LETTER Communicated by Bo Chang

State-Space Representations of Deep Neural Networks

Michael Hauser
mikebenh@gmail.com
Sean Gunn
sug375@psu.edu
Department of Mechanical Engineering, Pennsylvania State University, University Park, PA 16802, U.S.A.

Samer Saab Jr.
sys5880@psu.edu
Department of Electrical Engineering, Pennsylvania State University, University Park, PA 16802, U.S.A.

Asok Ray
axr2@psu.edu
Department of Mechanical Engineering, Pennsylvania State University, University Park, PA 16802, U.S.A.

This letter deals with neural networks as dynamical systems governed by finite difference equations. It shows that the introduction of k-many skip connections into network architectures, such as residual networks and additive dense networks, defines kth order dynamical equations on the layer-wise transformations. Closed-form solutions for the state-space representations of general kth order additive dense networks, where the concatenation operation is replaced by addition, as well as kth order smooth networks, are found. The developed provision endows deep neural networks with an algebraic structure. Furthermore, it is shown that imposing kth order smoothness on network architectures with d-many nodes per layer increases the state-space dimension by a multiple of k, and so the effective embedding dimension of the data manifold by the neural network is $k \cdot d$-many dimensions. It follows that network architectures of these types reduce the number of parameters needed to maintain the same embedding dimension by a factor of k^2 when compared to an equivalent first-order, residual network. Numerical simulations and experiments on CIFAR10, SVHN, and MNIST have been conducted to help understand the developed theory and efficacy of the proposed concepts.

Neural Computation **31, 538–554** (2019)
doi:10.1162/neco_a_01165

1 Introduction

The way in which deep learning was initially used to transform data representations was by nested compositions of affine transformations followed by nonlinear activations. The affine transformation can be, for example, a fully connected weight matrix or convolution operation. Residual networks (He, Zhang, Ren, & Sun, 2016) introduce an identity skip connection that bypasses these transformations, thus allowing the nonlinear activation to act as a perturbation term from the identity. Veit, Wilber, and Belongie (2016) introduced an algebraic structure showing that residual networks can be understood as the entire collection of all possible forward pass paths of subnetworks, although this algebraic structure ignores the intuition that the nonlinear activation is acting as a perturbation from identity. Lin and Jegelka (2018) showed that a residual network with a single node per layer and ReLU activation can act as a universal approximator, where it is learning something similar to a piecewise linear finite-mesh approximation of the data manifold.

Recent work consistent with the original intuition of learning perturbations from the identity has shown that residual networks, with their first-order perturbation terms, can be formulated as finite difference approximations of first-order differential equations (Hauser & Ray, 2017). This has the interesting consequence that residual networks are $\mathcal{C}^1$ smooth dynamic equations through the layers of the network. In addition, one may then define entire classes of $\mathcal{C}^k$ differentiable transformations over the layers and then induce network architectures from their finite difference approximations.

Chang, Meng, Haber, Tung, and Begert (2017) considered residual neural networks as forward-difference approximations to $\mathcal{C}^1$ transformations as well. This work has been extended to develop new network architectures by using central differencing, as opposed to forward differencing, to approximate the set of coupled first-order differential equations, called the midpoint network (Chang, Meng, Haber, Ruthotto et al., 2017). Similarly, other researchers have used different numerical schemes to approximate the first-order ordinary differential equations, such as the linear multistep method to develop the linear multistep-architecture (Lu, Zhong, Li, & Dong, 2017). This is different from previous work (Hauser & Ray, 2017), where entire classes of finite-differencing approximations to kth order differential equations are defined. Haber and Ruthutto (2017) considered how stability techniques from finite difference methods can be applied to improve first- and second-order smooth neural networks. For example, they suggest requiring that the real part of the eigenvalues from the Jacobian transformations be approximately equal to zero. This ensures that little information about the signal is lost and that the input data not diverge in progressing through the network.

In current work set out in section 2, closed-form solutions are found for the state-space representations for both general $\mathcal{C}^k$ network architectures as well as general additive densely connected network architectures (Huang, Liu, Weinberger, & van der Maaten, 2017), where a summation operation replaces the concatenation operation. The reason for this is that the concatenation operation explicitly increases the embedding dimension, while the summation operation implicitly increases the embedding dimension. We then show in section 3 that the embedding dimension for a $\mathcal{C}^k$ network is increased by a factor of k when compared to an equivalent $\mathcal{C}^0$ (standard) network and $\mathcal{C}^1$ (residual) network, and thus the number of parameters needed to learn is reduced by a factor of k^2 to maintain transformations on the same embedding dimension. Section 4 presents the results of experiments for validation of the proposed theory, and the details are provided in the appendix. The paper concludes in section 5, along with recommendations for future research.

2 Smooth Network Architectures

This section develops a relation between skip connections in network architectures and algebraic structures of dynamical systems of equations. The network architecture can be thought of as a map $x : M \times I \to \mathbb{R}^d$, where M is the data manifold, $x^{(0)}(M)$ is the set of input data/initial conditions, and I is the set $I = \{0, 1, 2, \ldots, L-1\}$ for an L-layer deep neural network. We write $x^{(l)} : M \to \mathbb{R}^d$ to denote the coordinate representation for the data manifold M at layer $l \in I$. In fact, the manifold is a Riemannian manifold (M, g) as it has the additional structure of possessing a smoothly varying metric g on its cotangent bundle (Hauser & Ray, 2017); however for the current purpose, we will only consider the manifold's structure to be M.

In order to reduce notational burdens, as well as to keep the analysis as general as possible, we will denote the lth-layer nonlinearity as the map $f^{(l)} : x^{(l)} \mapsto f^{(l)}\big(x^{(l)}\big)$ where $x^{(l)}$ is the output of layer l. For example, if it is a fully connected layer with bias and sigmoid nonlinearity, then $f^{(l)}(x^{(l)}) := \sigma\big(W^{(l)} \cdot x^{(l)} + b^{(l)}\big)$, or if it is a convolution block in a residual network, then

$$f^{(l)}\big(x^{(l)}\big) := \mathrm{BN}\big(W_2^{(l)} * \mathrm{LReLU}\big(\mathrm{BN}\big(W_1^{(l)} * x^{(l)}\big)\big)\big),$$

where the $*$ is the convolution operation, $W_1^{(l)}$ and $W_2^{(l)}$ are the learned filter banks, and LReLU and BN are the leaky-ReLU activation and batch-normalization functions. The nonlinear function $f^{(l)}$ can be thought of as a forcing function, from dynamical systems theory.

A standard architecture without skip connections has the following form:

$$x^{(l+1)} = f^{(l)}\big(x^{(l)}\big). \tag{2.1}$$

Section 2.1 defines and reviews smooth $\mathcal{C}^1$ residual (He et al., 2016) architectures. Section 2.2 expands on the section 2.1 to define and study the entire class of $\mathcal{C}^k$ architectures (Hauser and Ray, 2017) and develop the state-space formulation for these architectures to show that the effective embedding dimension increases by a multiple of k for architectures of these types. Similarly, section 2.3 develops the state-space formulation for densely connected networks (Huang et al., 2017) and shows that for these dense networks with k-many layer-wise skip connections, the effective embedding dimension again increases by a multiple of k.

2.1 Residual Networks as Dynamical Equations. The residual network (He et al., 2016) has a single skip connection and is therefore simply a $\mathcal{C}^1$ dynamic transformation:

$$x^{(l+1)} = x^{(l)} + f^{(l)}\big(x^{(l)}\big)\Delta l. \tag{2.2}$$

The term Δl on the right-hand side of equation 2.2 is explicitly introduced here to remind us that this is a perturbation term. The accuracy of this assumption is verified by experiment in section 4.2.

If the equation is defined over $[0, d]$, then the partitioning of the dynamical system (Hauser & Ray, 2017) takes the following form:

$$\mathcal{P} = \big\{0 = l(0) < l(1) < l(2) < \ldots < l(n) < \ldots < l(L-1) = d\big\}, \tag{2.3}$$

where $\Delta l(n) := l(n+1) - l(n)$ can in general vary with n as the $\max_n \Delta l(n)$ still goes to zero as $L \to \infty$. To reduce notation, this letter writes $\Delta l := \Delta l(n)$ for all $n \in \{0, 1, 2, \ldots, L-1\}$. Notations are slightly changed here by taking $l = n\Delta l$ and indexing the layers by the fractional index l instead of the integer index n; however, this is inherent to switching notations between finite difference equations and continuous differential equations.

2.2 Architectures Induced from Smooth Transformations. Following the work of Hauser and Ray (2017), we call network architectures as being $\mathcal{C}^k$ architectures depending on how many times the finite difference operators have been applied to the map $x : M \times I \to \mathbb{R}^d$.

We define the forward and backward finite difference operators to be $\delta^+ : x^{(l)} \mapsto x^{(l+1)} - x^{(l)}$ and $\delta^- : x^{(l)} \mapsto x^{(l)} - x^{(l-1)}$, respectively. Furthermore, to see the various order derivatives of x at layer l, we use these finite difference operators to make the finite difference approximations for $k = 1, 2$ and general $k \in \mathbb{N}$, while explicitly writing the perturbation term in terms of Δl:

$$\delta^+ x^{(l)} = x^{(l+1)} - x^{(l)} = f^{(l)}\big(x^{(l)}\big)\Delta l \qquad \text{for } k = 1, \tag{2.4a}$$

$$\delta^+\delta^- x^{(l)} = x^{(l+1)} - 2x^{(l)} + x^{(l-1)} = f^{(l)}\big(x^{(l)}\big)\Delta l^2 \qquad \text{for } k = 2, \tag{2.4b}$$

$$\delta^+(\delta^-)^{k-1}x^{(l)} = \sum_{l'=0}^{k}\left[(-1)^{l'}\binom{k}{l'}x^{(l+1-l')}\right] = f^{(l)}\big(x^{(l)}\big)\Delta l^k \qquad k \in \mathbb{N}. \tag{2.4c}$$

The notation $(\delta^-)^{k-1} := \delta^-\delta^-\cdots\delta^-$ is defined as $k-1$-many applications of the operator δ^-, and $\binom{k}{l}$ is the binomial coefficient, read as k-choose-l. We take one forward difference and the remaining $k-1$ as backward differences so that the next layer $x^{(l+1)}$ (forward) is a function of the k previous layers $x^{(l)}, x^{(l-1)}, \ldots, x^{(l-k+1)}$ (backward).

From this formulation, depending on the order of smoothness, the network is implicitly creating interior or ghost elements, borrowing language from finite difference methods, to properly define the initial conditions. One can view a ghost element as a pseudo-element that lies outside the domain used to control the gradient. For example with a $k = 2$ architecture from equation 2.4b, one needs the initial position and velocity in order to be able to define $x^{(2)}$ as a function of $x^{(0)}$ and $x^{(1)}$. In the section 2.3, we show that the dense network (Huang et al., 2017) can be interpreted as the interior or ghost elements needed to initialize the dynamical equation.

To see the equivalent state-space formulation of the kth order equation defined by equation 2.4c, first we define the states as the various-order finite differencing of the transformation x at l:

$$q_1^{(l)} := x^{(l)}, \tag{2.5a}$$

$$q_2^{(l)} := \delta^- x^{(l)}, \tag{2.5b}$$

$$q_n^{(l)} := (\delta^-)^{n-1}x^{(l)} \qquad \forall n = 1, 2, \ldots, k. \tag{2.5c}$$

We then have the recursive relation $q_{n+1}^{(l+1)} = q_n^{(l+1)} - q_n^{(l)}$, initialized at the $n = k$ base case $q_k^{(l+1)} - q_k^{(l)} = f^{(l)}\big(q_1^{(l)}\big)\Delta l^k$ from equation 2.4c, as the means to find the closed-form solution by induction. Assuming $q_{n+1}^{(l+1)} = \sum_{l'=n+1}^{k}\big[q_{l'}^{(l)}\big] + f\big(q_1^{(l)}\big)\Delta l^k$, we have the following:

$$\begin{aligned} q_n^{(l+1)} &= q_n^{(l)} + q_{n+1}^{(l+1)} = q_n^{(l)} + \sum_{l'=n+1}^{k}\big[q_{l'}^{(l)}\big] + f\big(q_1^{(l)}\big)\Delta l^k \\ &= \sum_{l'=n}^{k}\big[q_{l'}^{(l)}\big] + f\big(q_1^{(l)}\big)\Delta l^k. \end{aligned} \tag{2.6}$$

The first equality follows from the recursive relation and the second from the base case. This shows that the state-space formulation of the $\mathcal{C}^k$ neural network is given:

$$q_n^{(l+1)} = \sum_{l'=n}^{k} \left[q_{l'}^{(l)}\right] + f\left(q_1^{(l)}\right)\Delta l^k \qquad \forall n = 1, 2, \ldots, k. \tag{2.7}$$

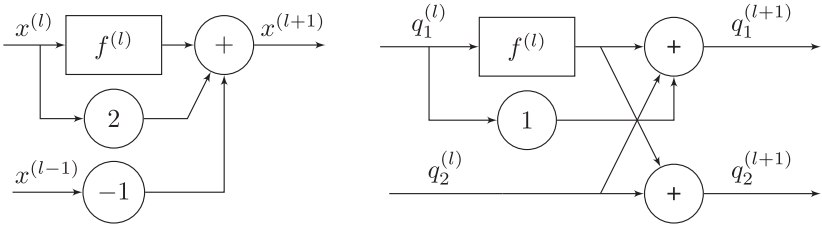
In matrix form, the state-space formulation is as follows:

$$\begin{pmatrix} q_1^{(l+1)} \\ q_2^{(l+1)} \\ q_3^{(l+1)} \\ \vdots \\ q_k^{(l+1)} \end{pmatrix} = \begin{pmatrix} \mathbb{1} & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} \\ \mathbb{0} & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} \\ \mathbb{0} & \mathbb{0} & \mathbb{1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbb{1} \\ \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{1} \end{pmatrix} \cdot \begin{pmatrix} q_1^{(l)} \\ q_2^{(l)} \\ q_3^{(l)} \\ \vdots \\ q_k^{(l)} \end{pmatrix} + \begin{pmatrix} \mathbb{1} & \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} \\ \mathbb{0} & \mathbb{1} & \mathbb{0} & \cdots & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{1} \end{pmatrix} \cdot \begin{pmatrix} f^{(l)}\left(q_1^{(l)}\right) \\ f^{(l)}\left(q_1^{(l)}\right) \\ f^{(l)}\left(q_1^{(l)}\right) \\ \vdots \\ f^{(l)}\left(q_1^{(l)}\right) \end{pmatrix} \Delta l^k. \tag{2.8}$$

We use the notation where $\mathbb{1}$ is the $d \times d$ identity matrix and $\mathbb{0}$ is the $d \times d$ matrix of all zeros. From equation 2.7 and, equivalently, equation 2.8, it is understood that if there are d-many nodes at layer l (i.e., $x^{(l)}$ maps to $\mathbb{R}^d$), then a kth-order smooth neural network can be represented in the state-space form as $q^{(l)} := \left[q_1^{(l)}; q_1^{(l)}; \cdots; q_k^{(l)}\right]$, which maps to $\mathbb{R}^{k \cdot d}$. Furthermore, it is seen that the k-many state variables are transformed by the shared activation function $f^{(l)}$, which has a $(d \times d)$-parameter matrix, as opposed to a full $(k \cdot d \times k \cdot d)$-parameter matrix, thus reducing the number of parameters by a factor of k^2.

The schematic of the $\mathcal{C}^2$ architecture, with its equivalent first-order state-space representation, is given in Figure 1. The $\mathcal{C}^2$ architecture is given by equation 2.4b, which can be conveniently rewritten as $x^{(l+1)} = x^{(l)} + \left(x^{(l)} - x^{(l-1)}\right) + f^{(l)}(x^{(l)})\Delta l^2$. Setting $q_1^{(l)} = x^{(l)}$ and $q_2^{(l)} = x^{(l)} - x^{(l-1)}$, the state-space model is updated as $q_1^{(l+1)} = q_1^{(l)} + q_2^{(l)} + f^{(l)}\left(q_1^{(l)}\right)$ and $q_2^{(l+1)} = q_2^{(l)} + f^{(l)}\left(q_1^{(l)}\right)$. Thus, given $x^{(l)}$ maps to $\mathbb{R}^d$, $q^{(l)} = \left[q_1^{(l)}; q_2^{(l)}\right]$ will map to $\mathbb{R}^{2d}$.

2.3 Additive Dense Network for General $k \in \mathbb{N}$. The additive dense network, which is inspired by the dense network (Huang et al., 2017), is defined for general k by the following system of equations:



(a) A $\mathcal{C}^2$ architecture is a second-order equation.

(b) The equivalent state-space model of the $\mathcal{C}^2$ network.

Figure 1: The block diagram of the $\mathcal{C}^2$ architecture (left), derived from $x^{(l+1)} - 2x^{(l)} + x^{(l-1)} = f^{(l)}(x^{(l)})$, and its equivalent first-order state-space model (right), where $q_1^{(l)} = x^{(l)}$ and $q_2^{(l)} = x^{(l)} - x^{(l-1)}$. It is seen that if the second-order model has d-many nodes (i.e., $x^{(l)}$ maps to $\mathbb{R}^d$), then its state-space representation is $q^{(l)} = [q_1^{(l)}; q_2^{(l)}]$ maps to $\mathbb{R}^{2d}$. The state-space model is updated as $q_1^{(l+1)} = q_1^{(l)} + q_2^{(l)} + f^{(l)}(q_1^{(l)})$ and $q_2^{(l+1)} = q_2^{(l)} + f^{(l)}(q_1^{(l)})$.

$$x^{(l+1-n)} = \sum_{l'=n}^{k-1} \left[f^{(l-l')}\left(x^{(l-l')}\right) \Delta l \right] + x^{(l+1-k)} \quad \forall n = 0, 1, \ldots, k-1. \tag{2.9}$$

To put this into a state-space form, we need to transform this into a system of finite difference equations. The general nth-order difference equation, with one forward difference and all of the remaining backward is used because from a dense network perspective, the value at $l+1$ (forward) is a function of $l, l-1, \ldots, l-n+1$ (backward):

$$\delta^+(\delta^-)^{n-1}x^{(l)} = \sum_{l'=0}^{n} \left[(-1)^{l'} \binom{n}{l'} x^{(l+1-l')} \right] \quad \forall n = 1, 2, \ldots, k. \tag{2.10}$$

Substituting equation 2.9 into equation 2.10 yields the following:

$$\delta^+(\delta^-)^{n-1}x^{(l)} = \sum_{l'=0}^{n} \left[(-1)^{l'} \binom{n}{l'} \left(\sum_{l''=l'}^{k-1} \left[f^{(l-l'')}\left(x^{(l-l')}\right) \Delta l \right] \right) \right]$$
$$\forall n = 1, 2, \ldots, k. \tag{2.11}$$

Notice that we used $\sum_{l'=0}^{n} \left[(-1)^{l'} \binom{n}{l'} x^{(l+1-k)} \right] = \sum_{l'=0}^{n} \left[(-1)^{l'} \binom{n}{l'} \right] x^{(l+1-k)} = 0$. Equation 2.11 is equivalent to the additive dense network formulation

from equation 2.9, but reformulated to a form that lends itself to interpretation using finite differencing. We then define the network states as the various-order finite differences across layers:

$$q_n^{(l)} := (\delta^-)^{n-1} x^{(l)} \qquad \forall n = 1, 2, \ldots, k. \tag{2.12}$$

We still need to find the representations of the $x^{(l-n)}$'s in terms of the states $q_1^{(l)}, q_2^{(l)}, \ldots, q_k^{(l)}$. To do this, we use the property of binomial inversions of sequences (Proctinger, 1993):

$$q_n^{(l)} = \sum_{l'=0}^{n-1} (-1)^{l'} \binom{n-1}{l'} x^{(l-l')} \quad \Rightarrow \quad x^{(l-n)} = \sum_{l'=0}^{n-1} (-1)^{l'} \binom{n-1}{l'} q_n^{(l)}. \tag{2.13}$$

The left-hand side of equation 2.13 is the definition of states from equation 2.12 written explicitly as the $n-1$th backward difference of a sequence $x^{(l)}$, and the implication arrow $\Rightarrow$ is the binomial inversion of sequences. This is the representation of the $x^{(l-n)}$'s in terms of the states $q_1^{(l)}, q_2^{(l)}, \ldots, q_k^{(l)}$.

It is now straightforward to find the state-space representation of the general kth-order dense network:

$$q_n^{(l+1)} = q_n^{(l)} + \sum_{l'=0}^{n} \left[(-1)^{l'} \binom{n}{l'} \left(\sum_{l''=l'}^{k-1} \left[f^{(l-l'')} \left(\sum_{l'''=0}^{l'-1} (-1)^{l'''} \binom{l'-1}{l'''} q_{l'}^{(l)} \right) \Delta l \right] \right) \right]. \tag{2.14}$$

Equation 2.14 is true $\forall n = 1, 2, \ldots, k$, and so it may be clearer when written as a matrix equation:

$$\begin{pmatrix} q_1^{(l+1)} \\ q_2^{(l+1)} \\ q_3^{(l+1)} \\ \vdots \\ q_k^{(l+1)} \end{pmatrix} = \begin{pmatrix} \mathbb{1} & \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} \\ \mathbb{0} & \mathbb{1} & \mathbb{0} & \cdots & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{1} \end{pmatrix} \cdot \begin{pmatrix} q_1^{(l)} \\ q_2^{(l)} \\ q_3^{(l)} \\ \vdots \\ q_k^{(l)} \end{pmatrix}$$

$$+\begin{pmatrix} \mathbb{1} & \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} \\ \mathbb{1} & -\mathbb{1} & \mathbb{0} & \cdots & \mathbb{0} \\ \mathbb{1} & -2\mathbb{1} & \mathbb{1} & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \mathbb{0} \\ \binom{k}{0}\mathbb{1} & -\binom{k}{1}\mathbb{1} & \binom{k}{2}\mathbb{1} & \cdots & (-1)^k\binom{k}{k}\mathbb{1} \end{pmatrix}$$

$$\cdot\begin{pmatrix} f^{(l)}\left(q_1^{(l)}\right) \\ f^{(l-1)}\left(q_1^{(l)} - q_2^{(l)}\right) \\ f^{(l-2)}\left(q_1^{(l)} - 2q_2^{(l)} + q_3^{(l)}\right) \\ \vdots \\ f^{(l-k+1)}\left(\sum_{n=0}^{k-1}(-1)^n\binom{k-1}{n}q_k^{(l)}\right) \end{pmatrix}\Delta l. \tag{2.15}$$

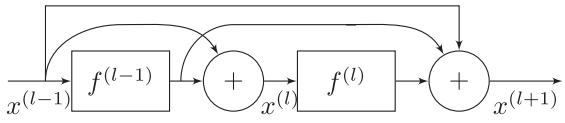
Remember that if there are d-many nodes per layer, then each $q_n^{(l)}$ maps to $\mathbb{R}^d$, and so these matrices are block matrices. For example, the entry $\binom{n}{l}\mathbb{1}$ is the $d \times d$ matrix with the number $\binom{n}{l}$ along all of the diagonals, for $n = 1, 2, \ldots, k$ and $l = 1, 2, \ldots, n$. Similarly, the matrix $\mathbb{0}$ is the $d \times d$ matrix of all zeros.

Equation 2.14 and, equivalently, equation 2.15, is the state-space representation of the additive dense network for general k. The schematic of the $k = 2$ additively dense network architecture, with its equivalent state-space representation, is given in Figure 2. By introducing k-many lags into the dense network, the dimension of the state space increases by a multiple of k for an equivalent first-order system, since we are concatenating all of the $q_n^{(l)}$'s to define the complete state of the system as $q^{(l)} := \left[q_1^{(l)}; q_2^{(l)}; \cdots; q_k^{(l)}\right]$, which maps to $\mathbb{R}^{k \cdot d}$.

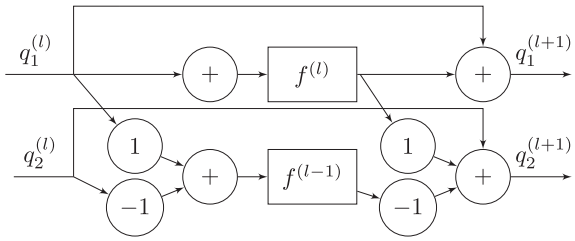
When we use the notation from dynamical systems and control theory, this can also be represented succinctly:

$$q_n^{(l+1)} = \mathbb{1} \cdot q_n^{(l)} + B_{n,k} \cdot u_{n,k}^{(l)}\left(q_1^{(l)}, q_2^{(l)}, \ldots, q_n^{(l)}\right) \qquad \forall n = 1, 2, \ldots, k, \tag{2.16}$$

where $B_{n,k}$ is defined as the nth row of the second block matrix of equation 2.15. It is seen that the neural network activations $u_{n,k}^{(l)}\left(q_1^{(l)}, q_2^{(l)}, \ldots, q_n^{(l)}\right)$ for all $n = 1, 2, \ldots, k$ acts as the controller of this system as the system moves forward in layers (analogous to time). In this sense, the gradient descent



(a) An additive dense network with $k = 2$.



(b) The equivalent state-space model of the $k = 2$ additive dense network.

Figure 2: The block diagram of the $k = 2$ additive dense network architecture architecture (top) and its equivalent state-space model (bottom), where $q_1^{(l)} = x^{(l)}$ and $q_2^{(l)} = x^{(l)} - x^{(l-1)}$. It is seen that if the $k = 2$ model has d-many nodes at each layer l (i.e. $x^{(l)}$ maps to $\mathbb{R}^d$), then its state-space representation $q^{(l)} := [q_1^{(l)}; q_2^{(l)}]$ maps to $\mathbb{R}^{2d}$. Note that the concatenation block in the standard dense network has been replaced with a summation block, although in the state-space form, it is seen that using a summation block still leads to the states being implicitly concatenated.

training process is learning a controller that maps the data from input to target.

Notice that in the state-space formulation in equation 2.15, it is immediate that the additive dense network, when $k = 1$, collapses to the residual network of equation 2.2. Also notice from equation 2.14 that additive dense networks have the form $\delta^+(\delta^-)^n x^{(l)} = (\delta^-)^n f^{(l)} \Delta l$ for $n = 1, 2, \ldots, k-1$.

3 Network Capacity and Skip Connections

The objective of this section is to partially explain why imposing high-order skip connections on the network architecture is likely to be beneficial. A first-order system has one state variable (e.g., position), while a second-order system has two state variables, (e.g., position and velocity). In general, a kth order system has k-many state variables, for $k \in \mathbb{N}$.

Recall that when $x^{(l)}$ maps to $\mathbb{R}^d$, the equivalent first-order system $q^{(l)} = \left[q_1^{(l)}; q_2^{(l)}; \ldots; q_k^{(l)}\right]$ maps to $\mathbb{R}^{k \cdot d}$, for a kth-order system. This holds since each of the k-many functions $x^{(l)}$ mapping to $\mathbb{R}^d$ operates independent of each

other through their independently learned weight matrices, and so their concatenation spans $\mathbb{R}^{k \cdot d}$.

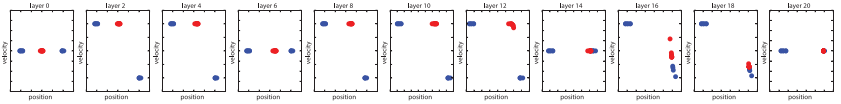
This immediately implies that the weight matrix for transforming the kth order system is $(d \times d)$, while the weight matrix for transforming the equivalent first-order system is $(k \cdot d \times k \cdot d)$. Therefore, by imposing k-many skip connections on the network architecture, from a dynamical systems perspective, we only need to learn up to $\frac{1}{k^2}$ as many parameters to maintain the same embedding dimension when compared to the equivalent zeroth or first-order system. Also notice that the $(k \cdot d \times k \cdot d)$ weight matrix for transforming the $x^{(l-n+1)}$'s to the state vectors $q_n^{(l)}$'s is a lower block diagonal matrix, and so it is full rank, and so state variables defined by this transformation matrix do not introduce degeneracies.

4 Numerical Experiments

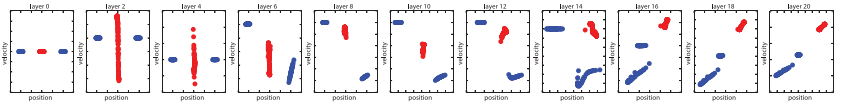
This section describes experiments designed to understand and validate the proposed theory. The simulations were run in tensorflow (Abadi et al., 2015) and trained via error backpropagation (Rumelhart, Hinton, & Williams, 1985) with gradients estimated by the Adam optimizer (Kingma & Ba, 2014).

4.1 Visualizing Implicit Dimensions. An experiment was conducted to visualize and understand these implicit dimensions induced from the higher-order dynamical system. The one-dimensional data were constructed such that 50% of the data are the red class and the other 50% are the blue class, and the blue data are separated with half to the left of the red data and half to the right. It might seem that there is no sequence of single-neuron transformations that would put these data into a form that can be linearly separated by hyperplane, and at best one could achieve an accuracy of 75%. This is the case with the standard $\mathcal{C}^1$ residual network (see Figure 3a). The $\mathcal{C}^1$ architecture has only one state variable, position, and therefore cannot place a hyperplane to linearly separate the data along the positional dimension.

In contrast, the $\mathcal{C}^2$ architecture has two state variables, position $q_1^{(l)} := x^{(l)}$ and velocity $q_2^{(l)} := x^{(l)} - x^{(l-1)}$, and therefore its equivalent first-order system is two-dimensional. When visualizing both state variables, one sees that the data in fact get shaped such that a hyperplane only along the positional dimension can correctly separate the data with 100% accuracy. If one were looking only at the positional state variable, that is, the output of the single node, it would seem as if the red and blue curves were passing through each other; however, in the entire two-dimensional space, we see that is not the case. Although this network has only a single node per layer and the weight matrices are just single scalars, the equivalent first-order dynamical system has two dimensions and therefore the one-dimensional data can be



(a) A $\mathcal{C}^1$ architecture achieves 75.0% accuracy.



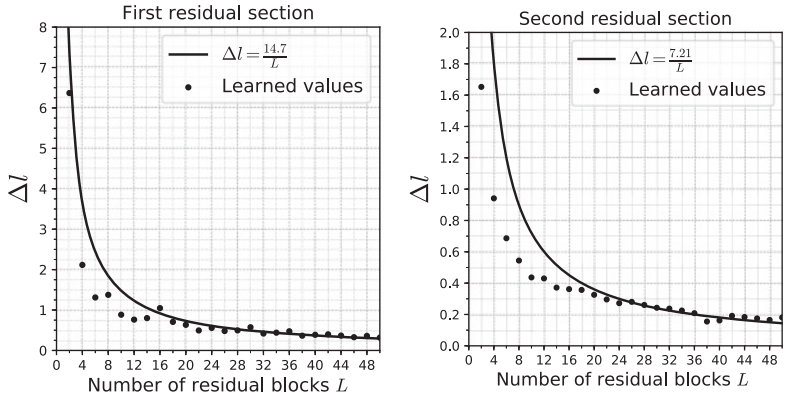
(b) A $\mathcal{C}^2$ architecture achieves 100% accuracy.

Figure 3: Experiments comparing how single-node per layer architectures linearly separate one-dimensional data. The x-axis is position $q_1^{(l)} = x^{(l)}$ (i.e., the value of the single node at layer l), while the y-axis is the velocity $q_2^{(l)} = x^{(l)} - x^{(l-1)}$; at $l = 0$, the velocity is set equal to zero. The $\mathcal{C}^1$ architecture has only one state variable, position, and is therefore unable to properly separate the data. In comparison, the $\mathcal{C}^2$ architecture, while still having only a single node per layer, has two state-space variables, position and velocity, and is therefore able to use both of these to correctly separate the data in the positional dimension of the single-node-per-layer architecture.

twisted in this two-dimensional phase space into a form such that it is linearly separable in only the one positional dimension.

4.2 Estimating the Magnitude of the Perturbations. This section seeks to quantify the magnitude of the perturbation, and therefore validate the perturbation approximations being made. In order for $x^{(l+1)} = x^{(l)} + f^{(l)}(x^{(l)})\Delta l + \mathcal{O}(\Delta l^2)$ to be a valid perturbation expansion from the transformation $\dot{x}^{(l)} = f^{(l)}(x^{(l)})$, we require $||f^{(l)}(x^{(l)})\Delta l||_2 << ||x^{(l)}||_2$. This implies that the magnitude of Δl should be such that $\frac{||f^{(l)}(x^{(l)})\Delta l||_2}{||x^{(l)}||_2} =<< 1$. In addition, assuming the image is traveling a constant distance d from input to output, one would expect the average size of the perturbation to be roughly $\Delta l \approx \frac{d}{L}$. That is, as one increases the number of layers, the average size of each partition region (mesh size) should get smaller as $\sim \frac{1}{L}$. Experiments were conducted on MNIST, measuring the size of the perturbation term for a $\mathcal{C}^1$ network with two sections of residual blocks of sizes $28 \times 28 \times 32$ and $14 \times 14 \times 64$, with the number of blocks in each section being $L = 2, 4, 6, \ldots, 50$. The results are shown in Figure 4. Details of this experiment are given in the appendix. Several conclusions are drawn from this experiment:

• The magnitude of the perturbation term, for sufficiently large L, is in fact much less than one. At least in this setting, this experimentally



(a) Average perturbation size from the first residual section.

(b) Average perturbation size from the second residual section.

Figure 4: Experiments on MNIST measuring the size of the perturbation term for a $\mathcal{C}^1$ (residual) network. The same basic network structure was used with two sections of feature maps of sizes 28×28 and 14×14. The magnitude of the perturbation term is measured against the number of blocks per section, with the number of blocks per section $L = 2, 4, 6, \ldots, 50$. With a total computational distance d, each image travels through the network; the average mesh size should go as $\Delta l \approx \frac{d}{L}$. The depth-invariant computational distance d was fit by linear regression, yielding $d = 14.7$ for the first block and $d = 7.21$ for the second.

validates the intuition that residual networks are learning perturbations from the identity function.

• With increasing the number of layers L, the magnitude of the perturbation goes as $\Delta l \approx \frac{d}{L}$, suggesting that there exists a total distance the image travels as it passes through the network. This implies that the image can be interpreted as moving along a trajectory from input to output, in which case the $\mathcal{C}^1$ network is a finite difference approximation to the differential equation governing this trajectory. Performing a linear regression on $\{(L, \frac{1}{d} \cdot L)\}$ yields that the image travels a "computational distance" of $d_1 = 14.7$ through the first section and $d_2 = 7.21$ through the second section. This may suggest that the first section is more important when processing the image than the second section. If taken literally, it would imply that the average MNIST image is traveling a total "computational distance" of $d_{total} = 21.9$ from the low-level input representation to the high-level abstract output representation. This measure is a depth-invariant computational distance the data travel through the network.

• This analytical approach suggests a systematic way of determining the depth of a network when designing new network architectures. If one

Table 1: Test Errors for Our Implementations of the Various Types of Architectures on CIFAR10 and SVHN.

	$\mathcal{C}^1$	$\mathcal{C}^2$	$\mathcal{C}^3$	$\mathcal{C}^4$	add-dense^2	add-dense^3	add-dense^4
CIFAR10	9.65%	9.59%	**9.46**%	13.08%	12.01%	12.59%	12.01%
SVHN	2.77%	**2.66**%	2.90%	6.64%	3.63%	3.66%	3.53%

Notes: All networks had three sections where the data are transformed to sizes $32 \times 32 \times 16$, $16 \times 16 \times 32$ and $8 \times 8 \times 64$ (denoted by height $\times$ width $\times$ number of channels), and each section having five residual blocks. Training procedures were kept constant for all experiments; only the skip connections were changed. The numbers in bold highlight the lowest test errors achieved per data set among the different architectures being compared. The C^3 network achieves the lowest test error of 9.46% on the CIFARIO detaset, and the C^2 network achieves the lowest test error of 2.66% on the SVHN data set.

requires a certain minimum mesh size, after estimating the d_i's, one can then calculate the minimum number of layers required to achieve a mesh of this size. For example, on this MNIST experiment, if one requires a minimum average mesh size of $\Delta l = 0.2$, then the first section should have about 74 layers while the second needs only 36 layers.

4.3 Comparison of Various Order Network Architectures. This section experimentally compares the classification performance of various order architectures that are described in this letter. The architectures that are tested are the $\mathcal{C}^k$ networks for $k = 1, 2, 3, 4$, as well as the additive dense network for $k = 2, 3, 4$; note that the $k = 1$ additive dense network is the same as the $\mathcal{C}^1$ network. In all of the experiments, first the $\mathcal{C}^1$ ResNet architecture was designed to work well; then, using these exact conditions, the described skip connections were introduced, changing nothing else. Further details of the experiments are in the appendix.

It is seen in Table 1 that in both CIFAR10 and SVHN, the $\mathcal{C}^1$, $\mathcal{C}^2$, and $\mathcal{C}^3$ architectures all perform similarly well, the $\mathcal{C}^4$ architecture performs much more poorly, and the three additive dense networks perform fairly well. On CIFAR10, the $\mathcal{C}^3$ architecture achieved the lowest test error, while on SVHN, this was achieved by the $\mathcal{C}^2$ architecture. A likely reason that the $\mathcal{C}^4$ architecture is performing significantly worse than the rest could be that this architecture imposes significant restrictions on how data flow through the network. Thus, the network does not have sufficient flexibility in how it can process the data.

5 Conclusion and Future Work

This letter has developed a theory of skip connections in neural networks in the state-space setting of dynamical systems with appropriate algebraic structures. This theory was then applied to find closed-form solutions for

the state-space representations of both $\mathcal{C}^k$ networks as well as dense networks. This immediately shows that these kth-order network architectures are equivalent, from a dynamical systems perspective, to defining k-many first-order systems. In the $\mathcal{C}^k$ design, this reduces the number of parameters needed to learn by a factor of k^2 while retaining the same state-space embedding dimension for the equivalent $\mathcal{C}^0$ and $\mathcal{C}^1$ networks.

Three experiments were conducted to validate and understand the proposed theory. The first had a carefully designed data set such that restricted to a certain number of nodes, the neural network is able to properly separate the classes only by using the implicit state variables in addition to its position, such as velocity. The second experiment on MNIST was used to measure the magnitude of the perturbation term with varying levels of layers, resulting in a depth-invariant computational distance the data travel, from low-level input representation to high-level output representation. The third experiment compared various-order architectures on benchmark image classification tasks. This letter explains in part why skip connections have been so successful and motivates the development of architectures of these types.

While there are many possible directions for further theoretical and experimental research, we suggest the following topics of future work:

- Rigorous design of network architectures from the algebraic properties of the space-space model, as opposed to engineering intuitions.
- Analysis of the topologies of data manifolds to determine relationships between data manifolds and minimum embedding dimension, in a similar manner to the Whitney embedding theorems.
- Investigations of the computational distance for different and more complex data sets. This invariant measure could be potentially used to systematically define the depth of the network, as well as to characterize the complexity of the data.

Appendix: Description of Numerical Experiments

For the experiment of section 4.2, no data augmentation was used and a constant batch size of 256 was used. In the network, each block has the form $x^{(l+1)} = x^{(l)} + W_2^{(l)} * \mathrm{LReLU}\big(\mathrm{BN}\big(W_1^{(l)} * x^{(l)}\big)\big)$, where the $*$ is the convolution operation, $W_1^{(l)}$ and $W_2^{(l)}$ are the learned filters, and LReLU and BN are the leaky-ReLU activation and batch-normalization functions. For specifying image sizes, we use the notation num_pixels_Y $\times$ num_pixels_X $\times$ num_channels. The first section of the network of constant feature map size operates on $28 \times 28 \times 32$ images, and a stride of 2 is then applied and mapped to $14 \times 14 \times 64$. After section 4.2, global average pooling was performed to reduce the size to 64 length vectors and fed into a fully connected layer for softmax classification.

For section 4.3, the batch size was updated automatically from 32, 64, . . . , 1024, when a trailing window of the validation error stopped decreasing. In CIFAR10, 5000 of the 60,000 training samples were used for validation, while in SVHN, a random collection of 80% of the training and extra data was used for training, while the remaining 20% was used for validation. The only data augmentation used during training was that the images were flipped left-right and padded with four zeros and randomly cropped to $32 \times 32 \times 3$.

In the networks of section 4.3, each section of constant feature map size contained five residual blocks, all having forcing functions:

$$f^{(l)}\big(x^{(l)}\big) := \mathrm{BN}\big(W_2^{(l)} * \mathrm{LReLU}\big(\mathrm{BN}\big(W_1^{(l)} * x^{(l)}\big)\big)\big).$$

The first, second, and third sections operate on images of sizes $32 \times 32 \times 16$, $16 \times 16 \times 32$, and $8 \times 8 \times 64$, respectively, with downsampling by convolution strides of two, and increasing the number of channels by using filter banks of size $1 \times 1 \times 16 \times 32$ and $1 \times 1 \times 32 \times 64$. Global average pooling was then performed on the last k layers to reduce the size to k-many 64-length vectors, and with each of the k vectors then fed into a fully connected layer of size 200, leaky-ReLu applied and then fully connected for softmax classification.

Acknowledgments

S. S. has been supported by the Walker Fellowship from the Applied Research Laboratory at the Pennsylvania State University. The work reported here has been supported in part by the U.S. Air Force Office of Scientific Research under grants FA9550-15-1-0400 and FA9550-18-1-0135 in the area of dynamic data-driven application systems. Any opinions, findings, and conclusions or recommendations expressed in this letter are our own and do not necessarily reflect the views of the sponsoring agencies.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. tensorflow.org.

Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., & Holtham, E. (2017). *Reversible architectures for arbitrarily deep residual neural networks*. arXiv:1709.03698.

Chang, B., Meng, L., Haber, E., Tung, F., & Begert, D. (2017). *Multi-level residual networks from dynamical systems view*. arXiv:1710.10348.

Haber, E., & Ruthotto, L. (2017). Stable architectures for deep neural networks. *Inverse Problems*, *34*(1), 014004.

Hauser, M., & Ray, A. (2017). Principles of Riemannian geometry in neural networks. In L. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan,

& R. Garnett (Eds.), *Advances in neural information processing systems*, (pp. 2804–2813). Red Hook, NY: Curran.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). Piscataway, NJ: IEEE.

Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (vol. 1, p. 3). Piscataway, NJ: IEEE.

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv:1412.6980.

Lin, H., & Jegelka, S. (2018). *Resnet with one-neuron hidden layers is a universal approximator*. arXiv:1806.10909.

Lu, Y., Zhong, A., Li, Q., & Dong, B. (2017). *Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations*. arXiv:1710.10121.

Proctinger, H. (1993). *Some information about the binomial transform*. CiteSeerX.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation* (Technical Report). San Diego: University of California San Diego, La Jolla Institute for Cognitive Science.

Veit, A., Wilber, M. J., & Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems*, (pp. 550–558). Red Hook, NY: Curran.

Received June 10, 2018; accepted October 28, 2018.