mathematics

MDPI

*Article*

# Solving a Class of High-Order Elliptic PDEs Using Deep Neural Networks Based on Its Coupled Scheme

**Xi'an Li [1], Jinran Wu [2,3], Lei Zhang [4,5,6,*] and Xin Tai [1]**

[1] Ceyear Technology Co., Ltd., Qingdao 266000, China
[2] School of Mathematical Sciences, Queensland University of Technology, Brisbane 4001, Australia
[3] The Institute for Learning Sciences and Teacher Education, Australian Catholic University, Brisbane 4000, Australia
[4] School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai 200240, China
[5] Institute of Natural Sciences, Shanghai Jiao Tong University, Shanghai 200240, China
[6] MOE-LSC, Shanghai Jiao Tong University, Shanghai 200240, China
[*] Correspondence: lzhang2012@sjtu.edu.cn

**Abstract:** Deep learning—in particular, deep neural networks (DNNs)—as a mesh-free and self-adapting method has demonstrated its great potential in the field of scientific computation. In this work, inspired by the Deep Ritz method proposed by Weinan E et al. to solve a class of variational problems that generally stem from partial differential equations, we present a coupled deep neural network (CDNN) to solve the fourth-order biharmonic equation by splitting it into two well-posed Poisson's problems, and then design a hybrid loss function for this method that can make efficiently the optimization of DNN easier and reduce the computer resources. In addition, a new activation function based on Fourier theory is introduced for our CDNN method. This activation function can reduce significantly the approximation error of the DNN. Finally, some numerical experiments are carried out to demonstrate the feasibility and efficiency of the CDNN method for the biharmonic equation in various cases.

**Keywords:** biharmonic equation; coupled scheme; DNN; variational form; Fourier mapping

**MSC:** 35J58; 65N35; 68T07

check for updates

**Citation:** Li, X.; Wu, J.; Zhang, L.; Tai, X. Solving a Class of High-Order Elliptic PDEs Using Deep Neural Networks Based on Its Coupled Scheme. *Mathematics* **2022**, *10*, 4186. https://doi.org/10.3390/math10224186

Academic Editor: Danilo Costarelli

Received: 19 October 2022
Accepted: 3 November 2022
Published: 9 November 2022

## 1. Introduction

In this paper, we consider the numerical solution of the biharmonic equation

$$\begin{cases} \Delta^2 u(\boldsymbol{x}) = f(\boldsymbol{x}), \text{ in } \Omega, \\ u(\boldsymbol{x}) = g(\boldsymbol{x}), \text{ on } \partial\Omega, \\ \Delta u(\boldsymbol{x}) = h(\boldsymbol{x}), \text{ on } \partial\Omega, \end{cases} \tag{1}$$

where $\Omega$ is a polygonal or polyhedral domain in Euclidean space $R^d$ ($d$ is the dimension) with a piecewise Lipschitz boundary that satisfies an interior cone condition, $f(\boldsymbol{x}) \in L^2(\Omega)$ is a given function and $\Delta$ is a standard Laplace operator. The operators $\Delta u(\boldsymbol{x})$ and $\Delta^2 u(\boldsymbol{x})$ are expressed as

$$\Delta u(\boldsymbol{x}) = \sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} \text{ and } \Delta^2 u(\boldsymbol{x}) = \sum_{i=1}^{d} \frac{\partial^4 u}{\partial x_i^4} + \sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial^4 u}{\partial x_i^2 x_j^2},$$

respectively.

A biharmonic equation is a class of common high-order partial differential equations (PDEs) that stems from the field of physics and applied mathematics, especially in elasticity theory and Stokes flow problems; for instance, scattered data fitting with thin plate splines

[1], fluid mechanics [2,3] and linear elasticity [4,5]. In the last few decades, many traditional numerical methods have been proposed for dealing with (1), and they can be classified into two categories: a direct (uncoupled) approach and coupled (splitting, mixed) approach.

In terms of the direct approach, there are the finite difference method (FDM) based on its uncoupled scheme [6–9], finite volume method (FVM) [10–12], finite element method (FEM) based on its variational formulation, such as non-conforming FEM [13–15], and conforming FEM [16,17]. The idea of the coupled approach is to introduce auxiliary variables and split the biharmonic equation into two coupled Poisson equations. Based on this coupled scheme, the finite difference technique [9,18,19], finite element technique and mixed element technique [20–24] are naturally used to solve the two second-order equations. In addition, the collocation method [25–27] and radial basis functions (RBF) method [28–30] are also approaches considered to solve (1).

However, the traditional methods will encounter the curse of an irregular domain and high dimension. Deep learning, especially deep neural networks (DNNs), have expressed a remarkable performance in solving mathematical problems in scientific computations and engineering applications based on its great potential in nonlinear approximation; for example, utilizing DNNs to solve partial differential equations [31–34], stochastic differential equations [35], inverse problems [36], molecular modeling [37], etc. Specially, the Deep Ritz method (DRM) [32] and physical information neural network (PINN) [36] have gained more and more attention in PDEs, and they have made wonderful grades for solving various PDEs. Within the architectures of DRM and PINN, some DNN-based methods have been proposed to directly deal with this biharmonic Equation (1) [38–43]. Since these DNN-based methods need to compute the second-order or fourth-order derivative of the solution in $\Omega$ and the second-order derivative of the solution in $\partial\Omega$, they may consume a large amount of time and computer resources when solving (1). In addition, the authors in [44] proposed a deep mixed residual method (MIM) to solve PDEs with high-order derivatives by splitting it into some first-order systems; however, the PDEs solution and its derivatives share nearly the same DNN and boundary conditions, are not consistent with the mechanism of PDEs and adversely affect the performance of MIM.

In this paper, we investigate addressing the high-order elliptic Equation (1) by combining the Deep Ritz method and the coupled scheme of the biharmonic equation, and then establish a coupled deep neural network architecture (CDNN). Motivated by double triangle series and Fourier expansion, a new activation function with sine and cosine are provided for our CDNN model; it will improve the capability of DNN for approximating a complex target because of it being smooth and local. The application of a trigonometric function as an activation function can also be found in [45,46]. The main contributions of this paper are as follows:

- Based on the coupled scheme of the fourth-order biharmonic Equation (1), we constructed a CDNN architecture for dealing with (1) by means of the Deep Ritz method used to solve variational problems; this architecture is composed of two independent DNNs. Compared with the existing DNN methods, the CDNN will reduce effectively the complexity of the algorithm, save the resources of the computer and make the neural networks easier to train. In the meantime, this model performs remarkably well and obtains considerable results.
- According to the property of spectral bias or frequency preference for the DNN, we introduced a Fourier mapping with sine and cosine as the activation function of the first layer for the DNN model; it can mitigate the pathology of the spectral bias of the DNN. In the viewpoint of function approximation, the DNN model with Fourier mapping mimics the Fourier expansion, in which, the first layer with Fourier mapping can be regarded as a series of Fourier basic functions and the output of the DNN is the (nonlinear) combination of those basis functions.
- By introducing some compared DRM models with different activation functions to solve the original form of the biharmonic equation, we show that our CDNN model performs better when solving (1) in various dimensional space.

The paper is structured as follows. In Section 2, we briefly introduce the framework of the deep neural network and the formulation of ResNet. Then, in Section 3, we construct the CDNN architecture to approximate the solution of the biharmonic equation based on its coupled scheme and provide the options of the activation function. In Section 4, some numerical examples are carried out to test the performance of the developed CDNN model. Section 5 discusses the merits and shortcomings of the CDNN, and provides the opportunity to address related works. Finally, some brief conclusions are made in Section 6.

## 2. Deep Neural Network and ResNet Architecture

In this section, the related concepts and mathematical formulation of the DNN are briefly introduced. At first, a standard neural unit of a DNN with an input $\boldsymbol{x} \in \mathbb{R}^d$ and an output $\boldsymbol{y} \in \mathbb{R}^m$ is in the form of

$$\boldsymbol{y} = \sigma \circ (\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{2}$$

where $\boldsymbol{W} = (w_{ij}) \in \mathbb{R}^{d\times m}$ and $\boldsymbol{b} \in \mathbb{R}^m$ are called a weight matrix and bias vector, respectively. Here and thereafter, $\sigma$ is a non-linear operator, commonly known as the activation function, and " $\circ$ " stands for the elementary-wise operation. Generally, the output of (2) will be transformed by a new weight and a new bias, and the new output will be fed into another activation function. Hence, (2) is also called the hidden layer of the DNN. In other word, the DNN is a nested composition of sequential single linear functions and nonlinear activation functions. Mathematically, the DNN with an input datum $\boldsymbol{x} \in \mathbb{R}^d$ and an output $\boldsymbol{y}(\boldsymbol{x};\boldsymbol{\theta})$ can be formulated as

$$\begin{cases} \boldsymbol{y}^{[0]} = \boldsymbol{x} \\ \boldsymbol{y}^{[l]} = \sigma \circ (\boldsymbol{W}^{[l]}\boldsymbol{y}^{[l-1]} + \boldsymbol{b}^{[l]}), \ \text{ for } \ l = 1,2,3,\cdots\cdots, L \\ \boldsymbol{y}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{y}^{[L]} \end{cases} \tag{3}$$

where $\boldsymbol{W}^{[l]} \in \mathbb{R}^{m_{l+1}\times m_l}, \boldsymbol{b}^{[l]} \in \mathbb{R}^{m_{l+1}}$ are the weight matrix and bias vector of the $l$-th hidden layer, respectively, $m_0 = d$ and $m_L$ is the dimension of the output for the DNN. For convenience, we denote the parameter set $\left(\boldsymbol{W}^{[L]},\cdots \boldsymbol{W}^{[1]}, \boldsymbol{W}^{[0]}, \boldsymbol{b}^{[L]},\cdots \boldsymbol{b}^{[1]}, \boldsymbol{b}^{[0]}\right)$ by $\boldsymbol{\theta}$ here and thereafter.

The residual neural network (ResNet) technique [47] has been widely used in deep neural networks for solving PDEs; it can not onl skilfully overcome the vanishing gradient problem of DNN in backpropagation, but can also well improve the capacity of the DNN to approximate the solution and high-order derivatives of PDEs [32,37]. The architecture of ResNet is depicted in the following Figure 1.

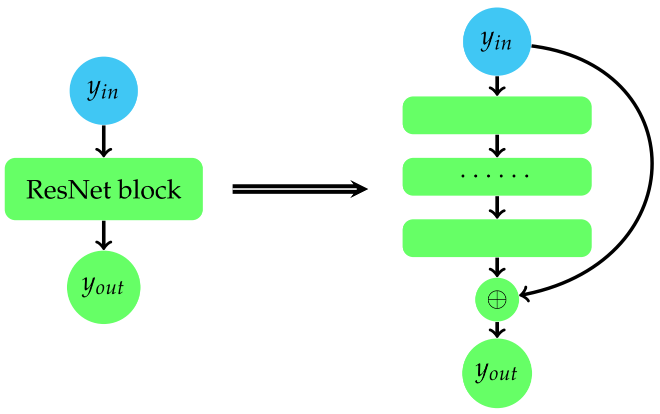


**Figure 1.** The architecture of ResNet.

Mathematically, a ResNet block with a one-step connection produces a filtered version $\boldsymbol{y}^{[\ell+1]}(\boldsymbol{x};\boldsymbol{\theta})$ for the input $\boldsymbol{y}^{[\ell]}(\boldsymbol{x};\boldsymbol{\theta})$, which is as follows:

$$\boldsymbol{y}^{[\ell+1]}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{y}^{[\ell]}(\boldsymbol{x};\boldsymbol{\theta}) + \sigma \circ (\boldsymbol{W}^{[\ell+1]}\boldsymbol{y}^{[\ell]}(\boldsymbol{x};\boldsymbol{\theta}) + \boldsymbol{b}^{[\ell+1]}).$$

In this work, we also employed the strategy of a one-step skip connection for two consecutive layers in the DNN if they have the same number of neurons. For those consecutive layers with different neuron numbers, the skip connection step is omitted.

## 3. Unified coupled DNNs Architecture to Biharmonic Equation

Skilfully decoupling the biharmonic Equation (1) into two Poisson equations, traditional numerical methods such as FEM and FDM obtain a favorable performance with a small number of computing resources and a lower time complexity. By introducing an auxiliary variable $w = -\Delta u$, one can rewrite the fourth-order Equation (1) into a couple of second-order equations:

$$\begin{cases} -\Delta w = f, & \text{in } \Omega \\ w = h, & \text{on } \partial\Omega \end{cases} \quad \text{and} \quad \begin{cases} -\Delta u = w, & \text{in } \Omega \\ u = g, & \text{on } \partial\Omega \end{cases}. \tag{4}$$

Then, we searched a couple of functions $(w, u)$ instead of finding a solution for the original problem (1). Generally, the strong solutions of (4) may be non-existent; one can obtain their weak solutions in the given domain $\Omega$. The weak function pair $(w, u)$ should satisfy $u, w \in \mathbb{H}_0^1(\Omega)$ and

$$\int_\Omega (\nabla u \cdot \nabla v - wv) d\boldsymbol{x} = 0 \quad \text{and} \quad \int_\Omega (\nabla w \cdot \nabla \psi - f\psi) d\boldsymbol{x} = 0 \quad \text{for all } v, \psi \in \mathbb{H}_0^1(\Omega).$$

They are equivalent to the weak version of the Euler–Lagrange equation for the following variational problems:

$$w = \min_{\psi \in \mathbb{H}_0^1(\Omega)} \mathcal{J}_1(\psi) = \frac{1}{2}\int_\Omega |\nabla \psi|^2 d\boldsymbol{x} - \int_\Omega f\psi d\boldsymbol{x} \tag{5}$$

and

$$u = \min_{v \in \mathbb{H}_0^1(\Omega)} \mathcal{J}_2(v) = \frac{1}{2}\int_\Omega |\nabla v|^2 d\boldsymbol{x} - \int_\Omega wv d\boldsymbol{x}, \tag{6}$$

respectively.

The Deep Ritz method based on a deep neural network is an efficient approach for solving variational problems that generally stem from PDEs [32]; it utilizes a parameterized neural network to replace the trial function and changes the original problems into the optimization of neural networks. In addition, some works concerning convergence analysis for DRM are proposed [48,49].

Based on the above results, we then suppose two ansatzes of DNN $y_1(:, \boldsymbol{\theta}_1)$ and $y_2(:, \boldsymbol{\theta}_2)$ as the functions $\psi$ and $v$, which minimize the variational problem (5) and (6), respectively, where $\boldsymbol{\theta}_1 \in \Theta$ and $\boldsymbol{\theta}_2 \in \Theta$ denote the parameters of underlying DNNs. Substituting $y_1(:, \boldsymbol{\theta}_1)$ into (5) for $\psi$, we can firstly obtain the following equation:

$$w(\boldsymbol{x}) = \operatorname*{arg\,min}_{y_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_1 \in \Theta} \left[\frac{1}{2}\int_\Omega |\nabla y_1(\boldsymbol{x}, \boldsymbol{\theta}_1)|^2 d\boldsymbol{x} - \int_\Omega f(\boldsymbol{x}) y_1(\boldsymbol{x}, \boldsymbol{\theta}_1) d\boldsymbol{x}\right] \quad \text{for } \boldsymbol{x} \in \Omega. \tag{7}$$

Since the minimization problem (5) is related to $w$ and $y_1(\boldsymbol{x}, \boldsymbol{\theta}_1)$ is an approximation of $w$, then we obtain the following equation by replacing $v$ and $w$ with $y_2(\boldsymbol{x}, \boldsymbol{\theta}_1)$ and $y_1(\boldsymbol{x}, \boldsymbol{\theta}_2)$, respectively, in (6):

$$u(\boldsymbol{x}) = \operatorname*{arg\,min}_{y_2(\boldsymbol{x}, \boldsymbol{\theta}_2), \boldsymbol{\theta}_2 \in \Theta} \left[\frac{1}{2}\int_\Omega |\nabla y_2(\boldsymbol{x}, \boldsymbol{\theta}_2)|^2 d\boldsymbol{x} - \int_\Omega y_1(\boldsymbol{x}, \boldsymbol{\theta}_1) y_2(\boldsymbol{x}, \boldsymbol{\theta}_2) d\boldsymbol{x}\right] \quad \text{for } \boldsymbol{x} \in \Omega. \tag{8}$$

Using the Monte Carlo method [50] to calculate the above integration in $\Omega$, we further have

$$\boldsymbol{\theta}_1^* = \operatorname*{arg\,min}_{\boldsymbol{\theta}_1 \in \Theta} L_{in1}(S_I; \boldsymbol{\theta}_1) \text{ and } \boldsymbol{\theta}_2^* = \operatorname*{arg\,min}_{\boldsymbol{\theta}_2 \in \Theta} L_{in2}(S_I; \boldsymbol{\theta}_2) \tag{9}$$

with

$$L_{in1}(S_I;\boldsymbol{\theta}_1) = \frac{1}{n_{in}}\sum_{i=1}^{n_{in}}\left[\frac{1}{2}\left|\nabla y_1(\boldsymbol{x}_I^i,\boldsymbol{\theta}_1)\right|^2 - f(\boldsymbol{x}_I^i)y_1(\boldsymbol{x}_I^i,\boldsymbol{\theta}_1)\right] \text{ for } \boldsymbol{x}_I^i \in S_I \tag{10}$$

and

$$L_{in2}(S_I;\boldsymbol{\theta}_2) = \frac{1}{n_{in}}\sum_{i=1}^{n_{in}}\left[\frac{1}{2}\left|\nabla y_2(\boldsymbol{x}_I^i,\boldsymbol{\theta}_2)\right|^2 - y_1(\boldsymbol{x}_I^i,\boldsymbol{\theta}_1)y_2(\boldsymbol{x}_I^i,\boldsymbol{\theta}_2)\right] \text{ for } \boldsymbol{x}_I^i \in S_I, \tag{11}$$

respectively; here, and thereafter, $S_I$ stands for the samples in $\Omega$ with probability density $\rho_I$.

Boundary conditions are important constraints for numerical methods such as FDM and FEM to solve PDEs, which ensure the uniqueness and accuracy of the solution for PDEs. Analogously, imposing boundary conditions is also an important issue in DNN representation. Under the boundary constraints of (4), the DNNs $y_1(\boldsymbol{x},\boldsymbol{\theta}_1)$ and $y_2(\boldsymbol{x},\boldsymbol{\theta}_2)$ on $\partial\Omega$ should satisfy

$$L_{bd1}(S_B;\boldsymbol{\theta}_1) = \frac{1}{n_{bd}}\sum_{j=1}^{n_{bd}}\left[y_1(\boldsymbol{x}_B^j,\boldsymbol{\theta}_1) - h(\boldsymbol{x}_B^j)\right]^2 \to 0 \text{ for } \boldsymbol{x}_B^j \in S_B \tag{12}$$

and

$$L_{bd2}(S_B;\boldsymbol{\theta}_1) = \frac{1}{n_{bd}}\sum_{j=1}^{n_{bd}}\left[y_2(\boldsymbol{x}_B^j,\boldsymbol{\theta}_2) - g(\boldsymbol{x}_B^j)\right]^2 \to 0 \text{ for } \boldsymbol{x}_B^j \in S_B, \tag{13}$$

here, and thereafter, $S_B$ stands for the samples on $\partial\Omega$ with probability density $\rho_B$.

To this end, two individual parameters of DNNs model are optimized by minimizing the following loss function:

$$L(S_I,S_B;\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) = L_1(S_I,S_B;\boldsymbol{\theta}_1) + L_2(S_I,S_B;\boldsymbol{\theta}_2) \tag{14}$$

with

$$L_1(S_I,S_B;\boldsymbol{\theta}_1) = L_{in1}(S_I,\boldsymbol{\theta}_1) + \gamma L_{bd1}(S_B;\boldsymbol{\theta}_1) \text{ and } L_2(S_I,S_B;\boldsymbol{\theta}_2) = L_{in2}(S_I,\boldsymbol{\theta}_2) + \gamma L_{bd2}(S_B;\boldsymbol{\theta}_2)$$

where $S_I = \{\boldsymbol{x}_I^i\}_{i=1}^{n_{in}}$ and $S_B = \{\boldsymbol{x}_B^j\}_{j=1}^{n_{bd}}$ represent the training data of $\Omega$ and $\partial\Omega$, respectively. In addition, we introduce a penalty parameter $\gamma$ to control the contribution of the boundary for the loss function; it increases gradually as the training process continues.

Our goal is to find two sets of parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ such that the approximate functions $y_1(\cdot,\boldsymbol{\theta}_1)$ and $y_2(\cdot,\boldsymbol{\theta}_2)$ minimize the functions $L_1(S_I,S_B;\boldsymbol{\theta}_1), L_2(S_I,S_B;\boldsymbol{\theta}_2)$ and $L(S_I,S_B;\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)$. If these loss functions are small enough, then $y_1(\boldsymbol{x},\boldsymbol{\theta}_1)$ and $y_2(\boldsymbol{x},\boldsymbol{\theta}_2)$ will be very close to the solution of (4), i.e.,

$$\boldsymbol{\theta}_1^*,\boldsymbol{\theta}_2^* = \begin{cases} \arg\min\limits_{\boldsymbol{\theta}_1\in\Theta} L_1(S_I,S_B;\boldsymbol{\theta}_1) \\ \arg\min\limits_{\boldsymbol{\theta}_2\in\Theta} L_2(S_I,S_B;\boldsymbol{\theta}_2) \\ \arg\min\limits_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2\in\Theta} L(S_I,S_B;\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) \end{cases} \iff w(\boldsymbol{x}) = y_1(\boldsymbol{x},\boldsymbol{\theta}_1^*) \text{ and } u(\boldsymbol{x}) = y_2(\boldsymbol{x},\boldsymbol{\theta}_2^*). \tag{15}$$

**Remark 1.** *In practical implementation, we bundled $L_1(S_I,S_B;\boldsymbol{\theta}_1)$, $L_2(S_I,S_B;\boldsymbol{\theta}_2)$ and $L(S_I,S_B;\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)$ together and trained them by means of a neural network optimizer. These functions in form are related, but they can be trained in parallel.*

**Remark 2.** *By directly transforming the biharmonic Equation* (1) *into a variational problem, one can easily employ DRM to solve it. The corresponding loss is given by*

$$L(S_I, S_B; \boldsymbol{\theta}) = L_{in}(S_I; \boldsymbol{\theta}) + \gamma L_{bd}(S_B; \boldsymbol{\theta})$$

*with*

$$L_{in}(S_I; \boldsymbol{\theta}) = \frac{1}{n_{in}} \sum_{i=1}^{n_{in}} \left[ \frac{1}{2} \left| \Delta y(\boldsymbol{x}_I^i; \boldsymbol{\theta}) \right|^2 - f(\boldsymbol{x}_I^i) y(\boldsymbol{x}_I^i; \boldsymbol{\theta}) \right] \ \textit{for} \ \boldsymbol{x}_I^i \in S_I,$$

*and*

$$L(S_B; \boldsymbol{\theta}) = \frac{1}{n_{bd}} \sum_{j=1}^{n_{bd}} \left[ y(\boldsymbol{x}_B^j; \boldsymbol{\theta}) - g(\boldsymbol{x}_B^j) \right]^2 + \frac{1}{n_{bd}} \sum_{j=1}^{n_{bd}} \left[ \Delta y(\boldsymbol{x}_B^j; \boldsymbol{\theta}) - h(\boldsymbol{x}_B^j) \right]^2 \to 0 \ \textit{for} \ \boldsymbol{x}_B^j \in S_B,$$

*where $y(\cdot; \boldsymbol{\theta})$ stands for the output of DRM.*

In order to obtain the $\boldsymbol{\theta}_1^*$ and $\boldsymbol{\theta}_2^*$, one can update the parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ by means of gradient descent (GD) or stochastic gradient descent (SGD) techniques over the training samples at each iteration. Regarding implementation, SGD is the common optimization method used for deep learning; it only requires the gradient information of a DNN over one or a few samples. In this context, the SGD is given by:

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha_k \nabla_{\boldsymbol{\theta}^k} L(\boldsymbol{x}; \boldsymbol{\theta}^k), \quad \boldsymbol{x} \in S_I \text{ or } S_B,$$

where the "learning rate" $\alpha_k$ decreases with $k$ increasing and $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$. Based on the above discussions, Figure 2 describes the schematic of the CDNN for solving biharmonic Equation (1).

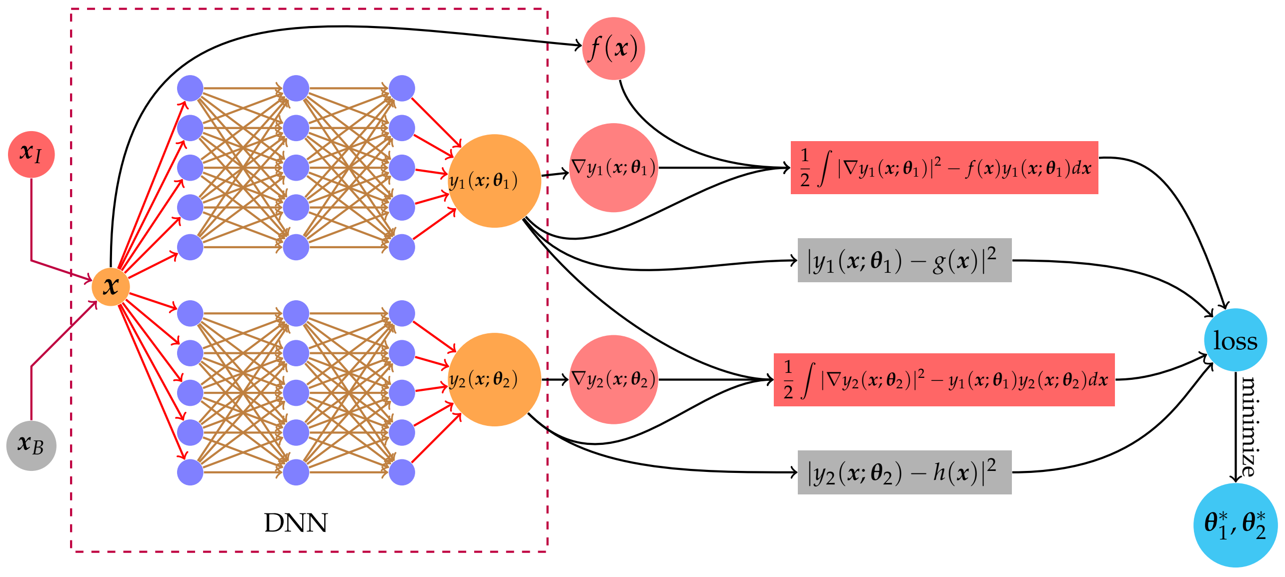

$\boldsymbol{x}_I$
$\boldsymbol{x}$
$\boldsymbol{x}_B$
$y_1(\boldsymbol{x}; \boldsymbol{\theta}_1)$
$y_2(\boldsymbol{x}; \boldsymbol{\theta}_2)$
DNN
$f(\boldsymbol{x})$
$\nabla y_1(\boldsymbol{x}; \boldsymbol{\theta}_1)$
$\nabla y_2(\boldsymbol{x}; \boldsymbol{\theta}_2)$
$\frac{1}{2}\int |\nabla y_1(\boldsymbol{x}; \boldsymbol{\theta}_1)|^2 - f(\boldsymbol{x}) y_1(\boldsymbol{x}; \boldsymbol{\theta}_1) dx$
$|y_1(\boldsymbol{x}; \boldsymbol{\theta}_1) - g(\boldsymbol{x})|^2$
$\frac{1}{2}\int |\nabla y_2(\boldsymbol{x}; \boldsymbol{\theta}_2)|^2 - y_1(\boldsymbol{x}; \boldsymbol{\theta}_1) y_2(\boldsymbol{x}; \boldsymbol{\theta}_2) dx$
$|y_2(\boldsymbol{x}; \boldsymbol{\theta}_2) - h(\boldsymbol{x})|^2$
loss
minimize
$\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^*$

**Figure 2.** Schematic of a CDNN for solving the biharmonic equation. The left two DNNs with the same framework(dotted line box) share the input $\boldsymbol{x} \in \mathbb{R}^d$ (including $\boldsymbol{x}_I$ and $\boldsymbol{x}_B$), and the upper and lower branches output $\boldsymbol{y}_1(\boldsymbol{x}; \boldsymbol{\theta}_1) \in \mathbb{R}$ and $\boldsymbol{y}_2(\boldsymbol{x}; \boldsymbol{\theta}_1) \in \mathbb{R}$, which are used to approximate the function $w$ and $u$, respectively. The right part handles the outputs of DNN and forms the total loss according to PDEs constraints and boundary conditions.

Many experiments have shown that choosing a suitable activation function is crucial for DNNs in various fields. As we have learned, nonlinear activation functions such as $\text{ReLU}(z) = \max\{0, z\}$, $\text{Sigmoid}(z)$ and $\tanh(z)$ are the common choice for the DNN model; they will improve the capacity of DNN to deal with various nonlinear problems, such as nonlinear PDEs and classification. Due to the biharmonic equation in this work being a class of high-order PDEs, an activation function with a low-order derivative possibly has an unfavourable effect for DNN when it is used to solve (1). We then consider the activation function with a good regularity, such as tanh. Figure 3 depicts the curves of the tanh function with first-order and second-order derivatives, which shows that the range of derivatives for tanh is small and stable.

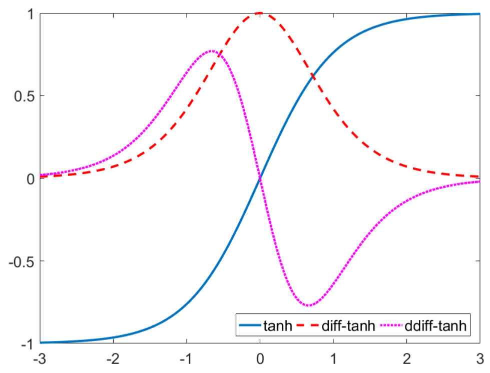


**Figure 3.** The curves for tanh function and its derivatives, respectively.

In the viewpoint of function approximation, the first layer with activation functions for the DNN can be regarded as a series of basic functions, and the output of the DNN is the (nonlinear) combination of those basis functions. Recently, the works [51,52] found the phenomenon of spectral bias or frequency preference for DNNs and show that the DNN will firstly capture the low-frequency component, and then some corresponding explanations are made by means of a neural tangent kernel (NTK) [45,53] and Fourier analysis [54,55]. Within this sense of spectral bias and Fourier approximation, a given real function $\mathcal{F}(x)$ can be expressed by the following sine and cosine expansions:

$$\mathcal{F}(x) = \sum_{n=1}^{\tilde{N}} \left( S(\cos(\omega_n x); \tilde{\boldsymbol{\theta}}) + T(\sin(\omega_n x); \bar{\boldsymbol{\theta}}) \right),$$

where $S(x, \tilde{\boldsymbol{\theta}}), T(x, \bar{\boldsymbol{\theta}})$ are fully connected DNNs or sub-modules of the DNN, respectively, $\{\omega_0, \omega_1, \omega_2, \cdots\}$ are the frequencies of interest in the target function and $\omega = 0$ will always be included. Obviously, it mimics the Fourier expansion, and the remaining blocks of the DNN (except for the first layer) are used to learn the coefficients of Fourier basis functions.

Choosing a Fourier feature mapping with sine and cosine as the activation function for the DNN model is reasonable; it can mitigate the pathology of spectral bias and enable networks to learn the target function well [45,46]. It is

$$\sigma(z) = \begin{bmatrix} \cos(2\boldsymbol{\kappa}\pi z) \\ \sin(2\boldsymbol{\kappa}\pi z) \end{bmatrix}, \tag{16}$$

where $\boldsymbol{\kappa}$ is a user-specified vector (it is not trainable) and is consistent with the number of neural units in the first hidden layer for the DNN. By performing the Fourier feature mapping for input points, the input points in $\Omega$ may be mapped to $[-1, 1]$; then, the subsequent modules of the DNN with different activation functions can easily deal with the feature information, such as sigmoid, tanh and ReLU, etc.

## 4. Numerical Experiments

### 4.1. Training Setup

In this section, we tested the performance of the CDNN model with the aforementioned activation functions (tanh and Fourier mapping) for solving the biharmonic equation in varying-dimensional spaces. In addition, two types of Deep Ritz methods with tanh and

Fourier mapping were introduced to serve as the baseline. These compared methods and their setups are as follows:

- **DRM**: A normal DNN model with tanh being its activation function for all hidden layers and its output layer being linear.
- **FDRM**: A normal DNN model with Fourier mapping as the activation function for its first hidden layer, where its activation functions for the remainder hidden layers are chosen as tanh and its output layer is linear. The vector $\boldsymbol{\kappa}$ as in (16) is set as $(0.25, 0.5, 0.75, 1, \cdots, 9.75, 10)$, and we will repeat it when the length of $\boldsymbol{\kappa}$ is less than the number of neural units for the first hidden layer.
- **CDNN**: A coupled DNN model with tanh being its activation function for all hidden layers and its output layer being linear.
- **FCDNN**: A coupled DNN model with Fourier mapping as the activation function for its first hidden layer, where its activation functions for the remainder hidden layers are chosen as tanh and its output layer is linear. The vector $\boldsymbol{\kappa}$ as in (16) is set as $(0.25, 0.5, 0.75, 1, \cdots, 9.75, 10)$, and we will repeat it when the length of $\boldsymbol{\kappa}$ is less than the number of neural units for the first hidden layer.

We provide the following criteria to evaluate the above models:

$$\text{REL} = \sum_{i=1}^{N'} \frac{|\tilde{u}(\boldsymbol{x}^i) - u^*(\boldsymbol{x}^i)|^2}{|u^*(\boldsymbol{x}^i)|^2}$$

where $\tilde{u}(\boldsymbol{x}^i)$ and $u^*(\boldsymbol{x}^i)$ are the approximate solution of the DNN and exact solution, respectively, for testing points $\{\boldsymbol{x}^i\}(i = 1, 2, \cdots, N')$, and $N'$ represents the number of testing points.

In our numerical experiments, all training and test data were generated with uniform distribution in Euclidean space $\mathbb{R}^d$, and all networks were trained by an Adam optimizer. The initial learning rate was set as $2 \times 10^{-4}$ with a decay rate of $5 \times 10^{-5}$ for each training epoch.

For the sake of viewing the training process, we tested our models for every 1000 epochs in the training process and recorded the result at the end. In our codes, the $\gamma$ was set as

$$\gamma = \begin{cases} \gamma_0, & \text{if } i_{\text{epoch}} < 0.1T_{\max}, \\ 10\gamma_0, & \text{if } 0.1T_{\max} <= i_{\text{epoch}} < 0.2T_{\max}, \\ 50\gamma_0, & \text{if } 0.2T_{\max} <= i_{\text{epoch}} < 0.25T_{\max}, \\ 100\gamma_0, & \text{if } 0.25T_{\max} <= i_{\text{epoch}} < 0.5T_{\max}, \\ 200\gamma_0, & \text{if } 0.5T_{\max} <= i_{\text{epoch}} < 0.75T_{\max}, \\ 500\gamma_0, & \text{otherwise}, \end{cases} \tag{17}$$

where the $\gamma_0 = 5$ in all our tests, and $T_{\max}$ represents the total number of epoch and was set as 100000 in our all experiments. We performed all neural network training and testing in TensorFlow (version 1.14.0) on a workstation (256-GB RAM, single NVIDIA GeForce GTX 2080Ti 12-GB).

### *4.2. Numerical Examples*

**Example 1.** *We solved the biharmonic Equation (1) on the unit square domain $\Omega = [0,1] \times [0,1]$. The exact solution and force-side are*

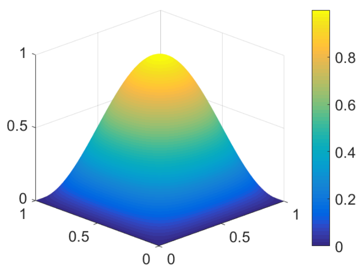
$$u(x_1, x_2) = \sin(\pi x_1)\sin(\pi x_2)$$

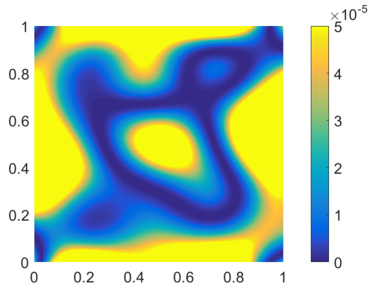
*and*

$$f(x_1, x_2) = 4\pi^4 \sin(\pi x_1)\sin(\pi x_2),$$

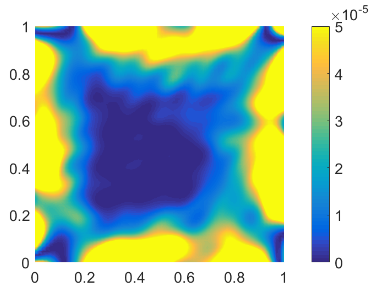
*respectively. The boundary conditions $g(x_1, x_2) = h(x_1, x_2) = 0$ on $\partial\Omega$.*

In the following tests, we obtained the solution of (1) by means of DRM, FDRM, CDNN and FCDNN, respectively. Their network size is (120, 60, 50, 50, 40), (60, 60, 50, 50, 40), (100, 50, 30, 30, 20) and (50, 50, 30, 30, 20), respectively. It is easy to know that the number of parameters for the methods is 14100, 10800, 14000 and 10000, respectively. At each training epoch, the training dataset was generated from $\Omega$ and $\partial\Omega$, which include 3000 interior points and 500 boundary points in $\Omega$, respectively. The testing dataset was uniformly sampled from $\Omega$ of mesh-size $h = 1/129$. We plot the numerical results in Figure 4 and list the final error results in Table 1.

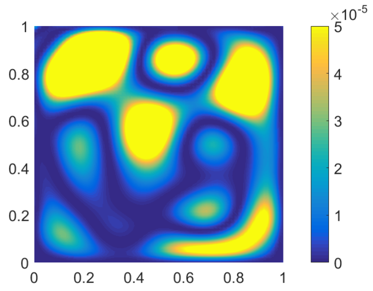

(**a**) Exact–solution



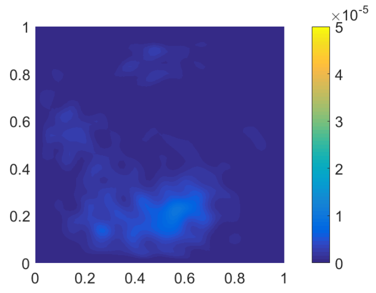
(**b**)Point–wise error for DRM



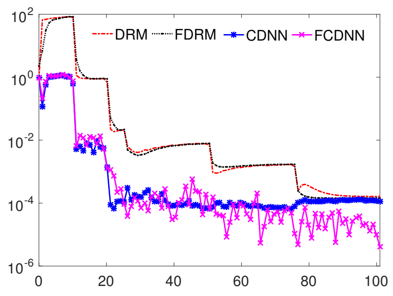
(**c**) Point–wise error for FDRM



(**d**) Point–wise error for CDNN



(**e**) Point–wise error for FCDNN



(**f**) REL

**Figure 4.** Testing results for Example 2.

**Table 1.** REL and running time of the aforementioned four models for Example 2.

| | **DRM** | **FDRM** | **CDNN** | **FCDNN** |
|---|---|---|---|---|
| REL | $1.64 \times 10^{-4}$ | $1.45 \times 10^{-4}$ | $1.12 \times 10^{-4}$ | $4.06 \times 10^{-6}$ |
| Time | 1.05 h | 1.37 h | 0.42 h | 0.55 h |

Based on the above results, we can see that the DRM, FDRM, CDNN and FCDNN are all able to approximate the solution of (1), in which, the FCDNN model performs best, and the performances of CDNN, DRM and FDRM are competitive. In terms of the overall errors, the CDNN is stable in all training processes, but the FCDNN is still descending with small oscillations. This means that the Fourier mapping activation function will improve the capability of the CDNN model. Table 1 shows that the CDNNs only cost approximately 0.5 h to solve Equation (1), but the DRMs cost more than 1.05 h. In summary, our CDNN methods not only have a high accuracy, but are also efficient.

**Example 2.** *We solved the biharmonic Equation* (1) *on the hexagram domain* $\Omega$ *derived from* $[0,1]\times[0,1]$. *The exact solution and force-side are*

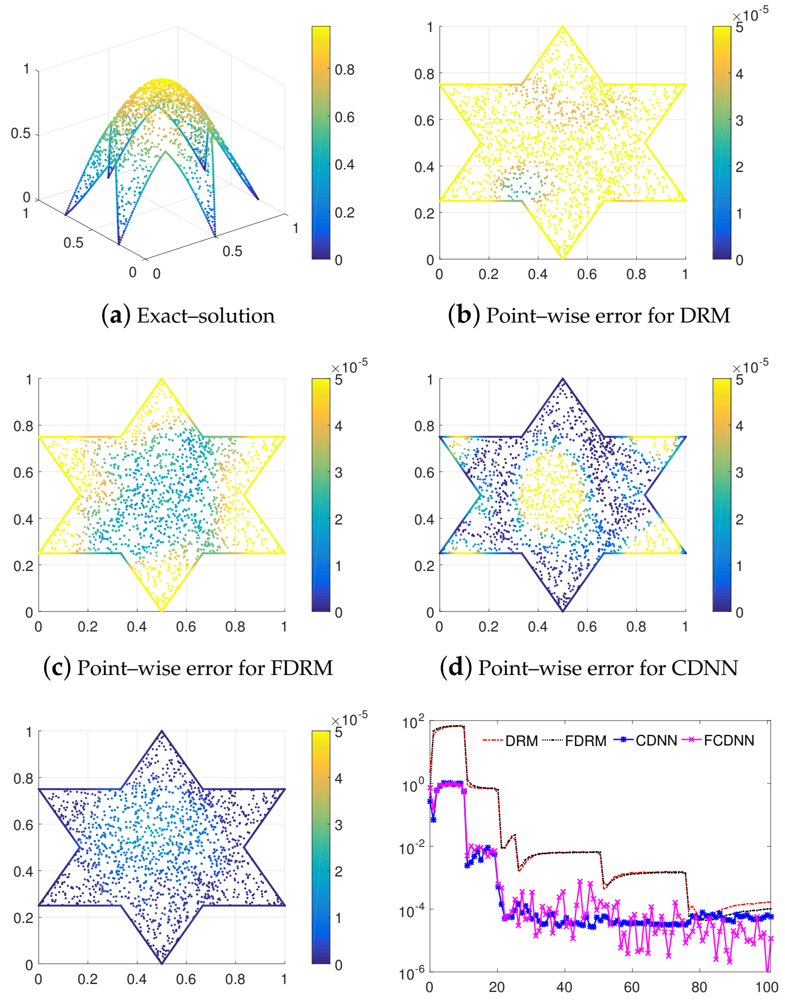
$$u(x_1,x_2) = 10x(1-2x_1^2+x_1^3)y(1-2x_2^2+x_2^3)$$

*and*

$$f(x_1,x_2) = 240x_2(1-2x_2^2+x_2^3) + 2880x_1(x_1-1)x_2(x_2-1) + 240x(1-2x_1^2+x_1^3),$$

*respectively. The functions* $g(x_1,x_2)$ *and* $h(x_1,x_2)$ *on* $\partial\Omega$ *are easy to obtain according to the exact solution; here, we omit it.*

In this example, the setups for DRM, FDRM, CDNN and FCDNN are the same as Example 4. At each training epoch, the training dataset was sampled from $\Omega$ and $\partial\Omega$, which include 3000 interior points and 500 boundary points, respectively. The testing dataset was randomly sampled from the hexagon domain in $[0,1]\times[0,1]$. We plot the numerical results in Figure 5 and list the final error results in Table 2.



(**a**) Exact–solution (**b**) Point–wise error for DRM (**c**) Point–wise error for FDRM (**d**) Point–wise error for CDNN (**e**) Point–wise error for FCDNN (**f**) REL

**Figure 5.** Testing results for Example 2.

**Table 2.** REL and running time of the aforementioned four models for Example 2.

| | **DRM** | **FDRM** | **CDNN** | **FCDNN** |
|---|---|---|---|---|
| REL | $1.65 \times 10^{-4}$ | $1.0310^{-4}$ | $5.7310^{-5}$ | $1.15 \times 10^{-5}$ |
| Time | 1.10 h | 1.26 h | 0.44 h | 0.55 h |

On the irregular domain, the CDNN and FCDNN are still able to obtain the solution of (1), and the performance of FCDNN is superior to that of DRM, FDRM and CDNN. In the meantime, the runtime of CDNNs is also approximately half of DRMs and the performance of FCDNN is still decreasing, with small oscillations when the other three methods become stable.

**Example 3.** *We solved the biharmonic Equation* (1) *on a unit cubic domain* $\Omega = [0,1] \times [0,1] \times [0,1]$ *with some holes. The exact solution is given by*

$$u(x_1, x_2, x_3) = 10x_1(1 - 2x_1^2 + x_1^3)x_2(1 - 2x_2^2 + x_2^3)x_3(1 - 2x_3^2 + x_3^3).$$

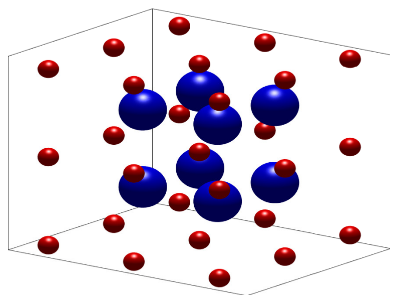
*It will naturally induce the boundary conditions* $g(x_1, x_2)$ *and* $h(x_1, x_2)$ *on* $\partial\Omega$*. By careful calculations, one can obtain the force side; here, we omit it.*

When the DRM, FDRM, CDNN and FCDNN are used to solve (1) in the three-dimension space, their network size is set as (200, 100, 80, 80, 60), (100, 100, 80, 80, 60), (100, 80, 60, 60, 40) and (50, 80, 60, 60, 40). The training data set, including 6000 interior points and 1000 boundary points, was randomly sampled from $\Omega$ and $\partial\Omega$. A testing dataset was given that includes 1600 random points distributed in $\Omega$. The testing results are plotted in Figure 6 and the final error results are listed in Table 3. For showing these results visually, we projected the point-wise error for the DRM, FDRM, CDNN and FCDNN evaluated on 1600 sample points into a rectangular region with mesh size $40 \times 40$, respectively. Note that the mapping is only for the purpose of visualization and is independent of the actual coordinates of those points.

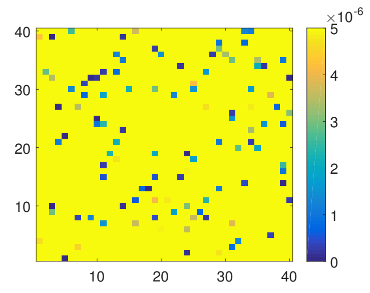
**Table 3.** REL and running time of the aforementioned four models for Example 3.

| | **DRM** | **FDRM** | **CDNN** | **FCDNN** |
|---|---|---|---|---|
| REL | 0.192 | $2.11 \times 10^{-4}$ | $1.99 \times 10^{-3}$ | $1.49 \times 10^{-5}$ |
| Time | 2.97 h | 4.16 h | 0.50 h | 0.54 h |

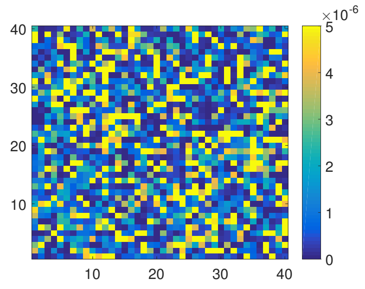
Based on the above results, we can see that the FCDNN still outperforms other models when solving the biharmonic problem (1) in a three-dimensional space, and that the performance of the CDNN becomes a bit weaker than the FDRM. In addition, the point-wise square error of the four models in Figure 6b,c, as well as the overall errors in Figure 6f, show that the performance of FCDNN is much better than that of the other three models. Compared with the case of 2D, the data in Table 3 show the run-time of CDNNs is almost kept unchanged, but the DRMs will cost more time.



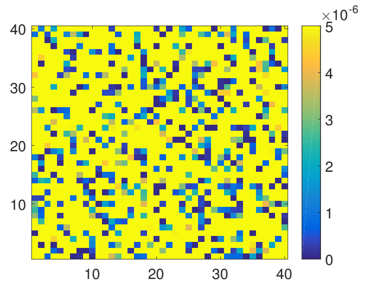
(**a**) Cube with some holes



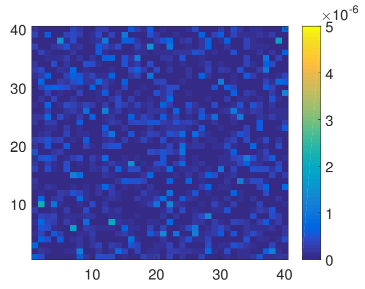
(**b**) Point–wise error for DRM



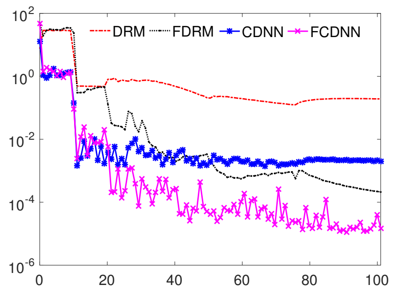
(**c**) Point–wise error for FDRM



(**d**) Point–wise error for CDNN



(**e**) Point–wise error for FCDNN



(**f**) REL

**Figure 6.** Testing results for Example 3.

**Example 4.** *We solved the biharmonic Equation* (1) *on the unit higher-dimensional domain* $\Omega = [0,1]^8$. *The exact solution and force-side are*

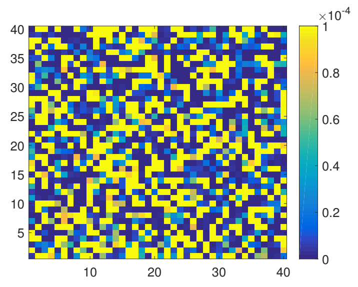
$$u(x_1, x_2, \dots, x_8) = \sin(\pi x_1)\sin(\pi x_2)\cdots\sin(\pi x_8)$$

*and*

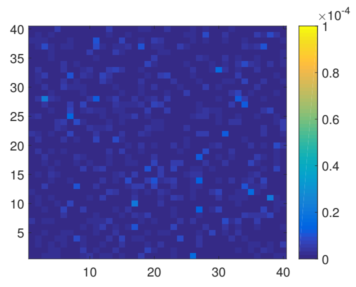
$$f(x_1, x_2, \dots, x_8) = 64\pi^4 \sin(\pi x_1)\sin(\pi x_2)\cdots\sin(\pi x_5),$$

*respectively. The boundaries of this problem satisfy* $g(x_1, x_2, \dots, x_8) = h(x_1, x_2, \dots, x_8) = 0$.

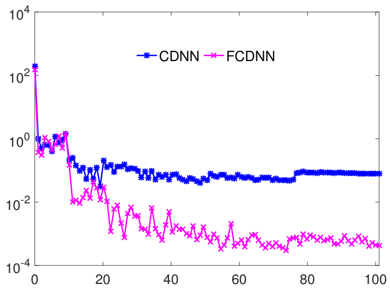
Since the DRM and FDRM needed a large amount of computing resources for this example, our station could not satisfy their requirements; as a result, we only employed the CDNN and FCDNN to solve the biharmonic Equation (1) in eight-dimensional space, where the sizes for the CDNN and FCDNN were set as (300, 200, 200, 100, 100) and (150, 200, 200, 100, 100), respectively. The training data set included 20000 interior points and 5000 boundary points randomly sampled from $\Omega$ and $\partial\Omega$. A testing dataset was given that included 1600 random points distributed in $\Omega$. The testing results are plotted in Figure 7 and the final error results are listed in Table 4. Additionally, the point-wise error for CDNN and FCDNN evaluated on 1600 sample points was projected into a rectangular region with mesh size $40 \times 40$, respectively.



(**a**) Point–wise error for CDNN



(**b**) Point–wise error for FCDNN



(**c**) REL

**Figure 7.** Testing results for Example 4.

The results in Figure 7 show that the FCDNN still maintains its good performance to approximate the exact solution of (1) in eight-dimensional space; however, the CDNN model will become slightly degenerated. In addition, the relative error of the FCDNN is much smaller than that of the CDNN based on Table 4.

**Table 4.** REL and running time of the CDNN and FCDNN for Example 4.

| | **CDNN** | **FCDNN** |
|---|---|---|
| REL | 0.0793 | 0.00042 |
| Time | 3.57 h | 4.03 h |

## 5. Discussion

Compared with the DRM algorithms, the proposed CDNN method can solve the fourth-order biharmonic equation well based on its coupled scheme. It not only effectively reduces the complexity of the DNN algorithm, but also save the resources of the computer. In the meantime, a novel activation function composed of sine and cosine is introduced; it will obviously improve the performance of DNNs in solving a complex target. In a lower dimensional space, the CDNN method costs the least time, and attains the best accuracy. Regarding a high dimensional space, the CDNN can still keep its favorable performance; however, the DRM method cannot work because of the tremendous computational burden. In addition, the idea of a coupled scheme can be extended to the PINN method; then, a coupled PINN method may be developed to solve the PDEs without a variational form or other high-order problems. In this paper, we considered the biharmonic equation with a Neumann boundary; it is suitable for our CDNN method. Thus, more complex boundary conditions, such as the Dirichlet boundary and Robin boundary or other mixed boundaries, may be considered. Different from the case of Neumann, the other boundaries will not naturally induce the boundaries for two networks; thus, it is necessary to carefully design the coupled framework and boundary constraints of the DNN. Further, the performance of the CDNN will be degenerated or even not convergent if all of the networks lack the appropriate boundary conditions. Finally, the first-order optimization methods of the

DNN and the sampling technique of training points may have an unfavorable effect on the accuracy and efficiency of DNN; this is an important issue to be solved.

## 6. Conclusions

By means of a coupled scheme of the biharmonic equation, we proposed a coupled DNN framework to solve this high-order problem in the work. As a class of a meshless method, the CDNN method does not rely on the initial guess and can approximate the solution of the biharmonic equation with a low complexity well. Additionally, a mixed loss function was designed that will enhance the stability and robustness for our model. Furthermore, a novel activation function based on Fourier approximation was introduced for the input layer, and the subsequent hidden layers were chosen as a good regularity function, such as tanh; this strategy can improve the accuracy and convergence rate for the CDNN method. Computational results show that the proposed method is feasible and efficient in solving the (1) in a complex domain and various dimension space. In the future, work is in progress to extend the neural network models to solve other high-order partial differential equations.

**Author Contributions:** Conceptualization, Methodology and Writing–original draft: X.L.; Methodology and Writing–review & editing: J.W.; Methodology, Supervision and Funding acquisition: L.Z.; Resources, Validation and Project administration: X.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by National Natural Science Foundation of China(NSFC) OF FUNDER grant number 1871339 and 1186113100.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** L.Z. is partially supported by the National Natural Science Foundation of China (NSFC 11871339, 11861131004). This work is also partially supported by HPC of School of Mathematical Sciences at Shanghai Jiao Tong University.

**Conflicts of Interest:** All authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

1. Wahba, G. *Spline Models for Observational Data*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1990.
2. Greengard, L.; Kropinski, M.C. An integral equation approach to the incompressible navier–stokes equations in two dimensions. *SIAM J. Sci. Comput.* **1998**, *20*, 318–336. [CrossRef]
3. Ferziger, J.H.; Peric, M. *Computational Methods for Fluid Dynamics*; Springer: Berlin/Heidelberg, Germany, 2002.
4. Christiansen, S. Integral equations without a unique solution can be made useful for solving some plane harmonic problems. *IMA J. Appl. Math.* **1975**, *16*, 143–159. [CrossRef]
5. Constanda, C. The boundary integral equation method in plane elasticity. *Proc. Am. Math. Soc.* **1995**, *123*, 3385–3396. [CrossRef]
6. Gupta, M.M.; Manohar, R.P. Direct solution of the biharmonic equation using noncoupled approach. *J. Comput. Phys.* **1979**, *33*, 236–248. [CrossRef]
7. Altas, I.; Dym, J.; Gupta, M.M.; Manohar, R.P. Multigrid solution of automatically generated high-order discretizations for the biharmonic equation. *SIAM J. Sci. Comput.* **1998**, *19*, 1575–1585. [CrossRef]
8. Ben-Artzi, M.; Croisille, J.P.; Fishelov, D. A fast direct solver for the biharmonic problem in a rectangular grid. *SIAM J. Sci. Comput.* **2008**, *31*, 303–333. [CrossRef]
9. Bialecki, B. A fourth order finite difference method for the Dirichlet biharmonic problem. *Numer. Algorithms* **2012**, *61*, 351–375.
10. Bi, C.J.; Li, L.K. Mortar finite volume method with Adini element for biharmonic problem. *J. Comput. Math.* **2004**, *22*, 475–488.
11. Wang, T. A mixed finite volume element method based on rectangular mesh for biharmonic equations. *J. Comput. Appl. Math.* **2004**, *172*, 117–130. [CrossRef]
12. Eymard, R.; Gallouët, T.; Herbin, R.; Linke, A. Finite volume schemes for the biharmonic problem on general meshes. *Math. Comput.* **2012**, *81*, 2019–2048. [CrossRef]
13. Baker, G.A. Finite element methods for elliptic equations using nonconforming elements. *Math. Comput.* **1977**, *31*, 45–59. [CrossRef]
14. Lascaux, P.; Lesaint, P. Some nonconforming finite elements for the plate bending problem. *Rev. Française D'automatique Inform. Rech. Opérationnelle Anal. Numérique* **1975**, *9*, 9–53. [CrossRef]

15. Morley, L.S.D. The triangular equilibrium element in the solution of plate bending problems. *Aeronaut. Q.* **1968**, *19*, 149–169. [CrossRef]
16. Zienkiewicz, O.C.; Taylor, R.L.; Nithiarasu, P. *The Finite Element Method for Fluid Dynamics*, 6th ed.; Elsevier Butterworth-Heinemann: Oxford, UK, 2005.
17. Ciarlet, P. *The Finite Element Method for Elliptic Problems*; North-Holland Publishing Company: Amsterdam, The Netherlands, 1978.
18. Smith, J. The coupled equation approach to the numerical solution of the biharmonic equation by finite differences. II. *SIAM J. Numer. Anal.* **1968**, *5*, 104–111. [CrossRef]
19. Ehrlich, L.W. Solving the biharmonic equation as coupled finite difference equations. *SIAM J. Numer. Anal.* **1971**, *8*, 278–287. [CrossRef]
20. Brezzi, F.; Fortin, M. *Mixed and Hybrid Finite Element Methods*; Springer: New York, NY, USA, 1991.
21. Cheng, X.L.; Han, W.; Huang, H.C. Some mixed finite element methods for biharmonic equation. *J. Comput. Appl. Math.* **2000**, *126*, 91–109. [CrossRef]
22. Davini, C.; Pitacco, I. An unconstrained mixed method for the biharmonic problem. *SIAM J. Numer. Anal.* **2000**, *38*, 820–836. [CrossRef]
23. Lamichhane, B.P. A stabilized mixed finite element method for the biharmonic equation based on biorthogonal systems. *J. Comput. Appl. Math.* **2011**, *235*, 5188–5197. [CrossRef]
24. Stein, O.; Grinspun, E.; Jacobson, A.; Wardetzky, M. A mixed finite element method with piecewise linear elements for the biharmonic equation on surfaces. *arXiv* **2019**, arXiv:1911.08029.
25. Mai-Duy, N.; See, H.; Tran-Cong, T. A spectral collocation technique based on integrated Chebyshev polynomials for biharmonic problems in irregular domains. *Appl. Math. Model.* **2009**, *33*, 284–299. [CrossRef]
26. Bialecki, B.; Karageorghis, A. Spectral Chebyshev collocation for the Poisson and biharmonic equations. *SIAM J. Sci. Comput.* **2010**, *32*, 2995–3019. [CrossRef]
27. Bialecki, B.; Fairweather, G.; Karageorghis, A.; Maack, J. A quadratic spline collocation method for the Dirichlet biharmonic problem. *Numer. Algorithms* **2020**, *83*, 165–199. [CrossRef]
28. Mai-Duy, N.; Tanner, R. An effective high order interpolation scheme in BIEM for biharmonic boundary value problems. *Eng. Anal. Bound. Elem.* **2005**, *29*, 210–223. [CrossRef]
29. Adibi, H.; Es'haghi, J. Numerical solution for biharmonic equation using multilevel radial basis functions and domain decomposition methods. *Appl. Math. Comput.* **2007**, *186*, 246–255. [CrossRef]
30. Li, X.; Zhu, J.; Zhang, S. A meshless method based on boundary integral equations and radial basis functions for biharmonic-type problems. *Appl. Math. Model.* **2011**, *35*, 737–751. [CrossRef]
31. E, W.; Han, J.; Jentzen, A. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun. Math. Stat.* **2017**, *5*, 349–380. [CrossRef]
32. Weinan, E.; Yu, B. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.* **2018**, *1*, 1–12.
33. Berg, J.; Nyström, K. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing* **2018**, *317*, 28–41. [CrossRef]
34. Sirignano, J.; Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **2018**, *375*, 1339–1364. [CrossRef]
35. Nabian, M.A.; Meidani, H. A deep learning solution approach for high-dimensional random differential equations. *Probabilistic Eng. Mech.* **2019**, *57*, 14–25. [CrossRef]
36. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
37. Zou, Z.; Zhang, H.; Guan, Y.; Zhang, J. Deep residual neural networks resolve quartet molecular phylogenies. *Mol. Biol. Evol.* **2020**, *37*, 1495–1507. [CrossRef] [PubMed]
38. Guo, H.; Zhuang, X.; Rabczuk, T. A deep collocation method for the bending analysis of Kirchhoff plate. *CMC-Comput. Mater. Contin.* **2019**, *59*, 433–456. [CrossRef]
39. Samaniego, E.; Anitescu, C.; Goswami, S.; Nguyen-Thanh, V.M.; Guo, H.; Hamdia, K.; Zhuang, X.; Rabczuk, T. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Comput. Methods Appl. Mech. Eng.* **2020**, *362*, 112790. [CrossRef]
40. Li, W.; Bazant, M.Z.; Zhu, J. A physics-guided neural network framework for elastic plates: Comparison of governing equations-based and energy-based approaches. *Comput. Methods Appl. Mech. Eng.* **2021**, *383*, 113933. [CrossRef]
41. Mohammad, V.; Ehsan, H.; Maryam, K.; Nasser, K. A physics-informed neural network approach to solution and identification of biharmonic equations of elasticity. *J. Eng. Mech.* **2021**, *148*, 04021154.
42. Zhongmin, H.; Zhen, X.; Yishen, Z.; Linxin, P. Deflection-bending moment coupling neural network method for the bending problem of thin plates with in-plane stiffness gradient. *Chin. J. Theor. Appl. Mech.* **2021**, *53*, 25–41.
43. Goswami, S.; Anitescu, C.; Rabczuk, T. Adaptive fourth-order phase field analysis using deep energy minimization. *Theor. Appl. Fract. Mech.* **2020**, *107*, 102527. [CrossRef]
44. Lyu, L.; Zhang, Z.; Chen, M.; Chen, J. MIM: A deep mixed residual method for solving high-order partial differential equations. *J. Comput. Phys.* **2022**, *452*, 110930. [CrossRef]

45. Wang, S.; Wang, H.; Perdikaris, P. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Comput. Methods Appl. Mech. Eng.* **2021**, *384*, 113938. [CrossRef]
46. Tancik, M.; Srinivasan, P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J.; Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7537–7547.
47. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NE, USA, 27–30 June 2016; pp. 770–778.
48. Duan, C.; Jiao, Y.; Lai, Y.; Li, D.; Yang, Z. Convergence rate analysis for deep ritz method. *Commun. Comput. Phys.* **2022**, *31*, 1020–1048. [CrossRef]
49. Jiao, Y.; Lai, Y.; Lo, Y.; Wang, Y.; Yang, Y. Error analysis of deep Ritz methods for elliptic equations. *arXiv* **2021**, arXiv:2107.14478.
50. Robert, C.P.; Casella, G. *Monte Carlo Statistical Methods*; Springer: New York, NY, USA, 2004.
51. Xu, Z.Q.J.; Zhang, Y.; Luo, T.; Xiao, Y.; Ma, Z. Frequency principle: Fourier analysis sheds light on deep neural networks. *Commun. Comput. Phys.* **2020**, *28*, 1746–1767. [CrossRef]
52. Rahaman, N.; Baratin, A.; Arpit, D.; Draxler, F.; Lin, M.; Hamprecht, F.; Bengio, Y.; Courville, A. On the spectral bias of neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 5301–5310.
53. Jacot, A.; Gabriel, F.; Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *Adv. Neural Inf. Process. Syst.* **2018**, 8571–8580.
54. Xu, Z.J. Understanding training and generalization in deep learning by fourier analysis. *arXiv* **2018**, arXiv:1808.04295.
55. Xu, Z.Q.J.; Zhang, Y.; Xiao, Y. Training behavior of deep neural network in frequency domain. In Proceedings of the International Conference on Neural Information Processing, Vancouver, BC, Canada, 8–14 December 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 264–274.