SELECTNET: SELF-PACED LEARNING FOR HIGH-DIMENSIONAL PARTIAL DIFFERENTIAL EQUATIONS

YIQI GU

DEPARTMENT OF MATHEMATICS, NATIONAL UNIVERSITY OF SINGAPORE, 10 LOWER KENT RIDGE ROAD, SINGAPORE, 119076 (MATGUY@NUS.EDU.SG)

HAIZHAO YANG*

DEPARTMENT OF MATHEMATICS, PURDUE UNIVERSITY†, WEST LAFAYETTE, IN 47907, USA
DEPARTMENT OF MATHEMATICS, NATIONAL UNIVERSITY OF SINGAPORE‡, SINGAPORE, 119076
(HAIZHAO@PURDUE.EDU)

CHAO ZHOU

DEPARTMENT OF MATHEMATICS, NATIONAL UNIVERSITY OF SINGAPORE, 10 LOWER KENT RIDGE ROAD, SINGAPORE, 119076 (MATZC@NUS.EDU.SG)

Abstract. The residual method with deep neural networks as function parametrization has been applied to solve certain high-dimensional partial differential equations (PDEs) successfully; however, its convergence is slow and might not be guaranteed even within a simple class of PDEs. To improve the convergence of the network-based residual method, we introduce a novel self-paced learning framework, SelectNet, which quantifies the difficulty of training samples, chooses simpler samples in the early stage of training, and slowly explores more challenging samples, e.g., samples with larger residual errors, mimicking the human cognitive process for more efficient learning. In particular, a selection network and the PDE solution network are trained simultaneously; the selection network adaptively weighting the training samples of the solution network achieving the goal of self-paced learning. Numerical examples indicate that the proposed SelectNet model outperforms existing models on the convergence speed and the convergence robustness, especially for low-regularity solutions.

Key words. High-Dimensional PDEs; Deep Neural Networks; Self-Paced Learning; Selected Sampling; Minimal Residual Method; Convergence.

AMS subject classifications. 65M75; 65N75; 62M45;

1. Introduction.

High-dimensional PDEs are important tools in physical, financial, and biological models [29, 14, 48, 15, 45]. However, developing numerical methods for high-dimensional PDEs has been a challenging task due to the curse of dimensionality in the discretization of the problem. For example, in traditional methods such as finite difference methods and finite element methods, $O(N^d)$ degree of freedom is required for a d-dimensional problem if we set N grid points or basis functions in each direction. Even if d becomes moderately large, the exponential growth N^d in the dimension d makes traditional methods immediately computationally intractable.

Recent research of the approximation theory of deep neural networks shows that deep network approximation is a powerful tool for mesh-free function parametrization. The research on the approximation theory of neural networks traces back to the pioneering work [7, 18, 1] on the universal approximation of shallow networks with sigmoid activation functions. The recent research focus was on the approximation rate of deep neural networks for various function spaces in terms of the number of network parameters showing that deep networks are more powerful than shallow networks in terms of approximation efficiency. For example, smooth functions [33, 31, 46, 13, 35, 44, 12], piecewise smooth functions [38], band-limited functions [37], continuous functions [47, 41, 40]. The reader is referred to [40] for the explicit characterization of the approximation error for networks

*CORRESPONDING AUTHOR.
†CURRENT INSTITUTE.
‡THE PROJECT WAS STARTED IN SINGAPORE.

with an arbitrary width and depth.

In particular, deep network approximation can lessen or overcome the curse of dimensionality under certain circumstances, making it an attractive tool for solving high-dimensional problems. For a sufficiently smooth function that is the integral transform of a high dimensional (essentially) compactly supported function with a one-dimensional integral kernel, the no curse of dimensionality can be shown via establishing the connection of network approximation with the Monte Carlo sampling or equivalently the law of large numbers [1, 37]. For functions in the Korobov space, connecting deep network approximation and sparse grid approximation leads to the fact that deep network approximation can lessen the curse of dimensionality [35]. For general continuous functions, [36] proves that deep network approximation can significantly reduce the curse of dimensionality through the Kolmogorov-Arnold superposition theorem. Finally, if the approximation error is only concerned on a low-dimensional manifold, there is no curse of dimensionality for deep network approximation [5, 4, 40].

As an efficient function parametrization tool, neural networks have been applied to solve PDEs via various approaches. Early work in [28] applies neural networks to approximate PDE solutions defined on grid points. Later in [9, 26], deep neural networks are employed to approximate solutions in the whole domain and PDEs are solved by minimizing the discrete L^2 residual at prescribed collocation points. Deep neural networks coupled with boundary governing terms by design can satisfy boundary conditions [34]. Nevertheless, designing boundary governing terms is usually difficult for complex geometry. Another approach to enforcing boundary conditions is to add boundary residual errors to the loss function as a penalized term and minimize it as well as the PDE residual error [16, 27]. The second technique is in the same spirit of residual methods in finite element methods and is more convenient in implementation. Therefore, it has been widely utilized for PDEs with complex domains. However, network computation was usually expensive limiting the applications of network-based PDE solvers. Thanks to the development of GPU-based parallel computing over the last two decades, which greatly boosts the network computation, network-based PDE solvers were revisited recently and have become a popular tool especially for high-dimensional problems [42, 3, 49, 30, 11, 17, 2, 20, 19]. Nevertheless, most network-based PDE solvers suffer from robustness issues: their convergence is slow and might not be guaranteed even within a simple class of PDEs.

To ease the issue above, we introduce a novel self-paced learning framework, SelectNet, to adaptively choose training samples in the residual method. Self-paced learning [24] is a recently raised learning technique that can choose a part of the training samples for actual training over time. Specifically, for a training data set with n samplings, self-paced learning uses a vector $v \in \{0,1\}^n$ to indicate whether or not each training sample should be included in the current training stage. The philosophy of self-paced learning is to simulate the learning style of human beings, which tends to learn easier aspects of a learning task first and deal with more complicated samplings later. Based on self-paced learning, a novel technique for selected sampling is put forward, which uses a selection neural network instead of the 0-1 selection vector v. Hence, it helps learning avoid redundant training information and speeds up the convergence of learning outcomes. This idea is further improved in [21] by introducing a deep neural network to select training data for image classification. Among similar works, a state-of-the-art algorithm named as SelectNet is proposed in [32] for image classification, especially for imbalanced data problems. Based on the observation that samples near the singularity of the PDE solution are rare compared to samples from the regular part, we extend the SelectNet [32] to network-based residual methods especially for PDE solutions with certain irregularity. As we shall see later, numerical results show that the proposed method is competitive with the traditional (basic) residual method for analytic solutions, and it outperforms others for low-regularity solutions, in the aspect of the convergence speed.

The organization of this paper is as follows. In Section 2, we introduce the residual methods and formulate the corresponding optimization problem. In Section 3, we present the SelectNet model in detail. In Section 4, we discuss the network implementation in the proposed model. In Section 5, we present ample numerical experiments for various equations to validate our method. We conclude with some remarks in the final section.

2. Residual Methods for PDEs. In this work, we aim at solving the following (initial) boundary value problems, giving a bounded domain $\Omega \subset \mathbb{R}^d$:

- elliptic equations

$$\begin{aligned} &\mathcal{D}_x u(x) = f(x), \text{ in } \Omega, \\ &\mathcal{B}_x u(x) = g_0(x), \text{ on } \partial\Omega; \end{aligned} \tag{2.1}$$

- parabolic equations

$$\begin{aligned} &\frac{\partial u(x,t)}{\partial t} - \mathcal{D}_x u(x,t) = f(x,t), \text{ in } \Omega \times (0,T), \\ &\mathcal{B}_x u(x,t) = g_0(x,t), \text{ on } \partial\Omega \times (0,T), \\ &u(x,0) = h_0(x), \text{ in } \Omega; \end{aligned} \tag{2.2}$$

- hyperbolic equations

$$\begin{aligned} &\frac{\partial^2 u(x,t)}{\partial t^2} - \mathcal{D}_x u(x,t) = f(x,t), \text{ in } \Omega \times (0,T), \\ &\mathcal{B}_x u(x,t) = g_0(x,t), \text{ on } \partial\Omega \times (0,T), \\ &u(x,0) = h_0(x), \quad \frac{\partial u(x,0)}{\partial t} = h_1(x) \text{ in } \Omega; \end{aligned} \tag{2.3}$$

where u is the solution function; f, g_0, h_0, h_1 are given data functions; $\mathcal{D}_x$ is a spatial differential operator concerning the derivatives of x; $\mathcal{B}_x$ is a boundary operator specifying a Dirichlet, Neumann or Robin boundary condition.

In our methods, the temporal variable t will be regarded as an extra spatial coordinate, and it will not be dealt with differently from x. For simplicity, the PDEs in (2.1)-(2.3) are unified in the following form

$$\begin{aligned} &\mathcal{D}u(\boldsymbol{x}) = f(\boldsymbol{x}), \text{ in } Q, \\ &\mathcal{B}u(\boldsymbol{x}) = g(\boldsymbol{x}), \text{ in } \Gamma, \end{aligned} \tag{2.4}$$

where $\boldsymbol{x}$ includes the spatial variable and possible the temporal variable; $\mathcal{D}u = f$ represents a generic PDE; $\mathcal{B}u = g$ represents the governing conditions including the boundary condition and the possible initial condition; Q and Γ are the corresponding domains of the equations.

Now we seek a neural network $u(\boldsymbol{x};\boldsymbol{\theta})$ approximating the solution $u(\boldsymbol{x})$ of the PDE (2.4). Then the residuals for the PDE and the governing conditions can be written by

$$\mathcal{R}_Q(u(\boldsymbol{x};\boldsymbol{\theta})) := \mathcal{D}u(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x}), \quad \mathcal{R}_\Gamma(u(\boldsymbol{x};\boldsymbol{\theta})) := \mathcal{B}u(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x}). \tag{2.5}$$

One can solve the PDE by searching for the optimal parameters of the network that minimizes the square sum of these two residuals, i.e.

$$\min_{\boldsymbol{\theta}} \|\mathcal{R}_Q(u(\boldsymbol{x};\boldsymbol{\theta}))\|_Q^2 + \lambda \|\mathcal{R}_\Gamma(u(\boldsymbol{x};\boldsymbol{\theta}))\|_\Gamma^2, \tag{2.6}$$

where $\|\cdot\|_*$ is usually the L^2-norm and λ is a parameter for weighting the sum, e.g.,

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\in Q}\left[|\mathcal{D}u(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})|^2\right] + \lambda \mathbb{E}_{\boldsymbol{x}\in\Gamma}\left[|\mathcal{B}u(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})|^2\right]. \tag{2.7}$$

3. SelectNet for Residual Methods. The network-based residual method has been applied to solve certain high-dimensional PDEs successfully. However, its convergence is slow and might not be guaranteed even within a simple class of PDEs. To ease this issue, we introduce a novel self-paced learning framework, SelectNet, to adaptively choose training samples in the residual method. The basic philosophy is to mimic the human cognitive process for more efficient learning: learning first from easier examples and slowly exploring more complicated ones. The proposed method is related to selected sampling [6, 22], an important tool of deep learning for computer science applications. Nevertheless, the effectiveness of selected sampling in scientific computing has not been fully explored yet.

In particular, a selection network $\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)$ and the PDE solution network $u(\boldsymbol{x};\boldsymbol{\theta})$ are trained simultaneously; the selection network adaptively weighting the training samples of the solution network achieving the goal of self-paced learning. $\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)$ is a "mentor" helping to decide whether a sample $\boldsymbol{x}$ is important enough to train the "student" network $u(\boldsymbol{x};\boldsymbol{\theta})$. The "mentor" could significantly avoid redundant training information and help to speed up the convergence. This idea is originally from self-paced learning [25] and is further improved in [21] by introducing a DNN to select training data for image classification. Among similar works, a state-of-the-art algorithm named as SelectNet was proposed in [32] for image classification, especially for imbalanced data problem. Based on the observation that samples near the singularity of the PDE solution are rare compared to samples from the regular part, we extend the SelectNet [32] to network-based residual methods especially for PDE solutions with certain irregularity.

Originally in image classification, for a training data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i))\}_{i=1}^n$, self-paced learning uses a vector $\boldsymbol{v} \in \{0,1\}^n$ to indicate whether or not each training sample should be included in the current training stage ($v_i = 1$ if the ith sample is included in the current iteration). The overall target function including $\boldsymbol{v}$ is

$$\min_{\boldsymbol{\theta},\boldsymbol{v}\in\{0,1\}^n} \sum_{i=1}^n v_i \mathcal{L}(y_i, \phi(\boldsymbol{x}_i;\boldsymbol{\theta})) - \lambda \sum_{i=1}^n v_i, \tag{3.1}$$

where $\mathcal{L}(y_i, \phi(\boldsymbol{x}_i;\boldsymbol{\theta}))$ denotes the loss function of a DNN $\phi(\boldsymbol{x}_i;\boldsymbol{\theta})$ for classifying a sample $\boldsymbol{x_i}$ to y_i. When this model is relaxed to $\boldsymbol{v} \in [0,1]^n$ and the alternative convex search is applied to solve the relaxed optimization, a straightforward derivation easily reveals a rule for the optimal value for each entry $v_i^{(t)}$ in the t-th iteration as

$$v_i^{(t)} = 1, \text{ if } \mathcal{L}(y_i, \phi(\boldsymbol{x}_i;\boldsymbol{\theta}^{(t)})) < \lambda, \qquad \text{and} \qquad v_i^{(t)} = 0, \text{ otherwise.} \tag{3.2}$$

A sample with a smaller loss than the threshold λ is treated as an "easy" sample and will be selected in training. When computing $\boldsymbol{\theta}^{(t+1)}$ with a fixed $\boldsymbol{v}^{(t)}$, the classifier is trained only on the selected "easy" samples. This mechanism helps to reduce the generalization error for image classification when the training data distribution is usually different from the test data distribution.

When solving high dimensional PDEs, the training and test data distributions are the same and there is no limitation for sampling. Hence, the desired mechanism in this case is to: 1) consider no bias on choosing samples for fast convergence in the early stage of training; 2) exclude "easy" samples and focus on "difficult" samples for better convergence in the latter stage of training. In particular, the SelectNet $\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)$ should satisfy the following requirements. First of all, as a weight function, $\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)$ is set to be bounded and uniformly scaled as follows:

$$0 \le m_0 < \phi_s(\boldsymbol{x};\boldsymbol{\theta}_s) < M_0, \quad \forall \boldsymbol{x} \text{ and } \boldsymbol{\theta}_s, \tag{3.3}$$

$$\frac{1}{|Q|}\int_Q \phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)d\boldsymbol{x} = 1, \quad \frac{1}{|\Gamma|}\int_\Gamma \phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)d\boldsymbol{x} = 1. \tag{3.4}$$

Second, $\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)$ should generate no bias towards samples in the early stage of training. Third, the role of $\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)$ is to add higher weights to samples with larger point-wise residual errors in the latter stage of training.

Hence, based on the model of residual methods in (2.7) and the above requirements, we have the following SelectNet framework for the residual method:

$$\begin{aligned}(3.5)\quad \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\theta}_s} \mathbb{E}_{x\in Q}\left[\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)|\mathcal{D}u(\boldsymbol{x};\boldsymbol{\theta})-f(\boldsymbol{x})|^2\right] \\ + \lambda\mathbb{E}_{x\in\Gamma}\left[\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)|\mathcal{B}u(\boldsymbol{x};\boldsymbol{\theta})|^2\right] \\ - \varepsilon^{-1}\left[\left(\frac{1}{|Q|}\int_Q \phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)d\boldsymbol{x}-1\right)^2+\left(\frac{1}{|\Gamma|}\int_\Gamma \phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)d\boldsymbol{x}-1\right)^2\right],\end{aligned}$$

where the last penalty term with a small penalty parameter $\varepsilon>0$ is to enforce (3.4) to be satisfied approximately. Besides, the condition in (3.3) holds automatically if the last layer of activation functions of $\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)$ is bounded and the network output is properly rescaled and shifted as we shall discuss later in the next section. In the early stage of training, since $\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)$ is randomly initialized with mean zero random variables and hence it has a slowly varying and smooth function configuration with a high probability satisfying the second requirement above. In the latter stage of training, $\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)$ has been optimized to choose difficult samples and the third requirement above is satisfied.

4. Network Implementation.

4.1. Network Architecture. The proposed framework is independent of the choice of deep neural networks. Advanced network design may improve the accuracy and convergence of the proposed framework, which would be interesting future work.

In this paper, feedforward neural networks will be repeatedly applied. Let $\phi(\boldsymbol{x};\boldsymbol{\theta})$ denote such a network with an input $\boldsymbol{x}$ and parameters $\boldsymbol{\theta}$, then it is defined recursively as follows:

$$\begin{aligned}&\boldsymbol{x}^0=\boldsymbol{x},\\ &\boldsymbol{x}^{l+1}=\sigma(\boldsymbol{W}^l\boldsymbol{x}^l+\boldsymbol{b}^l),\quad l=0,1,\cdots,L-1,\\ &\phi(\boldsymbol{x};\boldsymbol{\theta})=\boldsymbol{W}^L\boldsymbol{x}^L+\boldsymbol{b}^L,\end{aligned}\tag{4.1}$$

where σ is an application-dependent nonlinear activation function, and $\boldsymbol{\theta}$ consists of all the weights and biases $\{\boldsymbol{W}^l,\boldsymbol{b}^l\}_{l=0}^L$ satisfying

$$\begin{aligned}&\boldsymbol{W}^0\in\mathbb{R}^{m\times d},\quad \boldsymbol{W}^L\in\mathbb{R}^{1\times m},\quad \boldsymbol{b}^L\in\mathbb{R},\\ &\boldsymbol{W}^l\in\mathbb{R}^{m\times m},\quad \text{for } l=1,\cdots,L-1,\\ &\boldsymbol{b}^l\in\mathbb{R}^{m\times 1},\quad \text{for } l=0,\cdots,L-1.\end{aligned}\tag{4.2}$$

The number m is called the width of the network and L is called the depth.

For simplicity, we deploy the feedforward neural network with the activation function $\sigma(\boldsymbol{x})=\sin(\boldsymbol{x})$ as the solution network that approximates the solution of the PDE. As for the selection network introduced in Section 3, since it is required to be bounded in $[m_0,M_0]$, it can be defined via

$$\phi_s(\boldsymbol{x};\boldsymbol{\theta})=(M_0-m_0)\sigma_s(\hat{\phi}(\boldsymbol{x};\boldsymbol{\theta}))+m_0,\tag{4.3}$$

where $\sigma_s(x)=1/(1+\exp(-x))$ is the sigmoidal function, and $\hat{\phi}$ is a generic network, e.g. a feedforward neural network with the ReLU activation $\sigma(\boldsymbol{x})=\max\{0,\boldsymbol{x}\}$.

4.2. Special Network for Dirichlet Boundary Conditions. In the case of homogeneous Dirichlet boundary conditions, it is worth mentioning a special network design that satisfies the boundary condition automatically as discussed in [26, 3].

Let us focus on the boundary value problem to introduce this special network stucture. It is straightforward to generalize this idea to the case of an initial boundary value problem and we omit this discussion. Assume a homogeneous Dirichlet boundary condition

$$u(\boldsymbol{x}) = 0, \quad \text{on } \Gamma. \tag{4.4}$$

Then a solution network automatically satisfying the above condition can be constructed by

$$u(\boldsymbol{x};\boldsymbol{\theta}) = h(\boldsymbol{x})\hat{u}(\boldsymbol{x};\boldsymbol{\theta}), \tag{4.5}$$

where $\hat{u}$ is a generic network as in (4.1), and h is a specifically chosen function such as $h = 0$ on Γ.

For example, if Γ is a d-dimensional unit sphere, then $u(\boldsymbol{x};\boldsymbol{\theta})$ can take the form

$$u(\boldsymbol{x};\boldsymbol{\theta}) = (|\boldsymbol{x}|^2 - 1)\hat{u}(\boldsymbol{x};\boldsymbol{\theta}). \tag{4.6}$$

For another example, if Γ is the boundary of the d-dimensional cube $[-1,1]^d$, then $u(\boldsymbol{x};\boldsymbol{\theta})$ can take the form

$$u(\boldsymbol{x};\boldsymbol{\theta}) = \prod_{i=1}^{d}(x_i^2 - 1)\hat{u}(\boldsymbol{x};\boldsymbol{\theta}). \tag{4.7}$$

Since the boundary condition $\mathcal{B}u = 0$ is always fulfilled, it suffices to solve the minmax problem

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\theta}_s} \mathbb{E}_{x\in Q}\left[\phi_s(\boldsymbol{x};\boldsymbol{\theta}_s)|\mathcal{D}u(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})|^2\right] - \varepsilon^{-1}\left(\frac{1}{|Q|}\int_Q \phi_s(\boldsymbol{x};\boldsymbol{\theta}_s) - 1\right)^2 \tag{4.8}$$

to identify the best solution network $u(\boldsymbol{x};\boldsymbol{\theta})$.

4.3. Derivatives of Networks. Note that the evaluation of the optimization problem in (3.5) involves the derivative of the network $u(\boldsymbol{x};\boldsymbol{\theta})$ in terms of $\boldsymbol{x}$. When the activation function of the network is differentiable, the network is differentiable and the derivative in terms of $\boldsymbol{x}$ can be evaluated efficiently via the backpropagation algorithm. Note that the network we adopt in this paper is not differentiable. Hence, numerical differentiation will be utilized to estimate the derivative of networks. For example, for the elliptic operator $\mathcal{D}u := \nabla\cdot(a(x)\nabla u)$, $\mathcal{D}u(\boldsymbol{x};\boldsymbol{\theta})$ can be estimated by the second-order central difference formula

$$\begin{aligned}\mathcal{D}u(\boldsymbol{x};\boldsymbol{\theta}) \approx \frac{1}{h^2}\sum_{i=1}^{d} & a(\boldsymbol{x} + \frac{1}{2}h\boldsymbol{e}_i)(u(\boldsymbol{x} + h\boldsymbol{e}_i,\boldsymbol{\theta}) - u(\boldsymbol{x};\boldsymbol{\theta})) \\ & - a(\boldsymbol{x} - \frac{1}{2}h\boldsymbol{e}_i)(u(\boldsymbol{x};\boldsymbol{\theta}) - u(\boldsymbol{x} - h\boldsymbol{e}_i,\boldsymbol{\theta})),\end{aligned} \tag{4.9}$$

up to an error of $O(dh^2)$.

4.4. Network Training. Once networks have been set up, the rest is to train the networks to solve the min-max problem in (3.5). The stochastic gradient descent (SGD) method or its variants (e.g., Adam [23]) is an efficient tool to solve this problem numerically. Although the convergence of SGD for the min-max problem is still an active research topic [39, 8, 43], empirical success shows that SGD can provide a good approximate solution.

Before completing the algorithm description of SelectNet, let us introduce the key setup of SGD and summarize it in Algorithm 1 below. In each training iteration, we first set uniformly distributed training points $\{\boldsymbol{x}_n^1\}_{n=1}^{N_1} \subset Q$ and $\{\boldsymbol{x}_i^2\}_{n=1}^{N_2} \subset \Gamma$, and define the empirical loss of these training points as

$$
\begin{aligned}
J(\boldsymbol{\theta}_s, \boldsymbol{\theta}) = & \frac{1}{N_1}\sum_{n=1}^{N_1}\phi(\boldsymbol{x}_n^1;\boldsymbol{\theta}_s)|\mathcal{D}u(\boldsymbol{x}_n^1,\boldsymbol{\theta}) - f(\boldsymbol{x}_n^1)|^2 \\
& + \frac{\lambda}{N_2}\sum_{n=1}^{N_2}\phi(\boldsymbol{x}_n^2;\boldsymbol{\theta}_s)|\mathcal{B}u(\boldsymbol{x}_n^2,\boldsymbol{\theta}) - g(\boldsymbol{x}_n^2)|^2 \\
& - \varepsilon^{-1}\left[\left(\frac{1}{N_1}\sum_{n=1}^{N_1}\phi(\boldsymbol{x}_n^1;\boldsymbol{\theta}_s) - 1\right)^2 + \left(\frac{1}{N_2}\sum_{n=1}^{N_2}\phi(\boldsymbol{x}_n^2;\boldsymbol{\theta}_s) - 1\right)^2\right].
\end{aligned} \tag{4.10}
$$

Next, $\boldsymbol{\theta}_s$ can be updated by the gradient descent via

$$
\boldsymbol{\theta}_s \leftarrow \boldsymbol{\theta}_s + \tau_s \nabla_{\boldsymbol{\theta}_s} J, \tag{4.11}
$$

and $\boldsymbol{\theta}$ can be updated by the gradient descent via

$$
\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \tau \nabla_{\boldsymbol{\theta}} J, \tag{4.12}
$$

with decent step sizes τ_s and τ. Note that training points are randomly renewed in each iteration. In fact, for the same set of training points in each iteration, the updates (4.11) and (4.12) can be performed n_1 and n_2 times, respectively.

Algorithm 1 Residual Method with SelectNet

Require: the PDE (2.4)
Ensure: the parameters $\boldsymbol{\theta}$ in the solution network $u(\boldsymbol{x};\boldsymbol{\theta})$
  Set iteration parameters n, n_1, n_2 and parameters N_1, N_2 for sample sizes
  Initialize $u(\boldsymbol{x};\boldsymbol{\theta}^{0,0})$ and $\phi(\boldsymbol{x};\boldsymbol{\theta}_s^{0,0})$
  for $k = 1, \cdots, n$ **do**
    Generate uniformly distributed sampling points
    $\{\boldsymbol{x}_i^1\}_{i=1}^{N_1} \subset Q$ and $\{\boldsymbol{x}_i^2\}_{i=1}^{N_2} \subset \Gamma$
    for $j = 1, \cdots, n_1$ **do**
      Update $\boldsymbol{\theta}_s^{k-1,j} \leftarrow \boldsymbol{\theta}_s^{k-1,j-1} + \tau_s^{(k)} \nabla_{\boldsymbol{\theta}_s} J(\boldsymbol{\theta}_s^{k-1,j-1}, \boldsymbol{\theta}^{k-1,0})$
    end for
    $\boldsymbol{\theta}_s^{k,0} \leftarrow \boldsymbol{\theta}_s^{k-1,n_1}$
    for $j = 1, \cdots, n_2$ **do**
      Update $\boldsymbol{\theta}^{k-1,j} \leftarrow \boldsymbol{\theta}^{k-1,j-1} - \tau^{(k)} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_s^{k,0}, \boldsymbol{\theta}^{k-1,j-1})$
    end for
    $\boldsymbol{\theta}^{k,0} \leftarrow \boldsymbol{\theta}^{k-1,n_2}$
    if Stopping criteria is satisfied **then**
      Return $\boldsymbol{\theta} = \boldsymbol{\theta}^{k,0}$
    end if
  end for

d	the dimension of the problem
m	the width of each layer in the solution network
m_s	the width of each layer in the selection network
L	the depth of the solution network
L_s	the depth of the selection network
M_0	the upper bound of the selection network
m_0	the lower bound of the selection network
N_k	number of iterations in the optimization
N_i	number of updates of the selection network in each iteration
N_j	number of updates of the solution network in each iteration
N_1	number of training points inside the domain in each iteration
N_2	number of training points on the domain boundary in each iteration
ε	penalty parameter to uniform the selection network
λ	summation weight of the boundary residual

Table 5.1: *Parameters in the model and algorithm.*

5. Numerical Examples. In this section, the proposed SelectNet method is tested on several PDE examples including elliptic/parabolic and linear/nonlinear high-dimensional problems. The traditional basic method (2.7) and the recently introduced weak adversarial network (WAN) [49] are also tested for comparison. For the basic and SelectNet models, we choose the feedforward architecture with sin activation for the solution network, and the feedforward architecture with Relu activation for the selection network. AdamGrad [10] is employed to solve the min-max optimization (3.5), with the learning rates

$$\tau_s^{(k)} = 0.01, \tag{5.1}$$

for the selection network, and

$$\tau^{(k)} = 10^{p_0 - j(p_0-p_1)/J}, \text{ if } N_k^{(j)} < k \le N_k^{(j+1)}, \quad \forall j = 0, \cdots, J, \tag{5.2}$$

for the solution network, where $0 = N_k^0 < N_k^1 < \cdots < N_k^{J+1} = N_k$ are equispaced segments of total iterations, and p_0, p_1 are the first and last powers of learning rates. Other parameters used in the model and algorithm are listed in Table 5.1. In all examples, the feedforward network with $m = 100$, $L = 7$, $L_s = 3$ are used for the basic and SelectNet models.

We take the (relative) discrete L^2 error at random sampling points as the metric to evaluate the accuracy of the method, namely,

$$e_2(\boldsymbol{\theta}) := \sqrt{\frac{\sum_i |u(\boldsymbol{x}_i;\boldsymbol{\theta}) - u(\boldsymbol{x}_i)|^2}{\sum_i |u(\boldsymbol{x}_i)|^2}}, \tag{5.3}$$

where $\{\boldsymbol{x}_i\}$ are uniformly distributed testing points in Q.

5.1. Poisson Equations with Low-Regularity Solutions. First, let us consider the Poisson equation inside a bounded domain

$$\begin{aligned} \Delta u &= f, \quad \text{in } \Omega, \\ u &= g, \quad \text{on } \partial\Omega. \end{aligned} \tag{5.4}$$

parameters	dimensions	SelectNet	Basic	WAN
N_1	$d = 5, 10, 15, 20$	10000	10000	10000
N_2	$d = 5, 10, 15, 20$	10000	10000	10000
N_k	$d = 5, 10$	200	200	10000
	$d = 15, 20$	400	400	10000
N_j	$d = 5, 10$	100	100	2
	$d = 15, 20$	200	200	2
λ	$d = 5, 10, 15, 20$	10	10	10
m_s	$d = 5, 10$	10	/	/
	$d = 15, 20$	20	/	/
N_i	$d = 5, 10, 15, 20$	100	/	/
ε	$d = 5, 10, 15, 20$	0.001	/	/
$[m_0, M_0]$	$d = 5$	$[0.5, 5]$	/	/
	$d = 10, 15, 20$	$[0.1, 10]$	/	/

Table 5.2: The parameters of the implementation for various methods in the first Poisson example.

Two cases of solutions are taken for a test in this example. In the first case, we specify the exact solution

$$u(\boldsymbol{x}) = \sin(\frac{\pi}{2}(1 - |\boldsymbol{x}|)^{2.5}) \tag{5.5}$$

in $\Omega = \{\boldsymbol{x} : |\boldsymbol{x}| < 1\}$, whose first derivative is singular at the origin and the third derivative is singular on the boundary. The problems of $d = 5$, 10, 15, 20 are solved. The approximate solutions are searched for by learning rates decaying from 10^{-3} to 10^{-4}. The algorithm of WAN is also implemented by the same ADAM optimizer for comparison. The common implementation parameters and the minimal relative L^2 errors obtained during the iterations by these methods are listed in Table 5.2 and 5.3. Also, the curves of error decay versus iterations(%) are shown in Fig. 5.1. From these results, it is observed the three methods perform quite differently within the same number of updates ($O(10^4)$). Due to the low regularity of the exact solution, the approximate solution of WAN does not converge (N.C.) for all dimensions. The basic model works for $d = 5$, 10 but does not produce convergent solutions as well. SelectNet has a similar performance to the basic model for $d = 5$ (having error 5.72×10^{-3} compared to 3.99×10^{-3}), and obtains a more accurate solution for $d = 10$ (having error 2.27×10^{-3} compared to 3.66×10^{-2}). Furthermore, for $d = 15$ and 20, SelectNet still obtains solutions of low errors within $O(10^{-2})$, which implies its robustness for low-regularity problems of higher dimensions. We also present in Fig. 5.2 the following surfaces on (x_1, x_2)-plane

- the selection network $\phi(x_1, x_2, 0, \cdots, 0; \boldsymbol{\theta})$
- the solution error $u(x_1, x_2, 0, \cdots, 0; \boldsymbol{\theta}) - u(x_1, x_2, 0, \cdots, 0)$

for $d = 20$. It shows the numerical error accumulates at the origin where the method converges slowly due to the low regularity. On the other hand the selection net attains its peak at the origin, implying the selection of training points is mainly distributed near where the error is the largest.

In the second case, we specify the domain $\Omega = [-1, 1]^d$ and the exact solution $u(\boldsymbol{x}) = \tilde{u}(x_1, x_2) + x_3^2 - x_4^2 + x_5 + \cdots + x_d$, with

$$\tilde{u}(x_1, x_2) = \frac{64}{\pi^4} \sum_{\substack{n,m=1 \\ n,m \text{ odd}}}^{\infty} (-1)^{\frac{n+m}{2}} \frac{\cos(\frac{n\pi x}{2}) \cos(\frac{m\pi y}{2})}{nm(n^2 + m^2)}, \tag{5.6}$$

dimension	SelectNet	Basic	WAN
$d = 5$	5.72×10^{-3}	3.99×10^{-3}	0.800 (N.C.)
$d = 10$	2.27×10^{-3}	3.66×10^{-2}	1.128 (N.C.)
$d = 15$	3.26×10^{-3}	7.98×10^{-1} (N.C.)	2.154 (N.C.)
$d = 20$	6.76×10^{-2}	7.18×10^{-1} (N.C.)	2.634 (N.C.)

Table 5.3: The minimal relative L^2 errors obtained by various methods in the first Poisson example.

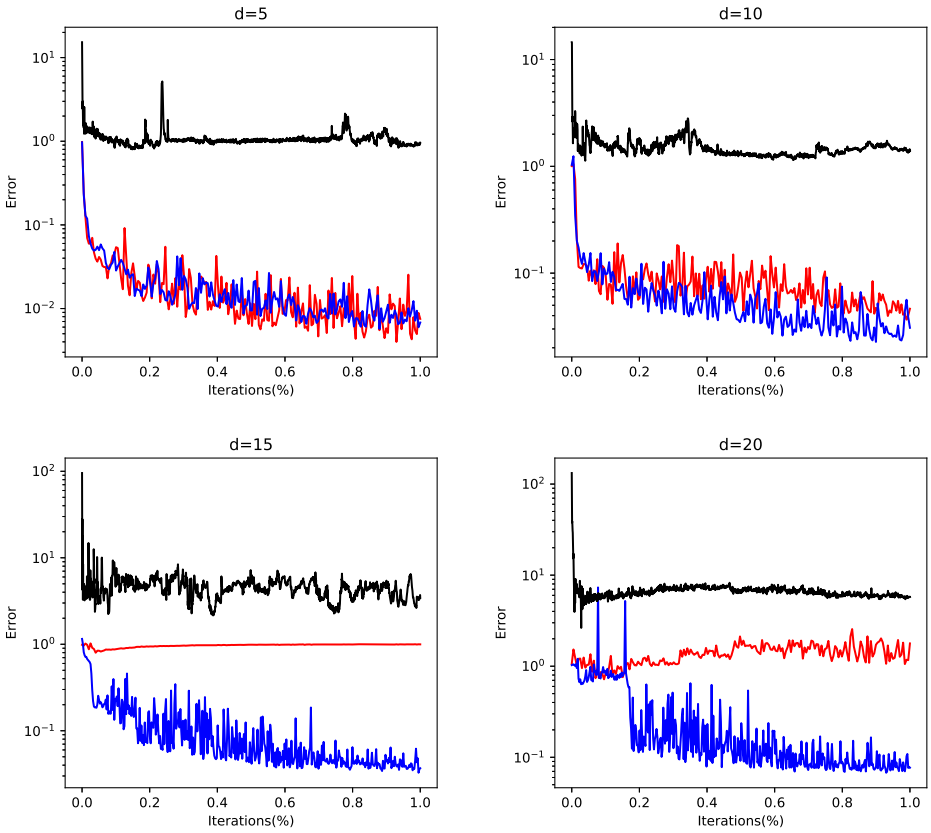

Fig. 5.1: *Relative L^2 errors v.s. iterations(%) in the first Poisson example (Blue: SelecNet model; Red: the basic model; Black: WAN).*

that leads to $f = 1$. Note $\tilde{u}(x_1, x_2)$ is singular at the four corners of $[-1, 1]^2$, which causes u has low regularity for x_1 and x_2. Similar to the preceding case, we solve the Poisson problem for $d = 5$, 10, 15, 20. The learning rates are set to decay from 10^{-2} to 10^{-4}. Parameters and minimal obtained errors by SelectNet model, the basic model and WAN are listed in Table 5.4 and 5.5. In Fig. 5.3 the error curves are shown. It can be seen from the results the basic model and WAN only converge to the exact solution for $d = 5$, obtaining relative L^2 errors 8.91×10^{-3} and 2.96×10^{-2}. For higher dimensions ($d = 10$, 15, 20), the error curves of the basic model fall down in the very beginning then turn back up and stay at high levels, while the curves of WAN stay at high levels and never descend, which shows neither converges to the exact solution. However, the error curves of SelectNet are essentially decreasing during the iterations, finally obtaining small errors of $O(10^{-3}) \sim O(10^{-2})$,

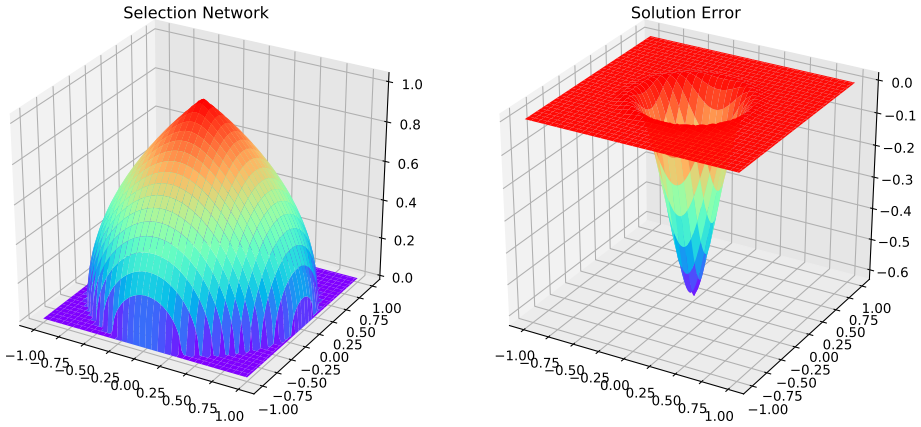

Fig. 5.2: *The (x_1, x_2)-surfaces of the selection network and the solution error by SelectNet (d=20) in the first Poisson example.*

parameters	dimensions	SelectNet	Basic	WAN
N_1	$d = 5, 10, 15$	1000	1000	1000
	$d = 20$	10000	10000	10000
N_2	$d = 5, 10, 15$	1000	1000	1000
	$d = 20$	10000	10000	10000
N_k	$d = 5, 10, 15$	100	100	10000
	$d = 20$	200	200	10000
N_j	$d = 5, 10, 15, 20$	100	100	2
λ	$d = 5, 10, 15, 20$	10	10	10
m_s	$d = 5, 10, 15, 20$	10	/	/
N_i	$d = 5, 10, 15, 20$	100	/	/
ε	$d = 5, 10, 15, 20$	0.001	/	/
$[m_0, M_0]$	$d = 5, 10, 15$	$[0.5, 5]$	/	/
	$d = 20$	$[0.1, 10]$	/	/

Table 5.4: The parameters of the implementation for various methods in the second Poisson example.

that demonstrate the effectiveness and robustness of SelectNet for higher dimensions. Moreover, the surfaces of solution errors and selection networks on (x_1, x_2)-plane for $d = 20$ are presented in Fig. 5.4, from which it can be seen the absolute value of error increases from the central origin to the four corners (from 0.025 to 0.175). Accordingly, the selection net attains its maximum near one corner, implying in the current iteration the selection is focusing on this corner. During the process of algorithm, We find the selection networks are trained to focus on a random corner of the four by the stochastic optimizer in each iteration.

5.2. Nonlinear Elliptic Equations. In this section, we test the effectiveness of SelectNet model on nonlinear elliptic equations with non-constant coefficients. We solve the following problem

dimension	SelectNet	Basic	WAN
$d = 5$	7.85×10^{-3}	8.91×10^{-3}	2.96×10^{-2}
$d = 10$	1.94×10^{-2}	7.90×10^{-2} (N.C.)	4.39×10^{-1} (N.C.)
$d = 15$	6.26×10^{-2}	2.36×10^{-1} (N.C.)	9.76×10^{-1} (N.C.)
$d = 20$	7.54×10^{-2}	1.91×10^{-1} (N.C.)	9.79×10^{-1} (N.C.)

Table 5.5: The minimal relative L^2 errors obtained by various methods in the second Poisson example.

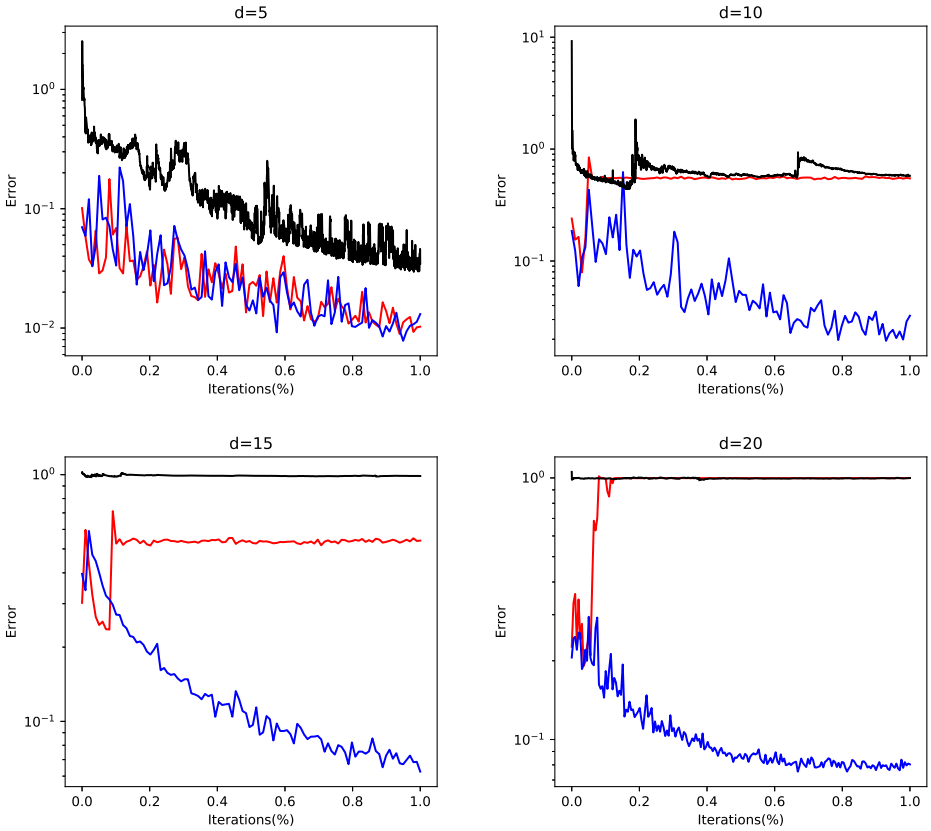

Fig. 5.3: *Relative L^2 errors v.s. iterations(%) in the second Poisson example (Blue: SelecNet model; Red: the basic model; Black: WAN).*

for $d = 10$ and 20 by the basic model, SelectNet model and WAN,

$$\begin{aligned} -\nabla \cdot (a(x)\nabla u) + |\nabla u|^2 &= f(x), \quad \text{in } \Omega := \{\boldsymbol{x} : |\boldsymbol{x}| < 1\}, \\ u &= g(x), \quad \text{on } \partial\Omega, \end{aligned} \tag{5.7}$$

with $a(x) = 1 + \frac{1}{2}|\boldsymbol{x}|^2$. The exact solution is set to be (5.5). As in the preceding section, the parameters, resulting errors, error surface and selection network surface are presented in Table 5.6-5.7 and Fig. 5.5-5.6. The results are similar to the preceding cases, from which we can observe SelectNet model converges more rapidly than the basic model for $d = 10$, and it keeps converging for $d = 20$ that neither of the basic model and WAN does. The final errors obtained by SelectNet

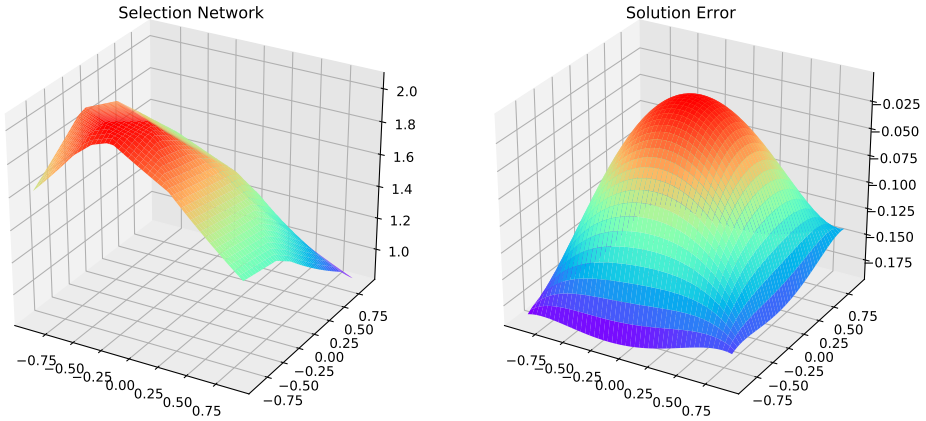

Fig. 5.4: *The (x_1, x_2)-surfaces of the selection network and the solution error by SelectNet (d=20) in the second Poisson example.*

parameters	dimensions	SelectNet	Basic	WAN
N_1	$d = 10, 20$	10000	10000	10000
N_2	$d = 10, 20$	10000	10000	10000
N_k	$d = 10$	200	200	10000
	$d = 20$	400	400	40000
N_j	$d = 10$	100	100	2
	$d = 20$	200	200	2
λ	$d = 10, 20$	10	10	10
m_s	$d = 10, 20$	10	/	/
N_i	$d = 10, 20$	100	/	/
ε	$d = 10, 20$	0.001	/	/
$[m_0, M_0]$	$d = 10, 20$	$[0.1, 10]$	/	/

Table 5.6: The parameters of the implementation for various methods in the nonlinear elliptic example.

is of $O(10^{-2})$.

5.3. Parabolic Equations. In this section, the SelectNet model is tested on an initial-boundary value problem of the parabolic (heat) equation, which is given by

$$\begin{aligned} \partial_t u(x,t) - \Delta u(x,t) &= f(x,t), \quad \text{in } Q := \Omega \times (0,1), \\ u(x,t) &= g(x), \quad \text{on } \partial\Omega \times (0,1), \\ u(x,0) &= h(x), \quad \text{in } \Omega, \end{aligned} \tag{5.8}$$

with $\Omega := [-1,1]^d$ and exact solution is set by $u(x,t) = (e^t - 1)\prod_{i=1}^{d} \cos(\pi x_i/2)$, which is analytic in the domain. In our method, time-discretization schemes are not utilized for time-dependent problems. Instead, we take t as an extra "spatial" variable in the SelectNet model. Hence the problem domain $\Omega \times (0,1)$ is an analog of a hypercylinder, and the "boundary" value is specified in the bottom $\Omega \times \{t = 0\}$ and the side $\partial\Omega \times (0,1)$. The learning rates are set to decay from 10^{-3}

dimension	SelectNet	Basic	WAN
$d = 10$	2.76×10^{-2}	5.32×10^{-2}	2.572 (N.C.)
$d = 20$	6.18×10^{-2}	9.75×10^{-1} (N.C.)	3.061 (N.C.)

Table 5.7: The minimal relative L^2 errors obtained by various methods in the nonlinear elliptic example.

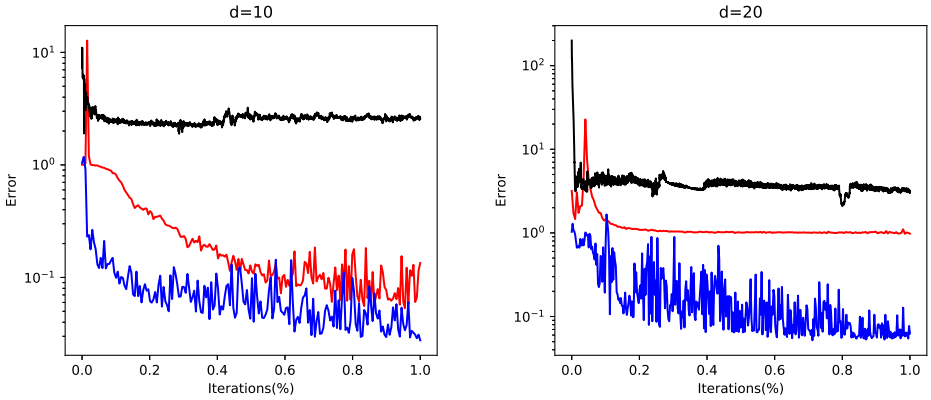

Fig. 5.5: *Relative L^2 errors v.s. iterations(%) in the nonlinear elliptic example (Blue: SelecNet model; Red: the basic model; Black: WAN)*

to 10^{-4}. Furthermore, due to $g = h = 0$, we build the network by

$$u(\boldsymbol{x}, t; \boldsymbol{\theta}) = t \prod_{i=1}^{d} (x_i^2 - 1)\tilde{u}(\boldsymbol{x}, t; \boldsymbol{\theta}) \tag{5.9}$$

where $\tilde{u}(\boldsymbol{x}, t; \boldsymbol{\theta})$ is a general feedforward neural network. Hence the (homogenous) initial and boundary conditions are always satisfied. As in the preceding sections, the parameters and computed results by the basic and SelectNet models for $d = 5$, 10, 15, 20 are listed in Table 5.8 and 5.9. It can be observed that even for analytic solutions, SelectNet can still obtain smaller errors than the basic model, especially for higher dimensions. Due to the specific architecture, the numerical solutions vanish at $t = 0$ and on $\partial\Omega$ where no error exists therefore. Instead, the solution error is mainly distributed near the origin. In Fig. 5.7 and 5.8 the (x_1, t)-surfaces of the numerical solutions and errors for $d = 20$ are displayed.

6. Conclusion. In this work, we improve the neural-based residual methods on generic PDEs by introducing a selection network for selected sampling in the optimization process. The objective is to place higher weights on the sampling points having larger point-wise residuals, and correspondingly we propose the SelectNet model that is a min-max optimization. In the implementation, both the solution and selection functions are approximated by feedforward neural networks, that are trained alternatively in the algorithm. The proposed SelectNet framework can solve high-dimensional PDEs that are intractable by traditional PDE solvers.

In the numerical examples, it is demonstrated the proposed SelectNet model works effectively for both elliptic and parabolic equations, even if in the case nonlinear equations. Furthermore, numerical results show that the proposed method outperforms the basic residual model and recently developed WAN when the target PDEs have low-regularity solutions. In such problems, the

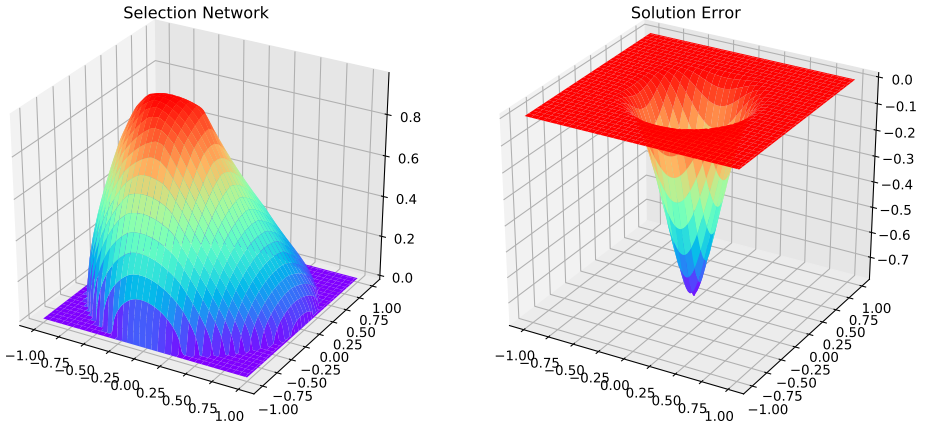

Fig. 5.6: *The (x_1, x_2)-surfaces of the selection network and the solution error by SelectNet (d=20) in the nonlinear elliptic example*

parameters	dimensions	SelectNet	Basic
N_1	$d = 5, 10, 15$	1000	1000
	$d = 20$	5000	5000
N_k	$d = 5, 10, 15, 20$	100	100
N_j	$d = 5, 10, 15, 20$	100	100
m_s	$d = 5, 10, 15, 20$	10	/
N_i	$d = 5, 10, 15, 20$	100	/
ε	$d = 5, 10, 15, 20$	0.001	/
$[m_0, M_0]$	$d = 5$	$[0.1, 10]$	/
	$d = 10, 15, 20$	$[0, 20]$	/

Table 5.8: The parameters of the implementation for various methods in the parabolic example

selection network always attains its maximum near one of the singularities that usually have larger point-wise residuals, hence improving the effect of convergence.

REFERENCES

[1] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, May 1993.

[2] Christian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. Deep splitting method for parabolic PDEs. *arXiv e-prints*, page arXiv:1907.03452, Jul 2019.

[3] Jens Berg and Kaj Nystrm. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28 – 41, 2018.

[4] Jian-Feng Cai, Dong Li, Jiaze Sun, and Ke Wang. Enhanced Expressive Power and Fast Training of Neural Networks by Random Projections. *arXiv e-prints*, page arXiv:1811.09054, Nov 2018.

[5] Charles K. Chui, Shao-Bo Lin, and Ding-Xuan Zhou. Construction of neural networks for realization of localized deep learning. *arXiv e-prints*, page arXiv:1803.03503, Mar 2018.

[6] Dominik Csiba and Peter Richtárik. Importance sampling for minibatches. *J. Mach. Learn. Res.*, 19(1):962–982, January 2018.

[7] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Feb 1989.

[8] Constantinos Daskalakis and Ioannis Panageas. The limit points of (optimistic) gradient descent in min-max op-

dimension	SelectNet	Basic
$d = 5$	2.98×10^{-3}	4.56×10^{-3}
$d = 10$	8.26×10^{-3}	8.84×10^{-3}
$d = 15$	1.51×10^{-2}	1.81×10^{-2}
$d = 20$	4.98×10^{-2}	9.58×10^{-2}

Table 5.9: The minimal relative L^2 errors obtained by various methods in the parabolic example

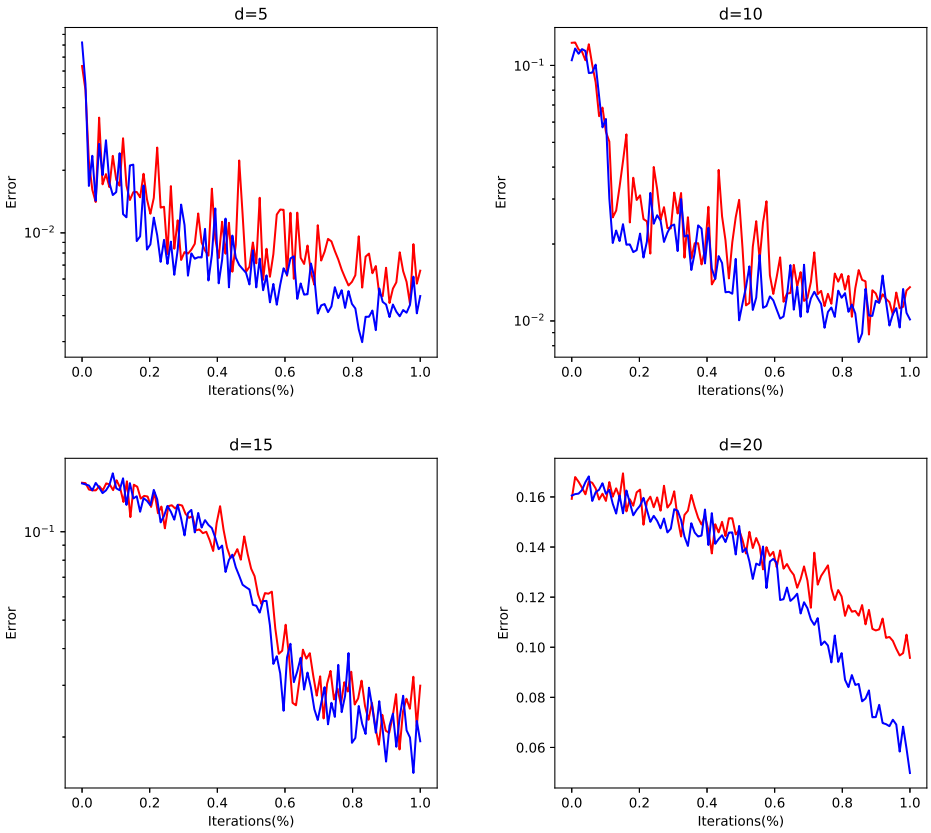

Fig. 5.7: *Relative L^2 errors v.s. iterations(%) in the parabolic example (Blue: SelecNet model; Red: the basic model.*

timization. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 9256–9266, USA, 2018. Curran Associates Inc.

[9] M. W. M. G. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

[10] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[11] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 11 2017.

[12] Weinan E, Chao Ma, and Lei Wu. Barron Spaces and the Compositional Function Spaces for Neural Network Models. *arXiv e-prints*, page arXiv:1906.08039, Jun 2019.

[13] Weinan E and Qingcan Wang. Exponential convergence of the deep neural network approximation for analytic functions. *CoRR*, abs/1807.00297, 2018.

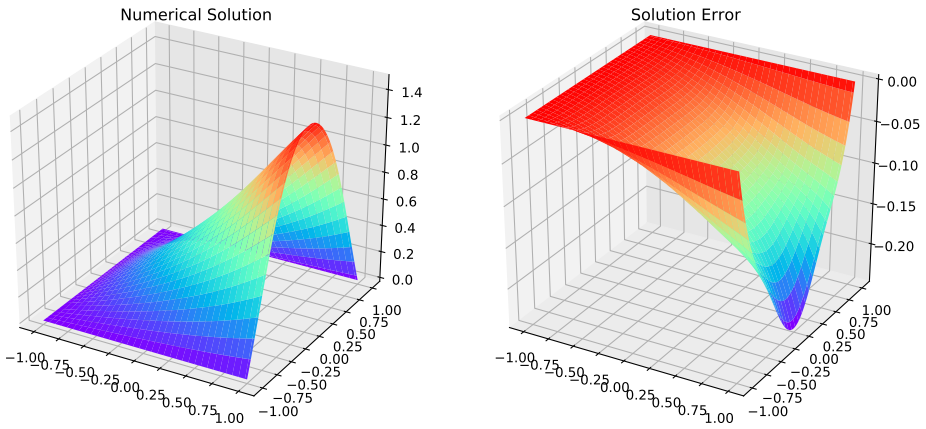

Fig. 5.8: *The (x_1, t)-surfaces of the numerical solution and the solution error by SelectNet (d=20) in the nonlinear elliptic example.*

[14] Matthias Ehrhardt and Ronald E. Mickens. A fast, stable and accurate numerical method for the blackscholes equation of american options. *International Journal of Theoretical and Applied Finance*, 11(05):471–501, 2008.

[15] Abhijeet Gaikwad and Ioane Muni Toke. Gpu based sparse grid technique for solving multidimensional options pricing pdes. In *Proceedings of the 2Nd Workshop on High Performance Computational Finance*, WHPCF '09, pages 6:1–6:9, New York, NY, USA, 2009. ACM.

[16] D. Gobovic and M. E. Zaghloul. Analog cellular neural network with application to partial differential equations with variable mesh-size. In *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*, volume 6, pages 359–362 vol.6, May 1994.

[17] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[18] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[19] Martin Hutzenthaler, Arnulf Jentzen, Thomas Kruse, and Tuan Anh Nguyen. A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations. *arXiv e-prints*, page arXiv:1901.10854, Jan 2019.

[20] Martin Hutzenthaler, Arnulf Jentzen, Thomas Kruse, Tuan Anh Nguyen, and Philippe von Wurstemberger. Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations. *arXiv e-prints*, page arXiv:1807.01212, Jul 2018.

[21] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Regularizing very deep neural networks on corrupted labels. *ArXiv*, abs/1712.05055, 2017.

[22] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. *CoRR*, abs/1803.00942, 2018.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, 2014.

[24] M. P. Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1189–1197. Curran Associates, Inc., 2010.

[25] M. P. Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1189–1197. Curran Associates, Inc., 2010.

[26] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, Sep. 1998.

[27] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, Sep. 2000.

[28] Hyuk Lee and In Seok Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110 – 131, 1990.

[29] T.T. Lee, F.Y. Wang, and R.B. Newell. Robust model-order reduction of complex biological processes. *Journal*

of Process Control, 12(7):807 – 821, 2002.

[30] Ke Li, Kejun Tang, Tianfan Wu, and Qifeng Liao. D3M: A deep domain decomposition method for partial differential equations. *arXiv e-prints*, page arXiv:1909.12236, Sep 2019.

[31] Shiyu Liang and R. Srikant. Why deep neural networks? *CoRR*, abs/1610.04161, 2016.

[32] Yunru Liu, Tingran Gao, and Haizhao Yang. SelectNet: Learning to Sample from the Wild for Imbalanced Data Training. *arXiv e-prints*, page arXiv:1905.09872, May 2019.

[33] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6231–6239. Curran Associates, Inc., 2017.

[34] A. Malek and R. Shekari Beidokhti. Numerical solution for high order differential equations using a hybrid neural network-optimization method. *Applied Mathematics and Computation*, 183(1):260 – 271, 2006.

[35] Hadrien Montanelli and Qiang Du. New error bounds for deep relu networks using sparse grids. *SIAM Journal on Mathematics of Data Science*, 1(1), Jan 2019.

[36] Hadrien Montanelli and Haizhao Yang. Error bounds for deep ReLU networks using the Kolmogorov–Arnold superposition theorem. *arXiv e-prints*, page arXiv:1906.11945, Jun 2019.

[37] Hadrien Montanelli, Haizhao Yang, and Qiang Du. Deep ReLU networks overcome the curse of dimensionality for bandlimited functions. *arXiv e-prints*, page arXiv:1903.00735, Mar 2019.

[38] Philipp Petersen and Felix Voigtlaender. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*, 108:296 – 330, 2018.

[39] Hassan Rafique, Mingrui Liu, Qihang Lin, and Tianbao Yang. Non-convex min-max optimization: Provable algorithms and applications in machine learning. *ArXiv*, abs/1810.02060, 2018.

[40] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep Network Approximation Characterized by Number of Neurons. *arXiv e-prints*, page arXiv:1906.05497, Jun 2019.

[41] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Nonlinear approximation via compositions. *Neural Networks*, 119:74 – 84, 2019.

[42] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339 – 1364, 2018.

[43] Christopher Srinivasa, Inmar Givoni, Siamak Ravanbakhsh, and Brendan J Frey. Min-max propagation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5565–5573. Curran Associates, Inc., 2017.

[44] Taiji Suzuki. Adaptivity of deep reLU network for learning in besov and mixed smooth besov spaces: optimal rate and curse of dimensionality. In *International Conference on Learning Representations*, 2019.

[45] David J. Wales and Jonathan P. K. Doye. Stationary points and dynamics in high-dimensional systems. *The Journal of Chemical Physics*, 119(23):12409–12416, 2003.

[46] Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103 – 114, 2017.

[47] Dmitry Yarotsky. Optimal approximation of continuous functions by very deep ReLU networks. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 639–649. PMLR, 06–09 Jul 2018.

[48] H. Yserentant. Sparse grid spaces for the numerical solution of the electronic schrödinger equation. *Numerische Mathematik*, 101(2):381–389, Aug 2005.

[49] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak Adversarial Networks for High-dimensional Partial Differential Equations. *arXiv e-prints*, page arXiv:1907.08272, Jul 2019.