Systems & Control Letters 175 (2023) 105508

Contents lists available at ScienceDirect

Systems & Control Letters

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

Approximation of compositional functions with ReLU neural networks[☆]

Qi Gong [a,*], Wei Kang [b], Fariba Fahroo [b]

[a] *Department of Applied Mathematics, University of California, Santa Cruz, CA, USA*
[b] *Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA, USA*

A R T I C L E   I N F O

Article history:
Received 28 July 2022
Received in revised form 2 January 2023
Accepted 14 March 2023
Available online xxxx

Keywords:
ReLU neural network
Compositional functions
The curse of dimensionality
Deep neural networks

A B S T R A C T

The power of DNN has been successfully demonstrated on a wide variety of high-dimensional problems that cannot be solved by conventional control design methods. These successes also uncover some fundamental and pressing challenges in understanding the representability of deep neural networks for complex and high dimensional input–output relations. Towards the goal of understanding these fundamental questions, we applied an algebraic framework developed in our previous work to analyze ReLU neural network approximation of compositional functions. We prove that for Lipschitz continuous functions, ReLU neural networks have an approximation error upper bound that is a polynomial of the network's complexity and the compositional features. If the compositional features do not increase exponentially with dimension, which is the case in many applications, the complexity of DNN has a polynomial growth. In addition to function approximations, we also establish ReLU network approximation results for the trajectories of control systems, and for a Lyapunov function that characterizes the domain of attraction.

1. Introduction

The curse of dimensionality (COD), a term coined by Richard Bellman in his book on dynamic programming [1], is a phenomenon in which the complexity of an approximate solution grows so fast, such as exponentially, with the state space dimension that the solution is computationally intractable for practical applications. COD arises as a major bottleneck in many applications of dynamical systems and nonlinear control. In recent years, deep neural network (DNN) has emerged as a promising tool to mitigate the COD for high dimensional problems that cannot be solved using conventional computational methods. The power of DNN has been successfully demonstrated on a wide variety of high-dimensional problems that cannot be solved by conventional control design methods [2–15]. These empirical successes uncover some fundamental and pressing challenges especially in understanding the representability of deep neural networks for complex and high dimensional input–output relations. Why are DNNs seemingly capable of mitigating the curse of dimensionality that is commonly suffered by other function approximation methods? Not surprisingly, such questions have attracted a large amount of research, for example, [16–27]. The work of this paper is based upon [19], where an algebraic framework is introduced to analyze DNN approximation of compositional functions. It is proved in [19] that, for any compositional function in C^1, the complexity of its DNN approximation is a polynomial of four compositional features. If these features do not depend on the function's input dimension exponentially, which is the case for many practical problems, the complexity of its DNN approximation is bounded by a polynomial of the input dimension rather than exponential function, i.e., the COD is mitigated. The main goal of this paper is to extend the idea in [19] to C^0 functions and to prove a polynomial complexity of DNN approximation for C^0 functions using ReLU networks.

It has been commonly observed in science and engineering that complicated input–output information relations consist of compositions of simple functions with low input dimensions; and the connections between the simple functions are sparse. Interestingly, deep neural networks and many iterative numerical algorithms for differential equations and optimization problems can also be viewed as compositional functions. Many nonlinear feedback control laws also have compositional structures. For example, the well-known backstepping [28,29] and adding a power integrator [30–32] approaches for lower triangular nonlinear systems, and forwarding [33] and saturation [34] approaches for upper-triangular systems construct nonlinear controllers recursively, which result in nonlinear controls with some compositional structures. When the state dimension is high, analytically

☆ This work was supported in part by the Air Force Office of Scientific Research (AFOSR), USA under grant no. FA9550-21-1-0113, the National Science Foundation (NSF), USA under grants no. 2134235 and no. 2202668, and Naval Research Laboratory - Monterey, California, USA.

* Corresponding author.
E-mail addresses: qgong@ucsc.edu (Q. Gong), wkang@nps.edu (W. Kang), fariba.fahroo@nps.edu (F. Fahroo).

https://doi.org/10.1016/j.sysconle.2023.105508

deriving nonlinear feedback control using such methods could be challenging. Numeric approximations of compositional feedback control by, for example, DNNs, present themselves as promising tools for such high-dimensional control problems.

In the recent work [19], it has been shown that DNNs of n number of neurons with smooth activation functions can approximate any C^m ($m \geq 1$) compositional functions with error bounded by $O(n^{-1/r})$, where r is the largest radio, d_i/m_i, between the input dimension d_i and the smoothness m_i among all nodes (components) in a compositional function. Since the approximate rate, $-1/r$, does not depend on the overall input dimension of a compositional function, the COD can be avoided if the dimensions of individual nodes remain small as the overall input dimension increases. Such situations appear in many examples that we have studied, including Lorenz-96 model [35], power system models [36–38], etc. As shown in [19], similar DNN approximation results hold for the solution of differential equations and optimal feedback control.

In this paper we extend the results in [19] in a few directions: (1) We relax the C^1 smoothness requirement on the compositional functions to Lipschitz continuous functions. Such extension not only includes a larger family of compositional functions, but also is important for control applications because many feedback controls are non-smooth due to the presence of constraints. (2) We extend the results in [19] from DNNs with smooth activation functions to deep ReLU networks. ReLU activation function has been a popular choice in many DNN applications. It enjoys desirable properties such as avoiding the gradient degeneracy in the training process. Recent studies on the expressiveness of ReLU DNNs also reveal important theoretical properties of ReLU DNNs for approximating continuous functions [21,22,39]. By combining the algebraic framework in [19] for analyzing compositional functions and the ReLU network approximation theory in [21] for continuous functions, we construct deep ReLU networks based on sparse structure of compositional functions. We show in this paper that the approximation error is a polynomial function of four compositional features. In particular, the approximation error is $O(W^{-2/d^{\mathbf{f}}_{max}})$, where W is the total number of parameters in a ReLU network and $d^{\mathbf{f}}_{max}$ is the maximum input dimension of nonlinear nodes in a compositional function. Thus, for high-dimensional compositional functions with small $d^{\mathbf{f}}_{max}$, the COD can be mitigated. In addition to function approximation, we further establish ReLU network results for the trajectories of control systems, and for a Lyapunov function that characterizes the domain of attraction of dynamical systems.

In the rest of this paper, we first review some key concepts of compositional functions represented as layered directed acyclic graphs (DAG) in Section 2. The main result on deep ReLU network approximation of Lipschitz continuous functions is presented in Section 3. It is followed by extensions of the function approximation results to the ReLU network approximation of trajectories of differential equations in Section 4, and the approximation of a Lyapunov function for characterizing the domain of attraction of dynamical systems in Section 5. We conclude the paper with a summary of contributions and some concluding remarks in Section 6.

2. Compositional functions

In this section, we review some key concepts from [19] on compositional functions. Readers are referred to [19] for more details and related results. Representing compositional functions using layered DAGs is essential in this paper. Consider a power system model in [37,38] as an illustrative example. The electric

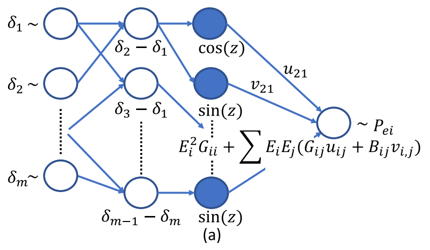

Fig. 1. Layered DAG of $(\mathbf{P}_e)_i$ defined in (1).

air-gap torque, $\mathbf{P}_e \in \mathbb{R}^m$, is a vector valued function. Its ith component is

$$(\mathbf{P}_e)_i = E_i^2 G_{ii} + \sum_{j=1, j\neq i}^{m} E_i E_j (G_{ij} \cos(\delta_i - \delta_j) + B_{ij} \sin(\delta_i - \delta_j)), \quad (1)$$

where m is the number of generators. For the ith generator, δ_i is the rotor angle in radian, E_i is the electromotive force or internal voltage of the generator, $G_{ij}+jB_{ij}$ is the ith row jth column element of the admittance matrix among all electromotive forces, and G_{ii} is the conductance representing the local load seen from E_i. If the system has $m = 20$ generators, then the model is a function $\mathbf{P}_e : \mathbb{R}^{20} \to \mathbb{R}^{20}$. It is a compositional function with simple nodes. Its layered DAG representation is shown in Fig. 1. Except for the layer of $\sin(z)$ and $\cos(z)$, all nodes are linear. For nonlinear nodes, they all have a single input. The layered DAG associated with a given function is not unique. In this study, two different layered DAGs are considered as different compositional functions even if they both represent the same input–output relationship. The compositional function is formally defined as follows.

Definition 1 (*Compositional Function [19]*)**.** A compositional function, denoted by a pair $(\mathbf{f}, \mathcal{G}^{\mathbf{f}})$, consists of a function, $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^q$, a layered DAG, $\mathcal{G}^{\mathbf{f}}$. We use subscripts to represent the layer number, i.e., $f_{i,j}$ is the jth node in the ith layer. Every node in $\mathcal{G}^{\mathbf{f}}$ that has at least one inward edge is a function, $f_{i,j} : \mathbb{R}^{d_{i,j}} \to \mathbb{R}$, where $d_{i,j}$ is the number of inward edges of the node. A node that has no inward edge is called an input node. It can take any value in the domain of $\mathbf{f}$. All input nodes in $\mathcal{G}^{\mathbf{f}}$ are labeled as layer *0*, called the input layer. A node that has no outward edge is called an output node. All output nodes are located in the final layer, called the output layer. All layers between input and output layers are called hidden layers. We always assume that the ranges and domains of all nodes are compatible for composition, i.e. if $(f_{i,j}, f_{l,k})$ is an edge in $\mathcal{G}^{\mathbf{f}}$, then the range of $f_{i,j}$ is contained in the interior of the domain of $f_{l,k}$.

In a compositional function $(\mathbf{f}, \mathcal{G}^{\mathbf{f}})$, $\mathbf{f} : [-R, R]^d \to \mathbb{R}^q$, the set of nodes in $\mathcal{G}^{\mathbf{f}}$ is denoted by $\mathcal{V}^{\mathbf{f}}$. If a node $f_{i,j}$ is a 1st order polynomial, then it is called a *linear node*. Otherwise, it is called a *general node*. All input nodes are treated as linear nodes. The set of linear nodes is denoted by $\mathcal{V}^{\mathbf{f}}_L$ and the set of general (nonlinear) nodes is denoted by $\mathcal{V}^{\mathbf{f}}_G$.

It has been shown in [19] that the family of compositional functions is closed under algebraic operations, including composition, linear combination, vector inner product and scalar function division. To analyze approximations of compositional functions, the concept of sensitivity with respect to (w.r.t.) a node is essential. This concept is defined based on an operation on DAGs called *compositional truncation*. We illustrate the concept of compositional truncation using the following simple example. Consider a compositional function with five layers shown in Fig. 2(a). Its truncation along the layer $l = 1$ is another compositional function, $\bar{\mathbf{f}}$, that is shown in Fig. 2(b). In the truncation process,

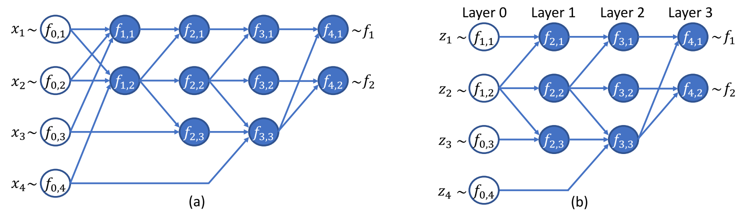

Fig. 2. Compositional truncation. The function in (b) is the compositional truncation of the function in (a) along layer $l = 1$.

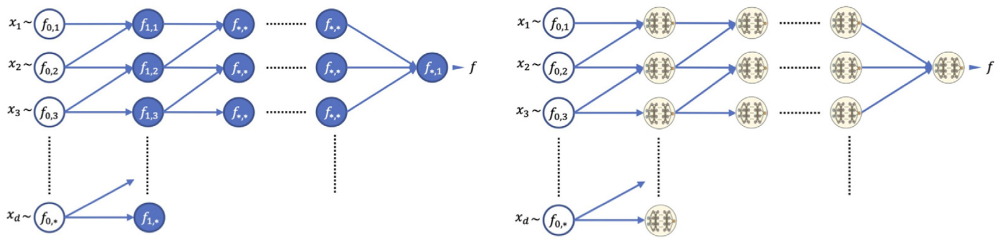

Fig. 3. A deep ReLU neural network approximation of compositional functions. Left figure: a compositional function with nodes $f_{i,j}$. Right figure: a deep ReLU neural network constructed by replacing each $f_{i,j}$ by its ReLU network approximation $f^{NN}_{i,j}$.

we first remove all edges that end at nodes in layers $l \leq 1$. Subsequently, $f_{0,1}$ and $f_{0,2}$ become isolated nodes, i.e., they have neither outward nor inward edge. All isolated nodes are removed from the graph. To form the input layer of the truncated function, we align all nodes in layers $l \leq 1$. In this case, they are $f_{1,1}, f_{1,2}, f_{0,3}$ and $f_{0,4}$. A set of dummy variables, (z_1, z_2, z_3, z_4), is introduced as the input of $\bar{\mathbf{f}}$, the truncated function. It is important to clarify that the domain of z_k is the intersection of the domains of all nodes that are directly connected with z_k in $\bar{\mathbf{f}}$. For example, z_2 in Fig. 2 can take any value that is in the domains of $f_{2,1}, f_{2,2}$ and $f_{2,3}$ simultaneously. The value is not necessarily in the range of the original node $f_{1,2}$ in $\mathcal{G}^{\mathbf{f}}$. The formal definition of compositional truncation can be found in [19].

Definition 2 (*The Sensitivity Associated with a Node [19]*)**.** Given a compositional function $(\mathbf{f}, \mathcal{G}^{\mathbf{f}})$ and a constant $1 \leq p \leq \infty$. Consider a node, $f_{i,j}$, in the ith layer, where $0 \leq i \leq l^{\mathbf{f}}_{max} - 1$ ($l^{\mathbf{f}}_{max}$ is the largest layer number in the DAG). Let $\bar{\mathbf{f}}(z_1, \ldots, z_j, \ldots)$ be the truncation of $\mathbf{f}$ along the ith layer. If $s_{i,j} > 0$ is a constant satisfying

$$\left\|\bar{\mathbf{f}}(z_1, \ldots, z_j + \Delta z, \ldots) - \bar{\mathbf{f}}(z_1, \ldots, z_j, \ldots)\right\|_p \leq s_{i,j}\,|\Delta z| \tag{2}$$

for all $(z_1, \ldots, z_j, \ldots)$ and $(z_1, \ldots, z_j+\Delta z, \ldots)$ within the domain of $\bar{\mathbf{f}}$, $s_{i,j}$ is called a sensitivity constant (under the p-norm) associated with $f_{i,j}$. In some discussions, we denote the constant by $s^{\mathbf{f}}_{i,j}$ to differentiate sensitivity constants associated with different functions.

Remark 1. Note that in [19] $s_{i,j}$ is called a Lipschitz constant associated with $f_{i,j}$. To avoid confusion with the standard Lipschitz constant of $f_{i,j}$ itself, we rename it as sensitivity constant.

3. ReLU network approximation of compositional functions

In this section, we apply the algebraic framework introduced in [19] for compositional functions to analyze the expressiveness of deep ReLU networks for Lipschitz continuous compositional functions satisfying the following assumption.

Assumption 1 (*Compositional Functions*)**.** Given a compositional function $(\mathbf{f}, \mathcal{G}^{\mathbf{f}})$. We assume that $\mathbf{f}$ is a function from $[-R, R]^d$ to $\mathbb{R}^q$ for some $R > 0$ and positive integers d and q. Assume that the nodes, $\{f_{i,j}\}$, are Lipschitz continuous functions with input dimension $d_{i,j} \geq 1$ and a Lipschitz constant $l_{i,j}$. The domain of $f_{i,j}$ is a hypercube with edge length $R_{i,j} > 0$. The ranges and domains of all nodes are compatible for composition, i.e., if $(f_{i,j}, f_{l,k})$ is an edge in $\mathcal{G}^{\mathbf{f}}$, then the range of $f_{i,j}$ is contained in the interior of the domain of $f_{l,k}$.

Our analysis is based on a recent result by Yarotsky [21] on the approximation of continuous functions by deep ReLU neural networks. In this work, an approximation error upper bound was given in terms of the modulus of continuity defined below.

Definition 3 (*Modulus of Continuity*)**.** Let $\omega : [0, \infty) \to [0, \infty)$ be an increasing real valued function vanishing at 0 and continuous at 0. A uniformly continuous function $f : [-R, R]^d \to R$ admits ω as modulus of continuity if and only if for all $x, y \in [-R, R]^d$

$$|f(x) - f(y)| \leq \omega\left(\|x - y\|_\infty\right).$$

The following Theorem 1 on the approximation error of ReLU neural networks was proved by Yarotsky [21] for continuous function defined on hypercube $[0, 1]^d$; and was extended to domains $[0, M]^d$ in [22]. Here, we restate the result on domain $[-R, R]^d$.

Theorem 1. *(Approximation of Continuous Functions by Relu Networks [21]) For any continuous function $f : [-R, R]^d \to R$ with the modulus of continuity ω_f, there exists a sequence of fully-connected ReLU network, f^{NN}, of constant width $2d+10$ and depth $D = O(W)$, where W is the size of the network (i.e., the total number of neural network parameters), such that*

$$\|f - f^{NN}\|_{L^\infty([-R,R]^d)} \leq c_1(d)\omega_f(c_2(d)2RW^{-2/d}), \tag{3}$$

where $c_1(d)$ and $c_2(d)$ are positive constants depending on the input dimension d but independent of f and W.

Now consider a compositional function $(\mathbf{f}, \mathcal{G}^{\mathbf{f}})$, $\mathbf{f} : [-R, R]^d \to \mathbb{R}^q$. Based on Theorem 1, each node, $f_{i,j}$, in this compositional function can be approximated by a ReLU network, $f^{NN}_{i,j}$. Substituting $f^{NN}_{i,j}$ for all nodes in $\mathcal{G}^{\mathbf{f}}$ results in a deep ReLU network that is built upon the compositional structure; see Fig. 3 for an illustration. Propagation of the approximation errors in each node can then be tracked using tools introduced in [19]. In this idea, a set of compositional features is essential.

Definition 4. In a compositional function $(\mathbf{f}, \mathcal{G}^{\mathbf{f}})$, $\mathbf{f} : [-R, R]^d \to \mathbb{R}^q$, the set of nodes in $\mathcal{G}^{\mathbf{f}}$ is denoted by $\mathcal{V}^{\mathbf{f}}$. The set of linear nodes is denoted by $\mathcal{V}_L^{\mathbf{f}}$ and the set of general (nonlinear) nodes is denoted by $\mathcal{V}_G^{\mathbf{f}}$. The following quantities are called compositional features,

$$\begin{aligned}
d_{max}^{\mathbf{f}} &= \max\{d_{i,j};\ \text{for } f_{i,j} \in \mathcal{V}_G^{\mathbf{f}}\},\\
L_{max}^{\mathbf{f}} &= \max\{l_{i,j};\ \text{for } f_{i,j} \in \mathcal{V}_G^{\mathbf{f}}\},\\
R_{max}^{\mathbf{f}} &= \max\{R_{i,j};\ \text{for } f_{i,j} \in \mathcal{V}_G^{\mathbf{f}}\},\\
S_{max}^{\mathbf{f}} &= \max\{s_{i,j};\ \text{for } f_{i,j} \in \mathcal{V}_G^{\mathbf{f}}\},\\
\left|\mathcal{V}_G^{\mathbf{f}}\right| &= \text{the total number of general nodes } f_{i,j} \in \mathcal{V}_G^{\mathbf{f}},
\end{aligned}$$

where $f_{i,j}$, $d_{i,j}$, $l_{i,j}$, $R_{i,j}$ and $s_{i,j}$ are introduced in Definitions 1 and 2 and Assumption 1.

Theorem 2. *Consider a compositional function* $(\mathbf{f}, \mathcal{G}^{\mathbf{f}})$, $\mathbf{f} : [-R, R]^d \to \mathbb{R}^q$, *satisfying Assumption 1. There exists a deep ReLU network of size W such that*

$$\left\|\mathbf{f}(\mathbf{x}) - \mathbf{f}^{NN}(\mathbf{x})\right\|_\infty \le CW^{-2/d_{max}^{\mathbf{f}}}, \quad \text{for all } \mathbf{x} \in [-R, R]^d, \tag{4}$$

where constant

$$C = C_1 L_{max}^{\mathbf{f}} R_{max}^{\mathbf{f}} S_{max}^{\mathbf{f}} \left|\mathcal{V}_G^{\mathbf{f}}\right|^{1+2/d_{max}^{\mathbf{f}}} \tag{5}$$

for some constant C_1 depending on input dimension, $d_{i,j}$, of each general node $f_{i,j} \in \mathcal{V}_G^{\mathbf{f}}$.

Proof. We first construct a ReLU network approximation, $\mathbf{f}^{NN}(\mathbf{x})$, of the compositional function $\mathbf{f}(\mathbf{x})$; then estimate the approximation error. The construction of f^{NN} is based on two steps;

1. approximating each node $f_{i,j}$ of the compositional function by appropriate ReLU networks, $f_{i,j}^{NN}$;
2. substituting $f_{i,j}^{NN}$ for all nodes in $\mathcal{G}^{\mathbf{f}}$.

This process generates a deep ReLU network denoted by f^{NN}; see Fig. 3 for an illustration. To construct $f_{i,j}^{NN}$, we distinguish two cases depending on if $f_{i,j}$ is a linear node or not.

- For all linear nodes $f_{i,j} \in \mathcal{V}_L^{\mathbf{f}}$, it is clear that $f_{i,j}$ can be exactly represented by two neurons with ReLU activation function as

  $$f_{i,j}(\mathbf{z}) = \sigma(f_{i,j}(\mathbf{z})) + \sigma(-f_{i,j}(\mathbf{z})) \triangleq f_{i,j}^{NN}(\mathbf{z}),$$

  where $\sigma(x) = \max\{x, 0\}$. The size (total number of parameters) of $f_{i,j}^{NN}$ is $2d_{i,j} + 2$.
- For each general (nonlinear) nodes $f_{i,j} \in \mathcal{V}_G^{\mathbf{f}}$, based on Theorem 1, there exists a fully-connected feedforward ReLU neural network, $f_{i,j}^{NN}$, of constant width $2d_{i,j} + 10$ and depth $D_{i,j} = O(W_{i,j})$, such that

  $$\left|f_{i,j} - f_{i,j}^{NN}\right| \le c_1(d_{i,j})\omega_{f_{i,j}}(c_2(d_{i,j})2R_{i,j}W_{i,j}^{-2/d_{i,j}}). \tag{6}$$

  We assign $W_{i,j}$ so that they are of the same order. More precisely, let

  $$W_{min} = \min\{W_{i,j}, f_{i,j} \in \mathcal{V}_G^{\mathbf{f}}\}, \quad W_{max} = \max\{W_{i,j}, f_{i,j} \in \mathcal{V}_G^{\mathbf{f}}\}.$$

  $W_{i,j}$ are chosen so that $k = W_{max}/W_{min} = O(1)$. Moreover, without lost of generality, we assume W_{min} is sufficiently large so that $\sum_{f_{i,j}\in\mathcal{V}_L^{\mathbf{f}}} \left(2d_{i,j} + 2\right) \le \sum_{f_{i,j}\in\mathcal{V}_G^{\mathbf{f}}} W_{i,j}$.

Substituting all $f_{i,j}^{NN}$ to the corresponding linear and general nodes leads to a deep ReLU network of size

$$\begin{aligned}
W &= \sum_{f_{i,j}\in\mathcal{V}_G^{\mathbf{f}}} W_{i,j} + \sum_{f_{i,j}\in\mathcal{V}_L^{\mathbf{f}}} \left(2d_{i,j} + 2\right)\\
&\le 2 \sum_{f_{i,j}\in\mathcal{V}_G^{\mathbf{f}}} W_{i,j}\\
&\le 2k \left|\mathcal{V}_G^{\mathbf{f}}\right| W_{min}.
\end{aligned} \tag{7}$$

Next we estimate the approximation error of $\mathbf{f}^{NN}$ with respect to the features of the compositional function, $(\mathbf{f}, \mathcal{G}^{\mathbf{f}})$, and W. Under Assumption 1, each general node, $f_{i,j}$, is Lipschitz continuous. Its modulus continuity $\omega_{f_{i,j}}$ can be set as a linear function. Thus, (6) implies that

$$\begin{aligned}
\left|f_{i,j} - f_{i,j}^{NN}\right| &\le 2l_{i,j}R_{i,j}c_1(d_{i,j})c_2(d_{i,j})W_{i,j}^{-2/d_{i,j}}\\
&\le 2l_{i,j}R_{i,j}c_1(d_{i,j})c_2(d_{i,j})W_{i,j}^{-2/d_{max}^{\mathbf{f}}}\\
&\le C_{max}L_{max}^{\mathbf{f}}R_{max}^{\mathbf{f}}W_{min}^{-2/d_{max}^{\mathbf{f}}},
\end{aligned} \tag{8}$$

where $C_{max} = \max\{2c_1(d_{i,j})c_2(d_{i,j}),\ f_{i,j} \in \mathcal{V}_G^{\mathbf{f}}\}$. From Proposition 3.9 in [19] and (8), the approximation error

$$\begin{aligned}
\left\|\mathbf{f}(\mathbf{x}) - \mathbf{f}^{NN}(\mathbf{x})\right\|_\infty &\le \sum_{f_{i,j}\in\mathcal{V}_L^{\mathbf{f}}} s_{i,j}^{\mathbf{f}} \left|f_{i,j} - f_{i,j}^{NN}\right| + \sum_{f_{i,j}\in\mathcal{V}_G^{\mathbf{f}}} s_{i,j}^{\mathbf{f}} \left|f_{i,j} - f_{i,j}^{NN}\right|\\
&= \sum_{f_{i,j}\in\mathcal{V}_G^{\mathbf{f}}} s_{i,j}^{\mathbf{f}} \left|f_{i,j} - f_{i,j}^{NN}\right|\\
&\le \sum_{f_{i,j}\in\mathcal{V}_G^{\mathbf{f}}} C_{max}L_{max}^{\mathbf{f}}R_{max}^{\mathbf{f}}S_{max}^{\mathbf{f}}W_{min}^{-2/d_{max}^{\mathbf{f}}}\\
&= C_{max}L_{max}^{\mathbf{f}}R_{max}^{\mathbf{f}}S_{max}^{\mathbf{f}} \left|\mathcal{V}_G^{\mathbf{f}}\right| W_{min}^{-2/d_{max}^{\mathbf{f}}}.
\end{aligned}$$

Substituting (7) to the above upper bound, we get

$$\begin{aligned}
&\left\|\mathbf{f}(\mathbf{x}) - \mathbf{f}^{NN}(\mathbf{x})\right\|_\infty\\
&\le (2k)^{2/d_{max}^{\mathbf{f}}} C_{max}L_{max}^{\mathbf{f}}R_{max}^{\mathbf{f}}S_{max}^{\mathbf{f}} \left|\mathcal{V}_G^{\mathbf{f}}\right|^{1+2/d_{max}^{\mathbf{f}}} W^{-2/d_{max}^{\mathbf{f}}}.
\end{aligned}$$

Thus (4)–(5) is proved with $C_1 = (2k)^{2/d_{max}^{\mathbf{f}}}C_{max}$. ♦

Remark 2. From the proof of Theorem 2, it is clear that the constructed ReLU network $\mathbf{f}^{NN}$ is a deep feedforward network with non-constant width. Such networks explore not only the expressiveness of ReLU networks (for each individual nodes $f_{i,j}$) but also the overall sparse compositional structure of $\mathbf{f}$. As a result, the approximation rate, $-2/d_{max}^{\mathbf{f}}$ in (4), depends on the input dimension of the individual general nodes, $f_{i,j} \in \mathcal{V}_G^{\mathbf{f}}$, not the input dimension of the overall compositional function $\mathbf{f}$. Therefore, if the input dimension of individual general nodes in a compositional function is bounded, the error upper bound of deep neural network approximation is a polynomial function of the compositional features, $\{d_{max}^{\mathbf{f}}, L_{max}^{\mathbf{f}}, R_{max}^{\mathbf{f}}, S_{max}^{\mathbf{f}}, \left|\mathcal{V}_G^{\mathbf{f}}\right|\}$. Thus, for compositional functions with low $d_{max}^{\mathbf{f}}$, the curse of dimensionality can be avoided. From the proof it is also clear that linear nodes $f_{i,j} \in \mathcal{V}_L^{\mathbf{f}}$ do not contribute to the approximation error.

4. ReLU approximation of ODEs and control systems

Consider a system of time-invariant ordinary differential equations (ODEs)

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, \mathbf{p}), \quad \mathbf{y} \in \mathbb{R}^d,\ \mathbf{f}(\mathbf{y}, \mathbf{p}) \in \mathbb{R}^d, \mathbf{p} \in \mathbb{R}^p,\ t \in [0, T], \tag{9}$$

where $\mathbf{y}$ is the state variable and $\mathbf{p}$ is the parameter in the model. For control systems, $\mathbf{p}$ represents the parameterized control input, for instance, the value of control input at grid points. The result in this section holds also for time-varying ODEs, since a time-varying ODE can be converted into a time-invariant one. We assume that $\mathbf{f}$ is a compositional function that satisfies Assumption 1. Thus, $\mathbf{f}$ is Lipschitz continuous in its domain with respect to $(\mathbf{y}, \mathbf{p})$, which ensures the existence and uniqueness of the solution to the initial value problem in (9).

Let $\boldsymbol{\phi}(t; \mathbf{x}, \mathbf{p})$ represent the solution of (9) satisfying $\mathbf{y}(0) = \mathbf{x}$. We are interested in approximating $\boldsymbol{\phi}(T; \cdot, \cdot) : (\mathbf{x}, \mathbf{p}) \to \boldsymbol{\phi}(T; \mathbf{x}, \mathbf{p})$, as a function of the initial condition and the parameter, by deep ReLU neural networks. Or equivalently, the trained DNN is a model of the system trajectory. Different from the ODE model, numerically evaluating a DNN is usually faster in real-time computation than solving ODEs. Note that although $\mathbf{f}$ is a compositional function, after integration, the compositional structure of the flow map, $\boldsymbol{\phi}(T; \mathbf{x}, \mathbf{p})$, becomes unknown. Therefore, we cannot apply Theorem 2 directly on $\boldsymbol{\phi}(T; \mathbf{x}, \mathbf{p})$. Instead, we approximate $\boldsymbol{\phi}(T; \mathbf{x}, \mathbf{p})$ by explicit numerical schemes, which can be viewed as iterations on the compositional function $\mathbf{f}$. In the following, we focus on forward Euler method. The result can be generalized to other explicit methods for differential equation.

Let $\mathbf{f}_E : \mathbb{R}^{d+p} \to \mathbb{R}^d$ represent the operator of the forward Euler method, i.e.,

$$\mathbf{f}_E(\mathbf{y}, \mathbf{p}) = \mathbf{y} + h\mathbf{f}(\mathbf{y}, \mathbf{p}), \tag{10}$$

where $h = T/K$ for some positive integer K. Applying (10) from initial condition $\mathbf{x}$ generates a numerical approximation of the solution

$$\boldsymbol{\phi}(T; \mathbf{x}, \mathbf{p}) \approx \overbrace{\mathbf{f}_E \circ \cdots \mathbf{f}_E \circ \mathbf{f}_E}^{K}(\mathbf{x}, \mathbf{p}) = (\mathbf{f}_E(\cdot, \cdot))^K(\mathbf{x}, \mathbf{p}).$$

It is clear that $(\mathbf{f}_E(\cdot, \cdot))^K$ is a compositional function with a DAG depending on the compositional structure of $\mathbf{f}$. Thus, a ReLU network approximation of $\mathbf{f}$ generates a deep ReLU approximation of $(\mathbf{f}_E(\cdot, \cdot))^K$, which further approximates the solution $\boldsymbol{\phi}(T; \mathbf{x}, \mathbf{p})$. The result is stated in the following theorem.

Theorem 3. *Consider the ODE (9) in which $(\mathbf{f}, \mathcal{G}^{\mathbf{f}})$ is a compositional function of $(\mathbf{x}, \mathbf{p})$ satisfying Assumption 1. Suppose $D^{\mathbf{f}} \subseteq [-R, R]^{d+p}$ is a compact set such that $\boldsymbol{\phi}(t; \mathbf{x}, \mathbf{p}) \in [-R, R]^d$ whenever $(\mathbf{x}, \mathbf{p}) \in D^{\mathbf{f}}$ and $t \in [0, T]$. There exists a deep ReLU network, $\mathbf{f}_E^{NN}$, of size W, such that for all $(\mathbf{x}, \mathbf{p}) \in D^{\mathbf{f}}$*

$$\|\boldsymbol{\phi}(T; \mathbf{x}, \mathbf{p}) - (\mathbf{f}_E^{NN}(\cdot, \cdot))^K(\mathbf{x}, \mathbf{p})\|_\infty \leq C_2^{\mathbf{f}}(T)\left(\epsilon^E(h) + \epsilon^{NN}(W)\right), \tag{11}$$

where $C_2^{\mathbf{f}}(T) = \frac{e^{LT}-1}{L}$ with L being a Lipschitz constant of $\mathbf{f}$, $h = T/K$, $\epsilon^E(h)$ is the truncation error of the Euler method satisfying $\lim_{h\to 0} \epsilon^E(h) = 0$ and

$$\epsilon^{NN}(W) = C_1 L^{\mathbf{f}}_{max} R^{\mathbf{f}}_{max} S^{\mathbf{f}}_{max} \left|\mathcal{V}_G^{\mathbf{f}}\right|^{1+2/d^{\mathbf{f}}_{max}} W^{-2/d^{\mathbf{f}}_{max}}$$

is the error of the ReLU network approximation of $\mathbf{f}$ defined in Theorem 2. The complexity of the ReLU network $(\mathbf{x}, \mathbf{p}) \to (\mathbf{f}_E^{NN}(\cdot, \cdot))^K(\mathbf{x}, \mathbf{p})$ is KW.

Proof. Based on Theorem 2, there exists a ReLU network, $\mathbf{f}^{NN}$ of size W, such that for all $(\mathbf{x}, \mathbf{p}) \in [-R, R]^{d+p}$

$$\left\|\mathbf{f}(\mathbf{x}, \mathbf{p}) - \mathbf{f}^{NN}(\mathbf{x}, \mathbf{p})\right\|_\infty \leq C_1 L^{\mathbf{f}}_{max} R^{\mathbf{f}}_{max} S^{\mathbf{f}}_{max} \left|\mathcal{V}_G^{\mathbf{f}}\right|^{1+2/d^{\mathbf{f}}_{max}} W^{-2/d^{\mathbf{f}}_{max}}$$

for some constant C_1 depending on input dimension $d_{i,j}$ of each node $f_{i,j}$. Let $\mathbf{f}_E^{NN}$ be the neural network obtained by substituting $\mathbf{f}^{NN}$ for $\mathbf{f}$ in (10). Then,

$$\begin{aligned}\left\|\mathbf{f}_E(\mathbf{x}, \mathbf{p}) - \mathbf{f}_E^{NN}(\mathbf{x}, \mathbf{p})\right\|_\infty &= h\left\|\mathbf{f}(\mathbf{x}, \mathbf{p}) - \mathbf{f}^{NN}(\mathbf{x}, \mathbf{p})\right\|_\infty \\ &\leq hC_1 L^{\mathbf{f}}_{max} R^{\mathbf{f}}_{max} S^{\mathbf{f}}_{max} \left|\mathcal{V}_G^{\mathbf{f}}\right|^{1+2/d^{\mathbf{f}}_{max}} W^{-2/d^{\mathbf{f}}_{max}}.\end{aligned}$$

Applying Proposition 3.10 in [19], we have

$$\begin{aligned}&\left\|(\mathbf{f}_E(\cdot, \cdot))^K(\mathbf{x}, \mathbf{p}) - (\mathbf{f}_E^{NN}(\cdot, \cdot))^K(\mathbf{x}, \mathbf{p})\right\|_\infty \\ &\quad\leq \frac{(1+hL)^K - 1}{(1+hL) - 1} hC_1 L^{\mathbf{f}}_{max} R^{\mathbf{f}}_{max} S^{\mathbf{f}}_{max} \left|\mathcal{V}_G^{\mathbf{f}}\right|^{1+2/d^{\mathbf{f}}_{max}} W^{-2/d^{\mathbf{f}}_{max}} \\ &\quad\leq \frac{e^{KhL} - 1}{L} C_1 L^{\mathbf{f}}_{max} R^{\mathbf{f}}_{max} S^{\mathbf{f}}_{max} \left|\mathcal{V}_G^{\mathbf{f}}\right|^{1+2/d^{\mathbf{f}}_{max}} W^{-2/d^{\mathbf{f}}_{max}} \\ &\quad= C_2^{\mathbf{f}}(T) C_1 L^{\mathbf{f}}_{max} R^{\mathbf{f}}_{max} S^{\mathbf{f}}_{max} \left|\mathcal{V}_G^{\mathbf{f}}\right|^{1+2/d^{\mathbf{f}}_{max}} W^{-2/d^{\mathbf{f}}_{max}}.\end{aligned} \tag{12}$$

On the other hand, it is well known that the global approximation error of the forward Euler method is (see, for instance, [40])

$$\|\boldsymbol{\phi}(T; \mathbf{x}, \mathbf{p}) - (\mathbf{f}_E(\cdot))^K(\mathbf{x}, \mathbf{p})\|_\infty \leq C_2^{\mathbf{f}}(T)\epsilon^E(h), \tag{13}$$

where the truncation error $\epsilon^E(h) \to 0$ as $h \to 0$. Based on the triangle inequality and (12)–(13),

$$\begin{aligned}&\|\boldsymbol{\phi}(T; \mathbf{x}, \mathbf{p}) - (\mathbf{f}_E^{NN}(\cdot, \cdot))^K(\mathbf{x}, \mathbf{p})\|_\infty \\ &\quad\leq \|\boldsymbol{\phi}(T; \mathbf{x}, \mathbf{p}) - (\mathbf{f}_E(\cdot, \cdot))^K(\mathbf{x}, \mathbf{p})\|_\infty \\ &\qquad+ \|(\mathbf{f}_E(\cdot, \cdot))^K(\mathbf{x}, \mathbf{p}) - (\mathbf{f}_E^{NN}(\cdot, \cdot))^K(\mathbf{x}, \mathbf{p})\|_\infty \\ &\quad\leq C_2^{\mathbf{f}}(T)\epsilon^E(h) + C_2^{\mathbf{f}}(T)\epsilon^{NN}(W),\end{aligned}$$

where $\epsilon^{NN}(W) = C_1 L^{\mathbf{f}}_{max} R^{\mathbf{f}}_{max} S^{\mathbf{f}}_{max} \left|\mathcal{V}_G^{\mathbf{f}}\right|^{1+2/d^{\mathbf{f}}_{max}} W^{-2/d^{\mathbf{f}}_{max}}$. ♦

Remark 3. We would like to emphasize that Theorem 3 is about the existence of DNNs whose error upper bound does not suffer from the curse of dimensionality. This theorem does not facilitate a practical algorithm of finding the DNN. The Euler method for ODEs is a convenient discretization for the purpose of proving the existence theorem. However, designing a DNN strictly following the compositional structure of the Euler algorithm may require an unnecessarily large number of layers.

5. Charactering the domain of attraction

In this section we apply the approximation results in Theorem 3 to analyze the domain of attraction of a dynamical system. To this end, we first review a key result from [41], where the domain of attraction is characterized by an explicit construction of a solution to Zubov's equation [42].

Consider a system of ODEs

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \quad \mathbf{x} \in \mathbb{R}^d. \tag{14}$$

Its solution is denoted by $\boldsymbol{\phi}(t, \mathbf{x})$, which satisfies the initial condition $\boldsymbol{\phi}(0, \mathbf{x}) = \mathbf{x}$. Let $\mathcal{E} \subset \mathbb{R}^d$ be a closed invariant subset of the system. Given a positive number δ, the δ-neighborhood of $\mathcal{E}$ is defined by

$$B_\delta(\mathcal{E}) = \{\mathbf{x} \in \mathbb{R}^n;\ d(\mathbf{x}, \mathcal{E}) < \delta\},$$

where $d(\cdot, \cdot)$ represents the distance under Euclidean norm. Although the results in [41] hold true for any uniformly asymptotically stable system, for the sake of simplicity we assume that $\mathcal{E}$ is exponentially stable in a neighborhood, i.e., there exist $\delta > 0$, $M > 0$ and $\lambda > 0$ such that

$$d(\boldsymbol{\phi}(t, \mathbf{x}), \mathcal{E}) \leq Md(\mathbf{x}, \mathcal{E})e^{-\lambda t}$$

for arbitrary $\mathbf{x} \in B_\delta(\mathcal{E})$.

The construction of a solution to Zubov's equation, which is also a Lyapunov function, needs a continuous function, $U : \mathbb{R}^n \to \mathbb{R}$, satisfying

$$U(\mathbf{x}) = 0, \quad \text{if } \mathbf{x} \in \mathcal{E}, \tag{15}$$

$$U(\mathbf{x}) > 0, \quad \text{otherwise}, \tag{16}$$

$$\lim_{d(\mathbf{x}, \mathcal{E}) \to 0} U(\mathbf{x}) = 0. \tag{17}$$

In addition, for any $\delta > 0$, there exists a $\gamma > 0$ such that

$$U(\mathbf{x}) > \gamma, \tag{18}$$

whenever $d(\mathbf{x}, \mathcal{E}) > \delta$. The following system is called the augmented system of (14),

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t)), \quad \mathbf{x}(0) \in \mathbb{R}^d, \\ \dot{z} &= U(\mathbf{x}), \quad z(0) = 0, \end{aligned} \tag{19}$$

where the initial condition of z is always zero. Given any initial state, $\mathbf{x}$, the last variable in the solution of (19) is denoted by $z(t, \mathbf{x})$. Its limit as $t \to \infty$ is denoted by $z(\infty, \mathbf{x})$. The right hand side of (19) is denoted by $\bar{\mathbf{f}}(\mathbf{x})$.

Theorem 4 (*[41]*). *Suppose that $\mathcal{E}$ is an exponentially stable and closed invariant set of (14). Assume that $\mathbf{f}(\mathbf{x})$ is locally Lipschitz on $\mathbb{R}^n$ and $\|\mathbf{f}(\mathbf{x})\|$ is bounded on $B_\delta(\mathcal{E})$ for some $\delta > 0$. Suppose $U(\mathbf{x})$ is Lipschitz on $B_\delta(\mathcal{E})$ satisfying (15)–(18). Then, for any constant $\alpha > 0$,*

$$V(\mathbf{x}) = \begin{cases} \tanh(\alpha z(\infty, \mathbf{x})), & \text{if } z(\infty, \mathbf{x}) < \infty, \\ 1, & \text{otherwise}, \end{cases} \tag{20}$$

is a Lyapunov function that characterizes the domain of attraction, i.e., $\mathbf{x}$ is in the domain of attraction of $\mathcal{E}$ if and only if $V(\mathbf{x}) < 1$.

Evaluating the function defined in (20) requires numerically integrating the system of ODEs (19). This can be computationally too expensive for many real-time applications. However, as pointed out in [41], one can generate data by solving (20) and then train a DNN that approximates $V(\mathbf{x})$. In this section, we study the complexity of the DNN. For a given bounded set $D^{\mathbf{f}} \subset \mathbb{R}^d$, the Lyapunov function can be approximated by

$$\bar{V}(\mathbf{x}) = \tanh(\alpha z(T, \mathbf{x}))$$

for some $T > 0$. The approximation error approaches zero as $T \to \infty$. In the following, we approximate $\bar{V}(\mathbf{x})$ using a DNN that is a ReLU network except that the output layer consists of a single activation function, $\tanh(\cdot)$, i.e.,

$$\begin{aligned} V^{NN} &= \tanh(\alpha z^{NN}) \\ z^{NN} &= \text{the output of a ReLU network.} \end{aligned} \tag{21}$$

Theorem 5. *Consider the ODE in (14) and its augmented system (19). Suppose the right hand side in (19) is a compositional function, $(\bar{\mathbf{f}}, \mathcal{G}^{\bar{\mathbf{f}}})$, satisfying Assumption 1. Assume that $\mathbf{f}$ and U satisfy the same assumptions as in Theorem 4. For any bounded set $D^{\mathbf{f}} \subset \mathbb{R}^d$, there exists a DNN, V^{NN}, in the form of (21) satisfying*

$$\|\bar{V}(\mathbf{x}) - V^{NN}(\mathbf{x})\|_\infty \le (1 - \tanh^2(\alpha\tilde{z}))C_2^{\bar{\mathbf{f}}}(T)\left(\epsilon^E(h) + \epsilon^{NN}(W)\right), \tag{22}$$

where $\tilde{z} = \min\{\left|z^{NN}\right|, |z(T, \mathbf{x})|\}$. The complexity of the neural network is KW. In (22), $C_2^{\bar{\mathbf{f}}}(T) = \frac{e^{LT}-1}{L}$ with L being a Lipschitz constant of $\bar{\mathbf{f}}$, $h = \frac{T}{K}$, $\epsilon^E(h)$ is the truncation error of the Euler method satisfying $\lim_{h\to 0}\epsilon^E(h) = 0$, and

$$\epsilon^{NN}(W) = C_1 L_{max}^{\bar{\mathbf{f}}} R_{max}^{\bar{\mathbf{f}}} S_{max}^{\bar{\mathbf{f}}} \left|\mathcal{V}_G^{\bar{\mathbf{f}}}\right|^{1+2/d_{max}^{\bar{\mathbf{f}}}} W^{-2/d_{max}^{\bar{\mathbf{f}}}}, \tag{23}$$

where the constant C_1 depends on the input dimensions of the general nodes in $\mathcal{G}^{\bar{\mathbf{f}}}$.

Proof. Let $z^{NN} : \mathbf{x} \to \mathbb{R}$ be a ReLU network, then

$$\begin{aligned} \|\bar{V}(\mathbf{x}) - V^{NN}(\mathbf{x})\|_\infty &= \|\tanh(\alpha z(T, \mathbf{x})) - \tanh(\alpha z^{NN}(\mathbf{x}))\|_\infty \\ &= \alpha\left|\tanh'(\alpha\xi)\right| \|z(T, \mathbf{x}) - z^{NN}(\mathbf{x})\|_\infty \end{aligned}$$

for some ξ between $z(T, \mathbf{x})$ and z^{NN}. Because $\tanh'(x) = 1 - \tanh^2(x)$ and because $\tanh(z)$ is an increasing function, we know

$$\left|\tanh'(\alpha\xi)\right| \le 1 - \tanh^2(\alpha\tilde{z}),$$

where $\tilde{z} = \min\{\left|z^{NN}\right|, |z(T, \mathbf{x})|\}$. Therefore,

$$\|\bar{V}(\mathbf{x}) - V^{NN}(\mathbf{x})\|_\infty \le \alpha(1 - \tanh^2(\alpha\tilde{z}))\|z(T, \mathbf{x}) - z^{NN}(\mathbf{x})\|_\infty. \tag{24}$$

Because $\bar{\mathbf{f}}$ of the augmented system satisfies the assumptions in Theorem 3, there exists a ReLU network, z^{NN}, that approximates $z(T, \mathbf{x})$ with an error upper bound (11). Substituting $\|z(T, \mathbf{x}) - z^{NN}(\mathbf{x})\|_\infty$ in (24) by this error upper bound yields (22). ♦

Remark 4. DNNs are trained through solving an optimization problem. In supervised learning, a typical data set has the following form

$$\{(\mathbf{x}_k, \mathbf{f}(\mathbf{x}_k));\ k = 1, 2, \ldots, N_{sample}\},$$

where N_{sample} is the number of samples and $\mathbf{f}$: $\mathbb{R}^d \to \mathbb{R}$ is a function whose value is to be predicted by a DNN. Let $f^{NN}(\theta, \mathbf{x})$ be a neural network in which θ is a vector of parameters to be determined through training. A widely used loss function is

$$l(\theta) = \sum_{i=k}^{N_{sample}} \left(\mathbf{f}(\mathbf{x}_k) - \mathbf{f}^{NN}(\mathbf{x}_k)\right)^2 + \lambda\|\theta\|_2^2. \tag{25}$$

A DNN is trained by finding a θ that minimizes the loss function. There exists a variety of optimization algorithms that are widely used in machine learning. In Example 1, we adopt a BFGS algorithm [43].

In the next, we introduce an example to validate the existence of a DNN that has a $O(W^{-2/d_{max}^{\bar{\mathbf{f}}}})$ error upper bound as implied by (23).

Example 1. Consider the following system of ODEs in $\mathbb{R}^2$,

$$\dot{\mathbf{x}} = \begin{bmatrix} \dfrac{\partial V}{\partial x_2} - (1 - V(\mathbf{x}))\dfrac{\partial V}{\partial x_1} \\ -\dfrac{\partial V}{\partial x_1} - (1 - V(\mathbf{x}))\dfrac{\partial V}{\partial x_2} \end{bmatrix}, \tag{26}$$

where

$$V(\mathbf{x}) = \begin{cases} x_1^{\frac{4}{3}} + x_2^{\frac{4}{3}} - \frac{1}{2}\tanh^2(x_1 + x_2), & \text{if } V(\mathbf{x}) < 1, \\ 1, & \text{otherwise.} \end{cases} \tag{27}$$

It can be shown that $V(\mathbf{x})$ is a Lyapunov function that fulfills the assumptions and properties in Theorem 4. The function characterizes the domain of attraction around $\mathbf{x} = 0$, i.e., the trajectory $\mathbf{x}(t)$ with initial condition $\mathbf{x}(0) = \mathbf{x}_0$ converges to the origin if and only if $V(\mathbf{x}_0) < 1$. The graph of $V(\mathbf{x})$ and the boundary of the domain of attraction are shown in Fig. 4. The functions in (26) are C^0 but not C^1. The domain of attraction is non-convex. The Lyapunov function is C^1 but not C^2.

The functions in (26) and (27) have simple compositional structures. The nonlinear nodes include $x_i^{\frac{4}{3}}$, $x_i^{\frac{1}{3}}$, $\tanh(z)$ ($z = x_1 + x_2$), and $\tanh^2(z)$. They all have a single input, which implies $d_{max}^{\bar{\mathbf{f}}} = 1$. From Theorem 5, there exists a sequence of DNNs that approximates $V(\mathbf{x})$. According to (23), we expect that the error of the DNN approximation is $O(W^{-2})$. For numerical validation, we train DNNs that approximate every nonlinear node in $V(\mathbf{x})$. Each DNN has $1 \le L \le 9$ layers and twelve neurons in each layer. Because the error upper bound in Theorem 5 is based on L^∞-norm, which is generally more difficulty to control than 2-norm in machine learning, the DNN approximation is further corrected by a single layer NN that has $n \le 225$ neurons. The complexity of the DNNs, W, is linearly dependent on $L = 1, 2, \ldots, 9$. On a data set that contains 10^4 samples, the maximum estimation error, defined as

$$\max\{|\bar{V}(\mathbf{x}) - V^{NN}(\mathbf{x})|;\ \mathbf{x} \in \text{data set}\}, \tag{28}$$

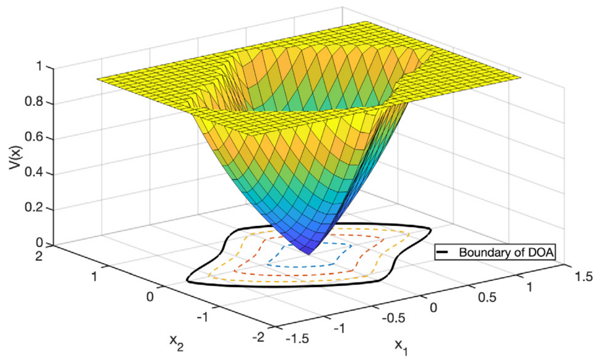

Fig. 4. The graph of $V(\mathbf{x})$ and the domain of attraction (DOA).

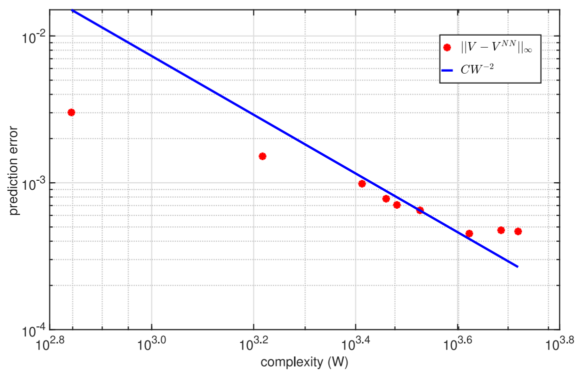

Fig. 5. The estimation error of $V^{NN}(\mathbf{x})$. The line is the graph of CW^{-2}. The dots are the error of $V^{NN}(\mathbf{x})$ for $L = 1, 2, \ldots, 9$.

is shown in Fig. 5. The error follows $O(W^{-2})$ closely when the number of layers is $L \leq 6$, or equivalently $W < 10^{3.6}$. When the complexity of the DNN increases beyond this point, the trend of decreasing error slows down.

This example is a numerical validation of the error upper bound in Theorem 5. The DNN design is based on an impractical assumption that the compositional structure and nonlinear nodes in $V(\mathbf{x})$ are known. How to find DNNs with guaranteed error bounded by (22)–(23), in general, is an open problem. This is not limited to the L^∞-norm in this paper. Finding DNNs that have guaranteed error upper bounds in L^p-space is widely recognized as an open problem in many theoretical studies and machine learning applications.

6. Conclusions

Applying an algebraic framework developed for the error analysis of approximating compositional functions, we prove three theorems on the neural network complexity in the approximation of functions, control system trajectories, and a Lyapunov function that characterizes the domain of attraction. Using conventional approximation methods such as a polynomial interpolation or a spectrum method, these problems suffer from the curse-of-dimensionality. The approximation complexity increases exponentially with the state space dimension. The main contributions of this paper, Theorems 2, 3 and 5, reveal that neural networks have an approximation error upper bound that is a polynomial of the network's complexity and the compositional features. If the compositional features do not increase exponentially with dimension, which is the case in all examples that we have studied in several published papers, the complexity of DNN has a polynomial growth. These results may, to some extend, explain the large number of empirical successes of deep learning in solving high dimensional problems of optimal control and PDEs. Although we focus on ReLU networks in all theorems, the properties of compositional features and the error estimation method for DAGs are applicable to other types of networks, provided that an error upper bound in the form of Theorem 1 can be proved.

The goal of the paper is to prove that deep neural networks can avoid an exponentially increasing complexity, which is an insuperable obstacle for conventional approximation methods for problems that have moderate or high dimensions. The error upper bounds proved in this paper could be conservative. For instance, we use the largest value of compositional features in the error upper bound, rather than the value associated with individual nodes. For systems of ODEs, the constructive proof is based on the Euler method, which has a low convergence rate. As a result, the ReLU network has many layers. In addition, the proof of the existence of a ReLU network does not mean it is easy to find the network. Training and validating DNNs taking advantage of the compositional features as well as their layered DAGs are some important and mathematically interesting open problems for next step research.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Appendix

The algebraic framework in [19] developed for compositional functions facilitate error analysis for regression problems that involves operations such as substitutions and iterative computation. Two of the propositions in [19] are used in the proof of theorems in this paper. They are given below without proof.

Proposition 1 (*Error Caused by Node Substitution*)**.** *Let $(\mathbf{f}, \mathcal{G}^{\mathbf{f}})$ be a compositional function. Let $\{h_1, h_2, \ldots, h_K\} \subseteq \mathcal{V}^{\mathbf{f}} \setminus \mathcal{V}_I^{\mathbf{f}}$ be a set of nodes. Under a p-norm, let $s_j^{\mathbf{f}} > 0$ be the sensitivity associated with h_j, $1 \leq j \leq K$. Suppose $\tilde{h}_j$, $1 \leq j \leq K$, is a set of functions in which $\tilde{h}_j$ has the same domain as h_j and a compatible range for substitution. Assume that*

$$\left|\tilde{h}_j(\mathbf{w}) - h_j(\mathbf{w})\right| \leq \epsilon_j, \quad \text{for all } \mathbf{w} \text{ in the domain of } h_j, \tag{29}$$

where ϵ_j, $j = 1, 2, \ldots, K$, are some positive numbers. Let $\tilde{\mathbf{f}}$ be the function obtained by substituting $\tilde{h}_j$ for h_j, $j = 1, 2, \ldots, K$. Then

$$\left\|\tilde{\mathbf{f}}(\mathbf{x}) - \mathbf{f}(\mathbf{x})\right\|_p \leq \sum_{j=1}^K L_j^{\mathbf{f}} \epsilon_j, \quad \text{for all } x \text{ in the domain of } \mathbf{f}. \tag{30}$$

Proposition 2 (*Error Propagation Through Composition*)**.** *Consider compositional functions $(\mathbf{f}, \mathcal{G}^{\mathbf{f}})$, $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$, $(\mathbf{g}, \mathcal{G}^{\mathbf{g}})$, $\mathbf{g} : \mathbb{R}^q \to \mathbb{R}^d$, and $(\mathbf{h}, \mathcal{G}^{\mathbf{h}})$, $\mathbf{h} : \mathbb{R}^d \to \mathbb{R}^q$. Suppose that the domains and ranges of them are compatible for compositions. Let $L^{\mathbf{f}}$ and $L^{\mathbf{h}}$ be Lipschitz constants of $\mathbf{f}$ and $\mathbf{h}$ under a p-norm. Suppose $\tilde{\mathbf{f}}$, $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ are functions satisfying*

$$\left\|\mathbf{f}(\mathbf{x}) - \tilde{\mathbf{f}}(\mathbf{x})\right\|_p \leq e_1, \quad \left\|\mathbf{g}(\mathbf{x}) - \tilde{\mathbf{g}}(\mathbf{x})\right\|_p \leq e_2, \quad \left\|\mathbf{h}(\mathbf{x}) - \tilde{\mathbf{h}}(\mathbf{x})\right\|_p \leq e_3,$$

for some $e_1 > 0$, $e_2 > 0$ *and* $e_3 > 0$ *and all* $\mathbf{x}$ *in corresponding domains. Given any integer* $K > 0$, *we have*

$$\begin{aligned} \left\|(\mathbf{f}(\cdot))^K \circ \mathbf{g}(\mathbf{x}) - (\tilde{\mathbf{f}}(\cdot))^K \circ \tilde{\mathbf{g}}(\mathbf{x})\right\|_p &\le \frac{(L^{\mathbf{f}})^K - 1}{L^{\mathbf{f}} - 1} e_1 + (L^{\mathbf{f}})^K e_2. \\ \left\|\mathbf{h} \circ (\mathbf{f}(\cdot))^K(\mathbf{x}) - \tilde{\mathbf{h}} \circ (\tilde{\mathbf{f}}(\cdot))^K(\mathbf{x})\right\|_p &\le L^{\mathbf{h}} \frac{(L^{\mathbf{f}})^K - 1}{L^{\mathbf{f}} - 1} e_1 + e_3. \end{aligned} \tag{31}$$

where $(\mathbf{f}(\cdot))^k(\mathbf{x}) := \mathbf{f} \circ \mathbf{f} \circ \cdots \circ \mathbf{f}(\mathbf{x})$.

References

[1] R.E. Bellman, Dynamic Programming, Rand Corporation Research Study, Princeton University Press, 1957.
[2] J. Darbon, G.P. Langlois, T. Meng, Overcoming the curse of dimensionality for some Hamilton–Jacobi partial differential equations via neural network architectures, Res. Math. Sci. 7 (2020).
[3] B. Hamzi, H. Owhadi, Learning dynamical systems from data: a simple cross-validation perspective, part I: Parametric kernel flows, Physica D 421 (2021).
[4] L. Grüne, Overcoming the curse of dimensionality for approximating Lyapunov functions with deep neural networks under a small-gain condition, IFAC-PapersOnline 31 (9) (2021) 317–322.
[5] D. Kalise, K. Kunisch, Polynomial approximation of high-dimensional Hamilton-Jacobi-Bellman equations and applications to feedback control of semilinear parabolic PDEs, SIAM J. Sci. Comput. 40 (2) (2018) A629–A652.
[6] J. Han, A. Jentzen, E. Weinan, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. USA 115 (34) (2018) 8505–8510.
[7] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[8] C. Sánchez-Sánchez, D. Izzo, Real-time optimal control via deep neural networks: Study on landing problems, J. Guid. Control, Dyna. 41 (5) (2018) 1122–1135.
[9] D. Tailor, D. Izzo, Learning the optimal state-feedback via supervised imitation learning, Astrodynamics 3 (4) (2019) 361–374.
[10] T. Nakamura-Zimmerer, Q. Gong, W. Kang, Adaptive deep learning for high-dimensional Hamilton–Jacobi–Bellman equations, SIAM J. Sci. Comput. 43 (2) (2021) A1221–A1247.
[11] B. Azmi, D. Kalise, K. Kunisch, Optimal feedback law recovery by gradient-augmented sparse polynomial regression, J. Mach. Learn. Res. 22 (48) (2021) 1–32.
[12] D. Izzo, E. Öztürk, Real-time guidance for low-thrust transfers using deep neural networks, J. Guid. Control, Dyna. (2021) 1–13.
[13] T. Nakamura-Zimmerer, Q. Gong, W. Kang, QRnet: Optimal regulator design with LQR-augmented neural networks, IEEE Control Syst. Lett. 5 (4) (2021) 1303–1308.
[14] D. Onken, L. Nurbekyan, X. Li, S.W. Fung, S. Osher, L. Ruthotto, A neural network approach for high-dimensional optimal control, 2021, [Online]. Available: arXiv:2104.03270.
[15] K. Kunisch, D. Walter, Semiglobal optimal feedback stabilization of autonomous systems via deep neural network approximation, ESAIM Control Optim. Calc. Var. 27 (16) (2021) 59.
[16] G. Cybenko, Approximation by superposition of a sigmoidal function, Math. Control Signals Systems 2 (4) (1989) 303–314.
[17] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
[18] J. Han, J. Arnulf, E. Weinan, Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning, 2017, arXiv preprint, arXiv:1707.02568.
[19] W. Kang, Q. Gong, Feedforward neural network and compositional functions with applications to dynamical systems, SIAM J. Control Optim. 60 (2) (2022) 786–813.
[20] A.R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Trans. Inform. Theory 39 (3) (1993) 930–945.
[21] D. Yarotsky, Optimal approximation of continuous functions by very deep ReLU network, in: Conference on Learning Theory, PMLR, 2018, pp. 639–649.
[22] H. Montanelli, H. Yang, Error bounds for deep ReLU networks using the Kolmogorov–Arnold superposition theorem, Neural Netw. 129 (2020) 1–6.
[23] E. Weinan, S. Wojtowytsch, On the Banach spaces associated with multi-layer ReLU networks: function representation, approximation theory and gradient descent dynamics, CSIAM Trans. Appl. Math. 1 (3) (2020) 387–440.
[24] P.C. Kainen, V. Kůrková, M. Sanguineti, Approximating multivariable functions by feedforward neural nets, in: M. Bianchini, M. Maggini, L. Jain (Eds.), Hanbook on Neural Information Processing, Intelligent Systems Reference Library, vol. 49, Springer, 2013, pp. 143–181.
[25] H.N. Mhaskar, T. Poggio, Deep vs. shallow networks: an approximation theory perspective, Anal. Appl. 14 (6) (2016) 829–848.
[26] H.N. Mhaskar, T. Poggio, Function Approximation By Deep Networks, vol. 19, (8) American Institute of Mathematics, 2020, http://dx.doi.org/10.3934/cpaa.2020181.
[27] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, Q. Liao, Why and when can deep - but not shallow - networks avoid the curse of dimensionality: a review, Int. J. Autom. Comput. 14 (2017) 503–519.
[28] R. Sepulchre, M. Jankovic, P.V. Kokotovic, Constructive Nonlinear Control, Springer, NewYork, 1997.
[29] A. Isidori, Nonlinear control systems, in: Communications and Control Engineering Series, third ed., Springer-Verlag, Berlin, 1995.
[30] C. Qian, W. Lin, W.P. Dayawansa, Smooth feedback, global stabilization and disturbance attenuation of nonlinear systems with uncontrollable linearization, SIAM J. Control Optim. 40 (1) (2001) 191–210.
[31] C. Qian, W. Lin, A continuous feedback approach to global strong stabilization of nonlinear systems, IEEE Trans. Automat. Control 46 (7) (2001) 1061–1079.
[32] J. Polendo, C. Qian, An expanded method to robustly stabilize uncertain nonlinear systems, Commun. Inf. Syst. 8 (1) (2008) 55–70.
[33] L. Praly, An introduction to forwarding, in: Control of Complex Systems, Springer, 2001, pp. 77–99.
[34] A.R. Teel, Global stabilization and restricted tracking for multiple integrators with bounded controls, Systems Control Lett. 18 (3) (1992) 165–171.
[35] E. Lorenz, Predictability – a problem partly solved, in: ECMWF Seminar on Predictability, 1996.
[36] T. Athay, R. Podmore, S. Virmani, A practical method for the direct analysis of transient stability, IEEE Trans. Power Appar. Syst. 98 (1979) 573–584.
[37] P.M. Anderson, A.A. Fouad, Power System Control and Stability, John Wiley & Sons, 2008.
[38] J. Qi, K. Sun, W. Kang, Optimal PMU placement for power system dynamic state estimation by using empirical observability gramian, IEEE Trans. Power Syst. 30 (4) (2014) 2041–2054.
[39] D. Yarotsky, Error bounds for approximations with deep ReLU network, Neural Netw. 94 (10) (2017) 103–114.
[40] E. Süli, D.F. Mayers, An Introduction To Numerical Analysis, Cambridge University Press, 2003.
[41] W. Kang, K. Sun, L. Xu, Data-driven computational methods for the domain of attraction and Zubov's Equation, 2021, arXiv:2112.14415v1.
[42] V.I. Zubov, Methods of A. M. Lyapunov and their Application, P Noordhoff LTD, Groningen, The Netherlands, 1964.
[43] E. Polak, Optimization - Algorithms and Consistent Approximations, Springer-Verlag, New York, 1997.