Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.OA-2022-0104

Vol. **16**, No. 1, pp. 58-78
February 2023

Sparse Deep Neural Network for Nonlinear Partial Differential Equations

Yuesheng Xu[1] and Taishan Zeng[2,3,*]

[1] *Department of Mathematics and Statistics, Old Dominion University, Norfolk, Virginia 23529, USA*
[2] *School of Mathematical Science, South China Normal University, Guangzhou 510631, China*
[3] *Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou 510006, China*

Received 29 June 2022; Accepted (in revised version) 22 July 2022

Abstract. More competent learning models are demanded for data processing due to increasingly greater amounts of data available in applications. Data that we encounter often have certain embedded sparsity structures. That is, if they are represented in an appropriate basis, their energies can concentrate on a small number of basis functions. This paper is devoted to a numerical study of adaptive approximation of solutions of nonlinear partial differential equations whose solutions may have singularities, by deep neural networks (DNNs) with a sparse regularization with multiple parameters. Noting that DNNs have an intrinsic multi-scale structure which is favorable for adaptive representation of functions, by employing a penalty with multiple parameters, we develop DNNs with a multi-scale sparse regularization (SDNN) for effectively representing functions having certain singularities. We then apply the proposed SDNN to numerical solutions of the Burgers equation and the Schrödinger equation. Numerical examples confirm that solutions generated by the proposed SDNN are sparse and accurate.

AMS subject classifications: 68T07, 65D15, 65F50, 47A52, 65N50
Key words: Sparse approximation, deep learning, nonlinear partial differential equations, sparse regularization, adaptive approximation.

1. Introduction

The goal of this paper is to develop a sparse regularization deep neural network model for numerical solutions of nonlinear partial differential equations whose solutions may have singularities. We will mainly focus on designing a sparse regularization model by employing multiple parameters to balance sparsity of different layers and the

*Corresponding author. *Email addresses:* zengtsh@m.scnu.edu.cn (T. Zeng), y1xu@odu.edu (Y. Xu)

overall accuracy. The proposed ideas are tested in this paper numerically to confirm our intuition and more in-depth theoretical studies will be followed in a future paper.

Artificial intelligence especially deep neural networks (DNN) has received great attention in many research fields. From the approximation theory point of view, a neural network is built by functional composition to approximate a continuous function with arbitrary accuracy. Deep neural networks are proven to have better approximation by practice and theory due to their relatively large number of hidden layers. Deep neural network has achieved state-of-the-art performance in a wide range of applications, including speech recognition [11], computer vision [28], natural language processing [14], and finance [8]. For an overview of deep learning the readers are referred to monograph [20]. Recently, there was great interest in applying deep neural networks to the field of scientific computing, such as discovering the differential equations from observed data [34], solving the partial differential equation (PDE) [21,29,30,35], and problem aroused in physics [16]. Mathematical understanding of deep neural networks received much attention in the applied mathematics community. A universal approximation theory of neural network for Borel measurable function on compact domain is established in [9]. Some recent research studies the expressivity of deep neural networks for different function spaces [15], for example, Sobolev spaces, Barron functions, and Hölder spaces. There are close connections between deep neural network and traditional approximation methods, such as splines [13,37], compressed sensing [1], and finite elements [22,26]. Convergence of deep neural networks and deep convolutional neural networks are studied in [40] and [41] respectively. Some work aims at understanding the training process of DNN. For instance, in paper [10], the training process of DNN is interpreted as learning adaptive basis from data.

Traditionally, deep neural networks are dense and over-parameterized. A dense network model requires more memory and other computational resources during training and inference of the model. Increasingly greater amounts of data and related model sizes demand the availability of more competent learning models. Compared to dense models, sparse deep neural networks require less memory, less computing time and have better interpretability. Hence, sparse deep neural network models are desirable. On the other hand, animal brains are found to have hierarchical and sparse structures [19]. The connectivity of an animal brain becomes sparser as the size of the brain grows larger. Therefore, it is not only necessary but also natural to design sparse networks. In fact, it was pointed out in [25] that the future of deep learning relies on sparsity. Furthermore, over-parameterized and dense models tend to lead to overfitting and weakening the ability to generalize over unseen examples. Sparse models can improve accuracy of approximation. Sparse regularization is a popular way to learn the sparse solutions [5,38,39,42]. The readers are referred to [24] for an overview of sparse deep learning.

Although much progress has been made in theoretical research of deep learning, it remains a challenging issue to construct an effective neural network approximation for general function spaces using as few neuron connections or neurons as possible. Most of existing network structures are specific for a particular class of functions. In this

paper, we aim to propose a multi-scale sparse regularized neural network to approximate the function effectively. A neural network with multiple hidden layers can be viewed as a multi-scale transformation from simple features to complex features. The layer-by-layer composite of functions can be seen as a generalization of wavelet transforms [7,12,33]. For neurons in different layers, corresponding to different transformation scales, the corresponding features have different levels of importance. Imposing different regularization parameters for different scales was proved to be an effective way to deal with multi-scale regularization problems [3, 6, 32]. Inspired by multi-scale analysis, we propose a sparse regularization network model by applying different sparse regularization penalties to the neuron connections in different layers. During the training process, the neural network adaptively learns matrix weights from given data. By sparse optimization, many weight connections are automatically zero. The remaining neural networks composed of non-zero weights form the sparse deep neural network that we desire.

This paper is organized in five sections. In Section 2, we describe a multi-parameter regularization model for solving partial differential equations by using deep neural networks. We study in Section 3 the capacity of the proposed multi-parameter regularization in adaptive representing functions having certain singularities. In Section 4, we investigate numerical solutions of nonlinear partial differential equations by using the proposed SDNN model. Specifically, we consider two equations: the Burgers equation and the Schrödinger equation since solutions of these two equations exhibit certain types of singularities. Finally, a conclusion is drawn in Section 5.

2. A sparse DNN model for solving partial differential equations

In this section, we propose a sparse DNN model for solving nonlinear partial differential equations (PDEs).

We begin with describing the PDE and its boundary, initial conditions to be considered in this paper. Suppose that Ω is an open domain in $\mathbb{R}^d$. By Γ we denote the boundary of the domain Ω. Let $\mathcal{F}$ denote a nonlinear differential operator, $\mathcal{I}$ the initial condition operator, and $\mathcal{B}$ the boundary operator. We consider the following boundary/initial value problem of the nonlinear partial differential equation:

$$\mathcal{F}\big(u(t,x)\big) = 0, \quad x \in \Omega, \quad t \in [0,T], \tag{2.1}$$

$$\mathcal{I}\big(u(0,x)\big) = 0, \quad x \in \Omega, \quad t = 0, \tag{2.2}$$

$$\mathcal{B}\big(u(t,x)\big) = 0, \quad x \in \Gamma, \quad t \in [0,T], \tag{2.3}$$

where $T > 0$, the data u on Γ and $t = 0$ are given and u in Ω is the solution to be learned. The formulation (2.1)-(2.3) covers a broad range of problems including conservation laws, reaction–diffusion equations, and Navier–Stokes equations. For example, the one-dimensional Burgers equation can be recognized as

$$\mathcal{F}(u) := \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2}.$$

The goal of this paper is to develop a sparse DNN model for solving problem (2.1). We will conduct numerical study of the proposed model by applying it to two equations, the Burgers equation and the Schrödinger equation, of practical importance.

Now, we present the sparse DNN model with multi-parameter regularization. We first recall the feed forward neural network (FNN). A neural network can be viewed as a composition of functions. A FNN of depth D is defined to be a neural network with an input layer, $D-1$ hidden layers, and an output layer. A neural network with more than two hidden layers is usually called a deep neural network (DNN). Suppose that there are d_i neurons in the i-th hidden layer. Let $W_i \in \mathbb{R}^{d_i \times d_{i-1}}$ and $b_i \in \mathbb{R}^{d_i}$ denote, respectively, the weight matrix and bias vector of the i-th layer. By $x_0 := x \in \mathbb{R}^{d_0}$ we denote the input vector and by $x_{i-1} \in \mathbb{R}^{d_{i-1}}$ we denote the output vector of the $(i-1)$-th layer. For the i-th hidden layer, we define the affine transform $L_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ by

$$L_i(x_{i-1}) := W_i x_{i-1} + b_i, \quad i = 1, 2, \ldots, D.$$

For an activation function σ_i, the output vector of the i-th hidden layer is defined as

$$x_i := \sigma_i\big(L_i(x_{i-1})\big).$$

Given nonlinear activation functions σ_i, $i = 1, 2, \ldots, D-1$, the feed forward neural network $\mathcal{N}_\Theta(x)$ of depth D is defined as

$$\mathcal{N}_\Theta(x) := L_D \circ \sigma_{D-1} \circ L_{D-1} \circ \cdots \circ \sigma_1 \circ L_1(x), \tag{2.4}$$

where $\circ$ denotes the composition operator and $\Theta := \{W_i, b_i\}_{i=1}^D$ is the set of trainable parameters in the network.

We first describe the physics-informed neural network (PINN) model introduced in [35] for solving the partial differential equation (2.1). We denote by $Loss_{PDE}$ the loss of training data on the partial differential equation (2.1). We choose N_f collocation points (t_f^i, x_f^i) by randomly sampling in domain Ω using a sampling method such as Latin hypercube sampling [23]. We then evaluate $\mathcal{F}(\mathcal{N}_\Theta(t_f^i, x_f^i))$ for $i = 1, 2, \ldots, N_f$ and define

$$Loss_{PDE} := \frac{1}{N_f} \sum_{i=1}^{N_f} \big|\mathcal{F}\big(\mathcal{N}_\Theta(t_f^i, x_f^i)\big)\big|^2,$$

where $\mathcal{F}$ is the operator for the partial differential equation (2.1).

We next describe the loss function for the boundary/initial condition. We randomly sample N_0 points x_0^i for the initial condition (2.2), N_b points $\{t_b^i, x_b^i\}$ for the boundary condition (2.3). The loss function $Loss_0$ related to the initial value condition is given by

$$Loss_0 := \frac{1}{N_0} \sum_{i=1}^{N_0} \big|\mathcal{I}\big(\mathcal{N}_\Theta(0, x_0^i)\big)\big|.$$

The loss function $Loss_b$ pertaining to the boundary value is given as

$$Loss_b := \frac{1}{N_b} \sum_{i=1}^{N_b} \big|\mathcal{B}\big(\mathcal{N}_\Theta(t_b^i, x_b^i)\big)\big|, \quad x_b^i \in \Gamma.$$

Adding the three loss functions $Loss_{PDE}$, $Loss_0$, and $Loss_b$ together gives rise to the PINN model

$$\min_{\Theta}\left\{\frac{1}{N_f}\sum_{i=1}^{N_f}|\mathcal{F}(\mathcal{N}_\Theta(t_f^i,x_f^i))|^2+\frac{1}{N_0}\sum_{i=1}^{N_0}|\mathcal{I}(\mathcal{N}_\Theta(0,x_0^i))|+\frac{1}{N_b}\sum_{i=1}^{N_b}|\mathcal{B}(\mathcal{N}_\Theta(t_b^i,x_b^i))|\right\}, \quad (2.5)$$

where $\Theta := \{W_i, b_i\}_{i=1}^D$.

The neural network learned from (2.5) is often dense and may be over-parameterized. Moreover, training data are often contaminated with noise. When noise presents, over-parameterized models may overfit training data samples and result in bad generalization to the unseen samples. The problem of over-fitting is often overcome by adding a regularization term

$$Loss := Loss_{PDE} + \beta(Loss_0 + Loss_b) + \text{Regularization}.$$

The ℓ_1- and ℓ_2-norms are popular choices for regularization. Design of the regularization often makes use of prior information of the solution to be learned. It is known [4, 17, 42] that the ℓ_1-norm can promote sparsity. Hence, the ℓ_1-norm regularization not only has many advantages over the ℓ_2-norm regularization, but also leads to sparse models which can be more easily interpreted. Therefore, we choose to use the ℓ_1-norm as the regularizer in this study. Furthermore, we observe that DNNs have an intrinsic multiscale structure whose different layers represent different scales of information, which will be validated later by numerical studies. In fact, we will demonstrate in the next section that a smooth function or smooth parts of a function can be represented by a DNN with sparse weight matrices. This is because a smooth part of a function contains redundant information, which can be described very well by a few parameters, and only non-smooth parts of a function require more parameters to describe them. In other words, by properly choosing regularization, DNNs can lead to adaptive sparse representations of functions having certain singularities. With this understanding, we construct an adaptive representation of a function, especially for a function having certain singularity by adopting a sparse regularization model. Our idea for the adaptive representation is to impose different sparsity penalties for different layers. Specifically, we propose a multiscale-like sparse regularization using the ℓ_1-norm of the weight matrix for each layer with a different parameter for a different layer. The regularization with multiple parameters allows us to represent a function in a multiscale-like neural network which is determined by sparse weight matrices having different sparsity at different layers. Such a regularization added to the loss function will enable us to robustly extract critical information of the solution of the PDE.

We now describe the proposed regularization. For layer i, we denote by $W_i^{k,j}$ the (k,j)-th entry of matrix W_i, the entry in the k-th row and the j-th column. For this reason, we adopt the ℓ_1-norm of matrix W_i defined by

$$\|W_i\|_1 := \sum_{k=1}^{d_i}\sum_{j=1}^{d_{i-1}}\left|W_i^{k,j}\right|$$

as our sparse regularization. Considering that different layers of the neural network play different roles in approximation of a function, we introduce here a multi-parameter regularization model

$$\text{Regularization} := \sum_{i=1}^{D} \alpha_i \|W_i\|_1, \tag{2.6}$$

where α_i are nonnegative regularization parameters. The use of different parameters for weight matrices of different layers in the regularization term (2.6) allows us to penalize the weight matrices at different layers of the neural network differently in order to extract the multiscale representation of the solution to be learned. That is, for a fixed i, parameter α_i determines the sparsity of weight matrix W_i. The larger the parameter α_i, the more sparse the weight matrix W_i is. The regularized loss function takes the form

$$Loss := Loss_{PDE} + \beta(Loss_0 + Loss_b) + \sum_{i=1}^{D} \alpha_i \|W_i\|_1. \tag{2.7}$$

The parameters $\Theta := \{W_i, b_i\}_{i=1}^{D}$ of the neural network $\mathcal{N}_\Theta(t,x)$ are learned by minimizing the loss function

$$\min_{\Theta} \left\{ \frac{1}{N_f} \sum_{i=1}^{N_f} \left| \mathcal{F}\big(\mathcal{N}_\Theta(t_f^i, x_f^i)\big) \right|^2 + \beta(Loss_0 + Loss_b) + \sum_{i=1}^{D} \alpha_i \|W_i\|_1 \right\}. \tag{2.8}$$

Truncating the weights of the layers close to the input layer has an impact on all subsequent layers. In practice, we usually set smaller regularization parameters in layers close to the input and larger regularization parameters in layers close to the output. The resulting neural network will exhibit denser weight matrices near the input layer and sparser weight matrices near the output layer. This network structure reflects the multi-scale nature of neural networks and is automatically learned by sparse regularization.

Appropriate choices of the regularization parameters are key to achieve good prediction results. We need to balance sparsity and prediction accuracy. Since there are multiple regularization parameters, the regularization parameters are chosen by grid search layer by layer in this paper. In practice, we first choose the regularization parameters close to the output layer, and then gradually choose the regularization coefficients close to the input layer.

We refer Eq. (2.8) as to the sparse DNN (SDNN) model for the partial differential equation. Upon solving the minimization problem (2.8), we obtain an approximate solution $u(t,x) := \mathcal{N}_\Theta(t,x)$ with sparse weight matrices. When the regularization parameters α_i are all set to 0, the SDNN model (2.8) reduces to the PINN model introduced in [35]. We will compare numerical performance of the proposed SDNN model with that of PINN model, for both the Burgers equation and the Schrödinger equation.

3. Function adaptive approximation by the SDNN model

We explore in this section the capacity of the proposed multi-parameter regularization in adaptive representing functions that have certain singularities. We will first reveal that a DNN indeed has an intrinsic multiscale-like structure which is desirable for representing non-smooth functions. We demonstrate in our numerical studies that the proposed SDNN model can reconstruct neural networks which approximate functions in the same accuracy order with nearly the same order of network complexity, regardless the smoothness of the functions. We include in this section a numerical study of reconstruction of black holes by the proposed SDNN model. In this section, we use the rectified linear unit (ReLU) function

$$\text{ReLU}(x) := \max\{0, x\}, \quad x \in \mathbb{R}$$

as an activation function.

We start with data fitting problem. Given training points (x_i, y_i), $i = 1, 2, \ldots, N$, a non-regularized neural network is determined by minimizing the regression error, that is,

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} |\mathcal{N}_\Theta(x_i) - y_i|^2. \tag{3.1}$$

The multi-parameter sparse regularization DNN model for the data fitting problem reads

$$\min_{\Theta} \left\{ \frac{1}{N} \sum_{i=1}^{N} |\mathcal{N}_\Theta(x_i) - y_i|^2 + \sum_{i=1}^{D} \alpha_i \|W_i\|_1 \right\}, \tag{3.2}$$

where α_i are nonnegative regularization parameters and W_i are weight matrices.

In examples to be presented in this section and the section that follows, the network structure is described by the number of neurons in each layer. Specifically, we use the notation $[d_0, d_1, \ldots, d_D]$ to describe networks that have one input layer, $D-1$ hidden layers and one output layer, with $d_0, d_1, \ldots, d_D$ number of neurons, respectively. The regularization parameters, which will be presented as a vector $\alpha := [\alpha_1, \alpha_2, \ldots, \alpha_D]$, are chosen so that best results are obtained. We will use the relative L_2 error to measure approximation accuracy. Suppose that y_i is the exact value of function f to be approximated at x_i, that is, $y_i = f(x_i)$, and suppose that $\hat{y}_i := \mathcal{N}_\Theta(x_i)$ is the output of the neural network approximation of f. We let

$$y := [y_1, y_2, \ldots, y_N], \quad \hat{y} := [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_N],$$

and define the error by $\|y - \hat{y}\|_2 / \|y\|_2$. Sparsity of the weight matrices is measured by the percentage of zero entries in the weight matrices W_i. In our computation, we set a weight matrix entry $W_i^{k,j} = 0$, if $|W_i^{k,j}| < \epsilon$, where ϵ is small positive number. In our numerical examples, we set $\epsilon := 0.001$ by default. For all numerical examples, the non-smooth, non-convex optimization problem (3.2) is solved by the Adam algorithm, which is an improved version of the stochastic gradient descent algorithm proposed in [27] for training deep learning models.

3.1. Intrinsic adaptivity of the SDNN model

We first investigate whether the SDNN model (3.2) can generate a network that has an intrinsic adaptive representation of a function. That is, a function generated by the model has a multiscale-like structure so that the reconstructed neural networks approximate functions in the same accuracy order with nearly the same order of network complexity, regardless the smoothness of the functions. In particular, when a function is singular a sparse network with higher layers is generated to capture the higher resolution information of the function. The complexity of the network is nearly proportional to the reciprocal of the approximation error regardless whether the function is smooth or not. In this experiment, we consider two examples: (1) one-dimensional functions and (2) two-dimensional functions.

In our first example, we consider approximation of the quadratic function

$$f(x) := x^2, \tag{3.3}$$

and the piecewise quadratic function

$$f_d(x) = \begin{cases} x^2 + 1, & x \geq 0, \\ x^2, & x < 0 \end{cases} \tag{3.4}$$

by SDNN. Note that the function defined by (3.3) is smooth and the function by (3.4) has a jump discontinuity at the point 0. We applied the sparse regularized network having the architecture $[1, 10, 10, 10, 10, 1]$ to learn these functions. We divide the interval $[-2, 2]$ by the nodes $x_j := -2 + jh$, for $j := 0, 1, \ldots, 200$, with $h := 1/50$, and sample the functions f at x_j. The test set is $\{(x_k, f(x_k))\}$, where $x_k := -2 + kh$, $h := 1/30$, $k = 0, 1, \ldots, 120$. The network is trained by the Adam algorithm with epochs $20,000$ and initial learning rate 0.001.

For function (3.3), regularization parameters are set to be $[0, 1e\text{-}4, 1e\text{-}4, 1e\text{-}3, 1e\text{-}3]$. We obtain the prediction error $5.94e\text{-}3$ for the test set. Sparsity of the resulting weight matrices is $[0.0\%, 87.0\%, 95.0\%, 98.0\%, 90.0\%]$ and the number of nonzero weight matrix entries is 31. Fig. 1 (left) shows the reconstructed SDNN for the function defined by (3.3).

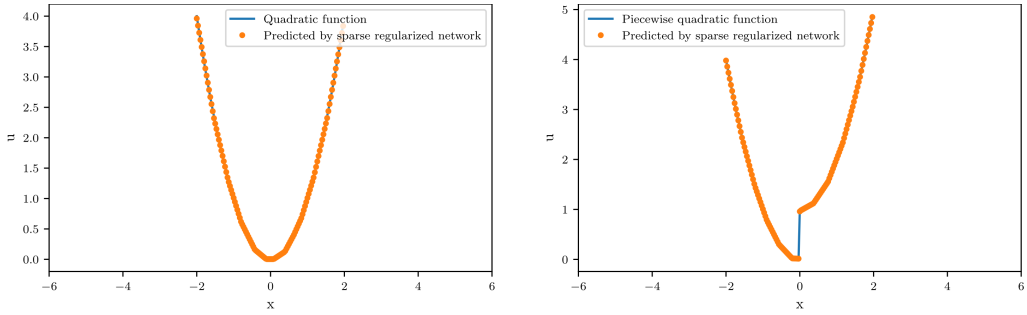

Figure 1: Numerical results of SDNN: for function (3.3) (left); for function (3.4) (right).

Table 1: Numerical result for quadratic function (3.3) and piecewise quadratic function (3.4) with network structure [1, 10, 10, 10, 10, 1].

Results for function (3.3)	Regularization parameters	$[0, 1e\text{-}4, 1e\text{-}4, 1e\text{-}3, 1e\text{-}3]$
	Relative L_2 error	$5.94e\text{-}3$
	Sparsity of weight matrices	$[0.0\%, 87.0\%, 95.0\%, 98.0\%, 90.0\%]$
	No. of nonzero entries	31
Results for function (3.4)	Regularization parameters	$[1e\text{-}5, 1e\text{-}4, 1e\text{-}4, 1e\text{-}4, 1e\text{-}3]$
	Relative L_2 error	$5.42e\text{-}3$
	Sparsity of weight matrices	$[50\%, 93.0\%, 88.0\%, 93.0\%, 90.0\%]$
	No. of nonzero entries	32

For function (3.4) the regularization parameters are chosen as $[1e\text{-}5, 1e\text{-}4, 1e\text{-}4, 1e\text{-}4, 1e\text{-}3]$. We obtain the prediction error $5.42e\text{-}3$ for the test set. Sparsity of the resulting weight matrices is $[50\%, 93.0\%, 88.0\%, 93.0\%, 90.0\%]$ and the number of nonzero weight matrix entries is 32. The reconstructed function is shown in Fig. 1 (right).

Numerical results for both functions (3.3) and (3.4) are summarized in Table 1. These results demonstrate that even though the function (3.4) has a jump discontinuity at the point 0, the proposed SDNN model can generate a network with nearly the same number of nonzero weight matrix entries and with the same accuracy as those for the smooth function (3.3). This shows that the proposed SDNN model has a good adaptive approximation property.

In our second example, we consider approximation of two-dimensional functions, once again one smooth function and one discontinuous function. We study smooth function

$$g(x, y) := e^{2x+y^2}, \tag{3.5}$$

whose image is illustrated in Fig. 2 (left), and piecewise function

$$g_d(x, y) = \begin{cases} e^{2x+y^2} + 1, & x \geq 0, \\ e^{2x+y^2}, & x < 0, \end{cases} \tag{3.6}$$

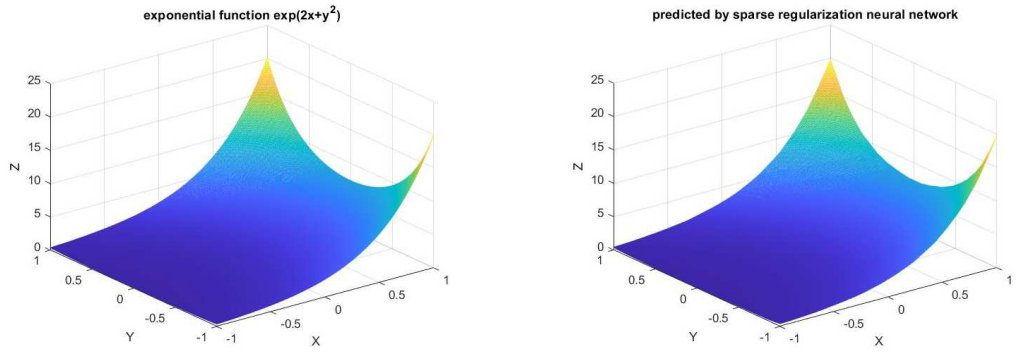

Figure 2: Left: Image of function e^{2x+y^2}. Right: Predicted by sparse regularized neural network.

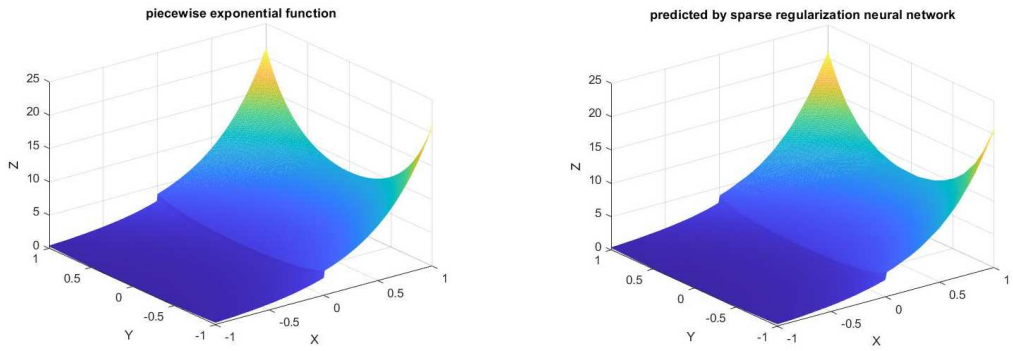

Figure 3: Left: Image of piecewise discontinuous function (3.6). Right: Predicted by sparse regularized neural network.

whose image is illustrated in Fig. 3 (left). Note that function (3.5) is smooth and function (3.6) has a jump discontinuity along $x = 0$.

For these two functions, the training data set is composed of grid points $[-1, 1] \times [-1, 1]$ uniformly discretized with step size $1/200$ on x- and y-direction, and the test set is composed of grid points $[-1, 1] \times [-1, 1]$ uniformly discretized with step size $1/300$ on the x- and y-directions. The network has 2 inputs, 4 hidden layers, and 1 output, with the architecture $[2, 20, 20, 20, 20, 1]$. For each hidden layer, there are 20 neurons. The initial learning rate for Adam is set to 0.001. The batch size is equal to 1024.

For function (3.5) we set the sparse regularization parameters as $[0, 1e\text{-}6, 1e\text{-}4, 1e\text{-}4, 1e\text{-}4]$. After $10,000$ epochs training, the sparsity of weight matrices is $[0.0\%, 68.5\%, 95.75\%, 97.75\%, 80.0\%]$ and the number of nonzero weight matrix entries is 178. The prediction error for the test set is $4.38e\text{-}3$. For function (3.6), the regularization parameters are set to be $[1e\text{-}4, 1e\text{-}5, 1e\text{-}5, 1e\text{-}4, 1e\text{-}4]$. The sparsity of weight matrices after regularization are $[60.0\%, 71.75\%, 81.75\%, 97.5\%, 90.0\%]$ and the number of nonzero weight matrix entries is 206. The prediction error for sparse regularized deep neural network is $4.27e\text{-}3$, which is even slightly better than that for function (3.5). The images of the reconstructed functions are shown respectively in Figs. 2 and 3 (right). Numerical results for this example are reported in Table 2.

Table 2: Numerical result for two-dimensional function (3.5) and (3.6) with network structure [2, 20, 20, 20, 20, 1].

Results for function (3.5)	Regularization parameters	$[0, 1e\text{-}6, 1e\text{-}4, 1e\text{-}4, 1e\text{-}4]$
	Relative L_2 error	$4.38e\text{-}3$
	Sparsity of weight matrices	$[0.0\%, 68.5\%, 95.75\%, 97.75\%, 80.0\%]$
	No. of nonzero connections	178
Results for function (3.6)	Regularization parameters	$[1e\text{-}4, 1e\text{-}5, 1e\text{-}5, 1e\text{-}4, 1e\text{-}4]$
	Relative L_2 error	$4.27e\text{-}3$
	Sparsity of weight matrices	$[60.0\%, 71.75\%, 81.75\%, 97.5\%, 90.0\%]$
	No. of nonzero connections	206

The numerical results presented in this subsection indicate that indeed the proposed SDNN model has an excellent adaptivity property in the sense that it generates networks with nearly the same number of nonzero weight matrix entries and the same order of approximation accuracy for functions regardless their smoothness.

3.2. An example of adaptive function approximation by the SDNN model

The second experiment is designed to test the sparsity of the network learned from the SDNN model (3.2) and the model's generalization ability. Specifically, in this example, we demonstrate that the sparse model (3.2) leads to a sparse DNN with higher accuracy in comparison to the standard DNN model (3.1). We consider the absolute value function

$$y = f(x) := |x| \quad \text{for} \quad x \in \mathbb{R}.$$

Note that function f is not differentiable at $x = 0$.

We adopt the same network architecture, that is, 1 input layer, 2 hidden layers, 1 output layer and each hidden layer containing 5 neurons, for both the standard DNN model (3.1) and the SDNN model (3.2). The training set is composed of equal-distance grid points laying in $[-2, 2]$ with step size 0.01. The test set is composed of equal-distance grid points in $[-5, 5]$ with step size 0.1. For the sparse regularized network, the regularization parameters are set as $[1e\text{-}4, 1e\text{-}3, 1e\text{-}3]$. For both the standard DNN model and the SDNN model, the number of epoch equals $10,000$. The initial learning rate is set to 0.001.

We present numerical results of this experiment in Table 3, where we compare errors and sparsity of the functions learned from the two models. Clearly, the network learned from the standard DNN model is non-sparse: all entries of its weight matrices are nonzero. While the network learned from the SDNN model has a good sparsity property: There are only 1 non-zero entries in W_3 and 2 non-zero entries in W_2 in the network learned from the SDNN model. Note that the absolution value function is the linear composition of two ReLU functions, that is $|x| = \text{ReLU}(x) + \text{ReLU}(-x)$. The SDNN model is able to find a linear combination of the two functions to represent the function $f(x) := |x|$ but the standard DNN model fails to do so.

We plot the graphs of the reconstructed functions by the standard DNN model (3.1) and the SDNN model (3.2) in Figs. 4 and 5, respectively. It can be seen from Fig. 4 that the function reconstructed by the standard DNN model (3.1) has large errors in the

Table 3: Approximation of the absolute value function by a SDNN with regularization parameters $[1e\text{-}4, 1e\text{-}3, 1e\text{-}3]$.

	Relative L_2 error	Sparsity of weight matrices $[W_1, W_2, W_3]$
Standard DNN model	$5.58e\text{-}2$	$[0\%, 0\%, 0\%]$
SDNN model	$1.87e\text{-}3$	$[20\%, 92\%, 80\%]$

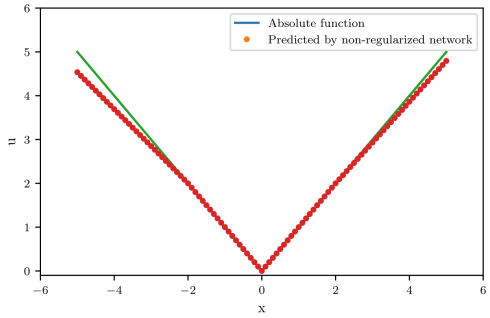

Figure 4: Reconstruction of function $f(x) := |x|$ by the standard DNN model (3.1).

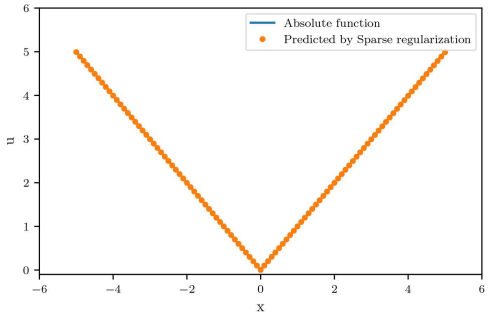

Figure 5: Reconstruction of function $f(x) := |x|$ by the SDNN model (3.2).

interval $[3, 5]$. Fig. 5 shows that the function reconstructed by the SDNN model (3.2) almost coincides with the original function. This example indicates that the SDNN model (3.2) has better generalization ability than the standard DNN model (3.1).

3.3. Reconstruction of a black hole

In this example, we consider reconstruction of the image of a black hole by the SDNN model. Specifically, we compare numerical results and reconstructed image quality of the SDNN model with those of the standard DNN model. We choose a color image of the black hole shown in Fig. 6 (left), which is turned into a gray image shown in Fig. 6 (right). The image has the size 128×128 and can be represented as a two-dimensional discrete function. The value of the gray image at the point (x_1, x_2) is defined as a $f_{image}(x_1, x_2)$, $x_1, x_2 = 1, 2, \ldots, 128$. The function clearly has singularities.

The network architecture that we used for the construction is

$$[2, 100, 100, 100, 100, 100, 100, 1].$$

We randomly choose $5,000$ points $(x_1^i, x_2^i, f_{image}(x_1^i, x_2^i))$ by uniform sampling, $i = 1, 2, \ldots, 5,000$, from the image of the black hole to train both the standard neural network and the sparse regularized network. The optimizer is chosen as the Adam algorithm with batch size $1,024$. The number of epoch is $40,000$. The patience parameter of early stopping is 200. Prediction results by the standard DNN model and by the SDNN model are shown respectively on Fig. 7 (left) and (right).

Error images of the two models are presented in Fig. 8, from which it can be seen that the sparse network has a smaller reconstruction error. The prediction error of the fully connected network is $9.66e$-3. For the sparse regularized neural network, the regularized parameters are set to be $[1e$-$9, 1e$-$9, 1e$-$9, 1e$-$9, 1e$-$8, 1e$-$8, 1e$-$8]$. The prediction error of the sparse regularized network is $9.28e$-3. The sparsity of the weight matrices are $[44.0\%, 78.3\%, 78.4\%, 80.3\%, 96.5\%, 98.2\%, 84.0\%]$. It shows that the sparse regularized network uses fewer neurons and has smaller prediction error. This indicates that by using the proposed multi-parameter sparse regularization, the deep neural network has the ability of multi-scale and adaptive learning.

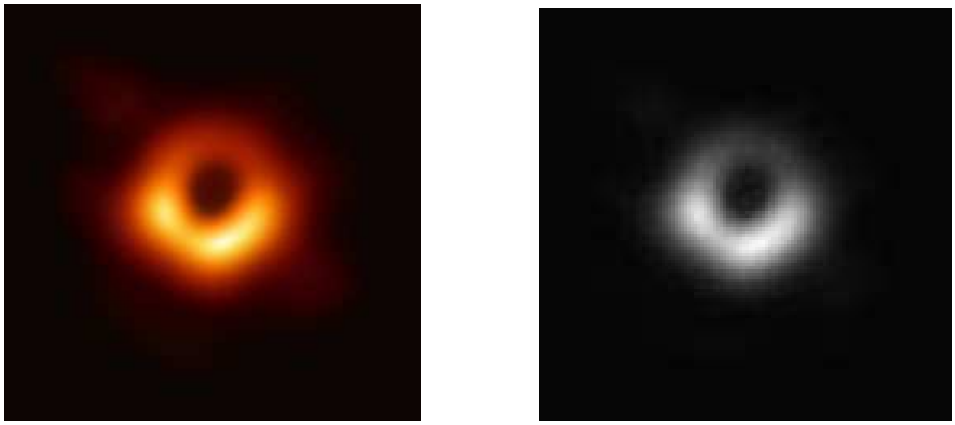

Figure 6: Left: Color image of the black hole. Right: Gray image of the black hole.

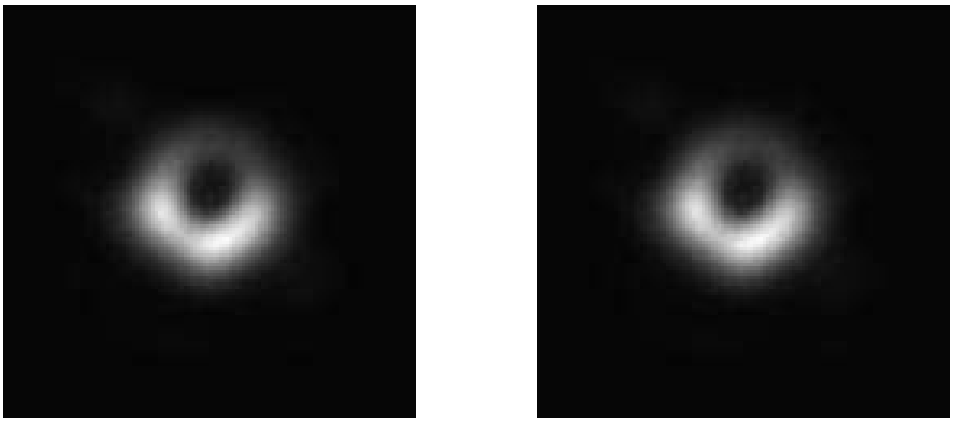

Figure 7: Images of the black hole reconstructed. By the standard DNN model (left) and by the SDNN model (right).

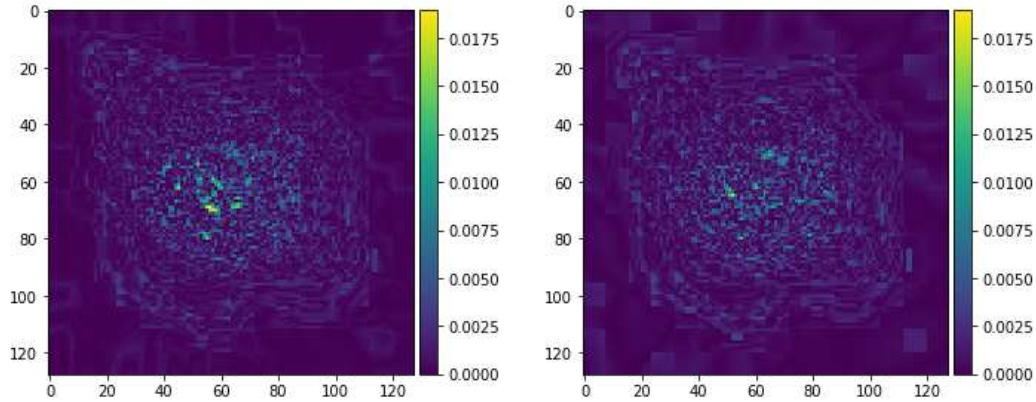

Figure 8: Reconstruction errors of the black hole. By the standard DNN model (left) and by the SDNN model (right).

Table 4: Numerical results of the black hole reconstructed by the standard DNN model vs. the SDNN model.

Standard DNN model	
Relative L_2 error	Sparsity of weight matrices
$9.66e$-3	$[0\%, 0\%, 0\%, 0\%, 0\%, 0\%, 0\%]$
SDNN model	
Relative L_2 error	Sparsity of weight matrices
$9.28e$-3	$[44.0\%, 78.3\%, 78.4\%, 80.3\%, 96.5\%, 98.2\%, 84.0\%]$

4. Numerical solutions of partial differential equations

We study in this section numerical performance of the proposed SDNN model for solving partial differential equations. We consider two equations: the Burgers equation and the Schrödinger equation. For both of these two equations, we choose the hyperbolic tangent (tanh) function defined by

$$\tanh(x) := \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad x \in \mathbb{R}$$

as the activation function to build networks for our approximate solutions due to its differentiability which is required by the differential equations.

4.1. The Burgers equation

The Burgers equation has attracted much attention since it is often used as simplified model for turbulence and shock waves [31]. It is well-known that the solution of this equation presents a jump discontinuity (a shock wave), even though the initial function is smooth.

In this example, we consider the following one-dimensional Burgers equation:

$$u_t(t,x) + u(t,x)u_x(t,x) - \frac{0.01}{\pi}u_{xx}(t,x) = 0, \quad t \in (0,1], \quad x \in (-1,1), \tag{4.1}$$

$$u(0,x) = -\sin(\pi x), \tag{4.2}$$

$$u(t,-1) = u(t,1) = 0. \tag{4.3}$$

The analytic solution of this equation, known in [2], will be used as our exact solution for comparison. Indeed, the analytic solution has the form

$$u(t,x) := -\frac{\int_{-\infty}^{+\infty} \sin\pi(x-\eta)h(x-\eta)\exp\left(-\eta^2/4\nu t\right)\mathrm{d}\eta}{\int_{-\infty}^{+\infty} h(x-\eta)\exp\left(-\eta^2/4\nu t\right)\mathrm{d}\eta}, \quad t \in [0,1], \quad x \in [-1,1],$$

where $\nu := 0.01/\pi$ and $h(y) := \exp(-\cos\pi y/2\pi\nu)$. A neural network solution of Eqs. (4.1)-(4.3) was obtained recently from the standard DNN model in [35].

We apply the setting (2.1)-(2.3) with

$$\mathcal{F}\big(u(t,x)\big) := u_t(t,x) + u(t,x)u_x(t,x) - \frac{0.01}{\pi}u_{xx}(t,x), \quad t \in (0,1], \quad x \in (-1,1).$$

Let $\{x_0^i, u_0^i\}_{i=1}^{N_0}$ denote the training data of u satisfying initial condition (4.2), that is, $u_0^i = -\sin(\pi x_0^i)$. Let $\{t_{b_1}^i\}_{i=1}^{N_{b_1}}$ and $\{t_{b_2}^i\}_{i=1}^{N_{b_2}}$ be the collocation points related to boundary condition (4.3) for $x = -1$ and $x = 1$ respectively. We denote by $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ the collocation points for $\mathcal{F}(u(t,x))$ in $[0,1] \times (-1,1)$. The sparse deep neural network $\mathcal{N}_\Theta(t,x)$ are learned by model (2.8) with

$$Loss_0 = \frac{1}{N_0}\sum_{i=1}^{N_0}\left|\mathcal{N}_\Theta(0,x_0^i) - u_0^i\right|^2,$$

$$Loss_b = \frac{1}{N_{b_1}}\sum_{i=1}^{N_{b_1}}\left|\mathcal{N}_\Theta(t_{b_1}^i,-1)\right| + \frac{1}{N_{b_2}}\sum_{i=1}^{N_{b_2}}\left|\mathcal{N}_\Theta(t_{b_2}^i,1)\right|.$$

In this experiment, 100 data points are randomly selected from boundary and initial data points, among which $N_{b_1} = 25$ points are located on the boundary $x = -1$, $N_{b_2} = 23$ points on the boundary $x = 1$, and $N_0 = 52$ points on the initial line $t = 0$. The distribution of random collocation points is shown in the top of Fig. 9. The number of

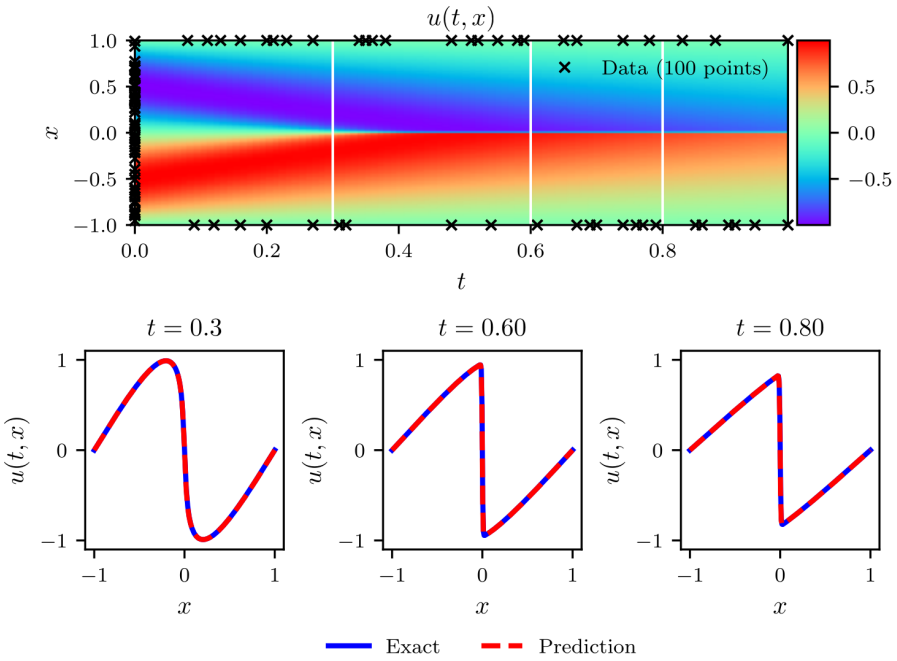

Figure 9: Burgers equation. Top: The training data and predicted solution $u(t,x)$ for sparse deep neural network with [2, 50, 50, 50, 1], regularization parameter $\alpha = [1e\text{-}6, 1e\text{-}6, 1e\text{-}6, 1e\text{-}4]$, $\beta = 20$. Bottom: Predicted solution $u(t,x)$ at time $t = 0.3$, $t = 0.6$, and $t = 0.8$.

Table 5: The Burgers equation. A neural network of 4 layers, with network architecture [2, 50, 50, 50, 1].

Algorithms	Parameters α sparsity of weight matrices	Relative L_2 error
PINN	No regularization $[0.0\%, 0.2\%, 0.5\%, 0.0\%]$	$2.45e$-2
SDNN ($\beta = 20$)	$[1e$-6, $1e$-6, $1e$-6, $1e$-4] $[13.0\%, 61.9\%, 71.2\%, 62.0\%]$	$1.68e$-3

Table 6: The Burgers equation. Neural networks of 8 layers with network architecture [2, 50, 50, 50, 50, 50, 50, 50, 1].

Algorithms	Parameters α & sparsity of weight matrices	Relative L_2 error
PINN	No regularization $[0.0\%, 0.8\%, 0.6\%, 0.6\%, 0.8\%, 0.6\%, 0.4\%, 0.0]$	$3.39e$-3
SDNN ($\beta = 10$)	$[0, 0, 0, 0, 1e$-7, $1e$-10, $1e$-6, $1e$-5] $[0.0\%, 0.7\%, 0.8\%, 0.7\%, 15.6\%, 0.5\%, 93.8\%, 94.0\%]$	$1.45e$-4
SDNN ($\beta = 10$)	$[1e$-6, $1e$-6, $1e$-6, $1e$-6, $1e$-6, $1e$-6, $1e$-5, $1e$-5] $[25.0\%, 78.6\%, 85.3\%, 82.8\%, 79.5\%, 84.0\%, 98.6\%, 94.0\%]$	$4.83e$-4

collocation points of the partial differential equation is $N_f = 10,000$ by employing the Latin hypercube sampling method. The test set is composed of grid points $[0,1]\times[-1,1]$ uniformly discretized with step size $1/100$ on the t-direction and step size $2/255$ on the x-direction.

We use two different network architectures $[2, 50, 50, 50, 1]$ and $[2, 50, 50, 50, 50, 50, 50, 50, 1]$ for DNNs. We choose Adam as the optimizer for both neural networks. The number of epoch is $30,000$. The initial learning rate is set to 0.001. Numerical results of these two networks presented respectively in Tables 5 and 6 show that the proposed SDNN model outperforms the PINN model in both weight matrix sparsity and approximation accuracy.

4.2. The Schrödinger equation

The Schrödinger equation is the most essential equation of non-relativistic quantum mechanics. It plays an important role in studying nonlinear optics, Bose-Einstein condensates, protein folding and bending. It is also a model equation for studying waves propagation and soliton [36]. In this subsection, we consider a one-dimensional Schrödinger equation with periodic boundary conditions

$$\begin{aligned} &iu_t(t,x) + 0.5u_{xx}(t,x) + |u(t,x)|^2u(t,x) = 0, \quad t \in (0,\pi/2], \quad x \in (-5,5),\\ &u(0,x) = 2\,\mathrm{sech}(x),\\ &u(t,-5) = u(t,5),\\ &u_x(t,-5) = u_x(t,5). \end{aligned} \tag{4.4}$$

Note that the solution u of problem (4.4) is a complex-valued function. The goal of this study is to test the effectiveness of the proposed SDNN model in solving complex-valued nonlinear differential equations with periodic boundary conditions, with a comparison to the standard DNN model recently developed in [35].

Problem (4.4) falls into the setting (2.1)-(2.3) with

$$\mathcal{F}\big(u(t,x)\big) := iu_t(t,x) + 0.5u_{xx}(t,x) + |u(t,x)|^2 u(t,x), \quad t \in (0,\pi/2], \quad x \in (-5,5).$$

Let ψ and ϕ be respectively the real part and imaginary part of the solution u of problem (4.4). We intend to approximate the solution $u(t,x)$ by a neural network $\mathcal{N}_\Theta(t,x)$ with two inputs (t,x) and two outputs which approximate $\psi(t,x)$ and $\phi(t,x))$, respectively. Let $\{x_0^i, u_0^i\}_{i=1}^{N_0}$ denote the training data to enforce the initial condition at time $t=0$, that is, $u_0^i = \mathrm{sech}(x_0^i)$, $\{t_b^i\}_{i=1}^{N_b}$ the collocation points on the boundary $x=-5$ and $x=5$ to enforce the periodic boundary conditions, and $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ the collocation points in $(0,\pi/2] \times (-5,5)$. These collocation points were generated by the Latin hypercube sampling method. We then learn the neural network $\mathcal{N}_\Theta(t,x)$ by model (2.8) with

$$Loss_0 := \frac{1}{N_0}\sum_{i=1}^{N_0} \left|\mathcal{N}_\Theta\left(0,x_0^i\right) - u_0^i\right|^2,$$

$$Loss_b := \frac{1}{N_b}\sum_{i=1}^{N_b}\left(\left|\mathcal{N}_\Theta\left(t_b^i,-5\right) - \mathcal{N}_\Theta\left(t_b^i,5\right)\right|^2 + \left|\frac{\partial \mathcal{N}_\Theta}{\partial x}\left(t_b^i,-5\right) - \frac{\partial \mathcal{N}_\Theta}{\partial x}\left(t_b^i,5\right)\right|^2\right).$$

A reference solution of problem (4.4) is solved by a Fourier spectral method using the Chebfun package [18]. Specifically, we obtain the reference solution by using 256 Fourier modes for space discretization and an explicit fourth-order Runge-Kutta method (RK4) with time-step $\Delta t := (\pi/2) \times 10^{-6}$ for time discretization. For more details of the discretization of Schrödinger equation (4.4), the readers are referred to [35].

For both the standard network and the sparse network, we used the network architecture $[2, 50, 50, 50, 50, 50, 50, 2]$. Both the networks were trained by the Adam algorithm with $30,000$ epochs. The initial learning rate is set to 0.001. The training set is composed of $N_0 := 50$ data points on $u(0,x)$, $N_b := 50$ sample points for enforcing the periodic boundaries, and $N_f := 20,000$ sample points inside the solution domain of Eq. (4.4). The test set is composed of grid points $(0,\pi/2] \times [-5,5]$ uniformly discretized with step size $\pi/400$ on the t-direction and step size $10/256$ on the x-direction.

Numerical results for this example are listed in Table 7. As we can see, the sparse network has a smaller prediction error than the standard network. When regularization parameters $\alpha = [0, 0, 0, 0, 5e\text{-}7, 1e\text{-}6, 1e\text{-}5]$, the relative L_2 error is smaller than the PINN method. When regularization parameters α are taken as $[9e\text{-}7, 5e\text{-}7, 6e\text{-}7, 7e\text{-}7, 8e\text{-}7, 1e\text{-}6, 1e\text{-}5]$, the sparsity of weight matrices are $[22.0\%, 50.5\%, 51.9\%, 50.6\%, 50.0\%, 64.5\%, 66.0\%]$. In other words, after removing more than half of the neural network connections, the sparse neural network still has a slightly higher prediction accuracy. The predicted solution of the SDNN is illustrated in Fig. 10. These numerical results clearly confirm that the proposed SDNN model outperforms the standard DNN model.

Table 7: The Schrödinger equation. The neural network of 7 layers with network architecture [2, 50, 50, 50, 50, 50, 2].

Algorithms	Parameters α & sparsity of weight matrices	Relative L_2 error
PINN	No regularization $[0.0\%, 0.3\%, 0.4\%, 0.6\%, 0.4\%, 0.68\%, 0.0\%]$	$1.41e$-3
SDNN ($\beta = 10$)	$\alpha = [0, 0, 0, 0, 5e\text{-}7, 1e\text{-}6, 1e\text{-}5]$ $[0.7\%, 0.3\%, 0.4\%, 50.6\%, 38.0\%, 74.7\%, 77.0\%]$	$8.15e$-4
SDNN ($\beta = 10$)	$\alpha = [9e\text{-}7, 5e\text{-}7, 6e\text{-}7, 7e\text{-}7, 8e\text{-}7, 1e\text{-}6, 1e\text{-}5]$ $[22.0\%, 50.5\%, 51.9\%, 50.6\%, 50.0\%, 64.5\%, 66.0\%]$	$1.38e$-3

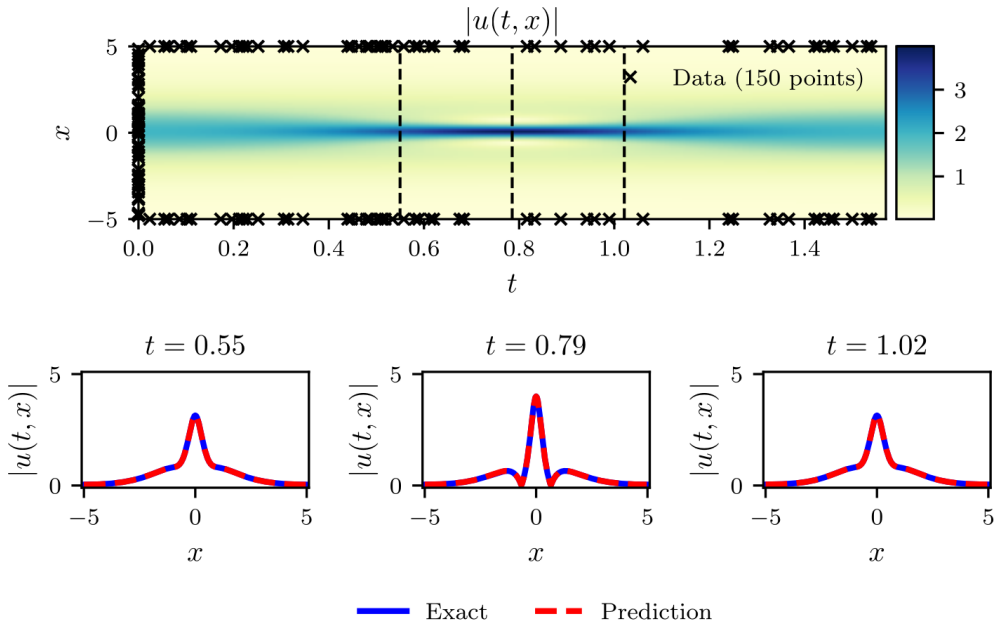

Figure 10: The Schrödinger equation. Top: The training data and predicted solution $|u(t,x)|$ by SDNN with network architecture [2, 50, 50, 50, 50, 50, 50, 2], regularization parameters $\alpha := [9e\text{-}7, 5e\text{-}7, 6e\text{-}7, 7e\text{-}7, 8e\text{-}7, 1e\text{-}6, 1e\text{-}5]$, and $\beta := 10$. Bottom: Predicted solutions at time $t := 0.55$, $t := 0.79$, and $t := 1.02$.

5. Conclusion

A sparse network requires less memory and computing time to operate it and thus it is desirable. We have developed a sparse deep neural network model by employing a sparse regularization with multiple parameters for solving nonlinear partial differential equations. Noticing that neural networks are layer-by-layer composite structures with an intrinsic multi-scale structure, we observe that the network weights of different layers have different weights of importance. Aiming at generating a sparse network structure while maintaining approximation accuracy, we proposed to impose different

regularization parameters on different layers of the neural network. We first tested the proposed sparse regularization model in approximation of singular functions, and discovered that the proposed model can not only generate an adaptive approximation of functions having singularities but also have better generalization than the standard network. We then developed a sparse deep neural network model for solving nonlinear partial differential equations whose solutions may have certain singularities. Numerical examples show that the proposed model can remove redundant network connections leading to sparse networks and has better generalization ability. Theoretical investigation will be performed in a follow-up paper.

Acknowledgments

Y. Xu is supported in part by US National Science Foundation under grant DMS-1912958. T. Zeng is supported in part by the National Natural Science Foundation of China under grants 12071160 and U1811464, by the Natural Science Foundation of Guangdong Province under grant 2018A0303130067, by the Opening Project of Guangdong Province Key Laboratory of Computational Science at the Sun Yat-sen University under grant 2021022, and by the Opening Project of Guangdong Key Laboratory of Big Data Analysis and Processing under grant 202101.

References

[1] B. ADCOCK AND N. DEXTER, *The gap between theory and practice in function approximation with deep neural networks*, SIAM J. Math. Data Sci. 3(2) (2021), 624–655.
[2] C. BASDEVANT, M. DEVILLE, P. HALDENWANG, J. LACROIX, J. OUAZZANI, R. PEYRET, P. ORLANDI, AND A. PATERA, *Spectral and finite difference solutions of the Burgers equation*, Comput. Fluids. 14 (1986), 23–41.
[3] M. BRENNER, Y. JIANG, AND Y. XU, *Multiparameter regularization for Volterra kernel identification via multiscale collocation methods*, Adv. Comput. Math. 31 (2009), 421–455.
[4] E. J. CANDÈS, J. ROMBERG, AND T. TAO, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory 52 (2006), 489–509.
[5] E. J. CANDÈS AND M. B. WAKIN, *An introduction to compressive sampling*, IEEE Signal Process. Mag. 25(2) (2008), 21–30.
[6] Z. CHEN, Y. LU, Y. XU, AND H. YANG, *Multi-parameter Tikhonov regularization for linear ill-posed operator equations*, J. Comput. Math. 26(1) (2008), 37–55.
[7] Z. CHEN, C. MICCHELLI, AND Y. XU, *Multiscale Methods for Fredholm Integral Equations*, Cambridge University Press, 2015.
[8] E. CHONG, C. HAN, AND F. C. PARK, *Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies*, Expert Syst. Appl. 83 (2017), 187–205.
[9] G. CYBENKO, *Approximation by superpositions of a sigmoidal function*, Math. Control Signals Systems. 2 (1989), 303–314.

[10] E. C. Cyr, M. A. Gulian, R. G. Patel, M. Perego, and N. A. Trask, *Robust training and initialization of deep neural networks: An adaptive basis viewpoint*, in: Proceedings of Machine Learning Research, Princeton University, Vol. 107, USA, 20-24 Jul 2020, PMLR, 512–536.
[11] G. E. Dahl, D. Yu, L. Deng, and A. Acero, *Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition*, IEEE Trans. Audio Speech Lang. Process. 20 (2012), 30–42.
[12] I. Daubechies, *Ten Lectures on Wavelets*, SIAM, 1992.
[13] I. Daubechies et al., *Nonlinear Approximation and (Deep) ReLU Networks*, Constr. Approx. 55 (2022), 127—172.
[14] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-Training of deep bidirectional transformers for language understanding*, arXiv:1810.04805v1.
[15] R. DeVore, B. Hanin, and G. Petrova, *Neural network approximation*, Acta Numer. 30 (2021), 327–444.
[16] M. Diefenthaler, A. Farhat, A. Verbytskyi, and Y. Xu, *Deeply learning deep inelastic scattering kinematics*, arXiv:2108.11638v1.
[17] D. Donoho, *Compressive sensing*, IEEE Trans. Inf. Theory. 52 (2006), 1289–1306.
[18] T. A. Driscoll, N. Hale, and L. N. Trefethen, *Chebfun Guide*, 2014.
[19] K. Friston, *Hierarchical models in the brain*, PLoS Comput. Biol. 4(11) (2008), e1000211.
[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.
[21] J. Han, A. Jentzen, and W. E, *Solving high-dimensional partial differential equations using deep learning*, in: Proceedings of the National Academy of Sciences, 115 (2018), 8505–8510.
[22] J. He, L. Li, J. Xu, and C. Zheng, *ReLU deep neural networks and linear finite elements*, J. Comput. Math. 38 (2020), 502–527.
[23] J. C. Helton and F. J. Davis, *Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems*, Reliab. Eng. Syst. Saf. 81(1) (2003), 23–69.
[24] T. Hoefler, D. A. Alistarh, T. Ben-Nun, N. Dryden, and E. A. Peste, *Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks*, J. Mach. Learn. Res. 22(241) (2021), 1–124.
[25] T. Hoefler, D. A. Alistarh, N. Dryden, and T. Ben-Nun, *The future of deep learning will be sparse*, SIAM News, May 03, 2021.
[26] J. Jung, K. Yoon, and P. Lee, *Deep learned finite elements*, Comput. Methods Appl. Mech. Engrg. 372 (2020), 113401.
[27] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in: Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015).
[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet classification with deep convolutional neural networks*, in: Advances in Neural Information Processing Systems, Curran Associates, Inc. (2012), 1097–1105.
[29] I. E. Lagaris, A. Likas, and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE trans. neural netw. 9 (1998), 987–1000.
[30] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, *Neural-network methods for boundary value problems with irregular boundaries*, IEEE trans. neural netw. 11 (2000), 1041–1049.
[31] T. Liu and K. Zumbrun, *Nonlinear stability of an undercompressive shock for complex Burgers equation*, Comm. Math. Phys. 168(1) (1995), 163–186.
[32] Y. Lu, L. Shen, and Y. Xu, *Multi-parameter regularization methods for high-resolution*

image reconstruction with displacement errors, IEEE Trans. Circuits Syst. I Regul. Pap. 54(8) (2007), 1788–1799.

[33] C. A. MICCHELLI AND Y. XU, *Using the matrix refinement equation for the construction of wavelets on invariant sets*, Appl. Comput. Harmon. Anal. 1(4) (1994), 391–401.

[34] M. RAISSI, *Deep hidden physics models: Deep learning of nonlinear partial differential equations*, J. Mach. Learn. Res. 19(1) (2018), 932–955.

[35] M. RAISSI, P. PERDIKARIS, AND 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.

[36] V. N. SERKIN AND A. HASEGAWA, *Novel soliton solutions of the nonlinear Schrödinger equation model*, Phys. Rev. Lett. 85(21) (2000), 4502.

[37] M. UNSER, *A representer theorem for deep neural networks*, J. Mach. Learn. Res. 20 (2019), 1–28.

[38] Y. XU, *Sparse regularization with the ℓ_0 norm*, arXiv:2111.08244, 2021.

[39] Y. XU AND Q. YE, *Generalized Mercer kernels and reproducing kernel Banach spaces*, Mem. Am. Math. Soc. 258 (2019), 1–122.

[40] Y. XU AND H. ZHANG, *Convergence of deep ReLU networks*, arXiv:2107.12530, 2021.

[41] Y. XU AND H. ZHANG, *Convergence of deep convolutional neural networks*, arXiv:2109.13542, 2021.

[42] H. ZHANG, Y. XU, AND J. ZHANG, *Reproducing kernel Banach spaces for machine learning*, J. Mach. Learn. Res. 10 (2009), 2741–2775.