Journal of Data Science 22 (1), 10–24
January 2024
DOI: 10.6339/23-JDS1093
Statistical Data Science

Neural Generalized Ordinary Differential Equations with Layer-Varying Parameters

Duo Yu[1], Hongyu Miao[2], and Hulin Wu[3,*]

[1]*Department of Population Health, The University of Texas at Austin, United States*
[2]*College of Nursing, Florida State University, United States*
[3]*Department of Biostatistics and Data Science, The University of Texas Health Science Center at Houston, United States*

Abstract

Deep residual networks (ResNets) have shown state-of-the-art performance in various real-world applications. Recently, the ResNets model was reparameterized and interpreted as solutions to a continuous ordinary differential equation or Neural-ODE model. In this study, we propose a neural generalized ordinary differential equation (Neural-GODE) model with layer-varying parameters to further extend the Neural-ODE to approximate the discrete ResNets. Specifically, we use nonparametric B-spline functions to parameterize the Neural-GODE so that the trade-off between the model complexity and computational efficiency can be easily balanced. It is demonstrated that ResNets and Neural-ODE models are special cases of the proposed Neural-GODE model. Based on two benchmark datasets, MNIST and CIFAR-10, we show that the layer-varying Neural-GODE is more flexible and general than the standard Neural-ODE. Furthermore, the Neural-GODE enjoys the computational and memory benefits while performing comparably to ResNets in prediction accuracy.

Keywords *B-splines; deep residual networks; neural ordinary differential equations*

1 Introduction

Deep learning (or deep neural networks) has been successfully applied in a variety of real-world areas, including computer vision (Krizhevsky et al., 2009; Goodfellow et al., 2014; Long et al., 2015), game playing (Silver et al., 2016, 2017), natural language processing (Graves et al., 2013; Bahdanau et al., 2014; Young et al., 2018), speech recognition (Noda et al., 2015; Yu and Deng, 2016), medical diagnosis (Esteva et al., 2017; Kim et al., 2018; Abdeltawab et al., 2019; Yu et al., 2020), and physical science (Sigaki et al., 2020). One type of deep learning model, called deep residual networks (ResNets), has shown state-of-the-art performance in image recognition (He et al., 2016a; Qiu et al., 2017; Zhang et al., 2017). By incorporating the shortcut connection, the ResNets improves learning performance with deeper and wider architectures (Bishop et al., 1995; Ripley, 2007). Therefore, it has been considered a default practice of the convolutional neural networks (CNN) models and a powerful tool to deal with complex image recognition problems. For example, ResNets-152 achieves 19.38% top-1 error on the ImageNet data with 152 layers (He et al., 2016a); and ResNets-1001 reaches 4.92% test error on the CIFAR-10 data with 1000 layers (He et al., 2016b).

*Corresponding author. Email: hulin.wu@uth.tmc.edu.

Received December 15, 2022; Accepted February 19, 2023

A ResNets transforms the input and hidden layers iteratively to filter the information:

$$h_{t+1} = h_t + \sigma(h_t, \theta_t), \quad \text{for} \quad t = 0, 1, \ldots, T-1, \tag{1}$$

where h_0 is the input data, $h_1, h_2, \ldots, h_T$ are the hidden layers, σ denotes the activation function, θ_t represents the parameters that link h_t and h_{t+1}. These iterative updates can be interpreted as an Euler discretization of a nonlinear ordinary differential equation (ODE) (Weinan, 2017; Chen et al., 2018; Haber and Ruthotto, 2017; Li et al., 2017; Lu et al., 2018):

$$h'(t) = \sigma(h(t), \theta(t)).$$

Motivated by such a link between ResNets and ODE, many deep learning architectures and training algorithms have been proposed, including differential equation-based neural networks (Chen et al., 2018; Chang et al., 2018; Haber and Ruthotto, 2017; Ruthotto and Haber, 2020; Lu et al., 2018; Li et al., 2017; Chen et al., 2018; Bai et al., 2019; Cranmer et al., 2020; Dupont et al., 2019; Zhang et al., 2019; Zhong et al., 2019; Greydanus et al., 2019) and continuous-time recurrent neural networks (RNN) (Chang et al., 2019; Lim, 2021; Lim et al., 2021; Rusch and Mishra, 2020, 2021). Among them, Chen et al. (2018) proposed the Neural-ODE model that replaced the multiple residual blocks with one ODE system in the model architecture. To train such a Neural-ODE model, the authors developed the publicly available software, torchdiffeq, which adopted various ODE solvers at the back-propagation step. Compared with the original ResNets, many advantages of the proposed Neural-ODE were demonstrated, including memory efficiency, adaptive computation of solving ODEs, scalable and invertible normalizing flow constructions, and building continuous time-series models.

Conceptually, the Neural-ODE model combines the strengths of both parametric and non-parametric approaches for model construction and training. Parametric dynamical models, such as ordinary differential equations (ODEs), have long been studied in, e.g., mathematics, physics, and engineering. Therefore, a rich amount of theories and application experiences have been accumulated (LaSalle, 1968; Simmons, 2016; Arnold, 2012; Yu et al., 2016, 2017, 2021). In general, the ODE models use parameterized mathematical functions to describe dynamic behaviors of the state variables for a given dynamic system, such as the logistic model for population growth (Yu et al., 2016), prey-predator dynamics in ecological studies (Tang et al., 2015), and Susceptible-Infectious-Recovered (SIR) model for modeling infectious disease transmissions (Yu et al., 2017). Then the parameters of the ODE model are estimated based on observed data so that the proposed ODE can describe the observed dynamic behavior of a system. However, this parametric approach is limited by its assumption of the underlying mechanisms, which may only partially capture the real dynamics. On the other hand, machine learning models, such as recurrent neural networks (RNNs), multilayer perceptron (MLP), and deep residual networks (ResNets), have been successfully applied in a variety of fields due to their universal approximation property. By connecting the deep residual networks and ordinary differential equations, the Neural-ODE model serves as a promising model to better capture the real-world dynamics.

1.1 Generalized Ordinary Differential Equation (GODE)

Previously, we have proposed a generalized ODE (GODE) to model the dynamics of both continuous and discrete data, including binary or categorical data types (Miao et al., 2014). Compared to the traditional ODE system where the variable is continuous, the GODE extends the ODE to model the outcome that follows an exponential family distribution, which refers to a family of

flexible distribution for both continuous and discrete random variables. Assume the outcome at time t, $y(t)$, which is a random variable that follows an overdispersed exponential distribution with the probability mass (or density) function of

$$f(y(t), \psi_t, \phi_t) = \exp\Big(\frac{y(t)\psi_t - b(\psi_t)}{a(\phi_t)} + c(y(t), \phi_t)\Big), \tag{2}$$

with respect to a σ-finite measure π, where $a(\cdot)$, $b(\cdot)$ and $c(\cdot)$ are some pre-specified functions; ψ_t and ϕ_t are natural parameter and dispersion parameter at time t, respectively. Then, the expectation of $y(t)$ can be derived as

$$E(y(t)) = \mu(t) = b'(\psi_t),$$

where $b'(\psi_t)$ is the first-order derivative of $b(\psi_t)$ with respect to ψ_t. In the GODE system, the expectation of the response variable, i.e., $\mu(t)$, is modeled as

$$\eta(t) = g(\mu(t)) = g^*(\mathbf{X}(t), Z(t), \boldsymbol{\beta}),$$

where $g(\cdot)$ denotes the link function; $g^*(\cdot)$ is a function of covariates, $\mathbf{X}(t)$, latent (unobserved) dynamic state variable, $Z(t)$, and the parameter vector $\boldsymbol{\beta}$. In particular, the latent state variable, $Z(t)$, is modeled by a time-varying parameterized ODE,

$$Z'(t) = \sigma(t, Z(t), \boldsymbol{\theta}(t)), \tag{3}$$

where $t \in [0, T]$, $Z(0)$ is the initial condition of the dynamic system which can be considered as an unknown parameter; σ is an explicit given function, $\boldsymbol{\theta}(t)$ denotes the vector of time-varying parameters.

Note that instead of an explicit function, if σ is a neural network activation function, then the latent time-varying variables, $Z(t)$ in equation (3), can be considered as hidden layers in neural networks. Moreover, it is equivalent to the hidden layer of ODE block in the Neural-ODE (Chen et al., 2018) when $\boldsymbol{\theta}(t)$ is independent of t. In this study, we further extend the GODE model to establish a general statistical framework for the deep learning neural network model.

1.2 Related Work

Recently, several studies have tried to extend the standard Neural-ODE with time-varying parameterized ODE systems (Zhang et al., 2019; Queiruga et al., 2020, 2021; Massaroli et al., 2020). Zhang et al. proposed ANODEV2, which extends the standard Neural-ODE with a coupled ODE system, in which the weights are time-varying and governed by an ODE. Queiruga et al. proposed ContinuousNet, a family of continuous-in-depth generalizations of ResNet architectures. A set of time-varying basis functions are used to model the neural network weights (Queiruga et al., 2020). Based on the CIFAR-10 and CIFAR-100 data sets, it has been shown that the ContinuousNet outperforms ANODEV2 and standard Neural-ODE while having competitive performances compared to ResNets in terms of accuracy. In the latter work of Queiruga et al. (2021), the stateful ODE-Nets were developed, which achieved state-of-the-art performance on CIFAR-10. Both the ContinuousNet and stateful ODE-Nets applied piecewise constant (or linear) basis functions in the time-varying ODEs. However, it is not clear if the basis function with higher orders, such as B-spline functions, can additionally improve the model performance. More recently, Günther et al. (2021) explored the B-spline parameterized neural networks, which

showed better performance compared to ResNets and standard Neural-ODE based on numerical tests. However, the Neural-ODE with B-spline parameterized ODEs have not been evaluated based on the benchmark data sets.

The time-varying parameterized differential equations are flexible (Chen and Wu, 2008; Xue et al., 2010) dynamic systems that have been widely applied in practice, and the ODE model fitting (training) methods and corresponding theoretical properties have been investigated in statistical communities in the past two decades (Chen and Wu, 2008; Xue et al., 2010; Liang et al., 2010). In this study, we couple the time-varying ODE and GODE modeling to generalize the standard Neural-ODE model proposed by Chen et al. (2018) and propose a neural generalized ODE (Neural-GODE) model. We can demonstrate that the aforementioned Neural-ODE or ODE-Nets models are special cases of the proposed Neural-GODE models. Thus, we establish a connection of deep neural network models and Neural-ODE models in machine learning community with the GODE model concept developed in statistical community many years ago. This lays a foundation to potentially leverage the statistical concepts and theories on ODE models to study the properties of deep learning algorithms in the future. In this study we also evaluated the performance of the proposed Neural-GODE and compared with existing deep residual networks (ResNet) (He et al., 2016a) and the standard Neural-ODE model (Chen et al., 2018) using benchmark datasets, MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky et al., 2009).

2 Method

Given a supervised learning problem with input $\mathcal{X}$ and output label $\mathcal{Y}$, the essential task is to find a mapping

$$F : \mathcal{X} \to \mathcal{Y},$$

where $\mathcal{X} \subset \mathbb{R}^p$, $\mathcal{Y} \subset \mathbb{R}$, such that $F(\mathbf{X}_i)$ can accurately predict y_i, and $(\mathbf{X}_i, y_i)$ is the i-th sample, $i = 1, 2, \ldots, n$. Usually, F is approximated by penalized regression, algorithms, and networks, such as LASSO, support vector machine (SVM), and multilayer perceptron (MLP) neural networks. Especially with the universal approximation property, multilayer feedforward networks, such as residual neural networks (ResNets), multilayer perceptron (MLP), and convolutional neural networks (CNN), are widely applied in complex prediction tasks (Hornik et al., 1989). Multilayer feedforward networks use the forward propagation technique that processes the inputs in a nonlinear and forward direction way to filter the information. For example, in a general residual neural networks (ResNets), the forward propagation of input $Z_0 \in R^{n \times p}$, with T layers is given by

$$Z_{(t+1)} = Z_t + h\sigma(Z_t K_t + b_t), \quad \text{for} \quad t = 0, 1, \ldots, T-1, \tag{4}$$

where $Z_0 = X$, and $Z_1, Z_2, \ldots, Z_T$ are the hidden layers, t is the layer index, σ is the activation function, and h is the scaling factor, K_t and b_t are the constant weights of the t-th hidden layer. The iterative updates of hidden layers Z_t, equation (4), can be interpreted as a discretized nonlinear ordinary differential equation (ODE):

$$\dot{Z}(t) = \sigma(Z(t)\beta(t) + b(t)),$$

where $Z(t)$ is a time-varying variable with initial $Z(0) = X$; $\beta(t)$ and $b(t)$ are time-varying parameters. Based on the GODE in Section 1.1, assume the outcome variable Y follows an exponential family distribution (2) with a mean of $E(Y) = \mu$ and link function involving a hidden variable Z_T:

$$\eta = g(\mu) = g^*(Z_T, \boldsymbol{\theta}),$$

where $Z_T \doteq Z(T)$ is the solution to the ODE in equation (2). The optimization problem with respect to both time-varying and constant parameters, $\beta(t)$, $b(t)$, $\boldsymbol{\theta}$, can be written as

$$\begin{aligned} &\min \frac{1}{n} L(\beta(t), b(t), \boldsymbol{\theta} \mid X, Y) + \lambda P(\beta(t), b(t), \boldsymbol{\theta}), \\ &\text{s.t.} \quad \dot{Z}(t) = \sigma(Z(t)\beta(t) + b(t)), \quad Z(0) = X, \\ &\qquad \eta = g(\mu) = g^*(Z_T, \boldsymbol{\theta}), \end{aligned} \tag{5}$$

where, $L(\beta(t), b(t), \boldsymbol{\theta} \mid X, Y)$ is the likelihood function and $P(\beta(t), b(t), \boldsymbol{\theta})$ is the penalty function, λ is the penalty parameter. Adopting the idea of the time-varying ODE model in Xue et al. (2010), we can approximate the time-varying parameters using B-splines, i.e.,

$$\begin{aligned} \beta(t) &= \boldsymbol{B}_1(t)'\boldsymbol{\zeta}_1, \\ b(t) &= \boldsymbol{B}_2(t)'\boldsymbol{\zeta}_2, \end{aligned}$$

where $\boldsymbol{B}_1(t)'$ and $\boldsymbol{B}_2(t)'$ are B-spline basis functions and $(\boldsymbol{\zeta}_1', \boldsymbol{\zeta}_2')$ are constant parameter vectors. A brief review of the B-spline is given below.

2.1 B-Spline Function

The spline functions have been widely used in nonparametric regression and varying-coefficient models in statistical research (Perperoglou et al., 2019). In particular, the splines are generally used for modeling the smooth functions of the interested variables, such as nonlinear effects of covariates, time-dependent effects in regression models, and time-series data modeling.

B-spline is a more general type of curve than Bézier curve (Bartels et al., 1995). A B-spline with degree k and m basis functions can be written as

$$\beta(t) = \sum_{i=0}^{m-1} B_{i,k}(t)\xi_i, \tag{6}$$

where $B_{i,k}(t)$ is a degree k (the highest power) basis function, and ξ_i is the coefficient of i-th basis function. The knot vector is $\{t_0, t_1, t_2, \ldots, t_{k+m}\}$. $B_{i,k}(t)$ is calculated recursively by the following formula:

$$\begin{aligned} B_{i,0} &= \begin{cases} 1, & t_i \leqslant t < t_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \\ B_{i,k} &= \frac{t - t_i}{t_{i+k} - t_i} B_{i,k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1}(t). \end{aligned}$$

In this study, we apply the B-spline to parameterize the time-varying parameters in the GODE. The degree of the basis function, k, controls the complexity of the time-varying effect; the number of the basis functions, m, controls the number of parameters in the GODE. If $k = 0$, $m = 1$, equation (6) can be rewritten as

$$\beta(t) = B_{0,0}(t)\xi_0, \tag{7}$$

where

$$B_{0,0}(t) = \begin{cases} 1, & t_0 \leqslant t < t_1, \\ 0, & \text{otherwise}. \end{cases} \tag{8}$$

In this case, the knot vector is $\{t_0, t_1\}$, i.e., the first and endpoints of the integration time interval. Thus, equation (7) and (8) are equivalent to $\beta(t) = \xi_0$, and $B_{0,0}(t) = 1$. Therefore, the Neural-GODE model is reduced to the standard Neural-ODE model with constant parameters when $k = 0$, $m = 1$. On the other hand, increasing the number of basis functions (knots), m, increases the number of parameters in the integration interval. If an ODE is solved by the simple Euler method with a small step size, the increased number of knots makes each Euler step have different parameters, i.e., the increased number of different θ_t in equation (1). Since the ResNets can be interpreted as nonlinear ODE as equation (1), the Neural-GODE with time-varying parameters can represent the ResNets in this case.

2.2 Model Architectures

We consider the residual networks (ResNets) architecture that has been used for comparison in Chen et al. (2018), see Figure 1. The ResNets first pre-processes the input three times with convolution layers, then followed by multiple standard residual blocks (He et al., 2016a), see Figure 1 (left panel). Each residual block consists of two convolution layers with a kernel size of 3. The standard neural-ODE model and neural-GODE model replace the residual blocks with an ODE block, see Figure 1 (right panel). However, the ODE block of the Neural-GODE is a system with time-varying parameters, which makes the model more flexible for training.

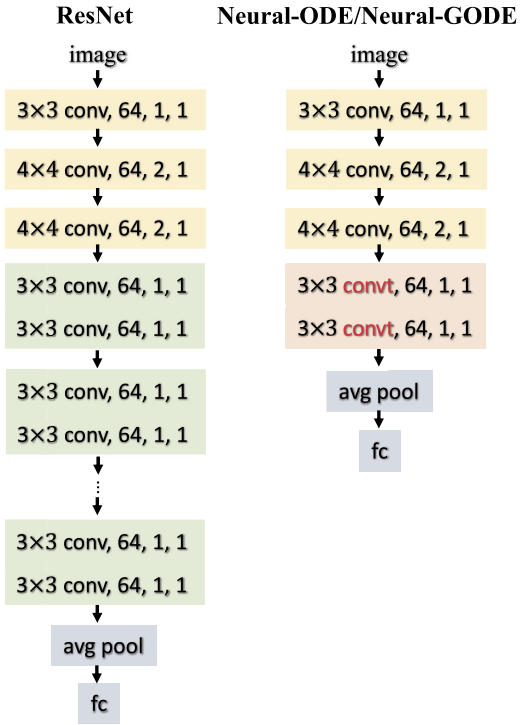

Figure 1: Model architectures comparison between ResNet, Neural-ODE and Neural-GODE for benchmark data. Neural-ODE and Neural-GODE replaces the residual blocks of ResNet with constant and time-varying parameterized ODE block (with two customized convolutional (convt) layers), respectively.

2.3 Implementation

The implementation of the ResNets, Neural-ODE, and Neural-GODE models follows the MNIST training in Chen et al. (2018). The training images are transformed by randomly cropping with padding of 4 on each border. We apply the group normalization over each mini-batch (Wu and He, 2018) right after each convolution layer and before the activation. A uniform knots sequence for B-splines is employed to parameterize the time-varying parameters. Specifically, given m basis functions with degree k in equation (6), the knot vector over time interval $[0, T]$ is $\{t_0, t_1, t_2, \ldots, t_{k+m}\}$, where $t_0 = 0$, $t_{i+1} - t_i = \frac{T}{k+m}$, $i = 0, \ldots, k+m-1$, $t_{k+m} = T$. We initialize the weights of the B-spline parameterized filter in the customized convolution layer (the module "convt" in Figure 1). We use SGD with a mini-batch size of 128 in training. The learning rate starts from 0.1 and is divided by 10 at epochs 60, 100, and 140. A total of 160 epochs are trained for each model. We use a momentum of 0.9. The dropout is not applied, following the practice in Ioffe and Szegedy (2015). We apply the Euler method with a step size of 0.05 for solving the ODE systems both in the Neural-ODE and Neural-GODE by using the package torchdiffeq Chen (2018). In testing, random cropping is not applied, and a mini-batch size of 1000 is used. The model performance is reported based on the testing dataset using accuracy as the metric. All the experiments are implemented on GPU (Tesla V100 with 16GB G-Memory), programming code can be found at https://github.com/Duo-Yu/Neural-GODE.

3 Experimental Results

We evaluate the Neural-GODE on two benchmark datasets, i.e., MNIST and CIFAR-10.

3.1 Model Performances

The MNIST and CIFAR-10 are standard benchmark datasets for computer vision and deep learning. In the MNIST, images are white-black handwritten digits with a size of 28×28 pixels. The MNIST classification aims to predict the ten handwritten digits. It has 70,000 images with 60,000 samples in the training dataset and 10,000 in the testing dataset. The CIFAR-10 dataset is 60,000 colored images with 10 classes. Each image has 32×32 pixels. Generally, the training set of CIFAR-10 consists of 50,000 samples. The ResNets is implemented with 6 and 20 residual blocks on MINIST and CIFAR-10, respectively. With such numbers of residual blocks, more than 99% accuracy can be reached in training on both datasets. The ODE systems are solved from 0 to 1 in the standard Neural-ODE and the proposed Neural-GODE. The number of basis functions of B-spline in Neural-GODE, m, is selected as 4 and 8, respectively, for MNIST and CIFAR-10 data training. The degree of the B-spline is selected as 1. The details of these hyperparameters' determination, including the B-spline degree, number of basis functions, and integration interval of ODE systems, are shown in the following Section 3.2. For test error, the Neural-GODE has the best performance (see Table 1). The ResNet has a slightly lower accuracy than that of Neural-GODE. The standard Neural-ODE has the worst performance compared to the other two models. Especially in the classification task based on CIFAR-10, which is more complex than the MNIST, the standard Neural-ODE model has about 2% lower accuracy than the Neural-GODE (Table 1). In terms of training efficiency, although the standard Neural-ODE and Neural-GODE are slower than ResNets, their memory efficiency can be seen clearly. Both Neural-ODE and Neural-GODE consist of a smaller number of training parameters than that of ResNets; see Table 1. In summary, the Neural-GODE model has the advantage in both predictive

Table 1: Model performance comparison.

	Model	Test error (%)	# params (M)	Time/iteration (s)
MNIST	Neural-GODE	0.31	0.43	0.035
	ResNets	0.33	0.57	0.012
	Neural-ODE	0.40	0.21	0.038
CIFAR-10	Neural-GODE	13.49	0.72	0.038
	ResNets	13.47	1.6	0.026
	Neural-ODE	15.32	0.21	0.041

accuracy and memory efficiency compared to the other two models.

The main difference between Neural-GODE and Neural-ODE is that the 'convt' layer weights are parameterized with B-spline functions of integration time (t) in Neural-GODE rather than constant in the Neural-ODE, Figure 1. As an illustration, we plot the weights of the first convolution layer of the residual block in the ResNets, the first 'convt' layer of the ODE blocks in the Neural-GODE and Neural-ODE in Figure 2. We observe that the kernel weights from the Neural-ODE are constant across the integration time (red lines in Figure 2) as expected. The weights in the Neural-GODE (blue lines in Figure 2) and ResNets (green lines in Figure 2) vary across layers or integration time, and the weights of the Neural-GODE are smoother than those of the ResNets, which might be the reason why the proposed Neural-GODE model could outperform the ResNets and Neural-ODE models in terms of computing efficiency and prediction accuracy.

3.2 Layer-Varying Parameters

To evaluate the effect of hyperparameters of Neural-GODE on prediction results, such as the number of basis functions (m) and degree (k) of B-spline as well as the integration interval endpoint (T) of ODE, we use different settings to train the Neural-GODE on CIFAR-10 data. We find that there exists an optimal number of basis functions for B-spline, which directly affects the Neural-GODE model size (the number of parameters). Based on CIFAR-10, the optimal number of basis functions is 8 when the ODE is solved by the Euler method with a step size of 0.05, see Table 2. If we increase the number of basis functions from 2 to 8, the test error is reduced by 1%. However, if the number of basis functions increases from 8 to 12, the test error increases slightly. According to the experiment, the linear B-spline fits the data well, i.e., when the degree of B-spline (k) is 1, the prediction accuracy reaches the highest. If we increase the endpoint (T), the training speed will significantly decrease. For example, given the linear B-spline with 8 basis functions, if the end integration time increases from 1 to 3, each iteration training time increases from 0.038s to 0.093s, which is about 2.4 times increase, see Table 2. At the same time, we do not find much benefit in terms of prediction accuracy when we increase the endpoint of ODE.

3.3 ODE Solvers

In the experiment, we mainly focus on the Euler method to solve the ODEs because of its first-order equivalence to the general residual network (see Method section). However, we could use

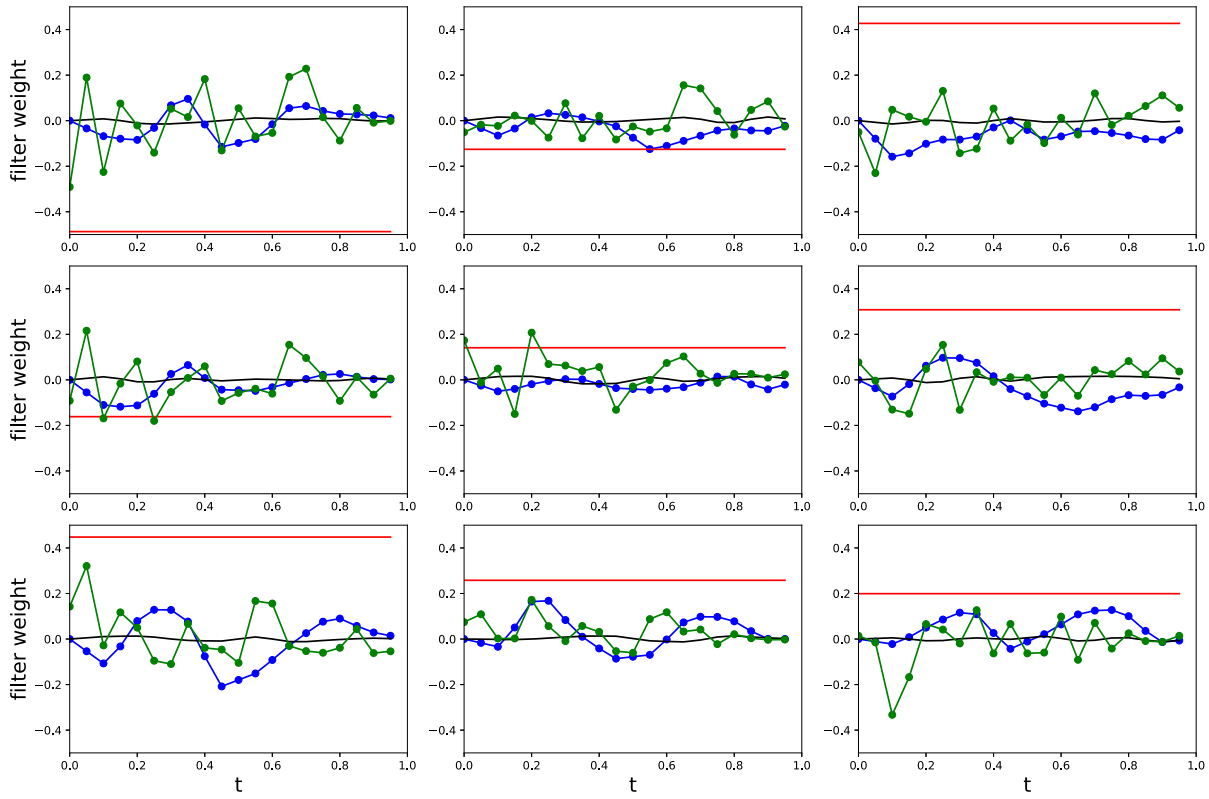

Figure 2: The patterns of the estimated weights of a 3×3 kernel from three models from the CIFAR-10 data example. The x-axis represents the index of layers in ResNets, and the integration time in Neural-GODE and Neural-ODE. The black line represents the initial value of the weights; the red line denotes the weight of the first 'convt' layer in the ODE block of the Neural-ODE; the blue line is the weight of the first 'convt' layer in the ODE block of the Neural-GODE; and the green line is the weight of the first convolution layer in the residual blocks of the ResNets.

alternative ODE solvers in the neural ODE models. For example, instead of using the fixed-step method such as Euler, we also implement another adaptive-step method, i.e., Runge-Kutta of order 5 of Dormand-Prince-Shampine (dopri5 in the package *torchidffeq*). Given the same model, we observe that the Runge-Kutta and Euler methods perform similarly in predictive accuracy, see Table 3. However, the Neural-GODE model shows lower test errors than the standard Neural-ODE using either the Runge-Kutta or the Euler methods. Due to its complex algorithm, the Runge-Kutta method is significantly slower than the Euler method in terms of training efficiency.

3.4 ODE Function Forms

In this section, we investigate the effect of the number of customized CNN layers inside the ODE block of Neural-GODE. The standard Neural-ODE directly replaces the residual blocks with one ODE block consisting of two convolution layers. The "time" t is considered as a separate channel and concatenated with other image channels. The concatenated channels are the input of the ODE block in which the convolution functions are standard. Instead of combining the "time" t with the convolution layer input, we parameterize t inside the convolution function. Specifically,

Table 2: Effect of time-varying parameters.

m	k	T	#params	Time(s)	Test error (%)
2	1	1	281,738	0.039	14.68
4	1	1	429,194	0.038	14.19
6	1	1	576,650	0.037	13.97
8	1	1	724,106	0.038	13.49
10	1	1	871,562	0.037	13.65
12	1	1	1,019,018	0.038	13.64
8	2	1	724,106	0.038	14.16
8	3	1	724,106	0.036	14.10
8	4	1	724,106	0.036	13.80
8	5	1	724,106	0.037	14.22
8	1	2	724,106	0.064	13.79
8	1	3	724,106	0.093	13.84

Table 3: ODE solver comparison.

Model		Test error (%)		Time (s)	
		Dopri5	Euler	Dopri5	Euler
MNIST	Neural-GODE	0.37	0.31	0.153	0.035
	Neural-ODE	0.37	0.40	0.063	0.038
CIFAR-10	Neural-GODE	13.99	13.49	0.224	0.038
	Neural-ODE	15.40	15.32	0.072	0.041

time-varying kernels are used rather than using the kernels with constant weights. Besides the number of basis functions of the B-spline, the number of customized CNN layers (i.e. the convt layer in Figure 1) can also affect the total number of parameters and the complexity of the ODE block. We compare the predictive performance given the different number of basis functions and the number of customized CNN layers based on CIFAR-10. The fixed parameters include the degree of B-spline ($k = 1$), the integration interval ($[0, 1]$), and the ODE solver (Euler method with a step size of 0.05). We observe a trade-off between the number of basis functions and the number of customized CNN layers of the ODE block, see Table 4. When the number of basis functions of the B-spline is small, a larger number of CNN layers may increase the predictive performance. For example, if the number of basis functions is 2 or 4, better predictive accuracy can be reached when the number of CNN layers is 3 (Table 4). However, if we further increase the number of basis functions, two layers of CNN have better performance. Since the ODE block with more CNN layers takes a longer time to solve, we apply two layers of CNN with eight basis functions of the B-spline for CIFAR-10 training.

Table 4: Effect of the number of the customized CNN layers.

m	# layers	#params	Time (s)	Test error(%)
2	1	207,754	0.024	15.83
	2	281,738	0.039	14.68
	3	355,594	0.047	14.31
	4	429,450	0.061	14.37
4	1	281,482	0.024	16.26
	2	429,194	0.038	14.19
	3	576,778	0.047	13.89
	4	724,362	0.062	14.40
6	1	355,210	0.025	15.67
	2	576,650	0.037	13.97
	3	797,962	0.048	14.07
	4	1,019,274	0.062	14.59
8	1	428,938	0.025	15.41
	2	724,106	0.038	13.49
	3	1,019,146	0.047	13.95
	4	1,314,186	0.063	14.02

4 Conclusion

The idea of bridging deep residual networks (ResNets) with the discretized ordinary differential equations (ODE) has raised much interest in the deep learning research field recently (Haber and Ruthotto, 2017; Chang et al., 2018; Lu et al., 2018; Li et al., 2017, 2019; Chen et al., 2018). With well-established ODE properties, theories, and numerical solutions, novel deep learning architectures and training algorithms have been proposed based on such connections. In this study, we further explore the performance of Neural-ODE based on two benchmark classification tasks, i.e., the classification problems of MNIST and CIFAR-10. We confirm that the Neural-ODE model has a training efficiency advantage compared to the deep ResNets in terms of training memory, as stated in Chen et al. (2018). However, it shows the drawbacks of predictive performance. The test error of Neural-ODE is 1.6% higher than that of the ResNets in CIFAR-10 (Table 1).

To overcome the predictive accuracy disadvantage of the standard Neural-ODE, we propose a time-varying Neural-GODE, which improves the model flexibility and the predictive performance. Instead of using constant weights of CNN layers from the standard Neural-ODE, we parameterize the kernel weights with time-varying parameters. Specifically, the weights are B-spline functions of "time" (layer) t. As a result, the proposed Neural-GODE model reaches similar or slightly better prediction accuracy than ResNets. Therefore, the time-varying parameterized ODE block is essential for Neural-ODE models to obtain state-of-the-art performance. Additionally, Queiruga et al. explored the benefits of multiple time-varying ODE blocks and stateful batch normalization layers for Neural-ODE (Queiruga et al., 2020, 2021). Based on those techniques, it has been shown that the time-varying parameterized Neural-ODE can reach

state-of-the-art performance for image classification problems (Queiruga et al., 2021).

Another key contribution of this work is that this is the first study to bridge the generalized ordinary differential equations (GODE) model proposed in statistical literature (Miao et al., 2014) with the deep learning algorithms and Neural-ODE models established in machine learning communities. We further extend the GODE model into a Neural-GODE model to establish a new statistical framework for general deep learning algorithms for both continuous and discrete outcome variables that follow a general exponential family distribution. This new statistical framework lays a good foundation to allow us to leverage theoretical properties and the asymptotic results for ODE models (including time-varying ODE models) (Xue et al., 2010) to further study the theoretical properties and improve the performance of deep learning algorithms in the future.

Supplementary Material

Programming code to reproduce our results and figures can be found at https://github.com/Duo-Yu/Neural-GODE. In the Supplementary Material, we list the code directories and corresponding results.

Funding

This work was supported in part by NIH grant R01 AI087135 and P03AI161943 (HW), grant from Cancer Prevention and Research Institute of Texas (PR170668) (HW), grant NSF/ECCS 2133106 (HM), and NSF/DMS 1620957 (HM).

References

Abdeltawab H, Shehata M, Shalaby A, Khalifa F, Mahmoud A, El-Ghar MA, et al. (2019). A novel cnn-based cad system for early assessment of transplanted kidney dysfunction. *Scientific Reports*, 9(1): 1–11. https://doi.org/10.1038/s41598-018-37186-2

Arnold VI (2012). *Geometrical Methods in the Theory of Ordinary Differential Equations*. Springer Science & Business Media.

Bahdanau D, Cho K, Bengio Y (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint https://arxiv.org/abs/1409.0473.

Bai S, Kolter JZ, Koltun V (2019). Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32.

Bartels RH, Beatty JC, Barsky BA (1995). *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann.

Bishop CM, et al. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Chang B, Chen M, Haber E, Chi EH (2019). Antisymmetricrnn: A dynamical system view on recurrent neural networks. arXiv preprint https://arxiv.org/abs/1902.09689.

Chang B, Meng L, Haber E, Ruthotto L, Begert D, Holtham E (2018). Reversible architectures for arbitrarily deep residual neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Chen J, Wu H (2008). Efficient local estimation for time-varying coefficients in deterministic dynamic models with applications to hiv-1 dynamics. *Journal of the American Statistical Association*, 103(481): 369–384. https://doi.org/10.1198/016214507000001382

Chen RT, Rubanova Y, Bettencourt J, Duvenaud DK (2018). Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31. https://doi.org/10.1007/978-3-030-04167-0

Chen RTQ (2018). torchdiffeq. https://github.com/rtqichen/torchdiffeq.

Cranmer M, Greydanus S, Hoyer S, Battaglia P, Spergel D, Ho S (2020). Lagrangian neural networks. arXiv preprint https://arxiv.org/abs/2003.04630.

Dupont E, Doucet A, Teh YW (2019). Augmented neural odes. *Advances in Neural Information Processing Systems*, 32.

Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, et al. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639): 115–118. https://doi.org/10.1038/nature21056

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27.

Graves A, Mohamed Ar Hinton G (2013). Speech recognition with deep recurrent neural networks. In: *2013 IEEE International Conference on Acoustics, Speech And Signal Processing*. 6645–6649.

Greydanus S, Dzamba M, Yosinski J (2019). Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32.

Günther S, Pazner W, Qi D (2021). Spline parameterization of neural network controls for deep learning. arXiv preprint https://arxiv.org/abs/2103.00301.

Haber E, Ruthotto L (2017). Stable architectures for deep neural networks. *Inverse Problems*, 34(1): 014004. https://doi.org/10.1088/1361-6420/aa9a90

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

He K, Zhang X, Ren S, Sun J (2016b). Identity mappings in deep residual networks. In: *European Conference on Computer Vision*. 630–645.

Hornik K, Stinchcombe M, White H (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5): 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Ioffe S, Szegedy C (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International Conference on Machine Learning*. 448–456.

Kim K, Kim S, Lee YH, Lee SH, Lee HS, Kim S (2018). Performance of the deep convolutional neural network based magnetic resonance image scoring algorithm for differentiating between tuberculous and pyogenic spondylitis. *Scientific Reports*, 8(1): 1–10. https://doi.org/10.1038/s41598-018-35713-9

Krizhevsky A, Hinton G, et al. (2009). Learning multiple layers of features from tiny images, Master's Thesis, University of Tront.

LaSalle JP (1968). Stability theory for ordinary differential equations. *Journal of Differential Equations*, 4(1): 57–65. https://doi.org/10.1016/0022-0396(68)90048-X

LeCun Y, Bottou L, Bengio Y, Haffner P (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324. https://doi.org/10.1109/5.726791

Li Q, Chen L, Tai C, et al. (2017). Maximum principle based algorithms for deep learning. arXiv preprint https://arxiv.org/abs/1710.09513.

Li Q, Lin T, Shen Z (2019). Deep learning via dynamical systems: An approximation perspective. arXiv preprint https://arxiv.org/abs/1912.10382.

Liang H, Miao H, Wu H (2010). Estimation of constant and time-varying dynamic parameters of hiv infection in a nonlinear differential equation model. *The Annals of Applied Statistics*,

4(1): 460. https://doi.org/10.1214/09-AOAS290

Lim SH (2021). Understanding recurrent neural networks using nonequilibrium response theory. *Journal of Machine Learning Research*, 22: 1–47.

Lim SH, Erichson NB, Hodgkinson L, Mahoney MW (2021). Noisy recurrent neural networks. *Advances in Neural Information Processing Systems*, 34: 5124–5137.

Long J, Shelhamer E, Darrell T (2015). Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3431–3440.

Lu Y, Zhong A, Li Q, Dong B (2018). Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In: *International Conference on Machine Learning*. 3276–3285.

Massaroli S, Poli M, Park J, Yamashita A, Asama H (2020). Dissecting neural odes. *Advances in Neural Information Processing Systems*, 33: 3952–3963.

Miao H, Wu H, Xue H (2014). Generalized ordinary differential equation models. *Journal of the American Statistical Association*, 109(508): 1672–1682. https://doi.org/10.1080/01621459.2014.957287

Noda K, Yamaguchi Y, Nakadai K, Okuno HG, Ogata T (2015). Audio-visual speech recognition using deep learning. *Applied Intelligence*, 42(4): 722–737. https://doi.org/10.1007/s10489-014-0629-7

Perperoglou A, Sauerbrei W, Abrahamowicz M, Schmid M (2019). A review of spline function procedures in r. *BMC Medical Research Methodology*, 19(1): 1–16. https://doi.org/10.1186/s12874-018-0650-3

Qiu Z, Yao T, Mei T (2017). Learning spatio-temporal representation with pseudo-3d residual networks. In: *Proceedings of the IEEE International Conference on Computer Vision*. 5533–5541.

Queiruga A, Erichson NB, Hodgkinson L, Mahoney MW (2021). Stateful ode-nets using basis function expansions. *Advances in Neural Information Processing Systems*, 34: 21770–21781.

Queiruga AF, Erichson NB, Taylor D, Mahoney MW (2020). Continuous-in-depth neural networks. arXiv preprint https://arxiv.org/abs/2008.02389.

Ripley BD (2007). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rusch TK, Mishra S (2020). Coupled oscillatory recurrent neural network (cornn): An accurate and (gradient) stable architecture for learning long time dependencies. arXiv preprint https://arxiv.org/abs/2010.00951.

Rusch TK, Mishra S (2021). Unicornn: A recurrent model for learning very long time dependencies. In: *International Conference on Machine Learning*, 9168–9178. PMLR.

Ruthotto L, Haber E (2020). Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62(3): 352–364. https://doi.org/10.1007/s10851-019-00903-1

Sigaki HY, Lenzi EK, Zola RS, Perc M, Ribeiro HV (2020). Learning physical properties of liquid crystals with deep convolutional neural networks. *Scientific Reports*, 10(1): 1–10. https://doi.org/10.1038/s41598-019-56847-4

Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587): 484–489. https://doi.org/10.1038/nature16961

Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676): 354–359.

https://doi.org/10.1038/nature24270

Simmons GF (2016). *Differential Equations with Applications and Historical Notes*. CRC Press.

Tang S, Tang B, Wang A, Xiao Y (2015). Holling ii predator–prey impulsive semi-dynamic model with complex poincaré map. *Nonlinear Dynamics*, 81(3): 1575–1596. https://doi.org/10.1007/s11071-015-2092-3

Weinan E (2017). A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 1(5): 1–11.

Wu Y, He K (2018). Group normalization. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 3–19.

Xue H, Miao H, Wu H (2010). Sieve estimation of constant and time-varying coefficients in nonlinear ordinary differential equation models by considering both numerical error and measurement error. *Annals of Statistics*, 38(4): 2351. https://doi.org/10.1214/09-AOS784

Young T, Hazarika D, Poria S, Cambria E (2018). Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3): 55–75. https://doi.org/10.1109/MCI.2018.2840738

Yu D, Deng L (2016). *Automatic Speech Recognition*, volume 1. Springer.

Yu D, Lin Q, Chiu AP, He D (2017). Effects of reactive social distancing on the 1918 influenza pandemic. *PloS One*, 12(7): e0180545. https://doi.org/10.1371/journal.pone.0180545

Yu D, Tang S, Lou Y (2016). Revisiting logistic population model for assessing periodically harvested closures. *Communications in Mathematical Biology and Neuroscience*, 2016: Article ID 14.

Yu D, Yaseen A, Luo X (2020). Neural network and deep learning methods for ehr data. In: *Statistics and Machine Learning Methods for EHR Data* (H Wu, JM Yamal, A Yaseen, V Maroufy, eds.), 253–271. Chapman and Hall/CRC.

Yu D, Zhu G, Wang X, Zhang C, Soltanalizadeh B, Wang X, et al. (2021). Assessing effects of reopening policies on COVID-19 pandemic in texas with a data-driven transmission model. *Infectious Disease Modelling*, 6: 461–473. https://doi.org/10.1016/j.idm.2021.02.001

Zhang K, Sun M, Han TX, Yuan X, Guo L, Liu T (2017). Residual networks of residual networks: Multilevel residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(6): 1303–1314. https://doi.org/10.1109/TCSVT.2017.2654543

Zhang T, Yao Z, Gholami A, Gonzalez JE, Keutzer K, Mahoney MW, et al. (2019). Anodev2: A coupled neural ode framework. *Advances in Neural Information Processing Systems*, 32.

Zhong YD, Dey B, Chakraborty A (2019). Symplectic ode-net: Learning hamiltonian dynamics with control. arXiv preprint: https://arxiv.org/abs/1909.12077.