robustness are proved. However, the fundamental question regarding the generalization accuracy of machine learning models in MPC has not been addressed.

Probably approximately correct (PAC) learning theory is a framework that mathematically analyze the generalization ability of machine learning models [22]. Specifically, in PAC learning, given a set of training data, the learner is supposed to choose the optimal hypothesis (i.e., machine learning model) that yields a low generalization error with high probability from a certain class of hypotheses. Therefore, PAC learning theory provides a useful tool that demonstrates under what conditions a learning algorithm will probably output an approximately correct hypothesis. For example, in [23], PAC learning theory was used to study the learnability of compression learning algorithm for the optimization problem of stochastic MPC using a finite number of realizations of the uncertainty. In [24], PAC learning was used to analyze the generalization performance of a convex piecewise linear classifier that classifies the thermal comfort in a HVAC system. However, to the best of our knowledge, the use of statistical machine learning theory in analyzing stability properties of machine learning models in MPC, and guiding machine learning model structure and training data collection have not been fully explored.

Many recent works have been developed characterizing learnability of neural networks in terms of sample complexity and generalization error [25–32]. Generalization error bound is a common methodology in statistical machine learning for evaluating the predictive performance of machine learning algorithms [33]. This bound depends on a number of factors such as the number of data samples, the number of layers and neurons, bounds of weight matrices, initialization method, among others. For example, in [29], a generalization error bound was developed for a family of RNN models including vanilla RNNs, long short term memory and minimal gated unit. The generalization error bound was established for multiclass classification problems, and was dependent on the total number of network parameters and the spectral norms of the weight matrices. In [27], a sample complexity bound that was fully independent of network depth and width under some assumptions was developed for feedforward neural networks. In [34], an expected risk bound was developed for RNNs that model single-output nonlinear dynamic systems. However, at this stage, generalization error bounds for RNNs that model multiple-input and multiple-output (MIMO) nonlinear dynamic systems using time-series data have not been studied.

Motivated by the above, in this work, we develop the methodological framework of generalization error bounds from machine learning theory for the development and verification of RNN models with specific theoretical accuracy guarantees and integrate these models into model predictive control system design for nonlinear chemical processes. Specifically, in Section 2, the class of nonlinear systems, the formulation of RNNs, along with some general assumptions on system stabilizability and RNN development are presented. In Section 3, preliminaries including some important definitions and lemmas are first presented, followed by the development of a probabilistic generalization error bound for RNN models accounting for the impact of training data size and the number of neurons and layers on accuracy and guiding network structure selection and training. In Section 4, the RNN models are incorporated in the MPC formulation, under which probabilistic closed-loop stability is derived based on the RNN generalization error bound. Finally, in Section 5, a chemical reactor example is used to demonstrate the impact of training sample size, RNN depth and width, input time length on its generalization error. Additionally, closed-loop simulations are carried out to analyze the probabilistic closed-loop stability and performance.

2. Preliminaries

2.1. Notation

The Frobenius norm of A is denoted by $\|A\|_F$. The Euclidean norm of a vector is denoted by the operator $|\cdot|$ and the weighted Euclidean norm of a vector is denoted by the operator $|\cdot|_Q$ where Q is a positive definite matrix. $\mathbf{R}_+$ denotes nonnegative real numbers. $\mathbf{x}^T$ denotes the transpose of $\mathbf{x}$. The notation $L_f V(\mathbf{x})$ denotes the standard Lie derivative $L_f V(\mathbf{x}) := \frac{\partial V(\mathbf{x})}{\partial x} f(\mathbf{x})$. Set subtraction is denoted by "\", i.e., $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$. A function $f(\cdot)$ is of class $\mathcal{C}^1$ if it is continuously differentiable. A continuous function $\alpha : [0, a) \to [0, \infty)$ belongs to class $\mathcal{K}$ if it is strictly increasing and is zero only when evaluated at zero. A function $f : \mathbf{R}^n \to \mathbf{R}^m$ is said to be L-Lipschitz, $L \geq 0$, if $|f(a) - f(b)| \leq L|a - b|$ for all $a, b \in \mathbf{R}^n$. $\mathbb{P}(A)$ denotes the probability that event A will occur. $\mathbb{E}[X]$ denotes the expected value of a random variable X.

2.2. Class of Systems

The class of continuous-time nonlinear systems considered is described by the following state-space form:

$$\dot{x} = F(x, u) := f(x) + g(x)u, \; x(t_0) = x_0 \tag{1}$$

where $x \in \mathbf{R}^n$ and $u \in \mathbf{R}^k$ are the sate vector, and the manipulated input vector. The control action is constrained by $u \in U := \{u_{\min} \leq u \leq u_{\max}\} \subset \mathbf{R}^k$, where $u_{\min}$ and $u_{\max}$ represent the minimum and the maximum value vectors of inputs allowed, respectively. $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$, and $n \times k$, respectively. Without loss of generality, the initial time t_0 is taken to be zero ($t_0 = 0$), and it is assumed that $f(0) = 0$, and thus, the origin is a steady-state of the system of Equation (1).

We assume the system of Equation (1) is stabilizable in the sense that there exists a stabilizing controller $u = \Phi(x) \in U$ that renders the origin exponentially stable. The stabilizability assumption implies that there exists a $\mathcal{C}^1$ control Lyapunov function $V(x)$ such that for all x in an open neighborhood D around the origin, the following inequalities hold:

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

$$\frac{\partial V(x)}{\partial x} F(x, \Phi(x)) \leq -c_3|x|^2, \tag{3}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq c_4|x| \tag{4}$$

where c_1, c_2, c_3 and c_4 are positive constants. Additionally, the Lipschitz property of $F(x, u)$ and the boundedness of u implies there exist positive constants M_F, L_x, $L_x^{'}$ such that the following inequalities hold for all $x, x' \in D$ and $u \in U$:

$$|F(x,u)| \leq M_F \tag{5}$$

$$|F(x,u) - F(x',u)| \leq L_x|x - x'| \tag{6}$$

$$\left|\frac{\partial V(x)}{\partial x}F(x,u) - \frac{\partial V(x')}{\partial x}F(x',u)\right| \leq L_x^{'}|x - x'| \tag{7}$$

Following the data generation method in [9], open-loop simulations of the nonlinear system of Equation (1) are first conducted to generate a large dataset that captures the system dynamics for $x \in \Omega_\rho$ and $u \in U$, where $\Omega_\rho := \{x \in \mathbf{R}^n \mid V(x) \leq \rho\}$, $\rho > 0$, is a compact set within which the system stability is guaranteed using the controller $u = \Phi(x) \in U$. Specifically, we sweep over all the values that (x, u) can take by running extensive open-loop simulations of the system of Equation (1) under various $x_0 \in \Omega_\rho$ and inputs u to generate a large number of dynamic trajectories. The open-loop simulation of the continuous system of Equation (1) under a sequence of inputs $u \in U$ is carried out in a sample-and-hold fashion (i.e., the inputs are fed into the system of Equation (1) as a piecewise constant function, $u(t) = u(t_k)$, $\forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$, and Δ is the sampling period). The nonlinear system of Equation (1) is integrated via explicit Euler method with a sufficiently small integration time step $h_c < \Delta$. Using the open-loop simulation data, recurrent neural network (RNN) models are developed to predict future states for (at least) one sampling period based on the current state measurements, and the manipulated inputs that will be applied for the next sampling period. In other words, the RNN model is developed to predict $x(t)$, $\forall t \in [t_k, t_{k+1})$ based on the measurements $x(t_k)$ and the inputs $u \in [t_k, t_{k+1})$. Finally, the time-series dataset is partitioned into three subsets for the purposes of training, validation and testing.

2.3. Recurrent Neural Network Model

Consider an RNN model that approximates the nonlinear dynamics of the system of Equation (1) with m sequences of T-time-length data points $(\mathbf{x}_{i,t}, \mathbf{y}_{i,t})$ where $\mathbf{x}_{i,t} \in \mathbf{R}^{d_x}$ is the RNN input, and $\mathbf{y}_{i,t} \in \mathbf{R}^{d_y}$ is the RNN output, $i = 1, ..., m$ and $t = 1, ..., T$ (Figure 1). It should be noted that the RNN inputs and outputs do not necessarily represent the nonlinear system inputs and states/outputs in Equation (1). Therefore, to differentiate the notations for RNN inputs/outputs from those for the nonlinear system of Equation (1), all the vectors for RNN models are written in boldface. Additionally, to simplify the discussion, the RNN model of Equations (8) and (9) is developed to predict states over one sampling period with total time steps $T = \frac{\Delta}{h_c}$ (i.e., the RNN model is to predict future states for all the integration time step h_c within one sampling period Δ). As a result, the RNN input $\mathbf{x}_{i,t}$ consists of the current state measurements and manipulated inputs that will be applied over $t = 1, ..., T$, and the RNN output $\mathbf{y}_{i,t}$ consists of the predicted states over $t = 1, ..., T$. Note that $\mathbf{x}_{i,t}$ remains unchanged over $t = 1, ..., T$ due to the sample-and-hold implementation of manipulated inputs.

The dataset is developed consisting of m data sequences drawn independently from some underlying distribution over $\mathbf{R}^{d_x \times T} \times \mathbf{R}^{d_y \times T}$. In this work, we consider a one-hidden-layer RNN with hidden states $\mathbf{h}_i \in \mathbf{R}^{d_h}$ computed as follows:

$$\mathbf{h}_{i,t} = \sigma_h(U\mathbf{h}_{i,t-1} + W\mathbf{x}_{i,t}) \tag{8}$$

where σ_h is the element-wise nonlinear activation function (e.g., ReLU). $U \in \mathbf{R}^{d_h \times d_h}$ and $W \in \mathbf{R}^{d_h \times d_x}$ are weight matrices connected to the hidden states and input vector, respectively. The output layer $\mathbf{y}_{i,t}$ is computed as follows:

$$\mathbf{y}_{i,t} = \sigma_y(V\mathbf{h}_{i,t}) \tag{9}$$

where $V \in \mathbf{R}^{d_y \times d_h}$ is the weight matrix, and σ_y is the element-wise activation function in the output layer (typically linear unit for regression problems).

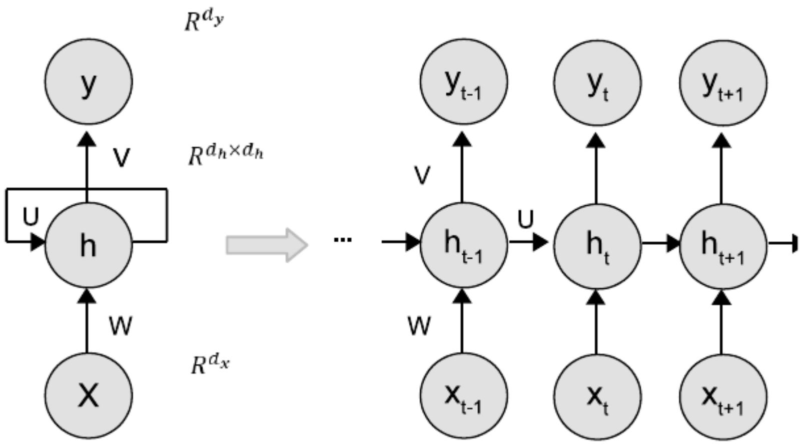

Figure 1. Recurrent neural network structure.

We consider the loss function $L(\mathbf{y}, \bar{\mathbf{y}})$ which calculates the squared difference between the true value $\bar{\mathbf{y}}$ and the predicted value $\mathbf{y}$ (i.e., L_2 loss). Without loss of generality, we have the following assumptions on the RNN model and dataset.

Assumption 1. *The RNN inputs are bounded, i.e.,* $|\mathbf{x}_{i,t}| \leq B_X$*, for all* $i = 1, ..., m$ *and* $t = 1, ..., T$.

Assumption 2. *The Frobenius norms of all the weight matrices are bounded as follows:*

$$\|U\|_F \leq B_{U,F}, \|V\|_F \leq B_{V,F}, \|W\|_F \leq B_{W,F} \tag{10}$$

Assumption 3. *Training, validation, and testing datasets are drawn from the same distribution.*

Assumption 4. *The nonlinear activation function* σ_h *is 1-Lipschitz continuous, and is positive-homogeneous, i.e.,* $\sigma_h(\alpha z) = \alpha \sigma_h(z)$ *for all* $\alpha \geq 0$ *and* $z \in \mathbf{R}$.

Remark 1. *All the assumptions made are standard in machine learning theory, and can be presented in system-theoretic language as follows. Assumption 1 assumes that the RNN inputs are bounded, which is consistent with the fact that the process states* x *and inputs* u *are bounded by* $x \in \Omega_\rho$ *and* $u \in U$*. Assumption 2 requires the RNN weight matrices to be bounded, which implies that only a finite class of neural network hypotheses are considered for modeling the nonlinear system of Equation (1). Assumption 3 is a natural and necessary assumption for generalization performance analysis. It implies that the machine learning models built from industrial operation data will be applied to the same process with the same data distribution. An example of activation function that satisfies Assumption 4 is Rectified Linear Unit (ReLu), which is a nonlinear activation function that has gained popularity in the machine learning domain.*

3. RNN Generalization Error

Since any learning algorithms are evaluated on finite training samples only, and do not provide any information on their predictive performance for unseen data, generalization error provides an important measure of how accurately a neural network model is able to predict output values for input data that has not been used in training. To implement machine learning models into real chemical processes, it is necessary to demonstrate that models are developed with a desired generalization error such that they can be applied for any reasonable operating conditions beyond those in the training dataset while maintaining a sufficiently small modeling error. In this section, we develop an upper bound for the generalization error of RNN models, and demonstrate that this error can be bounded with high probability provided that the training data samples and neural network structure meet a few requirements.

3.1. Preliminaries

We first present some important definitions and lemmas that will be used in the derivation of RNN generalization error. The random variables satisfying sub-Gaussian distribution, which is a probability distribution with strong tail decay, are defined as follows:

Definition 1. *A centered random variable $x \in \mathbf{R}$ is said to be sub-Gaussian with variance proxy σ^2, if $\mathbb{E}[x] = 0$, and the moment generating function satisfies*

$$\mathbb{E}[\exp(aX)] \leq \exp\left(\frac{a^2\sigma^2}{2}\right), \ \forall a \in \mathbf{R} \tag{11}$$

Lemma 1 (McDiarmid's inequality [35])**.** *Consider independent random variables $X_1, ..., X_n \in X$ and a function $f : X^n \to \mathbf{R}$ with bounded difference property, i.e., there exist positive numbers c_i such that the following inequality holds for all $x_1, ..., x_n, x_i' \in X$, $x_i \neq x_i'$ and $i \in \{1, ..., n\}$:*

$$|f(x_1, ..., x_i, ..., x_n) - f(x_1, ..., x_i', ..., x_n)| \leq c_i \tag{12}$$

then the following probability holds for any $a > 0$:

$$\mathbb{P}(f(X_1, ..., X_n) - \mathbb{E}[f(X_1, ..., X_n)] \geq a) \leq \exp\left(-\frac{2a^2}{\sum_{i=1}^n c_i^2}\right) \tag{13}$$

Let $L(\mathbf{y}_t, \bar{\mathbf{y}}_t)$ be the loss function, where $\mathbf{y}_t = h(\mathbf{x}_t)$ is the predicted RNN output, and $h(\cdot)$ represents the RNN functions in the hypothesis class $\mathcal{H}$ mapping input $\mathbf{x} \in \mathbf{R}^{d_x}$ to output $\mathbf{y} \in \mathbf{R}^{d_y}$. The following error definitions are commonly used in machine learning theory.

Definition 2. *Given a function h that predicts output values y for each input x, and an underlying distribution D, the* ***expected loss/error*** *or* ***generalization error*** *is*

$$L_D(h) \triangleq \mathbb{E}[L(h(x), y)] = \int_{X \times Y} L(h(x), y)\rho(x, y)dxdy \tag{14}$$

where $\rho(x, y)$ is joint probability distribution for x and y, and X, Y are the vector space of all possible inputs, and outputs, respectively.

Since in general the joint probability distribution ρ is unknown, we use the data samples drawn from this unknown probability distribution to compute empirical error, which is a proxy measure for the expected loss.

Definition 3. *Given a dataset with m data samples $S = (s_1, ..., s_m)$, where $s_i = (x_i, y_i)$, the* ***empirical error*** *or* ***risk*** *is*

$$\hat{\mathbb{E}}_S[L(h(x), y)] = \frac{1}{m}\sum_{i=1}^m L(h(x_i), y_i) \tag{15}$$

The RNN model is developed by minimizing the empirical risk of Equation (15) using a set of m data sequences. To ensure that the RNN model achieves a desired generalization performance in the sense that it well captures the nonlinear dynamics of the system of Equation (1) for various operation conditions, the objective of this work is to show that the generalization error $\mathbb{E}[L(h(\mathbf{x}), \mathbf{y})]$ can be bounded provided that the empirical risk is sufficiently small and bounded.

We consider the mean squared error (MSE) as loss function in this work. It is readily shown that the MSE loss function $L(\mathbf{y}, \bar{\mathbf{y}})$ is not Lipschitz continuous for all $\mathbf{y}, \bar{\mathbf{y}} \in \mathbf{R}^{d_y}$. However, since we consider a finite hypothesis class that satisfies Assumptions 1–4, we can show that the RNN output is bounded. This is consistent with the fact that the nonlinear system of Equation (1) is operated in the stability region Ω_ρ, and therefore, the RNN outputs are bounded within a compact set.

Let $r_t > 0$ denote the upper bound of $\mathbf{y}_t$, i.e., $|\mathbf{y}_t| \leq r_t$, $t = 1, ..., T$. Without loss of generality, we assume that the true outputs are also bounded by r_t. Therefore, the MSE loss function is a locally Lipschitz continuous function satisfying the following inequality for all $|\mathbf{y}_t|, |\bar{\mathbf{y}}_t| \leq r_t$.

$$|L(\mathbf{y}_1, \bar{\mathbf{y}}) - L(\mathbf{y}_2, \bar{\mathbf{y}})| \leq L_r |\mathbf{y}_1 - \mathbf{y}_2| \tag{16}$$

where L_r is the local Lipschitz constant.

The generalization error of a neural network function h_S chosen from a hypothesis class $\mathcal{H}$ based on a certain learning algorithm and a training dataset S drawn from distribution D can be decomposed into the approximation error and the estimation error as follows:

$$L_D(h_S) - L_D(h^*) = (\min_{h \in \mathcal{H}} L_D(h) - L_D(h^*)) + (L_D(h_S) - \min_{h \in \mathcal{H}} L_D(h)) \tag{17}$$

where the first and second terms in parentheses represent **approximation error** and **estimation error**, respectively. Specifically, $L_D(h_S)$ represents the error evaluated using the hypothesis h_S over the underlying data distribution D. h^* represents the optimal hypothesis (maybe outside of the finite hypothesis class $\mathcal{H}$) for the data distribution D. $\min_{h \in \mathcal{H}} L_D(h)$ is the optimal hypothesis within $\mathcal{H}$ that minimizes the loss functions over the distribution D. It can be seen that the approximation error depends on how close the hypothesis class $\mathcal{H}$ is to the optimal hypothesis h^*. In other words, a larger hypothesis class $\mathcal{H}$ generally leads to a lower approximation error since it is more likely that the optimal hypothesis h^* is included in $\mathcal{H}$. The estimation error depends on both the hypothesis class size and training data, and characterizes how good the selected hypothesis h_S associated with the training dataset S is with respect to the best hypothesis $\min_{h \in \mathcal{H}} L_D(h)$ within hypothesis class $\mathcal{H}$. As a result, a larger hypothesis class $\mathcal{H}$ may in turn lead to a higher estimation error since it is more difficult to find the optimal hypothesis within $\mathcal{H}$ over the distribution D. From the error decomposition of Equation (17), we demonstrate the dependencies of generalization error on the training dataset size and the complexity of hypothesis class. In the next section, we will take advantage of Rademacher complexity technique to derive a generalization error bound accounting for its dependencies on the above factors in a quantitative aspect. The results will also provide a guide for the design of neural network structures and the collection of training data in order to achieve a desired generalization performance for a specific modeling task.

3.2. Rademacher Complexity Bound

Rademacher complexity quantifies the richness of a class of functions, and is often used in machine learning theory to bound the generalization error. The definition of empirical Rademacher complexity is given below.

Definition 4 (Empirical Rademacher Complexity)**.** *Given a hypothesis class $\mathcal{F}$ of real-valued functions, and a set of data samples $S = \{s_1, ..., s_m\}$, the empirical Rademacher complexity of $\mathcal{F}$ is defined as*

$$\mathcal{R}_S(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\epsilon}} \left[\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \epsilon_i f(s_i) \right] \tag{18}$$

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

We also have the following contraction inequality for the hypothesis class $\mathcal{H}$ of vector-valued functions $h \in \mathbf{R}^{d_y}$.

Lemma 2 (c.f. Corollary 4 in [36])**.** *Consider a hypothesis class $\mathcal{H}$ of vector-valued functions $h \in \mathbf{R}^{d_y}$, and a set of data samples $S = \{s_1, ..., s_m\}$. Let $L(\cdot)$ be a L_r-Lipschitz function mapping $h \in \mathbf{R}^{d_y}$ to $\mathbf{R}$, then we have*

$$\mathbb{E}_\epsilon\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\epsilon_i L(h(\boldsymbol{x}_i), \boldsymbol{y}_i)\right] \le \sqrt{2}L_r\mathbb{E}_\epsilon\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\sum_{k=1}^{d_y}\epsilon_{ik}h_k(\boldsymbol{x}_i)\right] \tag{19}$$

where $h_k(\cdot)$ is the k-th component in the vector-valued function $h(\cdot)$, and ϵ_{ik} is an $m \times d_y$ matrix of independent Rademacher variables. In the following text, we will omit the subscript $\boldsymbol{\epsilon}$ of expectation for simplicity.

Since the RHS of Equation (19) is generally difficult to compute, we can reduce it to scalar classes, and derive the following bound [36]:

$$\mathbb{E}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\sum_{k=1}^{d_y}\epsilon_{ik}h_k(\mathbf{x}_i)\right] \le \sum_{k=1}^{d_y}\mathbb{E}\left[\sup_{h\in\mathcal{H}_k}\sum_{i=1}^{m}\epsilon_i h(\mathbf{x}_i)\right] \tag{20}$$

where $\mathcal{H}_k$, $k = 1, ..., d_y$, are classes of scalar-valued functions that correspond to the components of vector-valued functions in $\mathcal{H}$. Equation (20) will later be used in the derivation of the generalization error bound for RNN models approximating the nonlinear system of Equation (1).

Let $\mathcal{G}_t$ be the family of loss functions associated to $\mathcal{H}$ mapping the first t-time-step inputs $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t\} \in \mathbf{R}^{d_x\times t}$ to the t-th output $\mathbf{y}_t \in \mathbf{R}^{d_y}$.

$$\mathcal{G}_t = \{g_t : (\mathbf{x}, \bar{\mathbf{y}}) \to L(h(\mathbf{x}), \bar{\mathbf{y}}), h \in \mathcal{H}\} \tag{21}$$

where $\mathbf{x}$ is the RNN input vector, and $\bar{\mathbf{y}}$ is the true output vector. The following lemma characterizes the upper bound for the generalization error using Rademacher complexity $\mathcal{R}_S(\mathcal{G}_t)$.

Lemma 3 (c.f. Theorem 3.3 in [37])**.** *Given a set of m i.i.d. data samples, with probability at least $1-\delta$ over samples $S = (\boldsymbol{x}_{i,t}, \boldsymbol{y}_{i,t})_{t=1}^{T}$, $i = 1, ..., m$, the following inequality holds for all $g_t \in \mathcal{G}_t$:*

$$\mathbb{E}[g_t(\boldsymbol{x}, \boldsymbol{y})] \le \frac{1}{m}\sum_{i=1}^{m} g_t(\boldsymbol{x}_i, \boldsymbol{y}_i) + 2\mathcal{R}_S(\mathcal{G}_t) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{22}$$

Proof. While the full proof can be found in many machine learning books, e.g., [37], a proof sketch is presented below to help readers understand the derivation of Equation (22). To simplify the notations, let $\mathbb{E}[g_t]$ and $\hat{\mathbb{E}}_S[g_t]$ denote the expected loss $\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})]$ and the empirical loss $\frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i)$ based on a dataset S with m data samples, respectively. Additionally, we assume that $g_t(\mathbf{x}, \mathbf{y})$ is bounded in $[0, 1]$ (if not, we can scale the RNN output layer or loss function) without loss of generality. We define $\beta(S)$ to be a function of data samples $S = (s_1, s_2, ..., s_m)$ as follows, where s_i represents each data sample $(\mathbf{x}_{i,t}, \mathbf{y}_{i,t})$, $i = 1, ..., m$.

$$\beta(S) = \sup_{g_t\in\mathcal{G}_t}(\mathbb{E}[g_t] - \hat{\mathbb{E}}_S[g_t]) \tag{23}$$

Given two datasets $S = (s_1, ..., s_i, ..., s_m)$ and $S' = (s_1, ..., s'_i, ..., s_m)$ with only one different data point, i.e., $s_i \neq s'_i$, the following inequality holds for any $g_t(\mathbf{x}_i, \mathbf{y}_i) \in [0, 1]$:

$$\begin{aligned} \left|\beta(S) - \beta(S')\right| &= \left|\sup_{g_t \in \mathcal{G}_t} (\mathbb{E}[g_t] - \hat{\mathbb{E}}_S[g_t]) - \sup_{g_t \in \mathcal{G}_t} (\mathbb{E}[g_t] - \hat{\mathbb{E}}_{S'}[g_t])\right| \\ &\leq \left|\sup_{g_t \in \mathcal{G}_t} (\hat{\mathbb{E}}_{S'}[g_t] - \hat{\mathbb{E}}_S[g_t])\right| \\ &= \left|\sup_{g_t \in \mathcal{G}_t} \frac{g_t(s'_i) - g_t(s_i)}{m}\right| \\ &\leq \frac{1}{m} \end{aligned} \tag{24}$$

Then, using the McDiarmid's inequality in Lemma 1 and letting $a \geq \sqrt{\frac{\log(\frac{2}{\delta})}{2m}}$, we have

$$\mathbb{P}[\beta(S) - \mathbb{E}_S[\beta(S)] \geq a] \leq \exp\left(\frac{-2a^2}{\sum_{i=1}^{m} \frac{1}{m^2}}\right) = \exp(-2a^2 m) \leq \frac{\delta}{2} \tag{25}$$

where $\mathbb{E}_S[\beta(S)]$denotes the expectation of $\beta(S)$ with respect to the dataset S of m data samples. Equivalently, the following inequality holds with probability at least $1 - \frac{\delta}{2}$, for any $\delta > 0$, :

$$\beta(S) \leq \mathbb{E}_S[\beta(S)] + \sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{26}$$

Next, we derive the upper bound for $\mathbb{E}_S[\beta(S)]$ as follows:

$$\begin{aligned} \mathbb{E}_S[\beta(S)] &= \mathbb{E}_S\left[\sup_{g_t \in \mathcal{G}_t} (\mathbb{E}[g_t] - \hat{\mathbb{E}}_S[g_t])\right] \\ &\leq \mathbb{E}_{S,S'}\left[\sup_{g_t \in \mathcal{G}_t} (\hat{\mathbb{E}}_{S'}[g_t] - \hat{\mathbb{E}}_S[g_t])\right] \\ &= \mathbb{E}_{\epsilon,S,S'}\left[\sup_{g_t \in \mathcal{G}_t} \left(\frac{1}{m}\sum_{i=1}^{m} \epsilon_i (g_t(s'_i) - g_t(s_i))\right)\right] \\ &\leq \mathbb{E}_{\epsilon,S'}\left[\sup_{g_t \in \mathcal{G}_t} \left(\frac{1}{m}\sum_{i=1}^{m} \epsilon_i g_t(s'_i)\right)\right] + \mathbb{E}_{\epsilon,S}\left[\sup_{g_t \in \mathcal{G}_t} \left(\frac{1}{m}\sum_{i=1}^{m} -\epsilon_i g_t(s_i)\right)\right] \\ &= 2\mathbb{E}_{\epsilon,S}\left[\sup_{g_t \in \mathcal{G}_t} \frac{1}{m}\sum_{i=1}^{m} \epsilon_i g_t(s_i)\right] = 2\mathbb{E}_{\epsilon,S}[\mathcal{R}_S(\mathcal{G}_t)] \end{aligned} \tag{27}$$

where the first line is by substituting the definition of Equation (23) into $\mathbb{E}_S[\beta(S)]$. The second line is derived using the fact that $\mathbb{E}[g_t] = \mathbb{E}_{S'}[\hat{\mathbb{E}}_{S'}(g_t)]$ and the property of supremum function: $\sup_{g_t \in \mathcal{G}_t} \mathbb{E}_{S'}(f(S', g_t)) \leq \mathbb{E}_{S'}[\sup_{g_t \in \mathcal{G}_t} f(S', g_t)]$ for any function f. The third line is derived by introducing Rademacher variables ϵ_i, which do not affect its outcome since ϵ_i are i.i.d. randome variables taking values in $\{-1, +1\}$. The fourth line is obtained by separating the supremum function as $\sup(f + g) \leq \sup(f) + \sup(g)$, and the last line is derived using the fact that Rademacher variables ϵ_i have a symmetric distribution. Note that $\mathbb{E}_{\epsilon,S}[\mathcal{R}_S(\mathcal{G}_t)]$ in the last line of Equation (27) represents the expectation of the empirical Rademacher complexity, $\mathcal{R}_S(\mathcal{G}_t)$, over all samples of size m drawn from the same distribution. In order to bound this term, we apply McDiarmid's inequality again using confidence $\frac{\delta}{2}$, which yields a similar result as in Equation (25). Finally, using union bound which states that $\mathbb{P}(\cup_i A_i) \le \sum_i \mathbb{P}(A_i)$ holds for any finite or countable set of events A_i, $i = 1, 2, ...$, the following inequality holds with probability at least $1 - \delta$:

$$\begin{aligned} \beta(S) &\le 2\left(\mathcal{R}_S(\mathcal{G}_t) + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}\right) + \sqrt{\frac{\log \frac{2}{\delta}}{2m}} \\ &= 2\mathcal{R}_S(\mathcal{G}_t) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}} \end{aligned} \tag{28}$$

By substituting the definition of $\beta(S)$ of Equation (23) into the above equation, we obtain the result in Equation (22). This completes the proof of Lemma 3. □

It can be seen from Equation (22) that the generalization error bound depends on the empirical error (the first term), Rademacher complexity (the second term), and an error function associated with the 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 $\mathcal{R}_S(\mathcal{G}_t)$. Since most of the established results of Rademacher complexity are with respect to feedforward neural networks modeling real-valued functions only, we will start with a lemma for the hypothesis class of real-valued functions.

Lemma 4. *Given a hypothesis class $\mathcal{H}_k$ of real-valued functions corresponding to the k-th component of vector-valued function class $\mathcal{H}$, and a set of m i.i.d. data samples $S = (\boldsymbol{x}_{i,t}, \boldsymbol{y}_{i,t})_{t=1}^T$, $i = 1, ..., m$, the following inequality holds for the scaled empirical Rademacher complexity $m\mathcal{R}_S(\mathcal{H}_k) = \mathbb{E}[\sup_{h \in \mathcal{H}_k} \sum_{i=1}^m \epsilon_i h(\boldsymbol{x}_i)]$.*

$$\begin{aligned} m\mathcal{R}_S(\mathcal{H}_k) &= \frac{1}{\lambda} \log \exp\left(\lambda \mathbb{E}\left[\sup_{h \in \mathcal{H}_k} \sum_{i=1}^m \epsilon_i h(\boldsymbol{x}_i)\right]\right) \\ &\le \frac{1}{\lambda} \log\left(\mathbb{E}\left[\sup_{h \in \mathcal{H}_k} \exp(\lambda \sum_{i=1}^m \epsilon_i h(\boldsymbol{x}_i))\right]\right) \end{aligned} \tag{29}$$

where $\lambda > 0$ is an arbitrary parameter.

Proof. Equation (29) can be readily proved by using Jensen's inequality which states that given a random variable X and a convex function $\beta(\cdot)$, it holds that $\beta(\mathbb{E}[X]) \le \mathbb{E}[\beta(X)]$. Equation (29) will be used in the derivation of the upper bound for the Rademacher complexity $R_S(\mathcal{H})$ in Lemma 7. □

We can see from the definition of Rademacher complexity of Equation (18) that the value of $\mathcal{R}_S(\mathcal{G}_t)$ depends on the complexity of hypothesis class $\mathcal{G}_t$. However, since the RNN model of Equations (8) and (9) is a complex nonlinear function which is difficult to measure its learning capacity, we need to peel off the nonlinear activation functions and weight matrices through layers. The following lemma shows the "peeling" step used in the derivation of Rademacher complexity for the output layer of RNNs.

Lemma 5 (c.f. Lemma 1 in [27])**.** *Given a hypothesis class $\mathcal{H}$ of vector-valued functions that map the RNN inputs $\boldsymbol{x} \in \mathbf{R}^{d_x}$ to the hidden states $\boldsymbol{h} \in \mathbf{R}^{d_h}$, and any convex and monotonically increasing function $p : \mathbf{R} \to \mathbf{R}_+$, the following inequality holds for the RNN model of Equations (8) and (9) with a 1-Lipschitz, positive-homogeneous activation function $\sigma_y(\cdot)$:*

$$\mathbb{E}\left[\sup_{h \in \mathcal{H}, ||V||_F \le B_{V,F}} p\left(\left|\sum_{i=1}^m \epsilon_i \sigma_y(V\boldsymbol{h}_i)\right|\right)\right] \le 2 \cdot \mathbb{E}\left[\sup_{h \in \mathcal{H}} p\left(B_{V,F} \cdot \left|\sum_{i=1}^m \epsilon_i \boldsymbol{h}_i\right|\right)\right] \tag{30}$$

Proof. The proof is omitted here as it is similar to the proof for the next lemma, which will be presented in detail. Interested readers can refer to [27] for the proof of Lemma 5. □

Lemma 5 peels off the weight matrix V between the RNN hidden layer and output layer. To further peel off the weight matrices in the RNN hidden layers, we provide the following lemma.

Lemma 6. *Given a hypothesis class $\mathcal{H}$ of vector-valued functions that map the RNN inputs $\boldsymbol{x} \in \mathbf{R}^{d_x}$ to the hidden states $\boldsymbol{h} \in \mathbf{R}^{d_h}$, and any convex and monotonically increasing function $p : \mathbf{R} \to \mathbf{R}_+$, the following equation holds for the RNN model of Equations (8) and (9) with a 1-Lipschitz, positive-homogeneous activation function $\sigma_h(\cdot)$:*

$$\begin{aligned}
&\mathbb{E}\left[\sup_{h\in\mathcal{H},||U||_F\le B_{U,F},||W||_F\le B_{W,F},} p\left(\left|\sum_{i=1}^{m}\epsilon_i \boldsymbol{h}_{i,t}\right|\right)\right] \\
&= \mathbb{E}\left[\sup_{h\in\mathcal{H},||U||_F\le B_{U,F},||W||_F\le B_{W,F}} p\left(\left|\sum_{i=1}^{m}\epsilon_i \sigma_h(U\boldsymbol{h}_{i,t-1}+W\boldsymbol{x}_{i,t})\right|\right)\right] \\
&\le 2\mathbb{E}\left[\sup_{h\in\mathcal{H}} p\left(B_{U,F}\left|\sum_{i=1}^{m}\epsilon_i \boldsymbol{h}_{i,t-1}\right| + B_{W,F}\left|\sum_{i=1}^{m}\epsilon_i \boldsymbol{x}_{i,t}\right|\right)\right]
\end{aligned} \tag{31}$$

Proof. We first define an augmented weight matrix $Z = [U|W] \in \mathbf{R}^{d_h\times(d_h+d_x)}$, and an augmented vector $\bar{\mathbf{h}}_{i,t} = [\mathbf{h}_{i,t-1}|\mathbf{x}_{i,t}] \in \mathbf{R}^{d_h+d_x}$. To simplify the discussion, we assume that the Frobenius norm of the matrix Z is bounded by $||Z||_F \le B_{Z,F}$, given that both U and W are bounded by $||U||_F \le B_{U,F}$ and $||W||_F \le B_{W,F}$. Then, the hidden layer vector at t-th time step, $\mathbf{h}_{i,t}$, can be written as follows:

$$\mathbf{h}_{i,t} = \sigma_h(U\mathbf{h}_{i,t-1} + W\mathbf{x}_{i,t}) = \sigma_h(Z\bar{\mathbf{h}}_{i,t}) \tag{32}$$

Letting $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_h$ denote the rows of the matrix Z, we have

$$\left|\sum_{i=1}^{m}\epsilon_i \mathbf{h}_{i,t}\right|^2 = \sum_{j=1}^{d_h}|\mathbf{z}_j|^2\left(\sum_{i=1}^{m}\epsilon_i\sigma_h\Big(\frac{\mathbf{z}_j^T}{|\mathbf{z}_j|}\bar{\mathbf{h}}_{i,t}\Big)\right)^2 \tag{33}$$

The supremum of Equation (33) over all the weight matrix $Z = [\mathbf{z}_1\ \mathbf{z}_2\ ...\ \mathbf{z}_h]$ that satisfies $||Z||_F \le B_{Z,F}$ is obtained when $|\mathbf{z}_j| = B_{Z,F}$ for some j, and $|\mathbf{z}_i| = 0$ for all $i \ne j$. Therefore, we have

$$\begin{aligned}
&\mathbb{E}\left[\sup_{h\in\mathcal{H},||U||_F\le B_{U,F},||W||_F\le B_{W,F}} p\left(\left|\sum_{i=1}^{m}\epsilon_i \mathbf{h}_{i,t}\right|\right)\right] \\
&= \mathbb{E}\left[\sup_{h\in\mathcal{H},|\mathbf{z}|=B_{Z,F}} p\left(\left|\sum_{i=1}^{m}\epsilon_i \sigma_h(\mathbf{z}^T\bar{\mathbf{h}}_{i,t})\right|\right)\right]
\end{aligned} \tag{34}$$

Since $p(\cdot)$ is a convex and monotonically increasing function, $p(|a|) \le p(a) + p(-a)$ holds, and the above equation can be further bounded as follows:

$$\begin{aligned}
\mathbb{E}\left[\sup_{h\in\mathcal{H},|\mathbf{z}|=B_{Z,F}} p\left(\left|\sum_{i=1}^{m}\epsilon_i \sigma_h(\mathbf{z}^T\bar{\mathbf{h}}_{i,t})\right|\right)\right] &\le \mathbb{E}\left[\sup_{h\in\mathcal{H},|\mathbf{z}|=B_{Z,F}} p\left(\sum_{i=1}^{m}\epsilon_i \sigma_h(\mathbf{z}^T\bar{\mathbf{h}}_{i,t})\right)\right] \\
&\quad + \mathbb{E}\left[\sup_{h\in\mathcal{H},|\mathbf{z}|=B_{Z,F}} p\left(-\sum_{i=1}^{m}\epsilon_i \sigma_h(\mathbf{z}^T\bar{\mathbf{h}}_{i,t})\right)\right] \\
&= 2\mathbb{E}\left[\sup_{h\in\mathcal{H},|\mathbf{z}|=B_{Z,F}} p\left(\sum_{i=1}^{m}\epsilon_i \sigma_h(\mathbf{z}^T\bar{\mathbf{h}}_{i,t})\right)\right]
\end{aligned} \tag{35}$$

where the last equality is derived from the fact that the random variables ϵ_i have a symmetric distribution, i.e., $\mathbb{P}(\epsilon_i = 1) = \mathbb{P}(\epsilon_i = -1) = 0.5$. Following the proof in [27] and Theorem 4.12 in [38], the RHS of Equation (35) can be further bounded by

$$\begin{aligned} 2\mathbb{E}\left[\sup_{h\in\mathcal{H},|\mathbf{z}|=B_{Z,F}} p\left(\sum_{i=1}^{m}\epsilon_i\sigma_h(\mathbf{z}^T\bar{\mathbf{h}}_{i,t})\right)\right] &\le 2\mathbb{E}\left[\sup_{h\in\mathcal{H},|\mathbf{z}|=B_{Z,F}} p\left(\sum_{i=1}^{m}\epsilon_i\mathbf{z}^T\bar{\mathbf{h}}_{i,t}\right)\right] \\ &\le 2\mathbb{E}\left[\sup_{h\in\mathcal{H},|\mathbf{u}|=B_{U,F},|\mathbf{w}|=B_{W,F}} p\left(|\mathbf{u}|\left|\sum_{i=1}^{m}\epsilon_i\mathbf{h}_{i,t-1}\right| + |\mathbf{w}|\left|\sum_{i=1}^{m}\epsilon_i\mathbf{x}_{i,t}\right|\right)\right] \\ &= 2\mathbb{E}\left[\sup_{h\in\mathcal{H}} p\left(B_{U,F}\left|\sum_{i=1}^{m}\epsilon_i\mathbf{h}_{i,t-1}\right| + B_{W,F}\left|\sum_{i=1}^{m}\epsilon_i\mathbf{x}_{i,t}\right|\right)\right] \end{aligned} \tag{36}$$

□

Based on Lemmas 5 and 6, the following lemma provides an upper bound for the Rademacher complexity of the RNN hypothesis class.

Lemma 7. *Let $\mathcal{H}_{k,t}$, $k = 1, ..., d_y$, be the class of real-valued functions that corresponds to the k-th component of the RNN output at t-th time step, with weight matrices and activation functions satisfying Assumptions 1–4. Given a set of m i.i.d. data samples $S = (\boldsymbol{x}_{i,t}, \boldsymbol{y}_{i,t})_{t=1}^{T}$, $i = 1, ..., m$, the following equation holds for the Rademacher complexity:*

$$\mathcal{R}_S(\mathcal{H}_{k,t}) \le \frac{M(\sqrt{2\log(2)t}+1)B_X}{\sqrt{m}} \tag{37}$$

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

Proof. Let $\mathbf{v}_k$ be the k-th row in the weight matrix V. Using Equations (29) and (30), the scaled Rademacher complexity $m\mathcal{R}_S(\mathcal{H}_k)$ can be bounded as follows:

$$\begin{aligned} m\mathcal{R}_S(\mathcal{H}_{k,t}) &= \mathbb{E}\left[\sup_{h\in\mathcal{H}_{k,t},||V||_F\le B_{V,F}} \sum_{i=1}^{m}\epsilon_i\sigma_y(\mathbf{v}_k\mathbf{h}_{i,t})\right] \\ &\le \frac{1}{\lambda}\log\mathbb{E}\left[\sup_{h\in\mathcal{H}_{k,t},||V||_F\le B_{V,F}} \exp\left(\lambda\sum_{i=1}^{m}\epsilon_i\sigma_y(\mathbf{v}_k\mathbf{h}_{i,t})\right)\right] \\ &\le \frac{1}{\lambda}\log\mathbb{E}\left[\sup_{h\in\mathcal{H}_{k,t}} \exp\left(B_{V,F}\lambda\left|\sum_{i=1}^{m}\epsilon_i\mathbf{h}_{i,t}\right|\right)\right] \end{aligned} \tag{38}$$

where $\exp(\cdot)$ corresponds to the monotonically increasing function $p(\cdot)$ in Lemmas 5 and 6. Then, we use Equation (31) and further derive the bound for the RHS of the above equation as follows:

$$\begin{aligned} &\frac{1}{\lambda}\log\mathbb{E}\left[\sup_{h\in\mathcal{H}_{k,t}} \exp\left(B_{V,F}\lambda\left|\sum_{i=1}^{m}\epsilon_i\mathbf{h}_{i,t}\right|\right)\right] \\ &\le \frac{1}{\lambda}\log\left(2\cdot\mathbb{E}\left[\sup_{h\in\mathcal{H}_{k,t-1}} \exp\left(B_{V,F}\lambda\cdot\left(B_{U,F}\left|\sum_{i=1}^{m}\epsilon_i\mathbf{h}_{i,t-1}\right| + B_{W,F}\left|\sum_{i=1}^{m}\epsilon_i\mathbf{x}_{i,t}\right|\right)\right)\right]\right) \end{aligned} \tag{39}$$

Assuming that the initial hidden states $\mathbf{h}_{i,0} = 0$, by recursively applying Lemma 6 to the term $|\sum_{i=1}^{m}\epsilon_i\mathbf{h}_{i,t-1}|$ in Equation (39), we obtain that

$$\begin{aligned} m\mathcal{R}_S(\mathcal{H}_{k,t}) &\leq \frac{1}{\lambda}\log\left(2^t \cdot \mathbb{E}\left[\exp\left(B_{V,F}\lambda \cdot \left(B_{W,F} \cdot \left|\sum_{i=1}^{m} \epsilon_i \mathbf{x}_{i,t}\right| \cdot \sum_{j=0}^{t-1}(B_{U,F})^j\right)\right)\right]\right) \\ &= \frac{1}{\lambda}\log\left(2^t \cdot \mathbb{E}\left[\exp\left(B_{V,F}\lambda \cdot \left(B_{W,F} \cdot \left|\sum_{i=1}^{m} \epsilon_i \mathbf{x}_{i,t}\right| \cdot \frac{(B_{U,F})^t - 1}{B_{U,F} - 1}\right)\right)\right]\right) \end{aligned} \tag{40}$$

It is noted that the RNN model in this work is developed to predict one sampling time, for which the RNN inputs $\mathbf{x}_{i,t}$ remain the same. If the RNN inputs are varying over time, Equation (40) can be modified by taking the maximum value of $|\sum_{i=1}^{m} \epsilon_i \mathbf{x}_{i,t}|$ within the prediction period. Subsequently, we define the following random variable q

$$q = M\left|\sum_{i=1}^{m} \epsilon_i \mathbf{x}_{i,t}\right| \tag{41}$$

where the randomness comes from the Rademacher variables ϵ_i, and M denotes the product of all weight matrices, i.e., $M = B_{V,F}B_{W,F}\frac{(B_{U,F})^t-1}{B_{U,F}-1}$. Then, Equation (40) can be written as

$$\begin{aligned} m\mathcal{R}_S(\mathcal{H}_{k,t}) &\leq \frac{1}{\lambda}\log(2^t \cdot \mathbb{E}[\exp(\lambda q)]) \\ &= \frac{t\log(2)}{\lambda} + \frac{1}{\lambda}\log(\mathbb{E}[\exp(\lambda(q - \mathbb{E}[q]))]) + \mathbb{E}[q] \end{aligned} \tag{42}$$

Using Jensen's inequality, we can bound $\mathbb{E}[q]$ as follows:

$$\mathbb{E}[q] = \mathbb{E}\left[M\left|\sum_{i=1}^{m} \epsilon_i \mathbf{x}_{i,t}\right|\right] \leq M\sqrt{\mathbb{E}\left[\left|\sum_{i=1}^{m} \epsilon_i \mathbf{x}_{i,t}\right|^2\right]} = M\sqrt{\sum_{i=1}^{m}|\mathbf{x}_{i,t}|^2} \leq \sqrt{m}MB_X \tag{43}$$

where the second equality comes from the fact that ϵ_i are i.i.d. Rademacher random variables, and the last inequality is due to the assumption that $|\mathbf{x}_{i,t}| \leq B_X$. Subsequently, following the results in [38], we can show q is sub-Gaussian with the following variance factor v since q satisfies a bounded-difference condition with respect to its random variables ϵ_i, i.e., $q(\epsilon_1, ..., \epsilon_i, ..., \epsilon_m) - q(\epsilon_1, ..., -\epsilon_i, ..., \epsilon_m) \leq 2M|\mathbf{x}_{i,t}|$.

$$v = \frac{1}{4}\sum_{i=1}^{m}(2M|\mathbf{x}_{i,t}|)^2 = M^2\sum_{i=1}^{m}|\mathbf{x}_{i,t}| \tag{44}$$

According to the property of sub-Gaussian random variables in Definition 1, the following inequality holds for q:

$$\frac{1}{\lambda}\log(\mathbb{E}[\exp(\lambda(q - \mathbb{E}[q]))]) \leq \frac{\lambda M^2\sum_{i=1}^{m}|\mathbf{x}_{i,t}|}{2} \tag{45}$$

Let $\lambda = \frac{\sqrt{2\log(2)t}}{M\sqrt{\sum_{i=1}^{m}|\mathbf{x}_{i,t}|^2}} > 0$. The Rademacher complexity $m\mathcal{R}_S(\mathcal{H}_{k,t})$ in Equation (42) can be bounded as follows:

$$\begin{aligned} m\mathcal{R}_S(\mathcal{H}_{k,t}) &\leq \frac{t\log(2)}{\lambda} + \frac{1}{\lambda}\log(\mathbb{E}[\exp(\lambda(q - \mathbb{E}[q]))]) + \mathbb{E}[q] \\ &\leq M(\sqrt{2\log(2)t} + 1)\sqrt{\sum_{i=1}^{m}|\mathbf{x}_{i,t}|^2} \\ &\leq M(\sqrt{2\log(2)t} + 1)\sqrt{m}B_X \end{aligned} \tag{46}$$

□

Lemma 7 develops the Rademacher complexity upper bound for the hypothesis class $\mathcal{H}_k$ of real-valued functions that map RNN inputs to the k-th output. Subsequently, we derive the generalization bound for the loss function associated with the vector-valued functions that map the RNN inputs to the output vector by taking advantage of the contraction inequality of Equations (19) and (20).

Theorem 1. *Let $\mathcal{G}_t$ be the family of loss function associated to the hypothesis class $\mathcal{H}_t$ of vector-valued functions that map the RNN inputs to the RNN output at t-th time step, with weight matrices and activation functions satisfying Assumptions 1–4. Given a set of m i.i.d. data samples $S = (\boldsymbol{x}_{i,t}, \boldsymbol{y}_{i,t})_{t=1}^{T}$, $i = 1, ..., m$, with probability at least $1 - \delta$ over S, we have r*

$$\mathbb{E}[g_t(\boldsymbol{x}, \boldsymbol{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\boldsymbol{x}_i, \boldsymbol{y}_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{M(\sqrt{2\log(2)t}+1)B_X}{\sqrt{m}}\right) \tag{47}$$

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

Proof. Using the results in Lemma 7 and Equations (19) and (20), we can derive the following upper bound for the loss function $L(h(\mathbf{x}_i), \mathbf{y}_i)$ with $h(\mathbf{x}_i)$ being vector-valued functions:

$$\begin{aligned}\mathcal{R}_S(\mathcal{G}_t) = \mathbb{E}\left[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \epsilon_i L(h(\mathbf{x}_i), \mathbf{y}_i)\right] &\leq \sqrt{2}L_r \mathbb{E}\left[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{d_y} \epsilon_{ik} h_k(\mathbf{x}_i)\right] \\ &\leq \sqrt{2}L_r d_y \frac{M(\sqrt{2\log(2)t}+1)B_X}{\sqrt{m}}\end{aligned} \tag{48}$$

Then, substituting Equation (48) into Equation (22), we derive the generalization error bound in Equation (47). □

Remark 2. *As stated in [27], the assumption of positive-homogeneity for the nonlinear activation function can be loosened in some cases, under which a similar result of generalization error bound can be derived. Interested readers are referred to Lemma 2 and Theorem 2 in [27].*

Remark 3. *The generalization error bound of Equation (47) implies that the following attempts can be taken to reduce the generalization error: (1) minimize the empirical loss $\frac{1}{m}\sum_{i=1}^{m} g_t(\boldsymbol{x}_i, \boldsymbol{y}_i)$ 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 Equation (17), 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 last term $\mathcal{O}(\cdot)$ in Equation (47). Therefore, in practice, we generally start with a simple neural network and gradually increase it 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.*

Remark 4. *While the actual generalization error is difficult to obtain due to unknown data distribution and complexity of hypothesis class, Equation (47) characterizes the upper bound for the gap between the generalization error and empirical error by moving the term $\frac{1}{m}\sum_{i=1}^{m} g_t(\boldsymbol{x}_i, \boldsymbol{y}_i)$ to the LHS of Equation (47). Since the neural network training process itself is to minimize the training error only, this generalization gap is more useful in practice by showing how good the neural network will be for unseen data under the same data distribution. In terms of modeling the nonlinear system of Equation (1), this generalization gap provides an upper bound for the modeling error for all the states in the operating region, and can be used in the design of model-based controllers that probabilistically ensure closed-loop stability accounting for bounded modeling errors.*

Remark 5. *It is noticed that the generalization error bound also depends on the time length t of RNN inputs, which is different from the results derived for the feedforward neural networks in [27]. Additionally, unlike other deep neural networks which utilize different parameters for each hidden layer, RNNs share the same weight matrix U at each time step, and therefore, the bound for the product of weight matrices is derived in the form of $M = B_{V,F} B_{W,F} \frac{(B_{U,F})^t - 1}{B_{U,F} - 1}$. From Equation (47), it can be seen that as the data sequence length t increases, the network hypothesis becomes more complex, which leads to a larger generalization error bound. Therefore, a shorter time sequence prediction is preferred from the perspective of prediction accuracy. However, it does not necessarily mean a short prediction period is always desirable from the control perspective, especially in model predictive control (MPC) schemes. In Section 5, we will demonstrate that the RNN models predicting a short period of time achieved desired prediction performance in open-loop tests, but perform poorly in closed-loop simulation due to the error accumulated during successive execution of RNN predictions within MPC prediction horizon.*

4. RNN-Based MPC with Probabilistic Stability Analysis

In this section, we present the formulation of Lyapunov-based MPC (LMPC) that uses RNN models to predict evolution of future states, along with the closed-loop stability analysis showing the boundedness of closed-loop state of Equation (1) in the stability region for all times in probability.

4.1. Lyapunov-Based Control Using RNN Models

To simplify the discussion of RNN stability properties for the continuous-time nonlinear system of Equation (1), we represent the RNN model in the following continuous-time form [9]:

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u) := A\hat{x} + \Theta^T z \tag{49}$$

where $\hat{x} \in \mathbf{R}^n$ and $u \in \mathbf{R}^k$ are the RNN state vector and the manipulated input vector, respectively. $z = [z_1, ..., z_n, z_{n+1}, ..., z_{k+n}] = [\sigma(\hat{x}_1), ..., \sigma(\hat{x}_n), u_1, ..., u_k] \in \mathbf{R}^{n+k}$ is a vector of both the input u and the network state $\hat{x}$, where $\sigma(\cdot)$ represents the nonlinear activation function. A is a diagonal coefficient matrix with all diagonal elements being negative, and $\Theta = [\theta_1, ..., \theta_n] \in \mathbf{R}^{(k+n)\times n}$ with $\theta_i = b_i[w_{i1}, ..., w_{i(k+n)}]$, $i = 1, ..., n$, where w_{ij} denotes the weight connecting the jth input to the ith neuron, $i = 1, ..., n$ and $j = 1, ..., (k+n)$. The weight matrices and activation functions satisfy Assumptions 1–4. To simplify the notation, we use Equation (49) to represent one-hidden-layer RNN model, and bias terms are not explicitly included in Equation (49); however, it is noted that the results that we will derive in this section are not restricted to one-hidden-layer RNN models, and can be extended to deep RNNs with multiple hidden layers.

We assume that there exists a stabilizing feedback controller $u = \Phi_{nn}(x) \in U$ that can render the origin of the RNN model of Equation (49) exponentially stable in an open neighborhood $\hat{D}$ around the origin. The stabilizability assumption implies the existence of a $\mathcal{C}^1$ control Lyapunov function $\hat{V}(x)$ such that the following inequalities hold for all x in $\hat{D}$:

$$\hat{c}_1 |x|^2 \leq \hat{V}(x) \leq \hat{c}_2 |x|^2, \tag{50}$$

$$\frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x, \Phi_{nn}(x)) \leq -\hat{c}_3 |x|^2, \tag{51}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} \right| \leq \hat{c}_4 |x| \tag{52}$$

where $\hat{c}_1$, $\hat{c}_2$, $\hat{c}_3$, $\hat{c}_4$ are positive constants. The closed-loop stability region for the RNN model of Equation (49) is characterized as a level set of Lyapunov function embedded in $\hat{D}$ as follows: $\Omega_{\hat{\rho}} := \{x \in \hat{D} \mid \hat{V}(x) \leq \hat{\rho}\}$, where $\hat{\rho} > 0$. Additionally, there exist positive constants M_{nn} and L_{nn} such that the following inequalities hold for all $x, x' \in \Omega_{\hat{\rho}}$ and $u \in U$:

$$|F_{nn}(x,u)| \leq M_{nn} \tag{53}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x,u) - \frac{\partial \hat{V}(x')}{\partial x} F_{nn}(x',u) \right| \leq L_{nn}|x - x'| \tag{54}$$

Due to the model mismatch between the nonlinear system of Equation (1) and the RNN model of Equation (49), the following proposition is developed to demonstrate that the feedback controller $u = \Phi_{nn}(x) \in U$ is able to stabilize the system of Equation (1) with high probability if the modeling error is sufficiently small.

Proposition 1. *Consider the RNN model trained using a set of m i.i.d. data samples* $S = (\boldsymbol{x}_{i,t}, \boldsymbol{y}_{i,t})_{t=1}^{T}$, $i = 1, ..., m$, *and satisfying Assumptions 1–4. Under the assumption that the feedback controller* $u = \Phi_{nn}(x) \in U$ *renders the the origin of the RNN system of Equation (49) exponentially stable for all* $x \in \Omega_{\hat{\rho}}$, *if for all* $x \in \Omega_{\hat{\rho}}$ *and* $u \in U$, *the modeling error can be constrained by* $|F(x,u) - F_{nn}(x,u)| \leq \gamma|x|$, *where* γ *is a positive real number satisfying* $\gamma < \hat{c}_3/\hat{c}_4$, *then the controller* $u = \Phi_{nn}(x) \in U$ *also renders the origin of the nonlinear system of Equation (1) exponentially stable with probability at least* $1 - \delta$ *for all* $x \in \Omega_{\hat{\rho}}$.

Proof. To demonstrate that the origin of the nominal system of Equation (1) can be rendered exponentially stable $\forall x \in \Omega_{\hat{\rho}}$ with probability at least $1 - \delta$ under the controller $u = \Phi_{nn}(x) \in U$ designed for the RNN model of Equation (49), we prove that the time-derivative of $\hat{V}$ associated with the state x of Equation (1) can be rendered negative in probability under $u = \Phi_{nn}(x) \in U$. Based on Equations (51) and (52), $\dot{\hat{V}}$ is derived as follows:

$$\begin{aligned} \dot{\hat{V}} =& \frac{\partial \hat{V}(x)}{\partial x} F(x, \Phi_{nn}(x)) \\ =& \frac{\partial \hat{V}(x)}{\partial x} (F_{nn}(x, \Phi_{nn}(x)) + F(x, \Phi_{nn}(x)) - F_{nn}(x, \Phi_{nn}(x))) \\ \leq& -\hat{c}_3|x|^2 + \hat{c}_4|x| \cdot |(F(x, \Phi_{nn}(x)) - F_{nn}(x, \Phi_{nn}(x)))| \end{aligned} \tag{55}$$

where the last term $|F(x, \Phi_{nn}(x)) - F_{nn}(x, \Phi_{nn}(x))|$ represents the error between the RNN model and the process model of Equation (1). Since the RNN model is trained using sampled data with a sufficiently small time interval (i.e., integration time step h_c), the modeling error term for the same initial state $x(t) = \hat{x}(t)$ can be approximated as follows:

$$\begin{aligned} &|F(x, \Phi_{nn}(x)) - F_{nn}(x, \Phi_{nn}(x))| \\ &\qquad \leq \left| \frac{x(t+h_c) - x(t)}{h_c} - \frac{\hat{x}(t+h_c) - \hat{x}(t)}{h_c} \right| + \mathcal{O}(h_c) \\ &\qquad \leq \left| \frac{x(t+h_c) - \hat{x}(t+h_c)}{h_c} \right| + \mathcal{O}(h_c) \end{aligned} \tag{56}$$

where $\hat{x}$ is the predicted state by RNN model, and x is the state of actual nonlinear system of Equation (1). $\mathcal{O}(h_c)$ is the truncation error from finite difference method. Since $|x(t+h_c) - \hat{x}(t+h_c)|$ represents the Euclidean norm of the prediction error, while the generalization error bound is derived using MSE as loss function in Theorem 1, the modeling error can be bounded as follows:

$$|F(x, \Phi_{nn}(x)) - F_{nn}(x, \Phi_{nn}(x))| \leq E_M \tag{57}$$

where

$$E_M = \frac{1}{h_c}\sqrt{\frac{1}{m}\sum_{i=1}^{m} g(x_i, y_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{M(\sqrt{2\log(2)h_c}+1)B_X}{\sqrt{m}}\right)} + \mathcal{O}(h_c) \tag{58}$$

By choosing the number of samples $m \geq m_N(\delta, h_c, |x|)$, where $m_N(\delta, h_c, |x|)$ is the minimum data sample size satisfying $E_M \leq \gamma|x|$, $\gamma < \hat{c}_3/\hat{c}_4$, we have the following equation showing that $\dot{\hat{V}}$ can be rendered negative for all $x \in \Omega_\rho$ and $x \neq 0$ with probability at least $1-\delta$, i.e., $\mathbb{P}[\dot{\hat{V}} \leq 0] \geq 1-\delta$,

$$\begin{aligned}\dot{\hat{V}} \leq & -\hat{c}_3|x|^2 + \hat{c}_4|x| \cdot |F(x, \Phi_{nn}(x)) - F_{nn}(x, \Phi_{nn}(x))| \\ \leq & -\hat{c}_3|x|^2 + \hat{c}_4|x|\frac{\hat{c}_3|x|}{\hat{c}_4} \\ = & -\tilde{c}_3|x|^2 \\ \leq & 0\end{aligned} \tag{59}$$

where $\tilde{c}_3 = -\hat{c}_3 + \hat{c}_4\gamma < 0$ for any $\gamma < \hat{c}_3/\hat{c}_4$. Therefore, with probability at least $1-\delta$, the closed-loop state of the system of Equation (1) converges to the origin under $u = \Phi_{nn}(x) \in U$ for all $x_0 \in \Omega_{\hat{\rho}}$. □

Remark 6. *The modeling error constraint $E_M \leq \gamma|x|$, $\forall x \in \Omega_{\hat{\rho}}$ implies that more data is needed for states closer to the origin. This is because when x approaches the origin, the upper bound $\gamma|x|$ is close to zero, and therefore, the prediction of $\hat{x}$ should be more accurate in order to yield a desired approximation of system dynamics $\dot{x} = F(x, u)$ using numerical methods. As a result, it seems that an infinite number of data samples may be needed when state converges to the origin (i.e., x is infinitely close to zero). However, we will show in the next subsection that the requirement of such a large dataset for the states around a small neighborhood around the origin is not necessary for operation under MPC. This is because under sample-and-hold implementation of control actions, the states are forced to be bounded in a small ball around the origin, instead of converging to the exact steady-state. Therefore, the modeling error constraint $E_M \leq \gamma|x|$, $\forall x \in \Omega_{\hat{\rho}}$ can be loosened for states in this small ball, which could improve computational efficiency of training process.*

4.2. Stabilization of Nonlinear System under Lyapunov-Based Controller

Subsequently, the following propositions are developed to demonstrate the impact of sample-and-hold implementation of control actions on system stability. Specifically, Proposition 2 demonstrates that in the presence of mismatch between the plant model of Equation (1) and the RNN models of Equation (49), the error between the predicted state and the actual state is bounded in a finite period of time. Then, we consider the Lyapunov-based controller $u = \Phi_{nn}(x)$ applied to the nonlinear system of Equation (1) in sample-and-hold fashion, and demonstrate in Proposition 3 that with high probability, the nonlinear system of Equation (1) can be stabilized using the controller $u = \Phi_{nn}(x)$ designed for the RNN model of Equation (49).

Proposition 2 (c.f. Proposition 3 in [9]). *Consider the nonlinear system $\dot{x} = F(x, u)$ of Equation (1) and the RNN model $\dot{\hat{x}} = F_{nn}(\hat{x}, u)$ of Equation (49) with the same initial condition $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$. There exists a class $\mathcal{K}$ function $f_w(\cdot)$ and a positive constant κ such that the following inequalities hold $\forall x, \hat{x} \in \Omega_{\hat{\rho}}$:*

$$|x(t) - \hat{x}(t)| \leq f_w(t) := \frac{E_M}{L_x}(e^{L_x t} - 1) \tag{60}$$

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{61}$$

Proof. The proof can be found in [9], and is omitted here. Note that the proof in [9] considers the nonlinear system subject to bounded disturbances, while in this work, we consider the nominal system without disturbances only. However, the stability results derived in this section can be readily generalized to the disturbed systems provided that the disturbances are sufficiently small and bounded. Additionally, the modeling error term in [9] is replaced by E_M (see the definition of E_M in Equation (58)) in Equations (60) and (61) which accounts for the RNN generalization error derived in a probabilistic manner. □

The following proposition is developed to show probabilistic closed-loop stability of the nonlinear system of Equation (1) under sample-and-hold implementation of the controller $u = \Phi_{nn}(x) \in U$.

Proposition 3 (c.f. Proposition 4 in [9])**.** *Consider the nonlinear system of Equation (1) with the controller $u = \Phi_{nn}(\hat{x}) \in U$ that meets the conditions of Equations (50)–(52), and the RNN model of Equation (49) that meets all the conditions in Theorem 1. Under the sample-and-hold implementation of control actions, i.e., $u(t) = \Phi_{nn}(\hat{x}(t_k))$, $\forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$. there exist $\epsilon_w > 0$, $\Delta > 0$ and $\hat{\rho} > \rho_{min} > \rho_{nn} > \rho_s$ that satisfy*

$$-\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x' M_F \Delta \leq -\epsilon_w \tag{62}$$

and

$$\rho_{nn} := \max\{\hat{V}(\hat{x}(t+\Delta)) \mid \hat{x}(t) \in \Omega_{\rho_s}, u \in U\} \tag{63}$$

$$\rho_{min} \geq \rho_{nn} + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}} f_w(\Delta) + \kappa(f_w(\Delta))^2 \tag{64}$$

such that for any $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, with probability at least $1-\delta$, the following inequality holds:

$$\hat{V}(x(t)) \leq \hat{V}(x(t_k)),\ \forall t \in [t_k, t_{k+1}) \tag{65}$$

and the state $x(t)$ of the nonlinear system of Equation (1) is bounded in $\Omega_{\hat{\rho}}$ for all times and ultimately bounded in $\Omega_{\rho_{min}}$.

Proof. The key steps for the proof of Proposition 3 are presented below, and the full proof is omitted here as it is similar to the proof of Proposition 4 in [9]. The only difference is that Equation (65) now holds in probability due to the probabilistic nature of the modeling error bound.

To show that the state will move towards Ω_{ρ_s}, which is a sufficiently small level set of $\hat{V}$ around the origin, we show that the time derivative of $\hat{V}$ can be rendered negative for any $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ under $u = \Phi_{nn}(x) \in U$.

$$\begin{aligned}
\dot{\hat{V}}(x(t)) =& \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k))) \\
=& \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k))) + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k))) \\
&- \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)))
\end{aligned} \tag{66}$$

As shown in Proposition 1, by choosing the number of samples $m \geq m_N(\delta, h_c, |x|)$ such that $E_M \leq \gamma |x|$, where $\gamma < \hat{c}_3/\hat{c}_4$, it holds that $\mathbb{P}[\dot{\hat{V}} \leq 0] \geq 1-\delta$ under $u = \Phi_{nn}(x) \in U$.

Then, using the Lipschitz condition in Equations (5)–(7) and the condition for Lyapounov function in Equations (50)–(52), Equation (66) can be further bounded as follows:

$$\begin{aligned}\dot{\hat{V}}(x(t)) &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + \frac{\partial \hat{V}(x(t))}{\partial x}F(x(t),\Phi_{nn}(x(t_k))) - \frac{\partial \hat{V}(x(t_k))}{\partial x}F(x(t_k),\Phi_{nn}(x(t_k))) \\ &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'}|x(t)-x(t_k)| \\ &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'}M_F\Delta\end{aligned} \tag{67}$$

Therefore, if Equation (62) is satisfied, we can find a negative real number $-\epsilon_w$ that bounds the time derivative of $\hat{V}$. This implies that for any state $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, with probability at least $1-\delta$, the Lyapunov function value will decrease in one sampling time, and therefore, the state can ultimately reach the set Ω_{ρ_s} under $u = \Phi_{nn}(x) \in U$ with a certain probability. Additionally, since $\dot{\hat{V}}$ may not be rendered negative within Ω_{ρ_s} under sample-and-hold implementation of Lyapunov-based control law $u = \Phi_{nn}(x) \in U$, the predicted state of the RNN model of Equation (49) is only required to be bounded in $\Omega_{\rho_{nn}}$, which is a slightly larger level set that includes Ω_{ρ_s} (see definition of $\Omega_{\rho_{nn}}$ in Equation 63). In this case, we can show that the state of the actual nonlinear system of Equation (49) is bounded in $\Omega_{\rho_{min}}$, which is a superset of $\Omega_{\rho_{nn}}$ that accounts for the modeling error within one sampling period (see definition of $\Omega_{\rho_{min}}$ in Equation (64)). As a result, we do not impose any constraints on $\dot{\hat{V}}$ for $x \in \Omega_{\rho_s}$. This explains why the modeling error constraint $E_M \leq \gamma|x|$ is not necessary for $x \in \Omega_{\rho_s}$ as stated in Remark 6. □

4.3. Lyapunov-Based MPC Using RNN Models for Nonlinear Systems

The Lyapunov-based model predictive control design is given by the following optimization problem [9,10]:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L_{MPC}(\tilde{x}(t),u(t))dt \tag{68}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t),u(t)) \tag{69}$$

$$u(t) \in U,\ \forall\, t \in [t_k, t_{k+N}) \tag{70}$$

$$\tilde{x}(t_k) = x(t_k) \tag{71}$$

$$\dot{\hat{V}}(x(t_k),u) \leq \dot{\hat{V}}(x(t_k),\Phi_{nn}(x(t_k))),\ \text{if}\ x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{72}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn},\ \forall\, t \in [t_k, t_{k+N}),\ \text{if}\ x(t_k) \in \Omega_{\rho_{nn}} \tag{73}$$

where $\tilde{x}$, N and $S(\Delta)$ are the predicted states, the prediction horizon length, and the set of piecewise constant functions with period Δ, respectively. We use $\dot{\hat{V}}(x,u)$ to represent the time derivative of Lyapunov function $\hat{V}$, i.e., $\dot{\hat{V}}(x,u) = \frac{\partial \hat{V}(x)}{\partial x}(F_{nn}(x,u))$. After solving the optimization problem of Equations (68)–(73) at $t = t_k$, we apply the first control action $u(t)$, $t \in [t_k,\ t_{k+1})$ from the optimal input trajectory $u^*(t)$, $t \in [t_k,\ t_{k+N})$ to the system of Equation (1). Then the horizon is rolled one sampling period forward, and the LMPC is resolved at the next sampling time with new state measurements available at $t = t_{k+1}$.

The optimization problem of Equations (68)–(73) minimizes the objective function of Equation (68), which is the integral of $L_{MPC}(\tilde{x}(t),u(t))$ over the prediction horizon, subject to the constraints of Equations (69)–(73). The RNN model of Equation (49) is used to predict state evolution over $t \in [t_k, t_{k+N})$ given the state measurements at $t = t_k$ in Equation (71). In the constraint of Equation (69), the RNN model of Equation (49) is used to predict the states of the closed-loop system. The constraint of Equation (70) ensures that the input are bounded over the entire prediction horizon. Finally, the constraints of Equations (72)–(73) drives the predicted state towards the origin and ultimately maintain it inside $\Omega_{\rho_{nn}}$. It should be noted that despite the probabilistic nature of the RNN generalization error bound, the neural network prediction of Equation (69) is deterministic after training is completed. In other words, given the same initial state $x(t_k)$, and the manipulated inputs $u(t)$, $\forall t \in [t_k,\, t_{k+N})$, the RNN model of Equation (69) produces deterministic results that statistically approximate the evolution of states over $t \in [t_k,\, t_{k+N})$. This is different from stochastic MPC which uses a stochastic process model in the MPC formulation, and therefore, requires calculation of uncertainty prorogation and accounts for probabilistic constraint satisfaction. The LMPC formulation of Equations (68)–(73) is solved with a deterministic RNN model, based on which recursive feasibility is guaranteed, and probabilistic stability results can be developed.

The following theorem is established to demonstrate that LMPC ensures closed-loop stability for the nonlinear system of Equation (1) with high probability provided that the RNN model is well constructed that satisfies the modeling error constraint in Proposition 1.

Theorem 2. *Consider the closed-loop system of Equation (1) under the LMPC of Equations (68)–(73) based on the controller $\Phi_{nn}(x)$ that satisfies Equations (50)–(52). Let $\Delta > 0$, $\epsilon_w > 0$ and $\hat{\rho} > \rho_{min} > \rho_{nn} > \rho_s$ satisfy Equations (62)–(64). Then, given any initial state $x_0 \in \Omega_{\hat{\rho}}$, if the RNN model is developed satisfying the conditions in Proposition 2 and Proposition 3, there always exists a feasible solution for the optimization problem of Equations (68)–(73) . Additionally, by choosing the number of samples $m \geq m_N(\delta, h_c, |x|)$ such that $E_M \leq \gamma|x|$ holds, then for each time step, with probability at least $1-\delta$, closed-loop stability is guaranteed for the system of Equation (1) under the LMPC of Equations (68)–(73) in the sense that $x(t) \in \Omega_{\hat{\rho}}$, $\forall t \geq 0$, and $x(t)$ ultimately converges to $\Omega_{\rho_{min}}$.*

Proof. The proof consists of two parts. In the first part, we prove recursive feasibility of the LMPC optimization problem of Equations (68)–(73) . The proof of this part follows closely the proof of Theorem 2 in [9], which shows that the stabilizing controller $u(t) = \Phi_{nn}(x(t)) \in U$, $t = [t_k, t_{k+N})$ is a feasible solution to the LMPC optimization problem. Specifically, when $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$ at $t = t_k$, it is readily shown that the control action $u(t) = \Phi_{nn}(x(t_k))$ is a feasible solution that satisfies the constraint of Equation (72) by taking the equal sign. When $x(t_k) \in \Omega_{\rho_{nn}}$, as shown in [9], $u(t) = \Phi_{nn}(x(t)) \in U$, $t = [t_k, t_{k+N})$ again are feasible solutions that maintain predicted states within $\Omega_{\rho_{nn}}$ within the prediction horizon.

In the second part, we prove that closed-loop stability is guaranteed in probability for the nonlinear system of Equation (1) under LMPC. Specifically, when $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$ at $t = t_k$, we have shown in Proposition 3 that for each sampling time, $\hat{V}(x(t)) \leq \hat{V}(x(t_k))$ holds under $u(t) = \Phi_{nn}(x(t)) \in U$ for $t \in [t_k, t_{k+1})$ with probability at least $1-\delta$. This implies that the state of the actual nonlinear system of Equation (1) can be driven towards the origin under the LMPC using RNN models for prediction provided that the modeling error is sufficiently small and satisfies $E_M \leq \gamma|x|$, $\forall x \in \Omega_{\hat{\rho}}$. When $x(t_k) \in \Omega_{\rho_{nn}}$, the input sequences are optimized to minimize the objective function of Equation (68) while meeting the constraint of Equation (73). However, due to the existence of modeling error, the true states may leave $\Omega_{\rho_{nn}}$ while the predicted states remain inside $\Omega_{\rho_{nn}}$. In Proposition 3, we have shown that with probability at least $1-\delta$, the true state of the system of Equation (1) can be bounded within $\Omega_{\rho_{min}}$, which is a superset of $\Omega_{\rho_{nn}}$ designed accounting for the modeling error within one sampling period. Additionally, it is noted that depending on the prediction horizon of RNN models, we may need to perform RNN predictions successively to obtain the full prediction of the state trajectory over the entire prediction horizon, $t \in [t_k, t_{k+N})$. For example, in this work, the RNN model of Equation (49) is developed to predict one sampling period forward, and thus, in order to predict state trajectory over $t \in [t_k, t_{k+N})$, we need to carry out RNN predictions N times. After the initial prediction at $t = t_k$, each prediction uses the previous predicted state as the initial state, along with the manipulated input u to predict the state at the next sampling time. This inevitably accumulates the modeling error over calculation, which may lead to a probability lower than $1-\delta$ for the final state prediction error to be bounded by $E_M \leq \gamma|x|$.

As a result, the true states may further deviate from predicted states, and ultimately leave $\Omega_{\rho_{min}}$ within finite time. Despite the degradation of prediction performance over time, closed-loop stability is not affected since LMPC is implemented in a rolling horizon manner with feedback state measurements available every sampling time. The input sequences are re-optimized using new state measurements at every sampling time to meet desired closed-loop performance. Additionally, since the modeling error condition $E_M \leq \gamma|x|$ holds for the first sampling period, the state of the actual nonlinear system of Equation (1) is guaranteed to not leave $\Omega_{\rho_{min}}$ within one sampling period with probability at least $1-\delta$ as shown in Proposition 3. At the next sampling period, the constraints of Equation (72) and of Equation (73) will be activated depending on the measurement of $x(t_{k+1})$. Regardless of where $x(t_{k+1})$ is, the LMPC of Equations (68)–(73) will drive the predicted state into $\Omega_{\rho_{nn}}$, and correspondingly, maintain the true state within $\Omega_{\rho_{min}}$ in probability. Therefore, for any state $x(t_k) \in \Omega_{\hat{\rho}}$, with probability at least $1-\delta$, the closed-loop state of the system of Equation (1) is bounded in $\Omega_{\hat{\rho}}$ for each sampling time, and is ultimately bounded within $\Omega_{\rho_{min}}$. This completes the proof of Theorem 2.
□

Remark 7. *It is noted that in Theorem 2, the probability of closed-loop stability (i.e., at least $1-\delta$) is derived for each sampling time since the probability of the modeling error bounded by $\gamma|x|$ is at least $1-\delta$ for one sampling period only. It is difficult to compute the overall probability of closed-loop stability for the entire state trajectory because given an initial state $x_0 \in \Omega_{\hat{\rho}}$, we do not know how many times steps it will take to drive the state into $\Omega_{\rho_{min}}$ beforehand. Additionally, the actual probability of closed-loop stability for each time step could be higher than the lower bound $1-\delta$ due to many reasons. For example, 1) the RNN model is well trained that yields a modeling error far below its upper bound, and 2) closed-loop stability may be unaffected if the next state does not leave $\Omega_{\hat{\rho}}$ even if the modeling error exceeds its upper bound during one sampling period. Therefore, the probability $1-\delta$ is conservative in many cases, and only provides a lower bound for the probability of closed-loop stability.*

5. Application to a Chemical Process Example

We use the same chemical process example as in [10] to illustrate the application of LMPC using RNN models. However, in this work, we will primarily demonstrate the use of generalization error bound framework to provide estimates of their accuracy in the development of RNN models for nonlinear dynamic processes. Specifically, we carry out five case studies to evaluate the relation between RNN generalization error and a number of factors such as data sample size, RNN depth/width, and data time length that impact its performance. Additionally, after the RNN model is incorporated in the LMPC formulation, we will demonstrate the closed-loop performances under the RNN models developed with different data sample size and structures, and evaluate their probabilistic closed-loop stability properties. We consider a well-mixed, non-isothermal continuous stirred tank reactor (CSTR) with an irreversible second-order exothermic reaction in this example. The reaction transforms a reactant A to a product B ($A \rightarrow B$), where C_{A0}, T_0 and F denote the inlet concentration of A, the inlet temperature and feed volumetric flow rate of the reactor, respectively. A heating jacket is used to supply/remove heat to/from the CSTR at a rate Q. The CSTR dynamic model is represented by the following material and energy balance equations:

$$\begin{aligned} \frac{dC_A}{dt} &= \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT}} C_A^2 \\ \frac{dT}{dt} &= \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} \end{aligned} \tag{74}$$

where C_A and T are the concentration of reactant A and temperature in the reactor, respectively. Q denotes the heat input rate, and V is the volume of the reacting liquid in the reactor. F, T_0, and C_{A0} are the volumetric flow rate, the feed temperature and the feed concentration of reactant A, respectively. We assume that the reacting liquid has a constant density of ρ_L and a heat capacity of C_p. ΔH, k_0, E, and R represent the enthalpy of reaction, pre-exponential constant, activation energy, and ideal gas constant, respectively. The list of process parameter values can be found in [10].

The objective of LMPC is to stabilize the CSTR at its unstable equilibrium point $(C_{As}, T_s) = (1.95\, kmol/m^3, 402\, K)$ corresponding to $(C_{A0_s}\, Q_s) = (4\, kmol/m^3, 0\, kJ/hr)$ by manipulating the inlet concentration of species A and the heat input rate. All the process states (C_A, T) and manipulated inputs (C_{A0}, Q) are represented in the deviation variables form, i.e., $\Delta C_{A0} = C_{A0} - C_{A0_s}$, $\Delta Q = Q - Q_s$, $\Delta C_A = C_A - C_{As}$, and $\Delta T = T - T_s$. To simplify the notation, we use $x^T = [\Delta C_A\ \Delta T]$ and $u^T = [\Delta C_{A0}\ \Delta Q]$ to represent CSTR states and inputs, respectively. By using deviation variables, the equilibrium point of the CSTR of Equation (74) is at the origin of the state-space. The following positive definite P matrix is used to characterize the closed-loop stability region $\Omega_{\hat{\rho}}$ (i.e., a level set of Lyapunov function $V(x) = x^T P x$) with $\hat{\rho} = 368$:

$$P = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{75}$$

Additionally, the manipulated inputs are required to be bounded as follows: $|\Delta C_{A0}| \leq 3.5$ kmol/m^3 and $|\Delta Q| \leq 5 \times 10^5$ kJ/hr to meet physical constraints. The integration of RNN models in MPC follows the method in [10,39]. Specifically, the RNN models are developed offline using Keras (version 2.4) [40], and then used to predict future states based on the state measurement at each sampling time in the real-time implementation of MPC. Then, the nonlinear optimization problem of the LMPC of Equations (68)–(73) is solved under the sampling period $\Delta = 10^{-2}\, hr$ using PyIpopt, which is the python module of the IPOPT software package (version 3.9.1) [41]. The dynamic model of Equation (74) is integrated using numerical method, i.e., explicit Euler method, with a sufficiently small integration time step of $h_c = 10^{-4}\, hr$.

5.1. RNN Generalization Performance

In this section, we carry out a number of RNN trainings with different RNN structures and data samples to show the relation between RNN generalization performance and a number of factors such as RNN input length, width, depth, weight bounds and data sample size.

5.1.1. Case Study 1: Data Sample Size

In the first case study, we trained RNN models using different data sample sizes. Specifically, we follow data generation method in [10] to initially generate a large dataset from open-loop simulation of Equation (74) under various control actions $u \in U$ and initial conditions within the stability region, i.e., $x_0 \in \Omega_{\hat{\rho}}$. The dataset consists of 200,000 time-series data samples, and is separated into 140,000 training, 30,000 validation, and 30,000 testing samples. The RNN models are developed by gradually increasing the training sample size, and is tested using unseen data from the testing dataset. It should be noted that only the data sample size is changed in this case study, while all the other parameters such as the RNN structure (i.e., number of layers, neurons, and other hyper-parameters) and training algorithm remain the same for all RNN models. The RNN models are developed with one hidden layer of 50 neurons, and using mean squared errors (MSE) as loss function.

Figure 2 shows the variation of RNN training and testing performances with respect to the training sample size. In the top figure of Figure 2, it is observed that both the testing and training MSEs increase as training data becomes less; in the bottom figure, we show the generalization gap $\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] - \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i)$ in Equation (47), where the expected error $\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})]$ is approximated using the testing dataset. The trend in Figure 2 is consistent with the result in Theorem 1, which demonstrates that more training data is needed in order to obtain a lower generalization gap between expected loss and training loss. Additionally, it is noticed when the training sample size is greater than 3000, both training and testing MSEs approach zero, and no significant improvement is observed for the models using more training data. The trend in Figure 2 also follows the relation between generalization error and data sample size in Equation (47), i.e., the generalization gap $\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] - \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i)$ is roughly proportional to $\frac{1}{\sqrt{m}}$, which shows that the generalization gap initially decreases fast when the sample size m starts increasing from zero, and changes slowly when m becomes large.

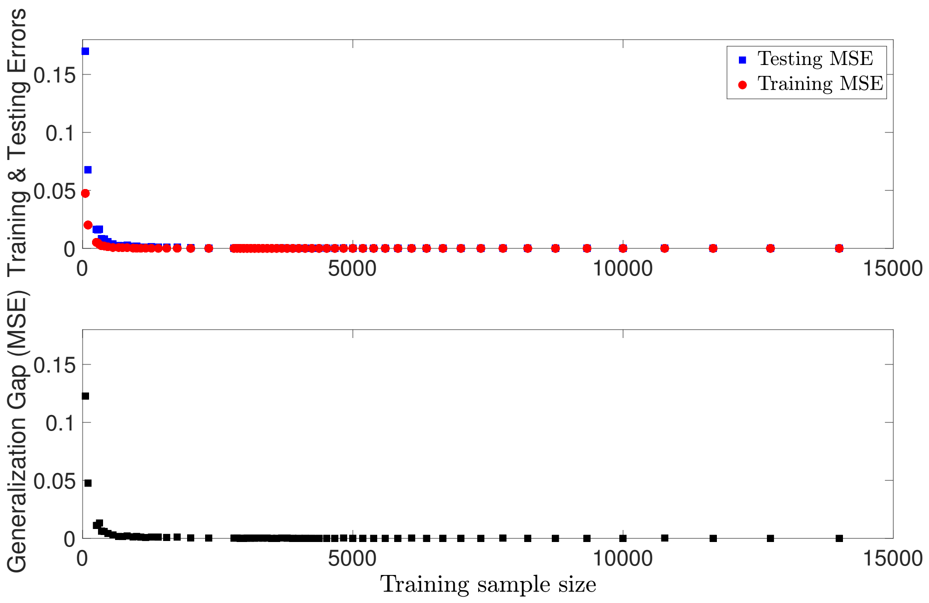

Figure 2. RNN generalization performance vs. training sample size.

5.1.2. Case Study 2 : RNN Depth and Width

In the second case study, we train RNN models with various depths and widths. 1400 training data, 300 validation data, and 300 testing data are used for all models. We first develop RNN models by fixing the network depth as one hidden layer, and increasing the number of neurons. As shown in Figure 3, both training and testing errors decrease as the network width increases up to 250 neurons. However, as more neurons are added (i.e., 270 and 280 neurons in Figure 3), the testing MSE increases while the training MSE remains close to zero all the time, which implies that overfitting has occurred during training. As a result, the generalization gap in Figure 3 shows a similar pattern, which decreases initially and increases again when a large number of neurons are used. While theoretically the expected error of Equation (47) does not explicitly depend on the network width, the results in Figure 3 are consistent with the fact that increasing the capacity of a model by adding more layers and/or more nodes to layers can improve the network learnability, but may also lead to overfitting.

Subsequently, we train RNN models by increasing the number of layers, and fixing five neurons each layer. Figure 4 shows that the testing MSE starts at around 0.02 for one hidden layer, gradually decreases with more layers, and finally increases again as the neural network becomes deeper. Meanwhile, the training MSE remains close to zero at the beginning, yet also slightly increases as the number of hidden layers increase. From Figure 4, it is concluded that one hidden layer is not sufficient to learn the process dynamics well, and with two, three, and four layers, the RNN models achieve the best training and generalization performance among all the models. Similar to Figure 3, the increase of generalization gap in Figure 4 implies that deeper RNN models are overfitting the training data. Additionally, it is also interesting to notice that the training error slightly increases in deeper networks. While in general, the generalization performance deteriorates and the training error remains unaffected when increasing the capacity of a model, the worse training performance in Figure 4 are actually common in neural network development due to the difficulty of training deep networks. Specifically, the optimization problem of neural network training is highly non-convex, and may get stuck at some local minima as the network becomes deeper. This is noticed during the training of RNN models in Figure 4, where both the training and validation losses exhibit a sharp increase at a certain epoch and then get stuck around that point until the end of epochs. Additionally, with more hidden layers, the number of parameters to be trained grows exponentially, which could lead to a poor training performance without a careful tuning of other hyperparameters.

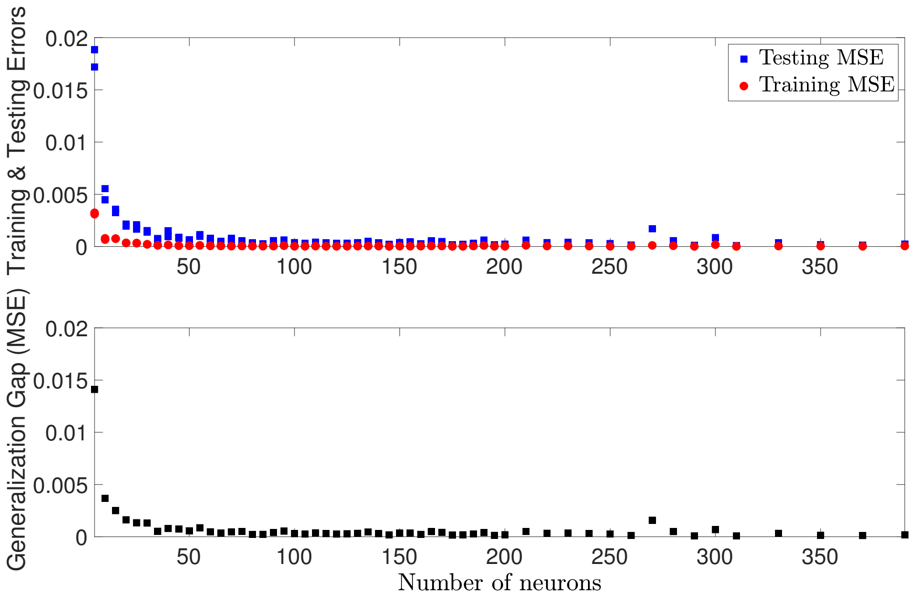

Figure 3. RNN generalization performance vs. RNN width (One hidden layer with increasing number of neurons).

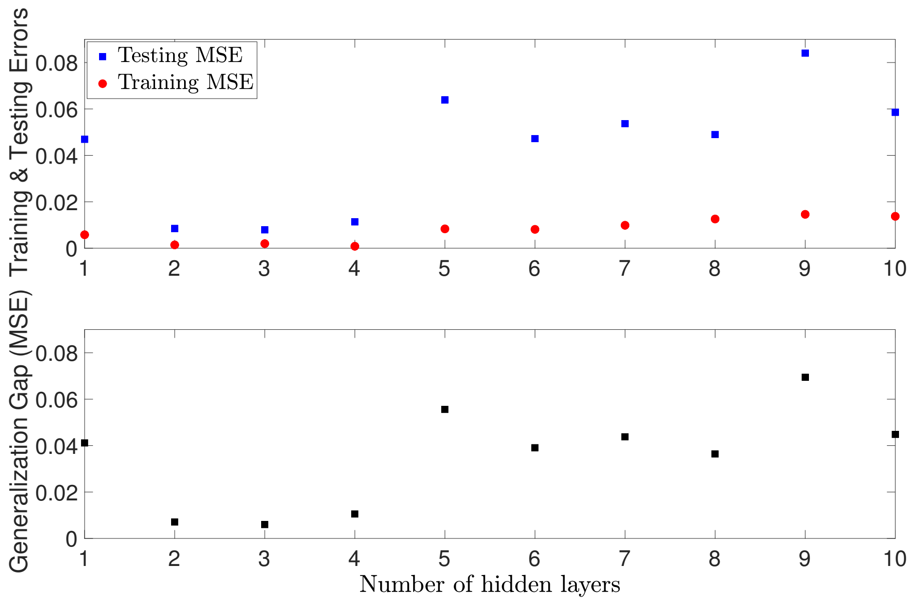

Figure 4. RNN generalization performance vs. RNN depth (Increasing the number of hidden layers and fixing 5 neurons for each layer).

Remark 8. *At first glance, the generalization error trend in Figures 3 and 4 seems in contrast to the results in Equation (47), which shows the generalization error bound is proportional to the complexity of RNN hypothesis class. However, it should be noted that Equation (47) only gives the upper bound for the generalization error of RNN models from the hypothesis class. It does not mean all the RNN models from the hypothesis class have a generalization error as large as* *its upper bound. From the error decomposition of Equation (17) showing the interplay between approximation and estimation errors, we have learned that as we enlarge the hypothesis class, the approximation error decreases, but the estimation error may increase. In this case study, by increasing the complexity of RNN hypothesis class in terms of more layers and neurons, overall the generalization performance improves; however, as the RNN models become deeper, overfitting also occurs due to a large estimation error. Therefore, in practice, we can do a grid search such as Figures 3 and 4 to determine the optimal number of layers and neurons.*

5.1.3. Case Study 3: Different Regions in $\Omega_{\hat{\rho}}$

As discussed in Remark 6, to meet the modeling error constraint $E_M \leq \gamma|x|, \forall x \in \Omega_{\hat{\rho}}$, more data is needed as the state approaches the origin, i.e., $x \to 0$. It is equivalent to show that under the same data density for different regions within the stability region $\Omega_{\hat{\rho}}$, a larger constant γ is needed to bound the modeling error $E_M \leq \gamma|x|$ for the states close to the origin. Therefore, in this case study, we develop multiple RNN models for different regions inside $\Omega_{\hat{\rho}}$ with the same data density, and demonstrate the variation of generalization performances. Specifically, we choose 9 level sets of Lyapunov function $\Omega_{\rho_i} := \{x \in \mathbf{R}^n \mid \hat{V}(x) \leq \rho_i\}$, $i = 0, \ldots, 8$, within $\Omega_{\hat{\rho}}$, with $\hat{\rho} = 368$ and $\rho_i = [40, 88, 115, 138, 159, 177, 195, 213, 244]$. For example, the first RNN model (model 0 with ρ_0) is developed and tested using the data within Ω_{ρ_0}, the second RNN model (model 1 with ρ_1) uses the data between Ω_{ρ_0} and Ω_{ρ_1}, and so on. Figure 5 shows a schematic of the training regions considered for the CSTR of Equation (74), where x_s is the steady-state, and $\Omega_{\hat{\rho}}$ is the stability region. The training datasets are generated for each region (i.e., elliptical annuli in Figure 5) with the same data density, where the data density is defined as the ratio of sample size to the area of each elliptical annulus. Similarly, in this case study, we use data from different regions within $\Omega_{\hat{\rho}}$ to build RNN models, while all the other parameters remain the same. The RNN models are developed with one hidden layer of 20 neurons, and using MSE as loss function.

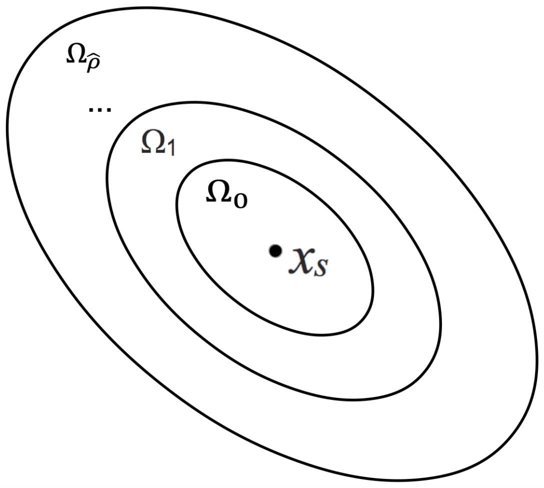

Figure 5. Schematic of different regions inside $\Omega_{\hat{\rho}}$.

To compute the modeling error $|F(x,u) - F_{nn}(x,u)| = |\frac{dx}{dt} - \frac{d\hat{x}}{dt}|$ where x and $\hat{x}$ denote the true state and predicted state, respectively, we carry out prediction for one integration time step, and use finite difference method to approximate the derivatives following Equation (56). Specifically, we first calculate the training and testing mean absolute errors (MAE) and divide them by the integration time step h_c, i.e., $|\frac{x(t+h_c)-\hat{x}(t+h_c)}{h_c}|$. Subsequently, to obtain an approximated value of γ for each model, i.e., $\frac{E_M}{|x|} \leq \gamma$, we divide those MAEs by the maximum value of $|x|$ in each elliptical annulus in Figure 5. Figure 6 shows the training and testing errors for the RNN models trained for different regions inside $\Omega_{\hat{\rho}}$.

It is observed that under the same data density, the models trained for the regions close to the origin (i.e., Models 0, 1, and 2 for Ω_{ρ_0}, Ω_{ρ_1} and Ω_{ρ_2}) produce larger generalization gaps. This implies a larger γ, or equivalently, more data is needed to meet the constraint $E_M \leq \gamma|x|$ for x in these regions. Additionally, it is observed that the generalization gap settles at around 2×10^{-5} for model 4 and after because those RNN models have achieved the best they can do under the current neural network training settings and data density.

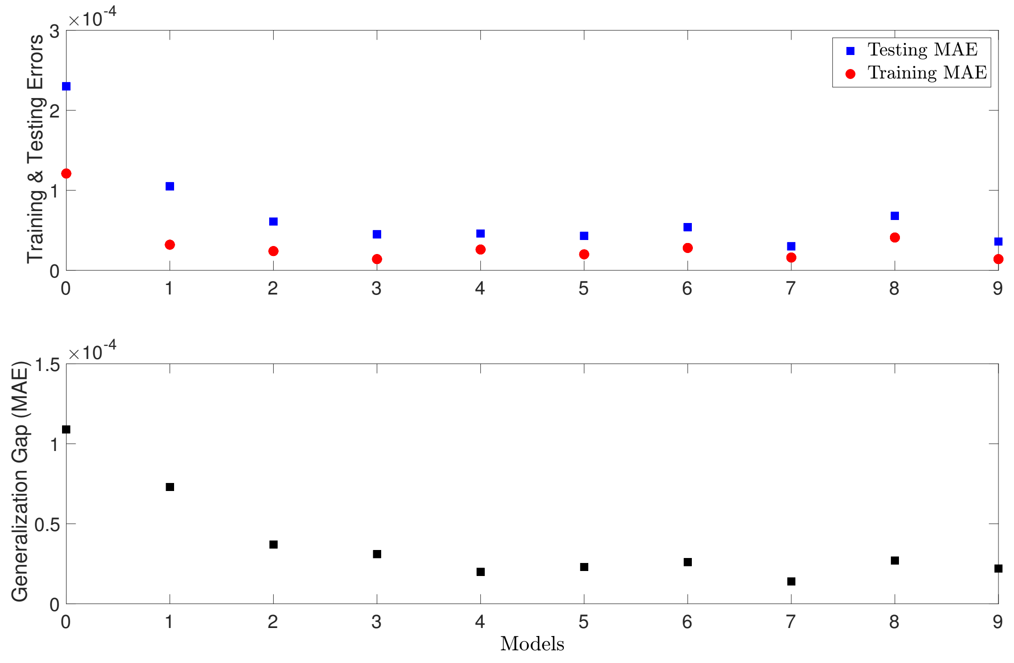

Figure 6. RNN generalization performance vs. different regions in $\Omega_{\hat{\rho}}$.

5.1.4. Case Study 4: Weight Matrix Bound

From Equation (48), it is seen that the generalization gap also depends on the weight matrix bound. To evaluate the relation between generalization performance and weight matrix bound, in this case study, we train RNN models with different weight matrix bounds. Specifically, we impose an upper bound constraint for each element in the RNN weight matrices with the following values $[0.8, 1.3, 1.8, 2.5, 3.0, 3.4, 3.9, 4.3]$.

The Frobenius norms of all the weight matrices are therefore also bounded. The training and testing errors are calculated following the approach in Case study 1, and are shown in Figure (7). It is observed that as the weight matrix bound becomes larger, the generalization gap gradually increases and settles at around 8×10^{-4}. This behavior implies that the RNN model is over-fitting when training with a large weight bound. The reason for the trend in Figure 7 is similar to that for Case study 2, which demonstrates that as the size of neural network hypothesis class becomes larger with increasing weight bounds, it is easier to find a hypothesis that fits training data well, but could also lead to large testing error (i.e., over-fitting).

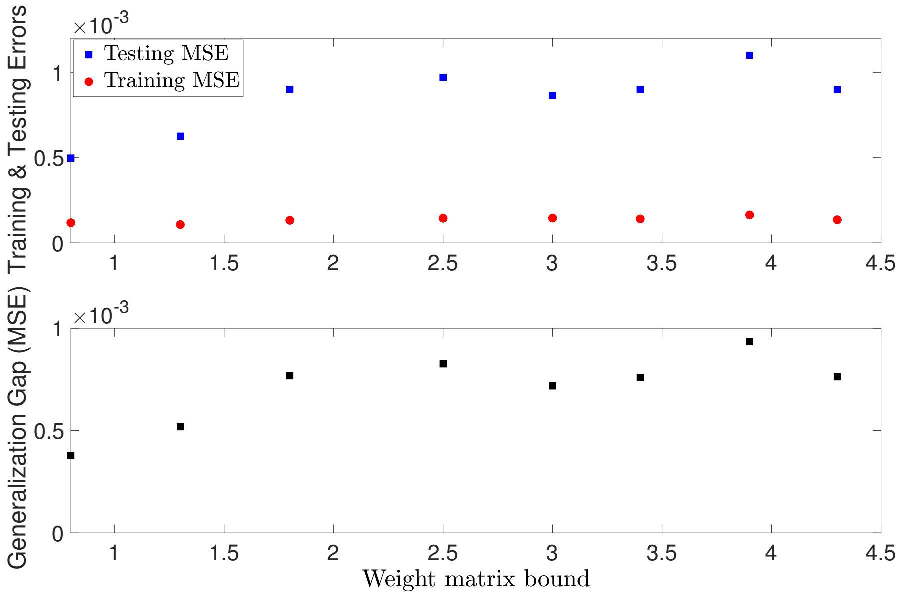

Figure 7. RNN generalization performance vs. weight matrix bound.

5.1.5. Case Study 5: RNN Input Length

Lastly, we study the dependency of RNN generalization error on the input time length t according to Equation (47). If we unfold a vanilla RNN over time to form a multi-layer feedfoward neural network, then this relation can also be interpreted in the way that a deep feedforward neural network has a large generalization error. In this example, we train RNN models with different input time length as follows: $t = 10^{-3} \times [1,2,3,4,5,6,7,8,9,10]$ hr.

Figure 8 shows the training and testing errors for different time lengths. Specifically, as RNN input time length increases, it is seen that the training error remains at a very low level for all models, but the testing error gradually increases and finally settles at around 6×10^{-3}. It is concluded from Figure 8 that a shorter input sequence yields better generalization performance, which is consistent with the theoretical result shown in Equation (47). However, it should be noted that a shorter input sequence does not necessarily yield better prediction in the formulation of MPC because as discussed in Theorem 2, in order to predict future states for a long prediction horizon, the RNN prediction needs to be executed successively, which inevitably accumulates the error during calculations. Therefore, when used in MPC, the RNN input length should be carefully chosen to account for MPC prediction horizon and maintain a desired generalization performance simultaneously.

Remark 9. *A small training dataset was chosen in Case studies 2–5 for demonstration purposes. Specifically, it was demonstrated in Case study 1 that with more than 3000 data samples, both training and testing errors are rendered sufficiently small. Therefore, to better demonstrate the relation between RNN generalization error bound and RNN depth/width, and data time length in other case studies, we chose a small training dataset such that significant differences can be observed by varying RNN depths, widths, time sequence length. However, it is noted that in practice, the sample size and all the other factors studied in this manuscript should be carefully chosen in order to improve the RNN generalization performance.*

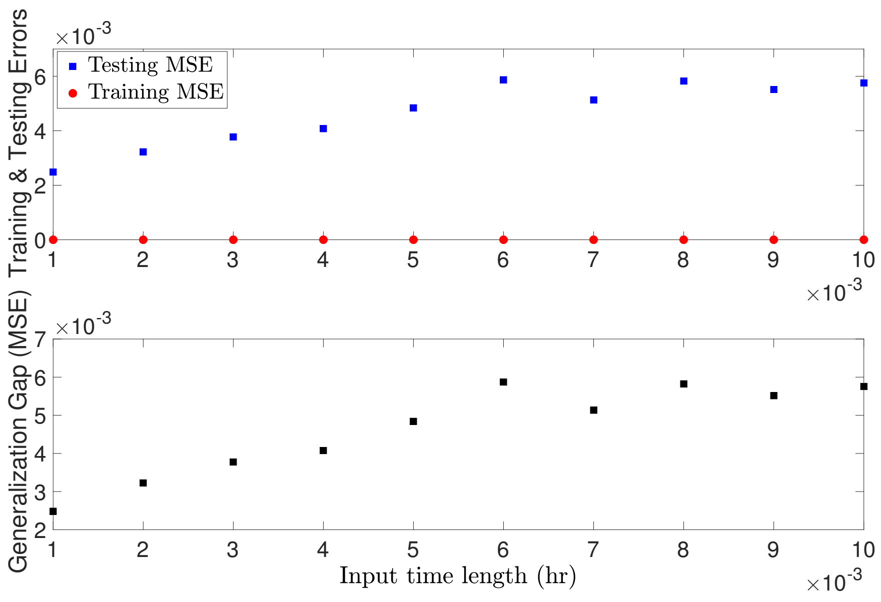

Figure 8. RNN generalization performance vs. input time length.

5.2. *Closed-Loop Performance Analysis*

In this section, we carry out closed-loop simulations of CSTR under the LMPC of Equations (68)–(73) using the different RNN models derived from the previous case studies. Additionally, we demonstrate the probabilistic closed-loop stability properties of RNN-based LMPC through extensive closed-loop simulations for the CSTR of Equation (74) with different initial conditions.

Figures 9–12 show the simulation results using 48 different initial conditions within $\Omega_{\hat{\rho}}$ for a few RNN models trained in Case study 1. Specifically, we first discretize the stability region $\Omega_{\hat{\rho}}$ and choose 48 initial conditions $x_0 \in \Omega_{\hat{\rho}}$ that are evenly spread within the stability region. Then, we run closed-loop simulations for all initial conditions using the following settings: (1) the whole simulation period t_p is twenty sampling periods (i.e., $20 \times 0.01 = 0.2$ hr), (2) the stability region $\Omega_{\hat{\rho}}$ and the terminal region $\Omega_{\rho_{min}}$ are characterized as $\hat{\rho} = 368$ and $\rho_{min} = 2$, respectively, and (3) the simulations are carried out using UCLA Hoffman 2 cluster and the optimization problem is solved using the python module of the IPOPT software package (i.e., PyIpopt). After obtaining the closed-loop profiles for each initial condition, the following policies are utilized to determine whether the closed-loop system is stable or not. Specifically, the closed-loop system is considered unstable if (1) the closed-loop state leaves the stability region $\Omega_{\hat{\rho}}$ at any point during the simulation, or (2) the closed-loop state remains inside $\Omega_{\hat{\rho}}$, but stays outside of $\Omega_{\rho_{min}}$ until the end of simulation or leaves $\Omega_{\rho_{min}}$ after entering for the first time.

Figure 9 shows the probability of closed-loop stability calculated following the above policies. It is seen that with more training data, the probability of the CSTR of Equation (74) being stabilized at its steady-state becomes higher, and the probability settles at around 0.78 for a sufficiently large dataset. The probability results in Figure 9 for RNN models in Case study 1 are consistent with its generalization performance plot in Figure 2, which shows that the generalization error decreases with more data used for training. In addition to the calculation of the probability for closed-loop stability, we also use the MPC cost function of Equation (68) as an indicator for comparing control performance in terms of the convergence speed and energy consumption. Specifically, the MPC cost function of Equation (68) in this example is designed in the following form:

$$L_{MPC}(x,u) = x^T P x + u^T Q u \tag{76}$$

where $P = [1000\ \ 0;\ 0\ \ 1]$ and $Q = [1\ \ 0;\ 0\ \ 3 \times 10^{-10}]$ are chosen such that the two states and the two inputs are in the same order of magnitude, respectively. Also, in this example, we put more penalty on the states x to allow the states to be driven to the steady-state more quickly. For each RNN model, we calculate the total costs $\int_{t=0}^{t_p} L_{MPC}(x,u)dt$ over the entire simulation period $t_p = 0.2$ hr, and sum up the cost values for all the trajectories initiated from 48 different initial conditions. Figure 10 shows the MPC total costs for the RNN models trained with different data sample sizes. It is demonstrated that with less training data, the MPC achieves a higher total cost, representing a slower convergence to the steady-state and/or a higher energy consumption. With a large number of training data (i.e., $\geq$ 6000), the MPC total costs remain at around 1420, and no significant improvement is noticed with more data added in training. Additionally, Figures 11 and 12 show the closed-loop state trajectory and state profiles for one of the initial condition out of 48 initial conditions. As shown in Figure 11, the state trajectory using the RNN model trained with 50 training data (dashed line) leaves the stability region due to poor predictions in solving the MPC optimization problem. On the contrary, the state trajectory using the RNN model with 14,000 training data (solid line) moves towards the steady-state smoothly and is ultimately bounded in the terminal set $\Omega_{\rho_{min}}$. This can also be seen in the closed-loop state profiles of Figure 12, where the temperature under 50 training data shows a sharp increase at 0.03 hr.

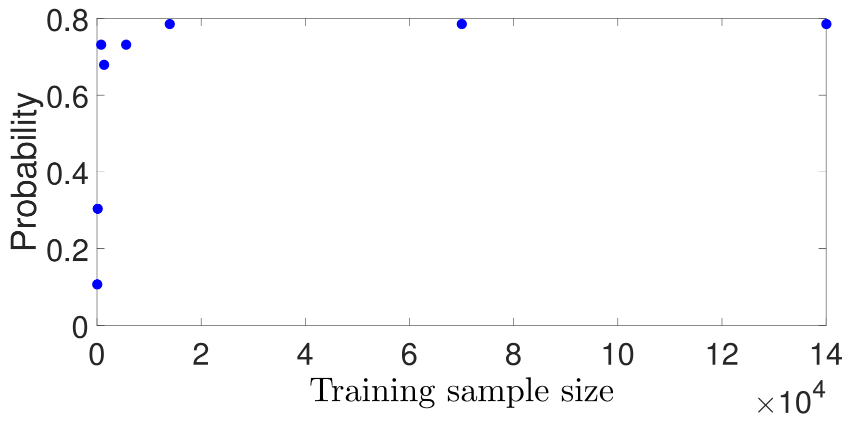

Figure 9. Probability of closed-loop stability vs. training sample sizes.

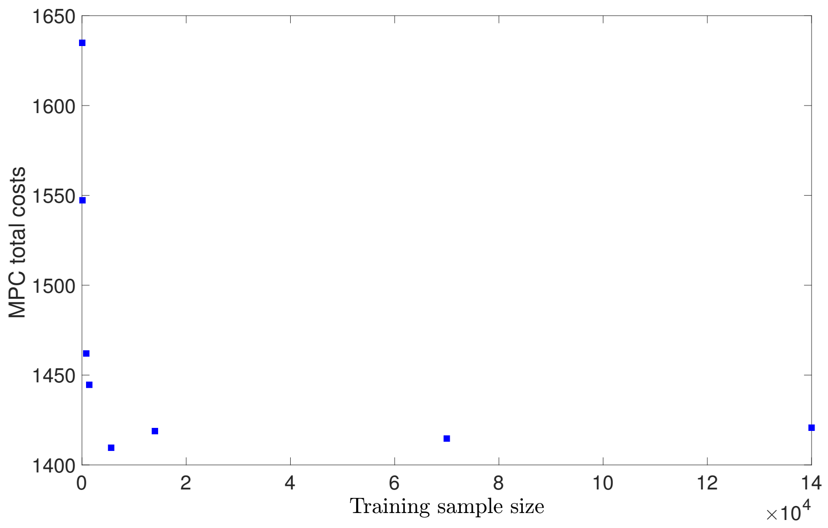

Figure 10. MPC total costs vs. training sample size.

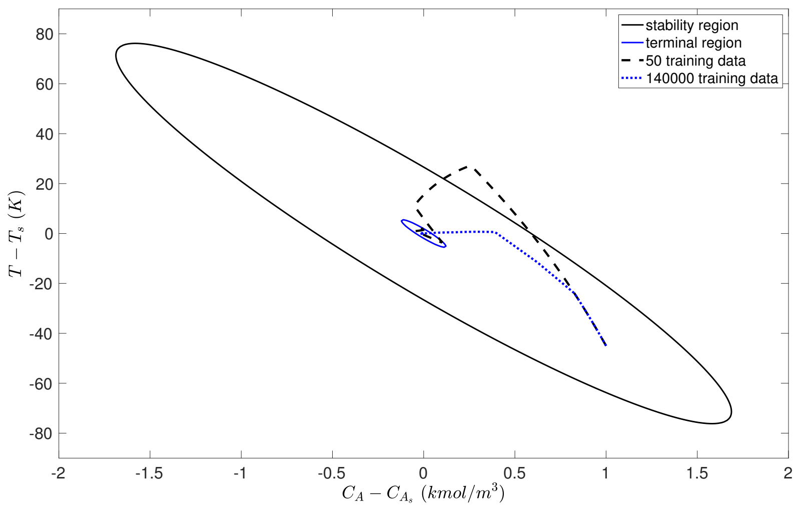

Figure 11. Closed-loop state trajectory under LMPC using two RNN models trained with different data sample sizes.

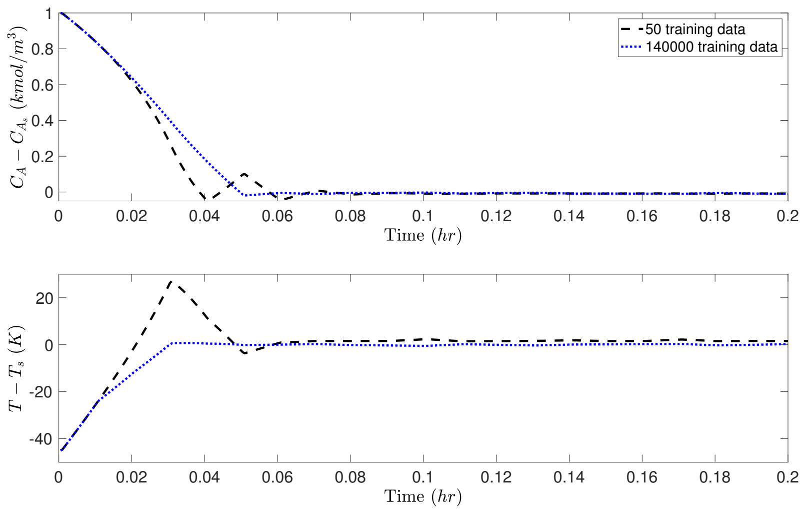

Figure 12. Closed-loop state profiles under LMPC using two RNN models trained with different data sample sizes.

Similar to the analysis for Case study 1, Figures 13–16 show the probability of closed-loop stability, MPC total costs, as well as the state-space trajectory and state profiles for one of the initial condition for the RNN models in Case study 2. In Figure 13, it is shown that the probability starts from 0.5, and settles at around 0.7 for wider RNN models (i.e., more neurons). Figure 14 shows the MPC total costs for different models, from which it is demonstrated that the first model with only 5 neurons has a extremely high value, and all the other models achieve a total cost around 1500. Figures 13 and 14 demonstrate that all the RNN models except the first one achieve desired closed-loop performance in terms of high probability of closed-loop stability and low total costs. This is due to the low generalization error (around 0.005) for nearly all the models in Figure 3. Figure 15 shows the comparison of the closed-loop state trajectories under the two RNN models using 5 and 350 neurons, respectively, from which it is demonstrated that the model with 5 neurons (dashed line) drives the state out of the stability region, while the one with 350 neurons successfully stabilizes the system in the terminal set. The corresponding state profiles (i.e., $C_A - C_{As}$ and $T - T_s$) can be found in Figure 16.

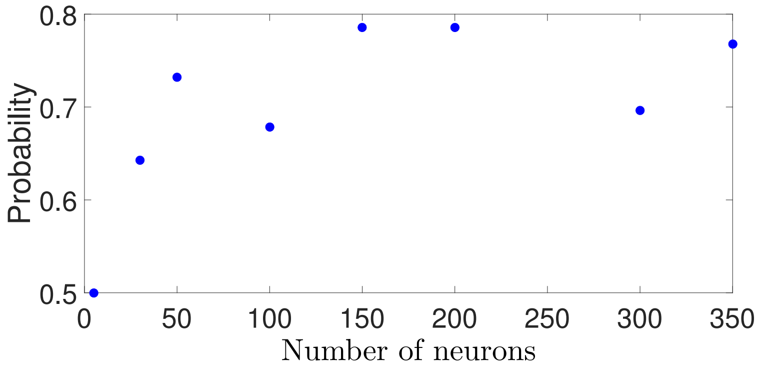

Figure 13. Probability of closed-loop stability vs. RNN width.

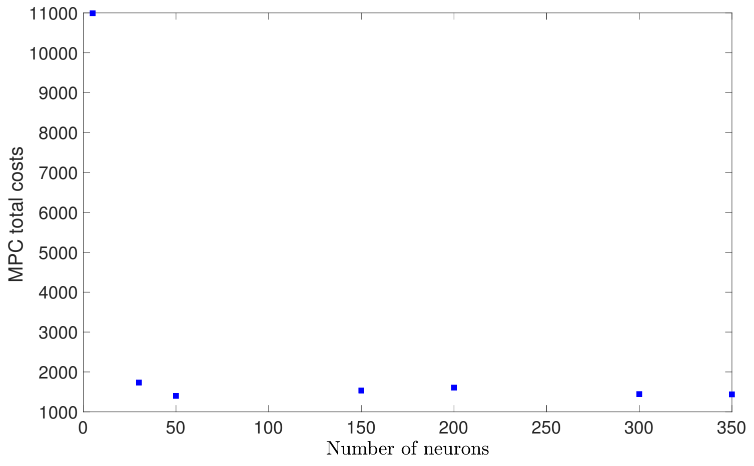

Figure 14. MPC total costs vs. RNN width.

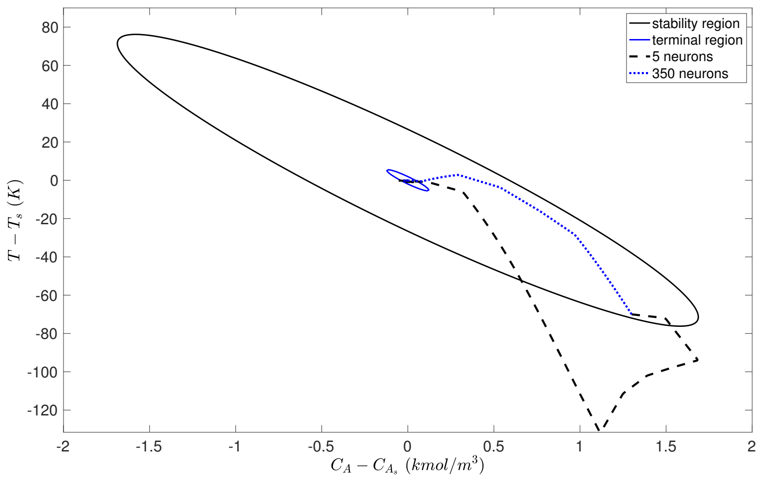

Figure 15. Closed-loop state trajectory under LMPC using two RNN models trained with different widths.

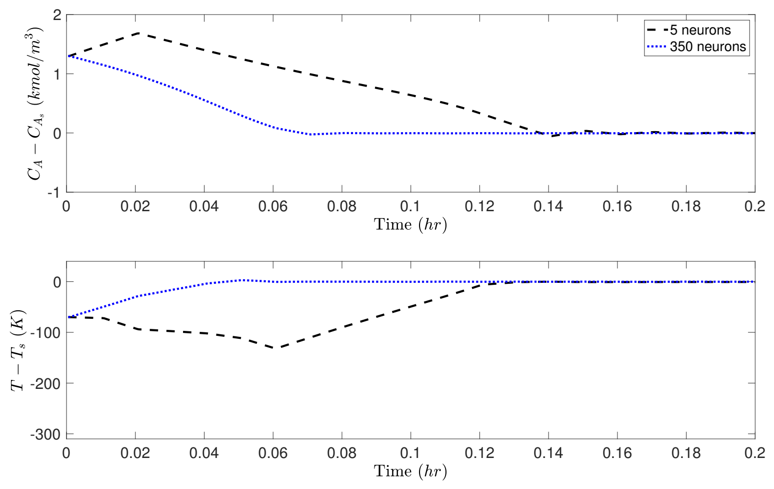

Figure 16. Closed-loop state profiles under LMPC using two RNN models trained with different widths.

To simplify the discussion for the remaining case studies, we will show the probability plot and MPC total cost plot only. Figure 17 shows the probability of closed-loop stability with respect to different RNN depths. It is demonstrated that the probability starts close to zero for one layer, increases up to 0.7 for four layers, and then decreases to almost zero for six layer and after. This trend follows exactly the generalization error plot in Figure 4, which shows the model with two, three and four layers achieve the lowest generalization error, and the models with more than five layers show worse generalization performance due to overfitting. Comparing to the closed-loop results for the RNNs with various widths in Figures 13 and 14, it is not surprising to see that the overall probability of closed-loop stability in this case study is worse because the open-loop generalization performance for the RNNs developed with different depths (Figure 4) is worse than that for the RNNs developed with different widths (Figure 3). Additionally, in Figure 18, we observe a similar pattern showing that the MPC total costs have the lowest values for two, three and four layers, and rise up for more layers.

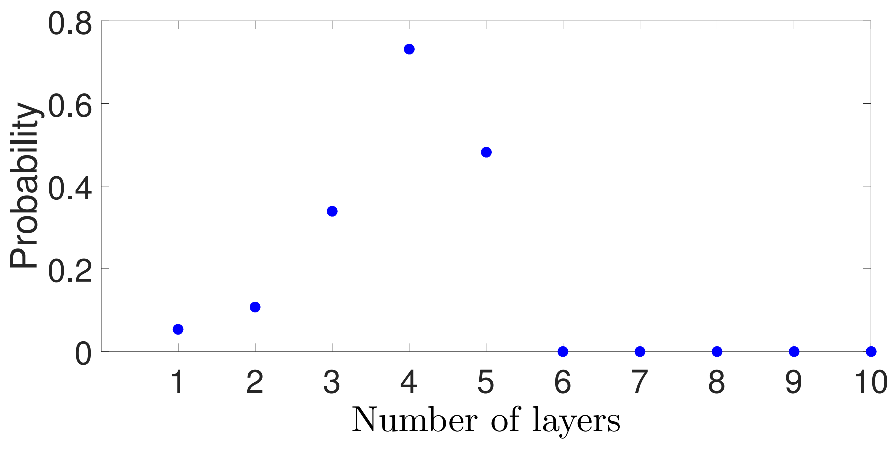

Figure 17. Probability of closed-loop stability vs. RNN depth.

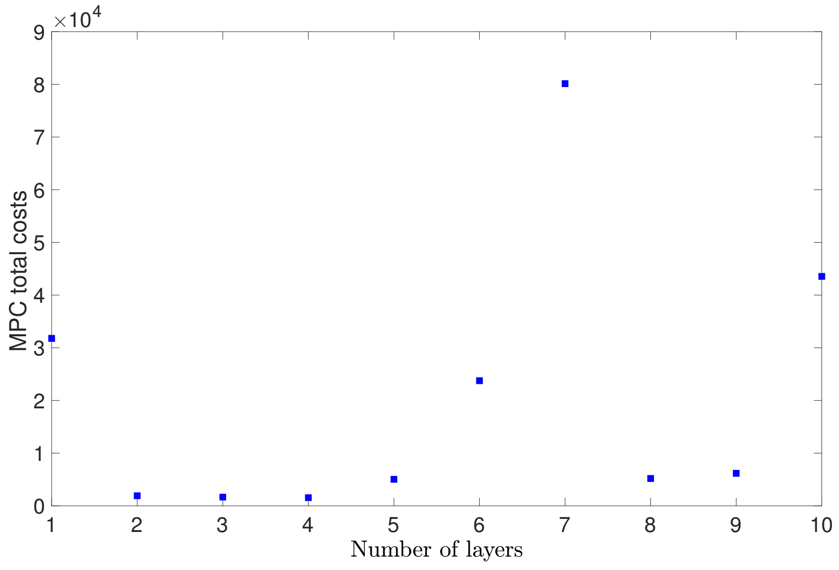

Figure 18. MPC total costs vs. RNN depth.

Closed-loop simulations for Case study 3 of different regions in $\Omega_{\hat{\rho}}$ are not carried out in this work, since the MPC formulation of Equations (68)–(73) only uses a single RNN model for prediction. Additionally, it is demonstrated from previous case studies that a single RNN model is sufficient to capture the process dynamics in the stability region, and therefore, there is no need to use different RNN models for different regions in $\Omega_{\hat{\rho}}$ from the control perspective.

Figure 19 shows the probability of closed-loop stability for the RNN models with different weight matrix bounds in Case study 4. It is shown that all the RNN models achieve a probability up to 0.7. The high probability of closed-loop stability is expected since in the open-loop generalization error plot in Figure 7, it is shown that all the models with different weight matrix bounds have a sufficiently small generalization error around 8×10^{-4}. As a result, the MPC total costs in Figure 20 are stable around 1000 for all models.

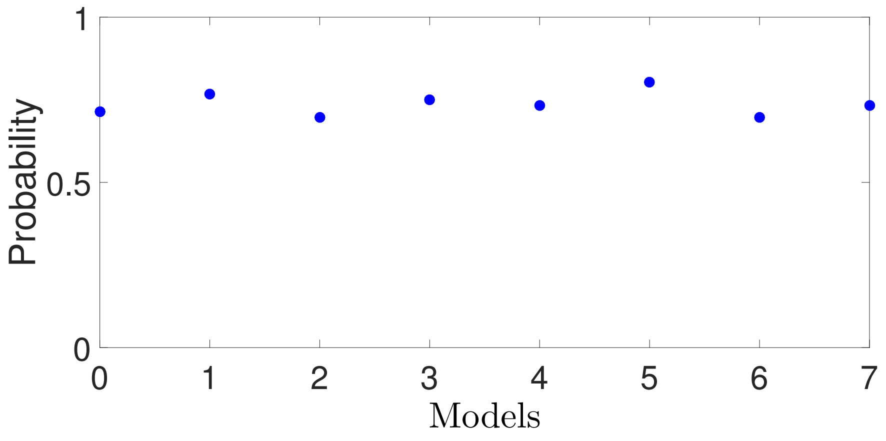

Figure 19. Probability of closed-loop stability vs. weight matrix bound.

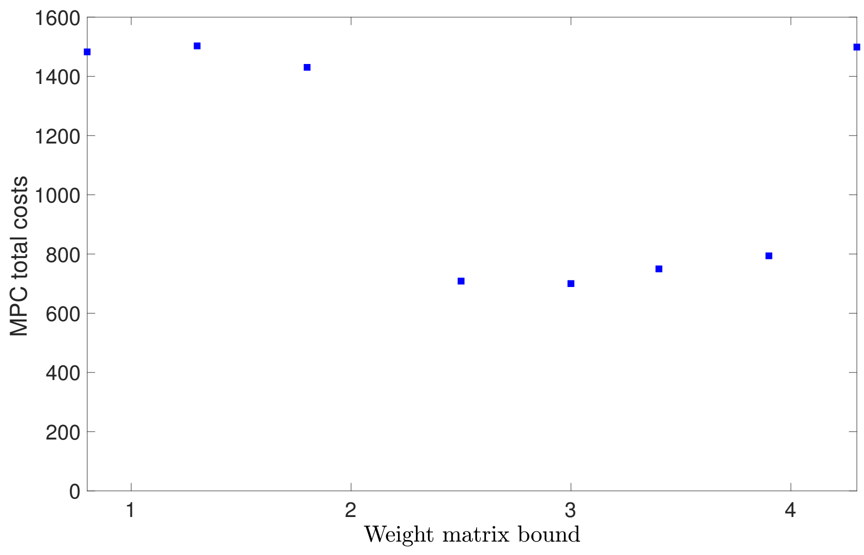

Figure 20. MPC total costs vs. weight matrix bound.

Lastly, Figures 21 and 22 show the closed-loop simulation results for Case study 5. As shown in Figure 21, the probability of closed-loop stability increases as the RNN input time length increases, and settles at around 0.9 for input time length greater than 6×10^{-3} hr. This seems inconsistent with the generalization performance of Figure 8 which shows the generalization error increases for longer input sequences at first glance. However, as we have discussed earlier, a low open-loop generalization error for short input sequences does not guarantee a desired closed-loop performance under MPC. Specifically, with shorter input sequences, the RNN prediction needs to be executed successively in each MPC iteration to predict all the future states within the prediction horizon. For example, in order to predict one sampling time $\Delta = 10^{-2}$ hr, the first RNN model with 1×10^{-3} input length in Figure 21 needs to run 10 times, and each time uses the previous predicted state as the initial state. The error accumulates during the calculation, which ultimately leads to poorer closed-loop performance. Therefore, for RNN models used in MPC, the input time length should be chosen carefully accounting for the system sampling time and MPC prediction horizon. Additionally, Figure 22 shows the MPC total costs with respect to different RNN input time lengths. It is seen that the first RNN model achieves the worst cost value, and all the other models have similar cost values around 2000. Through the closed-loop simulation of all the case studies investigated in the previous section, we demonstrate that the closed-loop performance is consistent with the open-loop generalization performance in the way that lower generalization errors typically leads to higher probability of closed-loop stability and lower MPC total costs. Therefore, the generalization error bound proposed in this work provides an efficient method for choosing neural network structure and data sample size to meet the closed-loop stability requirements.

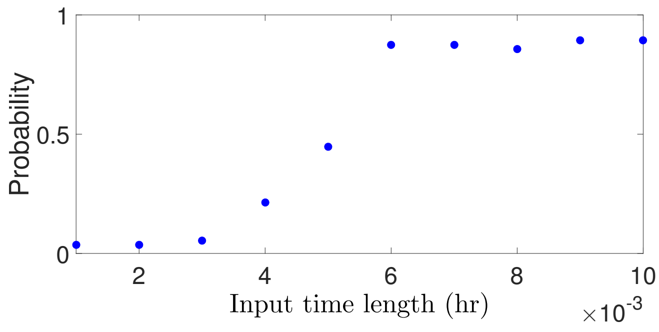

Figure 21. Probability of closed-loop stability vs. input time length.

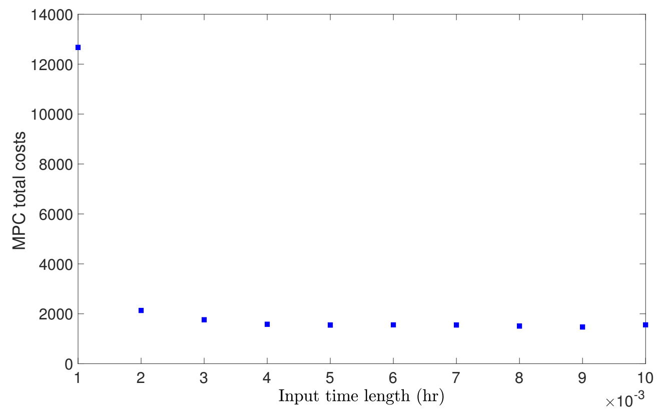

Figure 22. MPC total costs vs. input time length.

Remark 10. *The RNN models are trained offline, and the RNN-based MPC is solved in real time with new state measurements available at each sampling time. The averaged computation time for solving RNN-based MPC per sampling step is around 10 s, which is less than one sampling period* $\Delta = 0.01$ *hr* $= 36$ *s in this example. Therefore, the RNN-based MPC scheme can be implemented in real time without any computational issues.*

6. Conclusions

In this work, we developed a generalization probabilistic error bound for RNN models by taking advantage of the Rademacher complexity method for vector-valued functions. The RNN models were incorporated in the design of MPC, and probabilistic closed-loop stability properties were derived based on the RNN generalization error bounds. A number of case studies were simulated using a nonlinear chemical reactor example to demonstrate the impact of training sample size, the number of neurons and layers, regions where the data was generated, and input time length on the RNN generalization performance. Closed-loop simulation were carried out to further demonstrate the probabilistic closed-loop stability properties derived by the RNN-based LMPC.

Author Contributions: Z.W. developed the main results, performed the simulation studies and prepared the initial draft of the paper. D.R. contributed to the simulation studies in this manuscript. Q.G. and P.D.C. developed the idea of RNN generalization error, oversaw all aspects of the research and revised this manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare that they have no conflict of interest regarding the publication of the research article.

References

1. Cozad, A.; Sahinidis, N.V.; Miller, D.C. A combined first-principles and data-driven approach to model building. *Comput. Chem. Eng.* **2015**, *73*, 116–127. [CrossRef]
2. Wilson, Z.T.; Sahinidis, N.V. The ALAMO approach to machine learning. *Comput. Chem. Eng.* **2017**, *106*, 785–795. [CrossRef]
3. Ali, J.M.; Hussain, M.A.; Tade, M.O.; Zhang, J. Artificial Intelligence techniques applied as estimator in chemical process systems–A literature survey. *Expert Syst. Appl.* **2015**, *42*, 5915–5931.
4. Han, H.; Wu, X.; Qiao, J. Real-time model predictive control using a self-organizing neural network. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *24*, 1425–1436. [PubMed]
5. Wang, T.; Gao, H.; Qiu, J. A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 416–425. [CrossRef] [PubMed]

6. Wang, Y. A new concept using LSTM Neural Networks for dynamic system identification. In Proceedings of the American Control Conference 2017, Seattle, WA, USA, 24–26 May 2017; pp. 5324–5329.
7. Wong, W.; Chee, E.; Li, J.; Wang, X. Recurrent Neural Network-Based Model Predictive Control for Continuous Pharmaceutical Manufacturing. *Mathematics* **2018**, *6*, 242. [CrossRef]
8. Shahnazari, H.; Mhaskar, P.; House, J.M.; Salsbury, T.I. Modeling and fault diagnosis design for HVAC systems using recurrent neural networks. *Comput. Chem. Eng.* **2019**, *126*, 189–203. [CrossRef]
9. Wu, Z.; Tran, A.; Rincon, D.; Christofides, P.D. Machine Learning-Based Predictive Control of Nonlinear Processes. Part I: Theory. *AIChE J.* **2019**, *65*, e16729. [CrossRef]
10. Wu, Z.; Tran, A.; Rincon, D.; Christofides, P.D. Machine Learning-Based Predictive Control of Nonlinear Processes. Part II: Computational Implementation. *AIChE J.* **2019**, *65*, e16734. [CrossRef]
11. Pan, Y.; Wang, J. Nonlinear model predictive control using a recurrent neural network. In Proceedings of the IEEE International Joint Conference on Neural Networks 2008, Hong Kong, China, 1–8 June 2008; pp. 2296–2301.
12. Pan, Y.; Wang, J. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *IEEE Trans. Ind. Electron.* **2011**, *59*, 3089–3101. [CrossRef]
13. Xu, J.; Li, C.; He, X.; Huang, T. Recurrent neural network for solving model predictive control problem in application of four-tank benchmark. *Neurocomputing* **2016**, *190*, 172–178. [CrossRef]
14. Hoskins, J.; Himmelblau, D. Process control via artificial neural networks and reinforcement learning. *Comput. Chem. Eng.* **1992**, *16*, 241–251. [CrossRef]
15. Vepa, R. A review of techniques for machine learning of real-time control strategies. *Intell. Syst. Eng.* **1993**, *2*, 77–90. [CrossRef]
16. Hussain, M. Review of the applications of neural networks in chemical process control—Simulation and online implementation. *Artif. Intell. Eng.* **1999**, *13*, 55–68. [CrossRef]
17. Hewing, L.; Wabersich, K.; Menner, M.; Zeilinger, M. Learning-based model predictive control: Toward safe learning in control. *Annu. Rev. Control Robot. Auton. Syst.* **2020**, *3*, 269–296. [CrossRef]
18. Venkatasubramanian, V. The promise of artificial intelligence in chemical engineering: Is it here, finally? *AIChE J.* **2019**, *65*, 466–478. [CrossRef]
19. Mittal, M.; Gallieri, M.; Quaglino, A.; Salehian, S.; Koutník, J. Neural lyapunov model predictive control. *arXiv* **2020**, arXiv:2002.10451.
20. Limon, D.; Calliess, J.; Maciejowski, J. Learning-based nonlinear model predictive control. *IFAC-PapersOnLine* **2017**, *50*, 7769–7776. [CrossRef]
21. Aswani, A.; Gonzalez, H.; Sastry, S.; Tomlin, C. Provably safe and robust learning-based model predictive control. *Automatica* **2013**, *49*, 1216–1226. [CrossRef]
22. Valiant, L.G. A theory of the learnable. *Commun. ACM* **1984**, *27*, 1134–1142. [CrossRef]
23. Lygeros, J.; Margellos, K.; Prandini, M. Compression learning for chance constrained stochastic MPC. *IFAC-PapersOnLine* **2015**, *48*, 286–293. [CrossRef]
24. Zhou, Y.; Li, D.; Spanos, C. Learning optimization friendly comfort model for HVAC model predictive control. In Proceedings of the 2015 IEEE International Conference on Data Mining Workshop (ICDMW), Atlantic City, NJ, USA, 14–17 November 2015; pp. 430–439.
25. Bartlett, P.; Foster, D.J.; Telgarsky, M. Spectrally-normalized margin bounds for neural networks. *arXiv* **2017**, arXiv:1706.08498.
26. Zhang, Y.; Lee, J.; Wainwright, M.; Jordan, M.I. On the learnability of fully-connected neural networks. In Proceedings of the Artificial Intelligence and Statistics PMLR 2017, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 83–91.
27. Golowich, N.; Rakhlin, A.; Shamir, O. Size-independent sample complexity of neural networks. In Proceedings of the Conference On Learning Theory PMLR 2018, Stockholm, Sweden, 5–9 July 2018; pp. 297–299.
28. Cao, Y.; Gu, Q. Tight sample complexity of learning one-hidden-layer convolutional neural networks. *arXiv* **2019**, arXiv:1911.05059.
29. Chen, M.; Li, X.; Zhao, T. On generalization bounds of a family of recurrent neural networks. *arXiv* **2019**, arXiv:1910.12947.
30. Cao, Y.; Gu, Q. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *arXiv* **2019**, arXiv:1905.13210.
31. Zou, D.; Gu, Q. An improved analysis of training over-parameterized deep neural networks. *arXiv* **2019**, arXiv:1906.04688.
32. Akpinar, N.; Kratzwald, B.; Feuerriegel, S. Sample complexity bounds for recurrent neural networks with application to combinatorial graph problems. *arXiv* **2019**, arXiv:1901.10289.
33. Reid, M. Generalization bounds. In *Encyclopedia of Machine Learning*; Springer: Boston, MA, USA, 2010; pp. 447–454._328. [CrossRef]
34. Hanson, J.; Raginsky, M.; Sontag, E. Learning Recurrent Neural Net Models of Nonlinear Systems. *arXiv* **2020**, arXiv:2011.09573.
35. Sammut, C.; Webb, G.I. *Encyclopedia of Machine Learning*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2011.
36. Maurer, A. A vector-contraction inequality for rademacher complexities. In *Lecture Notes in Computer Science, Proceedings of the International Conference on Algorithmic Learning Theory, Bari, Italy, 19–21 October 2016*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 3–17.
37. Mohri, M.; Rostamizadeh, A.; Talwalkar, A. *Foundations of Machine Learning*; MIT Press: Cambridge, MA, USA, 2018.

38. Ledoux, M.; Talagrand, M. *Probability in Banach Spaces: Isoperimetry and Processes*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
39. Wu, Z.; Rincon, D.; Christofides, P.D. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. *J. Process Control* **2020**, *89*, 74–84. [CrossRef]
40. Keras. Available online: https://keras.io (accessed on 1 August 2015)
41. Wächter, A.; Biegler, L.T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* **2006**, *106*, 25–57. [CrossRef]