Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.OA-2021-0035

Vol. **15**, No. 1, pp. 42-67
February 2022

A Local Deep Learning Method for Solving High Order Partial Differential Equations

Jiang Yang[1,2,3] and Quanhui Zhu[2,*]

[1] *International Center of Mathematics, Southern University of Science and Technology, Shenzhen 518055, China*
[2] *Department of Mathematics, Southern University of Science and Technology, Shenzhen 518055, China*
[3] *Guangdong Provincial Key Laboratory of Computational Science and Material Design, Southern University of Science and Technology, Shenzhen 518055, China*

Received 8 March 2021; Accepted (in revised version) 12 August 2021

Abstract. At present, deep learning based methods are being employed to resolve the computational challenges of high-dimensional partial differential equations (PDEs). But the computation of the high order derivatives of neural networks is costly, and high order derivatives lack robustness for training purposes. We propose a novel approach to solving PDEs with high order derivatives by simultaneously approximating the function value and derivatives. We introduce intermediate variables to rewrite the PDEs into a system of low order differential equations as what is done in the local discontinuous Galerkin method. The intermediate variables and the solutions to the PDEs are simultaneously approximated by a multi-output deep neural network. By taking the residual of the system as a loss function, we can optimize the network parameters to approximate the solution. The whole process relies on low order derivatives. Numerous numerical examples are carried out to demonstrate that our local deep learning is efficient, robust, flexible, and is particularly well-suited for high-dimensional PDEs with high order derivatives.

AMS subject classifications: 35Q68

Key words: Deep learning, deep neural network, high order PDEs, reduction of order, deep Galerkin method.

1. Introduction

Partial differential equations (PDEs) play a significant role in the fields of physics, chemistry, biology, engineering, finance, and others. Classical numerical methods focus

[*]Corresponding author. *Email addresses:* `yangj7@sustech.edu.cn` (J. Yang), `11849393@mail.sustech.edu.cn` (Q. Zhu)

on designing efficient, accurate, and stable numerical schemes. Within the context of high-dimensional problems, however, the curse of dimensionality renders classical numerical methods impractical. As a result, many mathematicians have introduced neural networks into PDEs precisely because multilayer feedforward networks are proven to be universal approximators for the PDEs [15, 16]. More specifically, once the network structure is determined, any order derivatives of the neural network can be obtained analytically. Coupled with the automatic differentiation technique, neural networks can be applied to solve PDEs [2]. Depending upon different purposes, neural networks can be used to approximate the solution function, represent the solution solver, and even invert the equations.

In this paper, we consider the deep learning method as a means to solve the following k-th order initial boundary value problem (IBVP):

$$\begin{cases} u_t = \mathcal{L}(u), & x \in \Omega, \quad t \in [0,T], \\ u(x,0) = u_0(x), & x \in \Omega, \\ \mathcal{B}u = \mathbf{g}, & x \in \partial\Omega, \quad t \in [0,T], \end{cases} \tag{1.1}$$

where $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}_+$, $\mathcal{L}(u) = F(x,t,u,Du,\cdots,D^k u)$, F and $\mathbf{g}$ are linear or nonlinear functions, $\mathcal{B}$ is the boundary condition operator, and the p-th order derivative operator D^p consists of

$$\partial_{x_1}^{\alpha_1}\partial_{x_2}^{\alpha_2}\cdots\partial_{x_d}^{\alpha_d}u \quad \text{with} \quad \sum \alpha_i = p, \quad \alpha_i \in \mathbb{N}.$$

The neural network function $\varphi(x,t;\theta) : \mathbb{R}^{d+1} \times \Theta^M \mapsto \mathbb{R}^m$ is defined as follows:

$$\begin{aligned} &\varphi(x,t;\theta) = \mathcal{N}_{\text{out}} \circ \mathcal{N}_L \circ \cdots \circ \mathcal{N}_1 \circ \mathcal{N}_{\text{in}}(x,t), \\ &\mathcal{N}_{\text{in}}(x,t) = \sigma_{\text{in}}(\alpha x + \beta t + b), \quad \alpha \in \mathbb{R}^{n\times d}, \quad \beta, b \in \mathbb{R}^n, \\ &\mathcal{N}_{\text{out}}(y) = \sigma_{\text{out}}(\gamma y + c), \qquad \gamma \in \mathbb{R}^{m\times n}, \quad c \in \mathbb{R}^m, \end{aligned} \tag{1.2}$$

where d is the dimension of x, m is the dimension of the output, n is the width of the hidden layers, L is the number of the hidden layers (i.e., the network's depth) and $\mathcal{N}_i : \mathbb{R}^n \mapsto \mathbb{R}^n$ is the structure of the hidden layers. σ_{in} usually is the same nonpolynomial activation function as the hidden layers and σ_{out} is set as an equivalent function in most cases, i.e., $\sigma_{\text{out}}(x) = x$. For specific examples, a proper output transformation σ_{out} should be determined. Our goal is to find a suitable neural network $\varphi(x,t;\theta)$ to approximate a solution $u(x,t)$ to the problem (1.1).

[7] gives an overview of the progress that has been made in linking computational mathematics and machine learning. In most of existing literatures, the loss function is determined by either the PDEs or an equivalent formulation. For instance, the parabolic PDE is reformulated as a backward stochastic differential equation in [10, 11, 31], where the loss function is given by the solution of the backward stochastic differential equation, and the training process is shown to be a deep reinforcement learning process. In [28,29], the solution is approximated by a neural network. The proposal of a mesh-free algorithm makes high-dimensional calculations feasible. [21,30] provides