SOLVING PDES ON UNKNOWN MANIFOLDS WITH MACHINE LEARNING

A PREPRINT

Senwei Liang
Department of Mathematics, Purdue University, IN 47907, USA
liang339@purdue.edu

Shixiao W. Jiang
Institute of Mathematical Sciences, ShanghaiTech University, Shanghai, 201210, China
jiangshx@shanghaitech.edu.cn

John Harlim
Department of Mathematics, Department of Meteorology and Atmospheric Science,
Institute for Computational and Data Sciences
The Pennsylvania State University, University Park, PA 16802, USA
jharlim@psu.edu

Haizhao Yang
Department of Mathematics, Purdue University, IN 47907, USA
haizhao@purdue.edu

June 12, 2021

ABSTRACT

This paper proposes a mesh-free computational framework and machine learning theory for solving elliptic PDEs on unknown manifolds, identified with point clouds, based on diffusion maps (DM) and deep learning. The PDE solver is formulated as a supervised learning task to solve a least-squares regression problem that imposes an algebraic equation approximating a PDE (and boundary conditions if applicable). This algebraic equation involves a graph-Laplacian type matrix obtained via DM asymptotic expansion, which is a consistent estimator of second-order elliptic differential operators. The resulting numerical method is to solve a highly non-convex empirical risk minimization problem subjected to a solution from a hypothesis space of neural-network type functions. In a well-posed elliptic PDE setting, when the hypothesis space consists of feedforward neural networks with either infinite width or depth, we show that the global minimizer of the empirical loss function is a consistent solution in the limit of large training data. When the hypothesis space is a two-layer neural network, we show that for a sufficiently large width, the gradient descent method can identify a global minimizer of the empirical loss function. Supporting numerical examples demonstrate the convergence of the solutions and the effectiveness of the proposed solver in avoiding numerical issues that hampers the traditional approach when a large data set becomes available, e.g., large matrix inversion.

Keywords High-Dimensional PDEs · Diffusion Maps · Deep Neural Networks · Convergence Analysis · Least-Squares Minimization · Manifolds · Point Clouds.

1 Introduction

Solving PDEs on unknown manifolds is a challenging computational problem that commands a wide variety of applications. In physics and biology, such a problem arises in modeling of granular flow [60], liquid crystal [70], biomembranes [23]. In computer graphics [9], PDEs on surfaces have been used to restore damaged patterns on a surface [51], brain imaging [53], among other applications. By unknown manifolds, we refer to the situation where the parameterization of the domain is unknown. The main computational challenge arising from this constraint is on the approximation of the differential operator using the available sample data (point clouds) that are assumed to lie on (or close to) a smooth manifold. Among many available methods proposed for PDEs on surfaces embedded in $\mathbb{R}^3$, they typically parameterize the surface and subsequently use it to approximate the tangential derivatives along surfaces. For example, the finite element method represents surfaces [18, 11, 10] using triangular meshes. Thus its accuracy relies on the quality of the generated meshes which may be poor if the given point cloud data are randomly distributed. Another class of approach is to estimate the embedding function of the surface using e.g., level set representation [9] or closest point representation [61], and subsequently solve the embedded PDE on the ambient space. The key issue with this class of approaches is that since the embedded PDE is at least one dimension higher than the dimension of the two-dimensional surface (i.e., co-dimension higher than one), the computational cost may not be feasible if the manifold is embedded in high-dimensional ambient space. Another class of approaches is the mesh-free radial basis function (RBF) method [58, 25] for solving PDE on surfaces. This approach, however, may not be robust in high co-dimensional problems as pointed out in [25] and the convergence near the boundary can be problematic.

Recently in [26], an unsupervised learning method called the *Diffusion Map* (DM) algorithm [12] was proposed to directly approximate the second-order elliptic differential operator on point clouds data that lie on the manifolds. The proposed DM-based solver has been extended to elliptic problems with various types of boundary conditions that typically arise in applications [39], such as non-homogeneous Dirichlet, Neumann, and Robin types and to time-dependent advection-diffusion PDEs [71]. The main advantage of this approach is to avoid the tedious parameterization of sub-manifolds in a high-dimensional space. However, the estimated solution is represented by a discrete vector whose components approximate the function values on the available point clouds, analogous to standard finite-difference methods. With such a representation, one will need an interpolation method to find the solutions on new data points, which is a nontrivial task when the domain is an unknown manifold. This issue is particularly relevant if the available data points come sequentially. Another problem with the DM-based solver is that the size of the matrix approximating the differential operator increases as a function of the data size, which creates a computational bottleneck when the PDE solution involves a (pseudo) inversion operation or an eigenvalue decomposition.

One way to overcome these computational issues is to solve the PDE in a supervised learning framework using neural networks (NNs). On a Euclidean domain, where extensive research has been conducted [19, 29, 40, 69, 6, 76, 44, 5, 59, 24], NN-based PDE solvers reformulate the PDE problem as a regression problem where the governing PDE and boundary conditions are imposed. Subsequently, the PDE solution is approximated by a class of NN functions. An advantage of this mesh-free discretization method is that it has a good approximation properties [4, 20, 21, 56, 67, 55, 35, 36, 35, 37, 64, 65], which enable application of these PDE solvers to high-dimensional problems. With the advanced computational tools (e.g., TensorFlow and Pytorch) and the built-in optimization algorithms therein, developing mathematical software with parallel computing using neural networks is much simpler than conventional numerical techniques.

Building upon this encouraging result, we propose to solve PDEs on unknown manifolds by embedding the DM algorithm in NN-based PDE solvers. In particular, the DM algorithm is employed to approximate the second-order elliptic differential operator defined on the manifolds. Subsequently, a least-squares regression problem is formulated by imposing an algebraic equation that involves a graph-Laplacian type matrix, obtained by the discretization of the DM asymptotic expansion on the available point cloud training data. Numerically, we solve the resulting empirical loss function by finding the solution from a hypothesis space of neural-network type functions (e.g., with feedforward neural network, we consider the compositions of power of ReLU or polynomial-sine activation functions). Theoretically, we study the approximation and optimization aspects of the error analysis, induced by the training procedure that minimizes the empirical risk defined on available point cloud data. Under appropriate regularity assumptions, we show that when the hypothesis space has either an infinite width or depth, the global minimizer of the empirical loss function is a consistent solution in the limit of large training data. The corresponding error bound gives the relations between the desired accuracy and the required length of training data and width (or depth) of the network. Furthermore, when the hypothesis space is a two-layer neural network, we show that for a sufficiently large width, the gradient descent method can identify a global minimizer of the empirical loss function.

Numerically, we verify the proposed methods on several test problems on simple 2D and 3D manifolds with various co-dimensions and boundaries.

The paper will be organized as follows. In Section 2, we give a brief review on the DM-based PDE solver on closed manifolds and an overview of the ghost point diffusion maps (GPDM) for manifolds with boundaries. The proposed PDE solver is introduced in Section 3. The theoretical foundation of the proposed method is presented in Section 4. The numerical performance of the proposed method is illustrated in Section 5. We conclude the paper with a summary in Section 6. For reader's convenience, we present an algorithmic perspective for GPDM in Appendix A. We also include the longer proof for the optimization aspect of the algorithm in Appendix B and report all parameters used to generate the numerical results in Appendix C.

2 DM-based PDE Solver on Unknown Manifolds

To illustrate the main idea, let us discuss an elliptic problem defined on a d-dimensional closed sub-manifold $M \subseteq [0,1]^n \subseteq \mathbb{R}^n$. We will provide a brief discussion for the problem defined on compact manifolds with boundary to end this section. Let $u : M \to \mathbb{R}$ be a solution of the elliptic PDE,

$$(-a(\mathbf{x}) + \mathscr{L})u(\mathbf{x}) := -a(\mathbf{x})u(\mathbf{x}) + \mathrm{div}_g\big(\kappa(\mathbf{x})\nabla_g u(\mathbf{x})\big) = f(\mathbf{x}), \quad \mathbf{x} \in M. \tag{1}$$

Here, we have used the notations div_g and ∇_g for the divergence and gradient operators, respectively, defined with respect to the Riemannian metric g inherited by M from the ambient space $\mathbb{R}^n$. The real-valued functions a and κ are strictly positive such that $(-a(\mathbf{x}) + \mathscr{L})$ is strictly negative definite. The problem is assumed to be well-posed for $f \in C^{1,\alpha}(M)$, for $0 < \alpha < 1/2$. For $a \in C^{1,\alpha}(M)$ and $\kappa \in C^{3,\alpha}(M)$, a unique classical solution $u \in C^{3,\alpha}(M)$ is guaranteed. Here, we raise the regularity by one-order of derivative (compared to reported results in the literature [31, 27]) for the following reason.

The key idea of the DM-based PDE solver rests on the following asymptotic expansion,

$$G_\epsilon u(\mathbf{x}) := \epsilon^{-d/2}\int_M h\Big(\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\epsilon}\Big)u(\mathbf{y})dV(\mathbf{y}) = u(\mathbf{x}) + \epsilon(\omega(\mathbf{x})u(\mathbf{x}) + \Delta_g u(\mathbf{x})) + \mathscr{O}(\epsilon^2), \tag{2}$$

where the second equality is valid for any $u \in C^3(M)$ and $\mathbf{x} \in M$ with a high probability. Here, the function $h : [0,\infty) \to (0,\infty)$ is defined as $h(s) = \frac{e^{-s/4}}{(4\pi)^{d/2}}$ such that, effectively for a fixed bandwidth parameter $\epsilon > 0$, G_ϵ is a local integral operator. In (2), $\|\cdot\|$ denotes the standard Euclidean norm for vectors in $\mathbb{R}^n$ and we will use the same notation for arbitrary finite-dimensional vector space. Based on the asymptotic expansion in (2), one can approximate the differential operator $\mathscr{L}$ as follows,

$$\mathscr{L}u(\mathbf{x}) = \frac{\sqrt{\kappa}(\mathbf{x})}{\epsilon}\big(G_\epsilon(u(\mathbf{x})\sqrt{\kappa(\mathbf{x})}) - u(\mathbf{x})G_\epsilon\sqrt{\kappa(\mathbf{x})}\big) + \mathscr{O}(\epsilon) := L_\epsilon u(\mathbf{x}) + \mathscr{O}(\epsilon). \tag{3}$$

In our setup, we assume that we are given a set of point cloud data $X := \{\mathbf{x}_i \in M\}_{i=1,\dots,N}$, independent and identically distributed (i.i.d.) according to π, with an empirical measure defined as, $\pi_N(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^N \delta_{\mathbf{x}_i}(\mathbf{x})$. We use the notation $L^2(\pi)$ to denote the space of square-integrable functions with respect to the measure π. Accordingly, we define $L^2(\pi_N)$ as the space of functions $u : X \to \mathbb{R}^n$, endowed with the inner-product and norm-squared defined as,

$$\langle u, u\rangle_{L^2(\pi_N)} = \|u\|^2_{L^2(\pi_N)} = \int_M u^2(x)d\pi_N(x) = \frac{1}{N}\sum_{i=1}^N u^2(\mathbf{x}_i). \tag{4}$$

Given point cloud data, we first approximate the sampling density, $q = d\pi/dV$ evaluated at $\mathbf{x}_i$, with $Q_i := \epsilon^{-d/2}N^{-1}\sum_{j=1}^N h\big(\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{4\epsilon}\big)$. Define $\mathbf{W} \in \mathbb{R}^{N\times N}$ with entries,

$$\mathbf{W}_{ij} := \epsilon^{-d/2-1}N^{-1}h\Big(\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\epsilon}\Big)\sqrt{\kappa(\mathbf{x}_i)\kappa(\mathbf{x}_j)}Q_j^{-1}.$$

Define also a diagonal matrix $\mathbf{D} \in \mathbb{R}^{N\times N}$ with diagonal entries, $\mathbf{D}_{ii} = \sum_{j=1}^N \mathbf{W}_{ij}$. Then, we approximate the integral operator in (3) with the following matrix $\mathbf{L}_\epsilon$:

$$\mathbf{L}_\epsilon = \mathbf{W} - \mathbf{D}, \tag{5}$$

similarly to a discrete unnormalized graph Laplacian matrix. Here, the matrix $\mathbf{L}_\epsilon$ is self-adjoint and semi negative-definite with respect to the inner-product in $\langle u, v\rangle_Q := \frac{1}{N}\sum_{i=1}^N u(\mathbf{x}_i)v(\mathbf{x}_i)Q_i^{-1}$ such that it admits a non-positive

spectrum, $0 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_N$ with eigenvectors orthonormal in $\langle \cdot, \cdot \rangle_Q$. Since the kernel function h decays exponentially, the k-nearest-neighbor algorithm is usually used to impose sparsity to the estimator $\mathbf{L}_\epsilon$. The DM-based PDE solver approximates the PDE solution $u(\mathbf{x}_i)$ using the i-th component of the vector $\mathbf{u}_\epsilon \in \mathbb{R}^N$ that satisfies the linear system

$$(-\mathbf{a} + \mathbf{L}_\epsilon)\mathbf{u}_\epsilon = \mathbf{f} \tag{6}$$

of size N. Here, the i-th diagonal component of the diagonal matrix $\mathbf{a} \in \mathbb{R}^{N\times N}$ and the i-th component of $\mathbf{f} \in \mathbb{R}^N$ are $a(\mathbf{x}_i)$ and $f(\mathbf{x}_i)$, respectively. In [26], this approach has been theoretically justified and numerically extended to approximate the non-symmetric, uniformly elliptic second-order differential operators associated to the generator of Itô diffusions with appropriate local kernel functions.

Beyond the no boundary case, the approximation in (3), unfortunately, will not produce an accurate approximation when $\mathbf{x}$ is sufficiently closed to the boundary. To overcome this issue, a modified DM algorithm, following the classical ghost points method to obtain a higher-order finite-difference approximation of Neumann problems (e.g. [43]) was proposed in [39]. The proposed method, which is called the *Ghost Point Diffusion Maps* (GPDM), appends the point clouds with a set of ghost points away from the boundary along the outward normal collar such that the resulting discrete estimator is consistent in the $L^2(\mu_{N-N_b})$-sense, averaged over $N - N_b$ interior points on the manifold M [71]. To simplify the discussion in the remainder of this paper, we will use the same notation $\mathbf{L}_\epsilon$ to denote the discrete estimate of $\mathscr{L}$, obtained either via the classical or the ghost points diffusion maps. In Appendix A, we provide a brief review on the GPDM for Dirichlet boundary value problems.

3 Solving PDEs on Unknown Manifolds using Diffusion Maps and Neural Networks

We now present a new hybrid algorithm based on DM and NNs.

Deep neural networks (DNNs). Mathematically, DNNs are highly nonlinear functions constructed by compositions of simple nonlinear functions. For simplicity, we consider the fully connected feed-forward neural network (FNN), which is the composition of L simple nonlinear functions as follows: $\phi(\mathbf{x};\boldsymbol{\theta}) := \mathbf{a}^\top \mathbf{h}_L \circ \mathbf{h}_{L-1} \circ \cdots \circ \mathbf{h}_1(\mathbf{x})$, where $\mathbf{h}_\ell(\mathbf{x}) = \sigma(\mathbf{W}_\ell \mathbf{x} + \mathbf{b}_\ell)$ with $\mathbf{W}_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$, $\mathbf{b}_\ell \in \mathbb{R}^{N_\ell}$ for $\ell = 1, \ldots, L$, $\mathbf{a} \in \mathbb{R}^{N_L}$, σ is a nonlinear activation function, e.g., a rectified linear unit (ReLU) $\sigma(x) = \max\{x, 0\}$. Each $\mathbf{h}_\ell$ is referred as a hidden layer, N_ℓ is the width of the ℓ-th layer, and L is called the depth of the FNN. In the above formulation, $\boldsymbol{\theta} := \{\mathbf{a}, \mathbf{W}_\ell, \mathbf{b}_\ell : 1 \leq \ell \leq L\}$ denotes the set of all parameters in ϕ. For simplicity, we focus on FNN with a uniform width m, i.e., $N_\ell = m$ for all $\ell \neq 0$, in this paper.

Supervised Learning. Supervised learning approximates an unknown target function $f : \mathbf{x} \in \Omega \to y \in \mathbb{R}$ from training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i$'s are usually assumed to be i.i.d samples from an underlying distribution π defined on a domain $\Omega \subseteq \mathbb{R}^n$, and $y_i = f(\mathbf{x}_i)$. Consider the square loss $\frac{1}{2}\ell(\mathbf{x}, y; \boldsymbol{\theta}) = \left|\phi(\mathbf{x};\boldsymbol{\theta}) - y\right|^2$ of a given DNN $\phi(\mathbf{x};\boldsymbol{\theta})$ that is used to approximate $f(\mathbf{x})$, the population risk (error) and empirical risk (error) functions are, respectively,

$$\mathscr{J}(\boldsymbol{\theta}) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim\pi}\left[\left|\phi(\mathbf{x};\boldsymbol{\theta}) - f(\mathbf{x})\right|^2\right], \quad \hat{\mathscr{J}}(\boldsymbol{\theta}) = \frac{1}{2N}\sum_{i=1}^N \left|\phi(\mathbf{x}_i;\boldsymbol{\theta}) - y_i\right|^2. \tag{7}$$

The optimal set $\hat{\boldsymbol{\theta}}$ is identified via

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \hat{\mathscr{J}}(\boldsymbol{\theta}), \tag{8}$$

and $\phi(\cdot;\hat{\boldsymbol{\theta}}) : \Omega \to \mathbb{R}$ is the learned DNN that approximates the unknown function f.

The NN-based PDE solver with DM. Solving a PDE can be transformed into a supervised learning problem. Physical laws, like PDEs and boundary conditions, are used to generate training data in a supervised learning problem to infer the solution of PDEs. In the case when the PDE is defined on a manifold, we propose to use DM as an approximation to the differential operator according to (3) to obtain a linear system in (6), apply NN to parametrize the PDE solution, and adopt a least-square framework to identify the NN that approximates the PDE solution. For example, to solve the PDE problem in (1) on a d-dimensional closed, smooth manifold M identified with points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset M$, we minimize the following empirical loss

$$\boldsymbol{\theta}_S = \arg\min_{\boldsymbol{\theta}} \mathscr{J}_{S,\epsilon}(\boldsymbol{\theta}) := \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\|(-\mathbf{a} + \mathbf{L}_\epsilon)\boldsymbol{\phi}_{\boldsymbol{\theta}} - \mathbf{f}\|^2_{L^2(\pi_N)}, \tag{9}$$

where $\mathbf{L}_\epsilon \in \mathbb{R}^{N\times N}$ denotes the DM estimator for the differential operator $\mathscr{L}$, $\mathbf{a}$ and $\mathbf{f}$ are defined in (6), $\boldsymbol{\phi}_{\boldsymbol{\theta}} \in \mathbb{R}^N$ with the i-th entry as $\phi(\mathbf{x}_i;\boldsymbol{\theta})$. When $\mathbf{a} = \mathbf{0}$, we add a regularization term $\frac{\gamma}{2}\|\boldsymbol{\phi}_{\boldsymbol{\theta}}\|^2_{L^2(\pi_N)}$ in the loss function (9) to guarantee

well-posedness, where $\gamma > 0$ is a regularization parameter. When stochastic gradient descent is used to minimize (9), a small subset of the given point clouds is randomly selected in each iteration. Computationally, this amounts to randomly choosing batches, consisting of a few rows of $(-\mathbf{a}+\mathbf{L}_\epsilon)$ and a few entries of $\mathbf{f}$, to approximate the empirical loss function.

In the case of Dirichlet problems with non-homogeneous boundary conditions, $u(\mathbf{x}) = g(\mathbf{x})$, $\forall \mathbf{x} \in M$, given boundary points $\{\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_{N_b}\} \subset X \cap \partial M$ as the last N_b points of X, a penalty term is added to (9) to enforce the boundary condition as follows:

$$\boldsymbol{\theta}_S = \arg\min_{\boldsymbol{\theta}} \mathscr{J}_{S,\epsilon}(\boldsymbol{\theta}) := \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\|(-\mathbf{a}+\mathbf{L}_\epsilon)\boldsymbol{\phi}_{\boldsymbol{\theta}} - \mathbf{f}\|^2_{L^2(\pi_{N-N_b})} + \frac{\lambda}{2}\|\boldsymbol{\phi}^b_{\boldsymbol{\theta}} - \mathbf{g}\|^2_{L^2(\pi_{N_b})}, \tag{10}$$

where $\mathbf{L}_\epsilon \in \mathbb{R}^{(N-N_b)\times N}$ denotes the GPDM estimator for the differential operator $\mathscr{L}$ defined for problem with boundary and $\lambda > 0$ is a hyper-parameter. The construction of the matrix $\mathbf{L}_\epsilon$ is discussed in Appendix A. Accordingly, letting $\intercal$ denotes the transpose operator, we have also defined a column vector $\boldsymbol{\phi}^b_{\boldsymbol{\theta}} = \big(\phi(\bar{\mathbf{x}}_1;\boldsymbol{\theta}), \ldots, \phi(\bar{\mathbf{x}}_{N_b};\boldsymbol{\theta})\big)^\intercal \in \mathbb{R}^{N_b}$, whose components are also elements of the column vector $\boldsymbol{\phi}_{\boldsymbol{\theta}} = \big(\phi(\mathbf{x}_1;\boldsymbol{\theta}), \ldots, \phi(\mathbf{x}_N;\boldsymbol{\theta})\big)^\intercal \in \mathbb{R}^N$; the column vector $\mathbf{f} = \big(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{N-N_b})\big)^\intercal \in \mathbb{R}^{N-N_b}$ with function values on the interior points; and the column vector $\mathbf{g} = \big(g(\bar{\mathbf{x}}_1), \ldots, g(\bar{\mathbf{x}}_{N_b})\big)^\intercal \in \mathbb{R}^{N_b}$ with function values on the boundary points. In the case of other kinds of boundary conditions, a corresponding boundary operator can be applied to $\boldsymbol{\phi}^b_{\boldsymbol{\theta}}$ to enforce the boundary condition.

4 Theoretical Foundation of the Proposed Algorithm

The classical machine learning theory concerns with characterizations of the approximation error, optimization error estimation, and generalization error analysis. For the proposed PDE solver, the approximation theory involves characterizing: 1) the error of the DM-based discrete approximation of the differential operator on manifolds, and 2) the error of neural networks for approximating the PDE solution. In the optimization algorithm, a numerical minimizer (denoted as $\boldsymbol{\theta}_N$) provided by a certain algorithm might not be a global minimizer of the empirical risk minimization in (9) and (10). Therefore, designing an efficient optimization algorithm such that the optimization error $|\mathscr{J}_{S,\epsilon}(\boldsymbol{\theta}_N) - \mathscr{J}_{S,\epsilon}(\boldsymbol{\theta}_S)| \approx 0$ is important. In the generalization analysis, the goal is to quantify the error defined as $\|u - \phi(\cdot, \boldsymbol{\theta}_S)\|_{L^2(\pi)}$ over the distribution $\mathbf{x} \sim \pi$ that is unknown.

In approximation theory, the error analysis of DM is relatively well-developed, while the error of NNs is still under active development. Recently, there are two kinds of directions have been proposed to characterize the approximation capacity of NNs. The first one characterizes the approximation error in terms of the total number of parameters in an NN [73, 74, 54, 22, 56, 75, 57, 28, 67]. The second one quantifies the approximation error in terms of NN width and depth [63, 48, 64, 66, 65, 72]. In real applications, the width and depth of NNs are the required hyper-parameters to decide for numerical implementation instead of the total number of parameters. Hence, we will develop the approximation theory for the proposed PDE solver adopting the second direction.

In optimization theory, for regression problems without regularization (e.g., the right equation of (7)), it has been shown in [38, 14, 52, 15, 49] that (stochastic) gradient descent algorithms can converge to a global minimizer of the empirical loss function under the assumption of over-parametrization (i.e., the number of parameters are much larger than the number of samples). However, existing results for regression problems cannot be applied to the minimization problem corresponding to PDE solvers, which is much more difficult due to differential operators and boundary operators. A preliminary attempt was conducted in [50], but the results in [50] cannot be directly applied to our minimization problem in (9) and (10). Below, we will develop a new analysis to show that the gradient descent method can identify a global minimizer of (9).

The generalization analysis aims at quantifying the convergence of the generalization error $\|u - \phi(\cdot;\boldsymbol{\theta}_S)\|_{L^2(\pi)}$, i.e., showing that a global minimizer of the empirical risk minimization can also keep the population risk small. Let $u(X) \in \mathbb{R}^N$ be a column vector representing the evaluation of u on the training data set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and this notation is used similarly for other functions. Beyond the identification of $\boldsymbol{\theta}_S$ (a global minimizer of the empirical loss), which is addressed in the optimization theory, a typical approach is to estimate the difference between $\|u - \phi(\cdot;\boldsymbol{\theta}_S)\|_{L^2(\pi)}$ and $\|u(X) - \phi(X;\boldsymbol{\theta}_S)\|_{L^2(\pi_N)}$ via statistical learning theories. There have been several papers for the generalization error analysis of PDE solvers, e.g., [7, 30, 1, 50, 47, 16, 34, 46]. In most existing generalization analysis for PDE solvers, it is assumed that a good minimizer of the empirical risk minimization satisfies a certain norm constraint so that this minimizer can generalize well, e.g., the minimizer corresponds to a neural network with a small Lipschitz constant or norm. However, it is still an open problem to design numerical optimization algorithms to identify this nice minimizer. Another direction is to regularize the empirical risk minimization so that a global minimizer of the regularized loss can generalize well [50]. Nevertheless, there is no global convergence

analysis of the optimization algorithm for the regularized loss, which suggests that there is no guarantee that one can practically obtain the global minimizer of the regularized loss that can generalize well.

The theoretical analysis in this paper focuses on the approximation and optimization perspectives of the proposed PDE solver to develop an error analysis of $\|u(X)-\phi(X;\boldsymbol{\theta}_S)\|_{L^2(\pi_N)}$. In the discussion below, we restrict to the case of manifolds without boundaries to convey our main ideas for the theoretical analysis of the proposed algorithm. We further assume that the ReLU activation function, i.e., $\max\{x,0\}$, its power, e.g., ReLU^r, and FNNs are used in the analysis. An extension to other activation functions and network architectures might be possible. In Section 4.1, we show that the numerical solution of the proposed solver is consistent with the ground truth in appropriate limits, assuming that a global minimizer of our optimization problem is obtainable. In Section 4.2, we show that the gradient descent method can identify a global minimizer of our optimization problem for two-layer neural networks when their width is sufficiently large. The optimization theory together with the parametrization error analysis forms the theoretical foundation of the convergence analysis of the proposed PDE solver on manifolds.

4.1 Parametrization Error

The proposed PDE solver on manifolds applies DM to parametrize differential operators on manifolds and uses an NN to parametrize the PDE solution. We are going to quantify the parametrization error due to these two ideas, assuming that a global minimizer of the empirical loss minimization in (9) is achievable via a certain numerical optimization algorithm, i.e., estimating $\|\mathbf{u}-\boldsymbol{\phi}_S\|_{L^2(\pi_N)}$, where $\boldsymbol{\phi}_S\in\mathbb{R}^N$ with the i-th entry as $\phi(\mathbf{x}_i;\boldsymbol{\theta}_S)$ is the NN-solution of the PDE in (9). Our goal is to show the following convergence result.

Theorem 4.1 (Parametrization Error)**.** *Assume u solves* (1) *with $0<a_{\min}\le a(x)\le a_{\max}$ on $X=\{\mathbf{x}_1,\dots,\mathbf{x}_N\}$, randomly sampled from a distribution π on $M\subset[0,1]^n$, where M is a C^4-manifold with condition number τ_M^{-1}, volume V_M, and geodesic covering regularity G_M. For $u,\kappa\in C^4(M)\cap L^2(M)$ and $q\in C(M)$, where $q=d\pi/dV$, with probability higher than $1-N^{-2}$,*

$$\|u(X)-\phi(X;\boldsymbol{\theta}_S)\|_{L^2(\pi_N)}=\mathcal{O}\big(\epsilon,N^{-\frac{1}{2}}\epsilon^{-2-\frac{d}{4}},N^{-\frac{1}{2}}\epsilon^{-\frac{1}{2}-\frac{d}{4}},N^{1/2}\epsilon^{-1}m^{-8/(d\ln(n))}L^{-8/(d\ln(n))}\big),\tag{11}$$

as $\epsilon\to0$, after $N\to\infty$ and m or $L\to\infty$. Hence,

$$\lim_{\epsilon\to0}\lim_{N\to\infty}\lim_{m\to\infty}\|u(X)-\phi(X;\boldsymbol{\theta}_S)\|_{L^2(\pi_N)}=0.\tag{12}$$

Here $\phi(\mathbf{x};\boldsymbol{\theta}_S)$ has a width $\mathcal{O}(n\ln(n)m\log(m))$ and a depth $\mathcal{O}(L\log(L)+\ln(n))$ with $m\in\mathbb{N}^+$ and $L\in\mathbb{N}^+$ as two interger hyper-parameters.

In (11) and throughout this paper, the big-oh notation means $\mathcal{O}(f,g):=\mathcal{O}(f)+\mathcal{O}(g)$, as $f,g\to0$. As we shall see in our discussion below, the first three error terms come from the DM discretization, whereas the last error term is due to the approximation property of NNs. In (11), for a given sufficiently small ϵ, there exists a sufficiently large N_0 such that $\mathcal{O}\big(\epsilon,N^{-\frac{1}{2}}\epsilon^{-2-\frac{d}{4}},N^{-\frac{1}{2}}\epsilon^{-\frac{1}{2}-\frac{d}{4}}\big)=\mathcal{O}(\epsilon)$ for $N\ge N_0$. For this ϵ and N, there exists a sufficiently large m_0 such that $\mathcal{O}\big(\epsilon,N^{-\frac{1}{2}}\epsilon^{-2-\frac{d}{4}},N^{-\frac{1}{2}}\epsilon^{-\frac{1}{2}-\frac{d}{4}},N^{1/2}\epsilon^{-1}m^{-8/(d\ln(n))}L^{-8/(d\ln(n))}\big)=\mathcal{O}(\epsilon)$ when $m\ge m_0$ for any L. For example, in the case of uniformly sampled data, the second error term, induced by estimation of the sampling density q, becomes irrelevant and the factor $N^{1/2}$ in the last error term disappears due to symmetry. In such a case, the number of data points needed to achieve order-ϵ of the DM discretization for $N\ge N_0=\mathcal{O}(\epsilon^{-\frac{6+d}{2}})$, obtained by balancing the first and third error terms in (11). The NN width to achieve the same accuracy is $m\ge m_0=\mathcal{O}(\epsilon^{-\frac{d\ln(n)}{4}})$, obtained by balancing the first and last error terms in (11).

Before we prove Theorem 4.1, let us review relevant results that will be used for the proof.

The Parametrization Error of DM. For reader's convenience, we briefly summarize the pointwise error bound of the discrete estimator, which has been reported extensively (see e.g., [12, 68, 8, 17]). Our particular interest is to quantify the error induced by the matrix $\mathbf{L}_{a,\epsilon}:=-\mathbf{a}+\mathbf{L}_\epsilon$, where $\mathbf{L}_\epsilon$ is defined in (5) and $\mathbf{a}$ is a diagonal matrix with diagonal entries $\mathbf{a}_{ii}=a(\mathbf{x}_i)$.

First, let us quantify the Monte-Carlo error of the discretization of the integral operator, i.e., the error for introducing L_ϵ in (3). In particular, using the Chernoff bound (see Appendix B.2 in [8] or Appendix A in [26]), for any $\mathbf{x}_i\in X$ and any fixed $\epsilon,\eta>0$, and $u\in L^2(M)$, we have

$$\mathbb{P}(|(\mathbf{L}_\epsilon\mathbf{u})_i-L_\epsilon u(\mathbf{x}_i)|>\eta)<2\exp\Big(-C\frac{\eta^2\epsilon^{d/2+1}N}{\|\nabla_g u(\mathbf{x}_i)\|^2q(\mathbf{x}_i)^{-1}}\Big),$$

for some constant C that is independent of ϵ and N. Choosing $N^2 = 2\exp\Big(-C\frac{\eta^2\epsilon^{d/2+1}N}{\|\nabla_g u(\mathbf{x}_i)\|^2 q(\mathbf{x}_i)^{-1}}\Big)$, one can deduce that $\eta = C^{-1/2}(\frac{\log N}{N})^{1/2}\epsilon^{-1/2-d/4}\|\nabla_g u(\mathbf{x}_i)\|q(\mathbf{x}_i)^{-1/2}$, which means that with probability greater than $1-N^{-2}$,

$$(\mathbf{L}_\epsilon \mathbf{u})_i = L_\epsilon u(\mathbf{x}_i) + \mathcal{O}\Big(\frac{\sqrt{\log N}\|\nabla_g u(\mathbf{x}_i)\|q(\mathbf{x}_i)^{-1/2}}{N^{1/2}\epsilon^{1/2+d/4}}\Big),$$

as $N\to\infty$. When the density $q = d\pi/dV$ is non-uniform, one can use the same argument (e.g., see Appendix B.1 in [8]) to deduce the error induced by the estimation of the density, which is of order $\mathcal{O}(q(\mathbf{x}_j)(\frac{\log N}{N})^{1/2}\epsilon^{-2-d/4})$ with probability $1-N^{-2}$, to ensure a density estimation of order-ϵ^2. Together with (3), ignoring the $\sqrt{\log(N)}$ factor, we have the following pointwise error estimate.

Lemma 4.1. *Let $u,\kappa\in C^3(M)\cap L^2(M)$ and $q\in C(M)$, where $M\subseteq\mathbb{R}^n$ is a d-dimensional closed C^3-submanifold, then for any $\mathbf{x}_i\in X$, with probability higher than $1-N^{-2}$,*

$$(\mathbf{L}_{a,\epsilon}\mathbf{u})_i - \mathscr{L}_a u(\mathbf{x}_i) = (\mathbf{L}_\epsilon\mathbf{u})_i - \mathscr{L}u(\mathbf{x}_i) = \mathcal{O}\Big(\epsilon, \frac{q(\mathbf{x}_i)^{1/2}}{N^{1/2}\epsilon^{2+d/4}}, \frac{\|\nabla_g u(\mathbf{x}_i)\|q(\mathbf{x}_i)^{-1/2}}{N^{1/2}\epsilon^{1/2+d/4}}\Big), \tag{13}$$

as $\epsilon\to 0$ after $N\to\infty$.

If we allow for $u,\kappa\in C^4(M)$, then by Lemma 3.3 in [33], we can replace the first-order error term in (13) by $R_u(x)\epsilon^{4\beta-1}$, where $0<\beta<1/2$ such that

$$\|R_u\|_{L^2(M)} \le C\|u\|_{H^4(M)}\|\sqrt{\kappa}\|^2_{C^4(M)} < \infty$$

for some constant $C>0$ for $u,\kappa\in C^4(M)$. The term R_u will also appear in the second-error term as well since this error bound is to ensure the density estimation to achieve $\mathcal{O}(\epsilon^2)$. Also, with the given assumptions, it is immediate to show that $\|R_u q\|^2_{L^2(\pi)}, \|R_u\|^2_{L^2(\pi)} < \infty$.

For a fixed $\epsilon>0$ and using the fact that $\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^N f(\mathbf{x}_i)^2 = \int_M f(x)^2 q(x)dV(x) = \|f\|^2_{L^2(\pi)}$, we have,

$$\begin{aligned}
\lim_{N\to\infty}\|\mathbf{L}_{a,\epsilon}\mathbf{u} - \mathscr{L}_a u(X)\|^2_{L^2(\pi_N)} &= \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^N |(\mathbf{L}_{a,\epsilon}\mathbf{u})_i - \mathscr{L}_a u(x_i)|^2 \\
&\le C_1\epsilon^{8\beta-2}\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^N R_u(x_i)^2 + C_2\epsilon^{-4-\frac{d}{2}}\lim_{N\to\infty}\frac{1}{N^2}\sum_{i=1}^N (R_u(x_i)q(x_i)^2) \\
&\quad + C_3\epsilon^{-1-\frac{d}{2}}\lim_{N\to\infty}\frac{1}{N^2}\sum_{i=1}^N \|\nabla_g u(x_i)\|^2 (q(x_i)^{-1}) \\
&= C_1\epsilon^{8\beta-2}\|R_u\|^2_{L^2(\pi)} + \big(\lim_{N\to\infty}\frac{1}{N}\big)\big(C_2\epsilon^{-4-\frac{d}{2}}\|R_u q\|^2_{L^2(\pi)} + C_3\epsilon^{-1-\frac{d}{2}}\|u\|_{H^1(M)}\big) \\
&= C_1\epsilon^{8\beta-2}\|R_u\|^2_{L^2(\pi)}.
\end{aligned}$$

To conclude, we have the following lemma.

Lemma 4.2. *Let $u,\kappa\in C^4(M)$ and $q\in C(M)$ and let $M\in\mathbb{R}^n$ be a $d-$dimensional, closed, C^3-submanifold. Then, with probability higher than $1-N^{-2}$,*

$$\lim_{\epsilon\to 0}\lim_{N\to\infty}\|\mathbf{L}_{a,\epsilon}\mathbf{u} - \mathscr{L}_a u(X)\|_{L^2(\pi_N)} = 0. \tag{14}$$

This consistency estimate only holds when the limits are taken in the sequence as above.

The Parametrization Error of NNs. Let us denote the best possible empirical loss as $\mathscr{J}_{S,\epsilon}(\boldsymbol{\theta}_S)$, which depends only on the NN model and is independent of the optimization algorithm to solve the empirical loss minimization in (9), since $\boldsymbol{\theta}_S$ is a global minimizer of the empirical loss. The estimation of $\mathscr{J}_{S,\epsilon}(\boldsymbol{\theta}_S)$ can be derived from deep network approximation theory in Theorem 1.1 of [48] and its corollary in [13]. The (nearly optimal) error bound in Theorem 1.1 of [48] focuses on approximating functions in $C^s([0,1]^n)$ and hence suffers from the curse of dimensionality; namely, the total number of parameters in the ReLU FNN scales exponentially in n to achieve the same approximation accuracy. Fortunately, our PDE solution is only defined on a d-dimensional manifold embedded in $[0,1]^n$ with $d\ll n$. By taking advantage of the low-dimensional manifold, a corollary of Theorem 1.1 in [48] was proposed in [13] following the idea in [63] to conquer the curse of dimensionality. For completeness, we quote this corollary below.

Lemma 4.3 (Proposition 4.2 of [13]). *Given $m, L \in \mathbb{N}^+$, $\mu \in (0,1)$, $\nu \in (0,1)$. Let $M \subset \mathbb{R}^n$ be a compact d-dimensional Riemannian submanifold having condition number τ_M^{-1}, volume V_M, and geodesic covering regularity G_M, and define the μ-neighborhood as $M_\mu := \{\mathbf{x} \in \mathbb{R}^n : \inf_{\mathbf{y}\in M} \|\mathbf{x}-\mathbf{y}\|_2 \le \mu\}$. If $u \in C^s(M_\mu)$ with $s \in \mathbb{N}^+$, then there exists a ReLU FNN ϕ with width $17 s^{d_\nu+1} 3^{d_\nu} d_\nu (m+2)\log_2(8m)$ and depth $18 s^2 (L+2)\log_2(4L) + 2d_\nu$ such that for any $\mathbf{x} \in M_\mu$,*

$$|\phi(\mathbf{x}) - u(\mathbf{x})| \le 8\|u\|_{C^s(M_\mu)} \mu \Big((1-\nu)^{-1}\sqrt{n/d_\nu} + 1 \Big) + 170(s+1)^{d_\nu} 8^s (1-\nu)^{-1} \|u\|_{C^s(M_\mu)} m^{-2s/d_\nu} L^{-2s/d_\nu}, \quad (15)$$

where $d_\nu := \mathscr{O}\left(d \ln\left(n V_M G_M \tau_M^{-1}/\nu\right)/\nu^2\right) = \mathscr{O}\left(d\ln(n/\nu)/\nu^2\right)$ is an integer with $d < d_\nu < n$.

In Lemma 4.3, ReLU FNNs are used while in practice the learnable linear combination of a few activation functions might boost the numerical performance of neural networks [45]. The extension of Lemma 4.3 to the case of multiple kinds of activation functions is interesting future work. Lemma 4.3 can provide a baseline characterization to the approximation capacity of FNNs when ReLU is one of the choices of activation functions for a learnable linear combination. Hence, the following error estimation is still true for NNs constructed with the learnable linear combination of a few activation functions.

Using the same argument as the extension lemma for smooth functions (see e.g., Lemma 2.26 in [42]), one can extend any $u \in C^4(M)$ to $u \in C^4(M_{\mu_0})$, for any C^4 manifold $M \subseteq \mathbb{R}^n$ and a positive μ_0. Therefore, by Corollary 4.3, there exists a ReLU FNN ϕ with width $\mathscr{O}(n\ln(n)m\log(m))$ and depth $\mathscr{O}(L\log(L)+\ln(n))$ such that

$$|u(\mathbf{x}) - \phi(\mathbf{x})| \le C_{M,d,\nu}\|u\|_{C^3(M_{\mu_0})} \left(\mu\sqrt{\frac{n}{\ln n}} + n m^{-8/(d\ln(n))} L^{-8/(d\ln(n))} \right) \quad \text{for all } \mathbf{x} \in M,$$

for any $\mu \in (0, \mu_0)$, where $C_{M,d,\nu}$ is a constant depending only on M, d, and ν. Note that the curse of dimensionality has been lessened in the above approximation rate. Therefore, by taking $\mu \to 0$, we have the following error estimation

$$\|\mathbf{u} - \boldsymbol{\phi}\|_{L^2(\pi_N)} = \mathscr{O}(m^{-8/(d\ln(n))} L^{-8/(d\ln(n))}), \quad (16)$$

where the prefactor depends on M, d, ν, $\|u\|_{C^3(M_{\mu_0})}$, and n. By (13) and (16),

$$\begin{aligned} \|\mathbf{L}_{a,\epsilon}\boldsymbol{\phi}_S - \mathbf{f}\|_{L^2(\pi_N)} &\le \|\mathbf{L}_{a,\epsilon}\boldsymbol{\phi} - \mathbf{f}\|_{L^2(\pi_N)} \le \|\mathbf{L}_{a,\epsilon}\boldsymbol{\phi} - \mathbf{L}_{a,\epsilon}\mathbf{u}\|_{L^2(\pi_N)} + \|\mathbf{L}_{a,\epsilon}\mathbf{u} - \mathscr{L}_a u(X)\|_{L^2(\pi_N)} \\ &\le \|\mathbf{L}_{a,\epsilon}\|_2 \|\boldsymbol{\phi} - \mathbf{u}\|_{L^2(\pi_N)} + \mathscr{O}(\epsilon, N^{-\frac{1}{2}}\epsilon^{-2-\frac{d}{4}}, N^{-\frac{1}{2}}\epsilon^{-\frac{1}{2}-\frac{d}{4}}) \\ &= \|\mathbf{L}_{a,\epsilon}\|_2 \mathscr{O}(m^{-8/(d\ln(n))} L^{-8/(d\ln(n))}) + \mathscr{O}(\epsilon, N^{-\frac{1}{2}}\epsilon^{-2-\frac{d}{4}}, N^{-\frac{1}{2}}\epsilon^{-\frac{1}{2}-\frac{d}{4}}), \end{aligned} \quad (17)$$

where, for simplicity, we suppressed the functional dependence on $\mathbf{x}$ in the second error term above.

Recall that $\mathbf{L}_{a,\epsilon} = -\mathbf{a} + \mathbf{L}_\epsilon$, where $\mathbf{a}$ is a diagonal matrix with diagonal entries $0 < a_{\min} \le a(x_i) \le a_{\max}$ and $\mathbf{L}_\epsilon \mathbf{1} = \mathbf{0}$. By definition, $\mathbf{L}_\epsilon$ is diagonally dominant with diagonal negative entries and non-diagonal positive entries. This means

$$\|\mathbf{L}_{a,\epsilon}\|_\infty = \max_{1\le j\le N} \Big\{ a(\mathbf{x}_j) - \mathbf{L}_{\epsilon,jj} + \sum_{i\ne j} \mathbf{L}_{\epsilon,ij} \Big\} = \max_{1\le j\le N} \Big\{ a(\mathbf{x}_j) - 2\mathbf{L}_{\epsilon,jj} \Big\} = a_{\max} + C\epsilon^{-1},$$

for some constant C that depends on $\|\kappa\|_\infty$. Therefore, $\|\mathbf{L}_{a,\epsilon}\|_2 \le N^{1/2}\|\mathbf{L}_{a,\epsilon}\|_\infty \le CN^{1/2}\epsilon^{-1}$. Plugging this to (17), we obtain

$$\|\mathbf{L}_{a,\epsilon}\boldsymbol{\phi}_S - \mathbf{f}\|_{L^2(\pi_N)} \le \mathscr{O}\big(\epsilon, N^{-\frac{1}{2}}\epsilon^{-2-\frac{d}{4}}, N^{-\frac{1}{2}}\epsilon^{-\frac{1}{2}-\frac{d}{4}}, N^{1/2}\epsilon^{-1} m^{-8/(d\ln(n))} L^{-8/(d\ln(n))}\big), \quad (18)$$

as $\epsilon \to 0$ after $N \to \infty$, and m or $L \to \infty$. This concludes the upper bound for the best possible empirical loss.

Proof of Theorem 4.1. Now we derive the overall parametrization error considering DM and NN parametrization together to prove Theorem 4.1. Since $\mathbf{L}_\epsilon$ is an unnormalized discrete Graph-Laplacian matrix that is semi-negative definite, for $a \ge a_{\min} > 0$, one can see that,

$$\langle \boldsymbol{\xi}, \mathbf{L}_{a,\epsilon}\boldsymbol{\xi}\rangle_{L^2(\pi_N)} = \langle \boldsymbol{\xi}, \mathbf{L}_\epsilon\boldsymbol{\xi}\rangle_{L^2(\pi_N)} - \langle \boldsymbol{\xi}, \mathbf{a}\boldsymbol{\xi}\rangle_{L^2(\pi_N)} \le -a_{\min}\|\boldsymbol{\xi}\|^2_{L^2(\pi_N)},$$

for any $\boldsymbol{\xi} \in L^2(\pi_N)$. Letting $\boldsymbol{\xi} = \mathbf{u} - \boldsymbol{\phi}_S$, we have,

$$a_{\min}\|\mathbf{u} - \boldsymbol{\phi}_S\|^2_{L^2(\pi_N)} \le -\langle \mathbf{u} - \boldsymbol{\phi}_S, \mathbf{L}_{a,\epsilon}(\mathbf{u} - \boldsymbol{\phi}_S)\rangle_{L^2(\pi_N)} \le \|\mathbf{L}_{a,\epsilon}(\mathbf{u} - \boldsymbol{\phi}_S)\|_{L^2(\pi_N)}\|\mathbf{u} - \boldsymbol{\phi}_S\|_{L^2(\pi_N)}$$

and with probability higher than $1 - N^{-2}$,

$$\begin{aligned} \|\mathbf{u} - \boldsymbol{\phi}_S\|_{L^2(\pi_N)} &\le \frac{1}{a_{\min}}\|\mathbf{L}_{a,\epsilon}(\mathbf{u} - \boldsymbol{\phi}_S)\|_{L^2(\pi_N)} \\ &\le \frac{1}{a_{\min}}\Big(\|\mathbf{L}_{a,\epsilon}\mathbf{u} - \mathbf{f}\|_{L^2(\pi_N)} + \|\mathbf{L}_{a,\epsilon}\boldsymbol{\phi}_S - \mathbf{f}\|_{L^2(\pi_N)} \Big), \\ &\le \frac{1}{a_{\min}}\Big(\|\mathbf{L}_{a,\epsilon}\mathbf{u} - \mathscr{L}_a u(X)\|^2_{L^2(\pi_N)} + \|\mathbf{L}_{a,\epsilon}\boldsymbol{\phi}_S - \mathbf{f}\|_{L^2(\pi_N)} \Big), \\ &= \mathscr{O}\big(\epsilon, N^{-\frac{1}{2}}\epsilon^{-2-\frac{d}{4}}, N^{-\frac{1}{2}}\epsilon^{-\frac{1}{2}-\frac{d}{4}}, N^{1/2}\epsilon^{-1} m^{-8/(d\ln(n))} L^{-8/(d\ln(n))}\big), \end{aligned} \quad (19)$$

as $\epsilon \to 0$ after $N \to \infty$, and m or $L \to \infty$. Here, we have used (13) and (18) in the last equality above. Since the inequality in (15) is valid uniformly, together with (14), we achieve the convergence in (12) and the proof is completed. □

4.2 Optimization Error

In the parametrization error analysis, we have assumed that a global minimizer of the empirical loss minimization in (9) is achievable. In practice, a numerical optimization algorithm is used to solve this optimization problem and the numerical minimizer might not be equal to a global minimizer. Therefore, it is important to investigate the numerical convergence of an optimization algorithm to a global minimizer. The neural tangent kernel analysis originated in [38] and further developed in [3, 2] has been proposed to analyze the global convergence of the (stochastic) gradient descent method for the least-squares empirical loss function in (9). In the situation of solving PDEs, the global convergence analysis of optimization algorithms is vastly open. In [50], it was shown that the gradient descent method can identify a global minimizer of the least-squares optimization for solving second-order linear PDEs with two-layer neural networks under the assumption of over-parametrization. In this paper, we will extend this result in the context where the differential operator in the loss function is replaced by a discrete and approximate operator, $\mathbf{L}_{a,\epsilon}$, applied to the neural network solution (e.g., see (9) and (10)). Furthermore, the activation function considered here is ReLU^r, which is more general than the activation function in [50]. The extension to deeper neural networks follows the analysis in [2] and is left as future work.

To establish a general theorem applicable to various applications, we consider the following emprical risk $R_S(\boldsymbol{\theta})$ with an over-parametrized two-layer neural network optimized by the gradient descent method:

$$R_S(\boldsymbol{\theta}) := \frac{1}{2N}(\boldsymbol{A}\phi(X;\boldsymbol{\theta}) - f(X))^\intercal(\boldsymbol{A}\phi(X;\boldsymbol{\theta}) - f(X)), \tag{20}$$

where $X := \{\boldsymbol{x}_i\}_{i=1}^N$ is a given set of samples in $[0,1]^n$ of an arbitrary distribution (π in our specific application); $\boldsymbol{A}$ is a given matrix of size $N \times N$; and the two-layer neural network used here is constructed as

$$\phi(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{k=1}^m a_k \sigma(\boldsymbol{w}_k^\intercal \boldsymbol{x}), \tag{21}$$

where for $k \in [m] := \{1,\dots,m\}$, $a_k \in \mathbb{R}$, $\boldsymbol{w}_k \in \mathbb{R}^n$, $\boldsymbol{\theta} = \mathrm{vec}\{a_k, \boldsymbol{w}_k\}_{k=1}^m$, and $\sigma(x) = \mathrm{ReLU}^r(x)$, i.e., the r-th power of the ReLU activation function. We should point out that the matrix $\boldsymbol{A}$ in our specific application is $\mathbf{L}_{a,\epsilon}$ for (9); $\boldsymbol{A}$ is a block-diagonal matrix for (10) with one block for the differential equation and another block for the boundary condition; $\boldsymbol{A}$ is also a block-diagonal matrix when a regularization term $\|\boldsymbol{\phi}_{\boldsymbol{\theta}}\|^2_{L^2(\pi_N)}$ is applied to either (9) or (10). In the remainder of this section, we use the notation $\boldsymbol{A}$ since the result holds in general under some assumption discussed below. Without loss of generality, throughout our analysis, we also assume $|f| \le 1$, since the PDE defined on a compact domain can be normalized.

Recall that we use the two-layer neural network $\phi(\boldsymbol{x};\boldsymbol{\theta})$ in (21) with $\boldsymbol{\theta} = \mathrm{vec}\{a_k, \boldsymbol{w}_k\}_{k=1}^m$. In the gradient descent iteration, we use t to denote the iteration or the artificial time variable in the gradient flow. Hence, we define the following notations for the evolution of parameters at time t:

$$a_k^t := a_k(t), \quad \boldsymbol{w}_k^t := \boldsymbol{w}_k(t), \quad \boldsymbol{\theta}^t := \boldsymbol{\theta}(t) := \mathrm{vec}\{a_k^t, \boldsymbol{w}_k^t\}_{k=1}^m.$$

Similarly, we can introduce t to other functions or variables depending on $\boldsymbol{\theta}(t)$. When the dependency of t is clear, we will drop the index t. In the initialization of gradient descent, we set

$$a_k^0 := a_k(0) \sim \mathcal{N}(0,\gamma^2), \quad \boldsymbol{w}_k^0 := \boldsymbol{w}_k(0) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n), \quad \boldsymbol{\theta}^0 := \boldsymbol{\theta}(0) := \mathrm{vec}\{a_k^0, \boldsymbol{w}_k^0\}_{k=1}^m. \tag{22}$$

Then the empirical risk can be written as

$$\begin{aligned} R_S(\boldsymbol{\theta}) &= \frac{1}{2N}(\boldsymbol{A}\phi(X;\boldsymbol{\theta}) - f(X))^\intercal(\boldsymbol{A}\phi(X;\boldsymbol{\theta}) - f(X)) \\ &= \frac{1}{2N}(\phi(X;\boldsymbol{\theta}) - \boldsymbol{A}^{-1}f(X))^\intercal \boldsymbol{A}^\intercal \boldsymbol{A}(\phi(X;\boldsymbol{\theta}) - \boldsymbol{A}^{-1}f(X)) \\ &= \frac{1}{2N}\boldsymbol{e}^\intercal \boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{e}, \end{aligned}$$

where we denote $e_i = \phi(\boldsymbol{x}_i;\boldsymbol{\theta}) - (\boldsymbol{A}^{-1}f(X))_i$ for $i \in [N]$ and $\boldsymbol{e} = (e_1, e_2, \dots, e_N)^\intercal$. Hence, the gradient descent dynamics is

$$\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}), \tag{23}$$

or equivalently in terms of a_k and $\boldsymbol{w}_k$ as follows:

$$\begin{aligned}\dot{a}_k &= -\nabla_{a_k} R_S(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}(e^\intercal \boldsymbol{A}^\intercal \boldsymbol{A})_i \sigma(\boldsymbol{w}_k^\intercal \boldsymbol{x}_i),\\ \dot{\boldsymbol{w}}_k &= -\nabla_{\boldsymbol{w}_k} R_S(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}(e^\intercal \boldsymbol{A}^\intercal \boldsymbol{A})_i a_k \sigma'(\boldsymbol{w}_k^\intercal \boldsymbol{x}_i)\boldsymbol{x}_i.\end{aligned} \tag{24}$$

Adopting the neuron tangent kernel point of view [38], in the case of a two-layer neural network with an infinite width, the corresponding kernels $k^{(a)}$ for parameters in the last linear transform and $k^{(w)}$ for parameters in the first layer are functions from $M \times M$ to $\mathbb{R}$ defined by

$$\begin{aligned}k^{(a)}(\boldsymbol{x},\boldsymbol{x}') &:= \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I}_n)} g^{(a)}(\boldsymbol{w};\boldsymbol{x},\boldsymbol{x}'),\\ k^{(w)}(\boldsymbol{x},\boldsymbol{x}') &:= \mathbb{E}_{(a,\boldsymbol{w})\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I}_{n+1})} g^{(w)}(a,\boldsymbol{w};\boldsymbol{x},\boldsymbol{x}'),\end{aligned}$$

where

$$\begin{aligned}g^{(a)}(\boldsymbol{w};\boldsymbol{x},\boldsymbol{x}') &:= \left[\sigma(\boldsymbol{w}^\intercal \boldsymbol{x})\right]\cdot\left[\sigma(\boldsymbol{w}^\intercal \boldsymbol{x}')\right],\\ g^{(w)}(a,\boldsymbol{w};\boldsymbol{x},\boldsymbol{x}') &:= a^2\left[\sigma'(\boldsymbol{w}^\intercal \boldsymbol{x})\boldsymbol{x}\right]\cdot\left[\sigma'(\boldsymbol{w}^\intercal \boldsymbol{x}')\boldsymbol{x}'\right].\end{aligned}$$

These kernels evaluated at $N\times N$ pairs of samples lead to $N\times N$ Gram matrices $\boldsymbol{K}^{(a)}$ and $\boldsymbol{K}^{(\boldsymbol{w})}$ with $K^{(a)}_{ij} = k^{(a)}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ and $K^{(w)}_{ij} = k^{(w)}(\boldsymbol{x}_i,\boldsymbol{x}_j)$, respectively. Our analysis requires the matrix $\boldsymbol{K}^{(a)}$ to be positive definite, which has been verified for regression problems under mild conditions on random training data $X = \{\boldsymbol{x}_i\}_{i=1}^N$ and can be generalized to our case. Hence, we assume this together with the non-singularity of $\boldsymbol{A}$ as follows for simplicity.

Assumption 4.1. *Assume that: 1) The smallest eigenvalue of $\boldsymbol{K}^{(a)}$, denoted as λ_S, is positive. 2) The smallest eigenvalue of $\boldsymbol{A}\boldsymbol{A}^\intercal$, denoted as $\lambda_{\boldsymbol{A}}$, is positive.*

For a two-layer neural network with m neurons, define the $N\times N$ Gram matrix $\boldsymbol{G}^{(a)}(\boldsymbol{\theta})$ and $\boldsymbol{G}^{(w)}(\boldsymbol{\theta})$ using the following expressions for the (i,j)-th entry

$$\begin{aligned}\boldsymbol{G}^{(a)}_{ij}(\boldsymbol{\theta}) &:= \frac{1}{m}\sum_{k=1}^{m} g^{(a)}(\boldsymbol{w}_k;\boldsymbol{x}_i,\boldsymbol{x}_j),\\ \boldsymbol{G}^{(w)}_{ij}(\boldsymbol{\theta}) &:= \frac{1}{m}\sum_{k=1}^{m} g^{(w)}(a_k,\boldsymbol{w}_k;\boldsymbol{x}_i,\boldsymbol{x}_j).\end{aligned} \tag{25}$$

Clearly, $\boldsymbol{G}^{(a)}(\boldsymbol{\theta})$ and $\boldsymbol{G}^{(w)}(\boldsymbol{\theta})$ are both positive semi-definite for any $\boldsymbol{\theta}$. Let $\boldsymbol{G}(\boldsymbol{\theta}) = \boldsymbol{G}^{(a)}(\boldsymbol{\theta}) + \boldsymbol{G}^{(w)}(\boldsymbol{\theta})$, taking time derivative of (20) and using the equalities in (24) and (25), then we have the following evolution equation to understand the dynamics of the gradient descent method applied to (20):

$$\frac{\mathrm{d}}{\mathrm{d}t}R_S(\boldsymbol{\theta}) = -\|\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta})\|_2^2 = -\frac{m}{N^2}\boldsymbol{e}^\intercal \boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{G}(\boldsymbol{\theta})\boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{e} \le -\frac{m}{N^2}\boldsymbol{e}^\intercal \boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{G}^{(a)}(\boldsymbol{\theta})\boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{e}. \tag{26}$$

Our goal is to show that $R_S(\boldsymbol{\theta})$ converges to zero. These goals are true if the smallest eigenvalue $\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta})\right)$ of $\boldsymbol{G}^{(a)}(\boldsymbol{\theta})$ has a positive lower bound uniformly in t, since in this case we can solve (26) and bound $R_S(\boldsymbol{\theta})$ with a function in t converging to zero when $t\to\infty$ as shown in Lemma B.6. In fact, a uniform lower bound of $\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta})\right)$ can be $\frac{1}{2}\lambda_S$, which can be proved in the following three steps:

- (**Initial phase**) By Assumption 4.1 of $\boldsymbol{K}^{(a)}$, we can show that $\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(0))\right) \approx \lambda_S$ in Lemma B.5 using the observation that $\boldsymbol{K}^{(a)}_{ij}$ is the mean of $g(\boldsymbol{w};\boldsymbol{x}_i,\boldsymbol{x}_j)$ over the normal random variable $\boldsymbol{w}$, while $\boldsymbol{G}^{(a)}_{ij}(\boldsymbol{\theta}(0))$ is the mean of $g(\boldsymbol{w};\boldsymbol{x}_i,\boldsymbol{x}_j)$ with m independent realizations.
- (**Evolution phase**) The GD dynamics results in $\boldsymbol{\theta}(t)\approx\boldsymbol{\theta}(0)$ under the assumption of over-parametrization as shown in Lemma B.7, which indicates that
$$\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(0))\right) \approx \lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(t))\right).$$
- (**Final phase**) To show the uniform bound $\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(t))\right) \ge \frac{1}{2}\lambda_S$ for all $t\ge 0$, we introduce a stopping time t^* via
$$t^* = \inf\{t \mid \boldsymbol{\theta}(t)\notin\mathcal{M}(\boldsymbol{\theta}^0)\}, \tag{27}$$
where
$$\mathcal{M}(\boldsymbol{\theta}^0) := \left\{\boldsymbol{\theta} \mid \|\boldsymbol{G}^{(a)}(\boldsymbol{\theta}) - \boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0)\|_{\mathrm{F}} \le \frac{1}{4}\lambda_S\right\}, \tag{28}$$
and show that t^* is in fact equal to infinity in the final proof of Theorem 4.2 in Appendix B.

Let κ_A denote the condition number of $A^\intercal A$. Our main result of the global convergence of the gradient descent method for (20) is summarized in Theorem 4.2 below. The proof of Theorem 4.2 can be found in Appendix B.

Theorem 4.2 (Global Convergence of Gradient Descent: Two-Layer Neural Networks)**.** *Let $\boldsymbol{\theta}^0 := \mathrm{vec}\{a_k^0, \boldsymbol{w}_k^0\}_{k=1}^m$ at the gradient descent initialization for solving* (20)*, where $a_k^0 \sim \mathcal{N}(0,\gamma^2)$ and $\boldsymbol{w}_k^0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$ with any $\gamma \in (0,1)$. Let λ_S be a positive constant in Assumption 4.1. For any $\delta \in (0,1)$, if $m \geq \mathcal{O}(\kappa_A \mathrm{poly}(N, r, n, \frac{1}{\delta}, \frac{1}{\lambda_S}))$, then with probability at least $1-\delta$ over the random initialization $\boldsymbol{\theta}^0$, we have, for all $t \geq 0$,*

$$R_S(\boldsymbol{\theta}(t)) \leq \exp\left(-\frac{m\lambda_S\lambda_{\boldsymbol{A}} t}{N}\right) R_S(\boldsymbol{\theta}^0).$$

For the estimate of $R_S(\boldsymbol{\theta}^0)$, see Lemma B.4. In particular, if $\gamma = \mathcal{O}(\frac{1}{\sqrt{m}(\log m)^2})$, then $R_S(\boldsymbol{\theta}^0) = \mathcal{O}(1)$. For two-layer neural networks, Theorem 4.2 shows that, as long as the network width $m \geq \mathcal{O}(\kappa_A \mathrm{poly}(N, r, n, \frac{1}{\delta}, \frac{1}{\lambda_S}))$, the gradient descent method can identify a global minimizer of the empirical risk minimization in (20). For a quantitative description of $\mathcal{O}(\kappa_A \mathrm{poly}(N, r, n, \frac{1}{\delta}, \frac{1}{\lambda_S}))$, see (44) in Appendix B. In the case of FNNs with L layers, following the proof in [2], one can show the global convergence of gradient descent when $m \geq \mathcal{O}(\kappa_A \mathrm{poly}(L, N, r, n, \frac{1}{\delta}, \frac{1}{\lambda_S}))$, which is left as future work.

5 Numerical Examples

In this section, we numerically demonstrate the effectiveness and practicability of our proposed NN-based PDE solver on unknown manifolds. Our numerical examples will show that an NN-based solver can achieve low error on the given points and good generalization on the unseen data points. Also, we will include clock-time comparison to show that the proposed NN solver requires a much shorter clock-time compared with the DM-based solver (which directly solves the linear system in (6)) for the large point cloud size.

Three numerical examples are used for demonstration. First, we test our method on a two-dimensional torus embedded in $\mathbb{R}^3$ to show that the NN-based solver can achieve comparable error to the DM-based solver. We will see that the NN-based solver can handle large data set while the DM-based solver fails. Second, we demonstrate the ability of the NN-based solver to deal with manifolds of high co-dimension by validating it on a three-dimensional manifold embedded in $\mathbb{R}^{12}$. We will see that the NN-based solver can obtain a more accurate solution than DM. Finally, our method is applied to the equation on a two-dimensional semi-torus to verify the performance of the NN-based solver on problems with Dirichlet boundary conditions.

Devices and environments. The experiments of DM are conducted on the workstation with 32× Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz and 1 TB RAM and Matlab R2019a. The experiments of the NN-solver are conducted on the workstation with 16× Intel(R) Xeon(R) Gold 5122 CPU @ 3.60GHz and 93G RAM using Pytorch 1.0 and 1× Tesla V100.

Implementation detail. We summarize notations and list the hyperparameter setting for each numerical example in Appendix C. In our implementation, we use a 3-hidden-layer FNN with the same width m per hidden layer and the smooth Polynomial-Sine activation function in [45]. Polynomial-Sine is defined as $\alpha_1 \sin(\beta_1 x)+\alpha_2 x+\alpha_3 x^2$, where β_1, $\alpha_i, i = 1,2,3$ are trainable parameters initialized by normal distribution $\mathcal{N}(1, 0.01), \mathcal{N}(1, 0.01), \mathcal{N}(0, 0.01), \mathcal{N}(0, 0.01)$ respectively. As we shall demonstrate, our proposed NN solver is not sensitive to the numerical choices (e.g., activation functions, network types, and optimization algorithms) but a more advanced algorithm design may improve the accuracy and convergence. Particularly in Example 2, we will compare the performance of ReLU FNNs and ReLU^3 FNNs trained by the gradient descent method and the performance of the Polynomial-Sine FNNs trained with the Adam optimizer [41]. Though the ReLU or ReLU^3 FNNs with the gradient descent method enjoy theoretical guarantees in our analysis, the Polynomial-Sine FNNs with Adam can generalize better. So, the Polynomial-Sine activation and Adam optimizer will be used in all of our examples. In Adam, we use an initial learning rate of 0.01 for T iterations. The learning rate follows cosine decay with the increasing training iterations, i.e., the learning rate decays by multiplying a factor $0.5(cos(\frac{\pi t}{T}) + 1)$, where t is the current iteration. The NN results reported in this section are averaged over 5 independent experiments.

5.1 Example 1: 2D Torus

In this first example, we solve the elliptic PDE in (1) for $a = 0$ on a two-dimensional torus embedded in $\mathbb{R}^3$ with an embedding function defined as,

$$\iota(\theta_1,\theta_2) = \begin{pmatrix} (2+\cos\theta_1)\cos\theta_2 \\ (2+\cos\theta_1)\sin\theta_2 \\ \sin\theta_1 \end{pmatrix} \in \mathbb{R}^3 \text{ for } \begin{matrix} 0 \le \theta_1 \le 2\pi \\ 0 \le \theta_2 \le 2\pi \end{matrix}. \tag{29}$$

Here θ_1, θ_2 denote the intrinsic coordinates. For this numerical experiment, we set the diffusion coefficient $\kappa(\theta_1,\theta_2) = 1.1 + \sin^2\theta_1\cos^2\theta_2$, the true solution $u(\theta_1,\theta_2) = (\sin 2\theta_2 - 2\cos 2\theta_2/(2+\cos\theta_1))\cos\theta_1$, and analytically compute the right hand side function f. The point cloud data $X = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$ are uniformly sampled from intrinsic coordinates (θ_1,θ_2). We solve the PDE by optimizing the least-square problem given by,

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{2}\|\mathbf{L}_\epsilon \boldsymbol{\phi}_{\boldsymbol{\theta}} - \mathbf{f}\|^2_{L^2(\pi_N)} + \frac{\gamma}{2}\|\boldsymbol{\phi}_{\boldsymbol{\theta}}\|^2_{L^2(\pi_N)}. \tag{30}$$

where we add a regularization term with $\gamma > 0$ to overcome the ill-posedness induced by $a = 0$ (as discussed right after (9)).

The hyperparameter setting for the different N is summarized in Table 4 in Appendix C. To facilitate the training of FNN, we increase the NN width m and the number of iterations T as the number of training points N grows. We apply Adam optimizer to minimize (30). In particular, when $N = 80089$, Adam is used to minimize (30) since $\mathbf{L}_\epsilon \in \mathbb{R}^{80089\times 80089}$ is too large to compute in GPU directly. In our implementation, at each time, 8000 rows from $\mathbf{L}_\epsilon$ are randomly sampled and the submatrix $\mathbf{L}_{\text{sub}}$ of size 8000×80089 substitutes $\mathbf{L}_\epsilon$ in the loss (30). Then the network parameters $\boldsymbol{\theta}$ are updated for 100 iterations at each time. We repeat this procedure for 80 times.

Figure 1 shows various errors of DM and NN solutions as functions of training data size, N. We report the more precise numerical value of the errors in Table 5 in Appendix C. Since DM-based solver can only obtain the approximate solution at the point cloud data, we report the *forward error*, $\|\mathbf{L}_\epsilon\mathbf{u} - \mathbf{f}\|_\infty$, where $\mathbf{u} = u(X)$, which quantifies the accuracy of the approximation of the differential operator, and the *inverse error*, $\|\mathbf{u}_\epsilon - \mathbf{u}\|_\infty$, where $\mathbf{u}_\epsilon$ is the solution obtained by taking a pseudo-inverse of (6), which quantifies the accuracy of the approximate solution. As for the NN solution, since the solution is of the form $\phi(\cdot,\boldsymbol{\theta}_N)$, where $\boldsymbol{\theta}_N$ denotes the numerically obtained minimizer, we show the *testing error*, which is defined as the ℓ_∞ error on 300^2 Gauss-Legendre quadrature points that are not in the training data set. In Table 5 in Appendix C, we also report the the *training error* from NN-based solver, which is the ℓ_∞ error on training data points. From Figure 1, one can see both DM and NN provide convergent solutions. However, when N is large enough, e.g., over 40000, the Matlab software fails to compute the pseudo-inverse. Besides, from Figure 1, we see that the NN solution produces a good generalization on the unseen points.

We compare the clock-time, RAM, and GPU memory consumption for DM and NN in Table 1. We can see that the clock-time for pseudo-inverse of DM grows rapidly with the increasing N while that of NN remains much smaller when $N < 80089$. The rapid increase of NN clock-time for the case $N = 80089$ is attributed to the time consuming of the retrievals of the submatrix $\mathbf{L}_{\text{sub}}$ from $\mathbf{L}_\epsilon$. When conducting pseudo-inverse of DM, the Matlab occupies RAM to load the full matrix and do the computation. The NN-based solver utilizes RAM to load the matrix and uses GPU memory for model training. From Table 1, we see that the NN solver has larger memory consumption than DM when N is small but the NN solver uses less memory than DM when N is large. Since we set the batch size to be the total number of training points for all cases except $N = 80089$, the memory consumption will significantly decrease if a mini-batch is used in Adam.

	N	625	1225	2500	5041	10000	19881	40000	80089
DM	clock-time (sec.)	0.09	0.40	2.37	25.98	201.20	1294.06	N/A	N/A
	RAM (G)	-	-	0.34	0.61	2.26	9.20	37.50	148.00
	GPU Mem (G)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NN	clock-time (sec.)	10.70	14.94	20.45	23.99	29.96	59.86	166.78	1831.71
	RAM (G)	1.75	1.76	1.82	1.94	2.50	5.04	14.81	53.81
	GPU Mem (G)	1.01	1.02	1.05	1.17	1.52	2.87	8.07	9.02

Table 1: The comparison of clock-time, RAM, GPU memory for DM and NN solvers. In the DM case, we also report the require RAM space, estimated by the system process-manager, to solve the problem for large N, which we did not pursue due to the excessive wall-clock time.

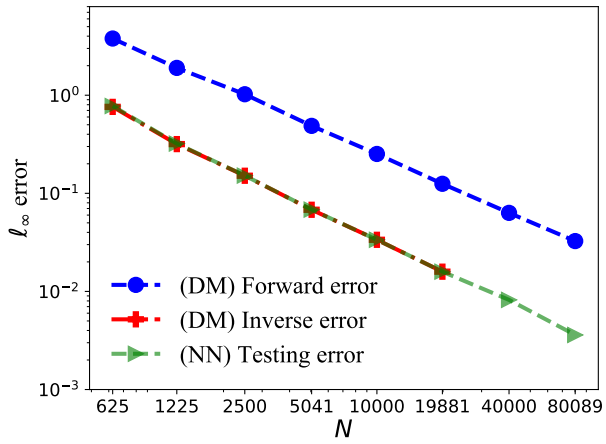

Figure 1: The comparison of the ℓ_∞ errors as functions of the number of training points N for DM and NN on a 2D torus embedded in $\mathbb{R}^3$.

5.2 Example 2: A 3D Manifold of High Co-Dimension

In this example, we consider the elliptic PDE in (1) with $a = 0, \kappa = 1$ on a closed manifold M, embedded by $\iota : M \hookrightarrow \mathbb{R}^{12}$, defined through the following embedding function,

$$\iota(t_1,t_2,t_3) := \big(\sin(t_1),\cos(t_1),\sin(2t_1),\cos(2t_1),\sin(t_2),\cos(t_2),\sin(2t_2),\cos(2t_2),\sin(t_3),\cos(t_3),\sin(2t_3),\cos(2t_3)\big),$$

for $t_1, t_2, t_3 \in [0, 2\pi)$. We manufacture the right hand data f by setting the true solution to be $u = \sin t_1 \cos t_2 \sin 2t_3$.

The training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ are generated by uniformly sampled points from the intrinsic coordinates (t_1, t_2, t_3). We solve the PDE problem by minimizing the loss function (30). The hyperparameter setting of DM and NN is presented in Table 6 in Appendix C. Figure 2 displays the error for DM and NN. We refer the readers to Table 7 for the detailed numerical values corresponding to this figure. The testing error refers to the ℓ_∞ error on 80^3 Gauss-Legendre quadrature points obtained from the intrinsic coordinates (t_1, t_2, t_3). From Figure 2, one can see that the NN solver produces convergent solutions. Besides, when N is large (e.g., $N = 4096$ or 12167), NN obtains a slightly more accurate solution than DM.

We compare the training and testing errors for different activation functions and optimizers in Table 2. The training errors of Polynomial-Sine with Adam are comparable to that of ReLU and ReLU^3 with gradient descent, but the Polynomial-Sine FNN optimized by Adam obtains the lowest testing error for most N. Therefore, we apply Polynomial-Sine and Adam in other examples.

Activation	Optimizer	N	512	1331	4096	12167	24389
Polynomial-Sine	Adam	training error	0.2665	0.1148	0.0297	0.0066	0.0023
		testing error	0.2715	0.1346	0.0302	0.0069	0.0024
ReLU^3	gradient descent	training error	0.2624	0.1137	0.0309	0.0058	0.0025
		testing error	0.2994	0.1977	0.0425	0.0147	0.0103
ReLU	gradient descent	training error	0.2637	0.1145	0.032	0.0066	0.0016
		testing error	0.4673	0.198	0.0326	0.0069	0.0016

Table 2: The comparison of the ℓ_∞ errors for different activation functions and optimizers on the 3D manifold embedded in $\mathbb{R}^{12}$.

5.3 Example 3: 2D Semi-Torus with Dirichlet Conditions

We consider solving the PDE in (1) with $a = 0$ on a two-dimensional semi-torus M with a Dirichlet boundary condition. Here, the embedding function is the same as in (29) except that the range of θ_2 is changed to $0 \le \theta_2 \le \pi$. Also, κ and the true solution u are defined as in Example 1 except for $0 \le \theta_2 \le \pi$, and the Dirichlet boundary condition is imposed by setting g to correspond to the solution u at $\theta_2 = 0, \pi$ and $0 \le \theta_1 \le 2\pi$.

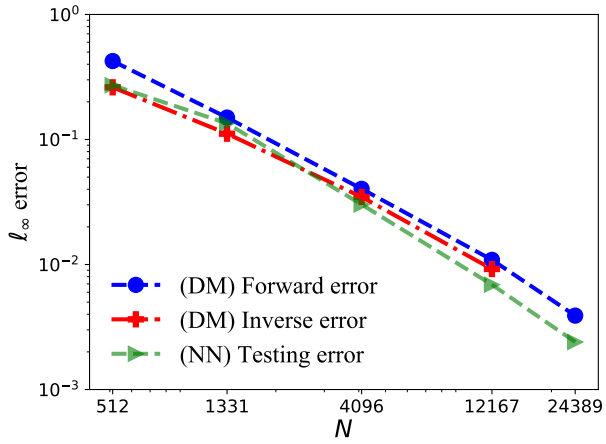

Figure 2: The comparison of the ℓ_∞ errors as functions of the number of training points N for DM and NN on the 3D manifold embedded in $\mathbb{R}^{12}$.

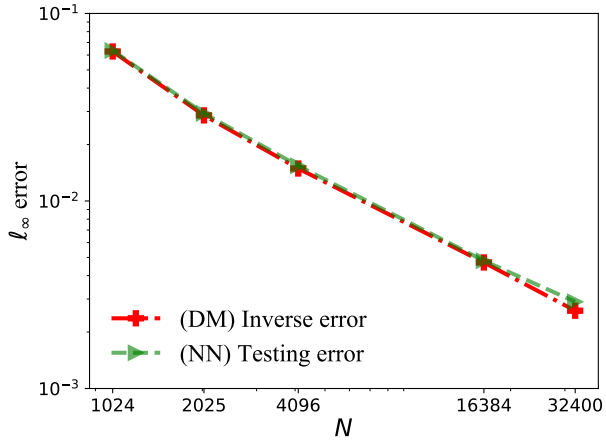

Figure 3: The comparison of the errors as functions of the number of training points for DM and NN on 2D semi-torus with Dirichlet condition.

We obtain the NN-based solution by solving the optimization problem in (10) with $\lambda = 5$ and $\mathbf{a} = 0$. The other hyperparameter setting can be found in Table 8 in Appendix C. The results (see Figure 3) show that our NN method works well on the equation with boundary condition. We refer the readers to Table 9 for the detailed numerical values corresponding to this figure.

6 Conclusion

This paper proposed a mesh-free computational framework and machine learning theory for solving PDEs on unknown manifolds given as a form of point clouds based on diffusion maps (DM) and deep learning. Parameterizing manifolds is challenging if the unknown manifold is embedded in a high-dimensional ambient Euclidean space, especially when the manifold is identified with randomly sampled data and has boundaries. First, a mesh-free DM algorithm was introduced to approximate differential operators on point clouds enabling the design of PDE solvers on manifolds with and without boundaries. Second, deep neural networks were applied to parametrize PDE solutions. Finally, we solved a least-squares minimization problem for PDEs, where the empirical loss function is evaluated using the DM discretized differential operators on point clouds and the minimizer is identified via stochastic gradient descent. The minimizer provides a PDE solution in a form of a neural network function on the whole unknown manifold with reasonably good accuracy. The mesh-free nature and randomization of the proposed solver enable efficient solutions to PDEs on manifolds arbitrary co-dimensional. New convergence and consistency

theories based on approximation and optimization analysis were developed to support the proposed framework. From the perspective of algorithm development, it is interesting to extend the proposed framework to various boundary conditions in future work. In terms of theoretical analysis, it is important to develop a generalization analysis of the proposed solver in the future and extend all of the analysis in this paper to manifolds with boundaries.

Acknowledgment

The research of J. H. was partially supported under the NSF grant DMS-1854299. S. L. and H. Y. were partially supported by the NSF grant DMS-1945029 and the NVIDIA GPU grant.

A GPDM algorithm for manifolds with boundaries

In this appendix, we give a brief overview of ghost point diffusion maps (GPDM) to construct the matrix $\mathbf{L}_\epsilon$ for manifolds with Dirichlet boundary conditions. As mentioned in [12, 32, 26, 39], the asymptotic expansion (2) in standard DM approaches is not valid near the boundary of the manifold. One way to overcome this boundary issue is the GPDM approach introduced in [39]. This approach extends the classical ghost point method [43] to solve elliptic and parabolic PDEs on unknown manifolds with boundary conditions [39, 71]. The GPDM approach can be summarized as follows (see [39, 71] for details).

1. **Estimation of normal vectors at boundary points** (see details in Section 2.2 and Appendix A of [71])**:** Assume the normal vector $\boldsymbol{\nu}$ is unknown and it will be numerically estimated. For well-sampled data, where data points are well-ordered along intrinsic coordinates, one can identify $\tilde{\boldsymbol{\nu}}$ as the tangent line approximation to $\boldsymbol{\nu}$. The error is $|\boldsymbol{\nu}-\tilde{\boldsymbol{\nu}}| = O(h)$, where the parameter h denotes the distance between consecutive ghost points (see Fig. 1(b) in [71] for a geometric illustration). For randomly sampled data, one can use the kernel method to estimate $\tilde{\boldsymbol{\nu}}$ and the error is $|\boldsymbol{\nu}-\tilde{\boldsymbol{\nu}}| = O(\sqrt{\epsilon})$ (see Fig. 1(c) in [71] for a geometric illustration and Appendix A in [71] for the detailed discussion).
2. **Specification of ghost points:** The basic idea of ghost points, as introduced in [39], is to specify the ghost points as data points that lie on the exterior normal collar, ΔM, along the boundary. Then all interior points whose distances are within ϵ^r from the boundary ∂M are at least ϵ^r away from the boundary of the extended manifold $M \cup \Delta M$. Theoretically, it was shown that, under appropriate conditions, the extended set $M \cup \Delta M$ can be isometrically embedded with an embedding function that is consistent with the embedding $M \hookrightarrow \mathbb{R}^m$ when restricted on M (see Lemma 3.5 in [39]).
   Technically, for randomly sampled data, the parameter h can be estimated by the mean distance from the boundary $\bar{\mathbf{x}}_b$ to its P (around 10 in simulations) nearest neighbors. Then, given the distance parameter h and the estimated normal vector $\tilde{\boldsymbol{\nu}}$, the approximate ghost points are given by,
   $$\tilde{\mathbf{x}}_{b,k} = \bar{\mathbf{x}}_b + kh\tilde{\boldsymbol{\nu}}, \quad \text{for } k = 1,\dots,K \text{ and } b = 1,\dots,N_b. \tag{31}$$
   In addition, one layer of interior ghost points are supplemented as $\tilde{\mathbf{x}}_{b,0} = \bar{\mathbf{x}}_b - h\tilde{\boldsymbol{\nu}}$. For well-sampled data, the interior estimated ghost point coincides with one of the interior points on the manifold when the tangent line is used. However, for randomly sampled data, the estimated interior ghost points will not necessarily coincide with an interior point (see Fig. 1 in [71] for comparison).
3. **Estimation of function values on the ghost points:**
   The main goal here is to estimate the function values $\{u(\tilde{\mathbf{x}}_{b,k})\}_{b,k=1}^{N_b,K}$ on the exterior ghost points by extrapolation, where the ghost points $\mathbf{x}_{b,k}$ lie exactly on the collar manifold ΔM (corresponding to the estimates in (31)). We assume that we are given the components of the column vector,
   $$\mathbf{u}_M := (u(\mathbf{x}_1),\dots,u(\tilde{\mathbf{x}}_{b,0}),\dots,u(\mathbf{x}_N)) \in \mathbb{R}^N. \tag{32}$$
   Here, we stress that the function values $\{u(\tilde{\mathbf{x}}_{b,0})\}_{b=1}^{N_b}$ are given exactly like the $u(\mathbf{x}_i)$ for any $\mathbf{x}_i \in M$, even when the ghost points $\tilde{\mathbf{x}}_{b,0}$ do not lie on the manifold M. Then we will use the components of the column vector,
   $$\mathbf{U}_G := (U_{1,1},\dots,U_{N_b,K}) \in \mathbb{R}^{N_bK}, \tag{33}$$
   to estimate the components of function values on ghost points, $\mathbf{U}_G := (u(\tilde{\mathbf{x}}_{1,1}),\dots,u(\tilde{\mathbf{x}}_{N_b,K})) \in \mathbb{R}^{N_bK}$. Numerically, we will obtain the components of $\mathbf{U}_G$ by solving the following linear algebraic equations for each $b = 1,\dots,N_b$,
   $$\begin{aligned} U_{b,1} - 2u(\mathbf{x}_b) + u(\tilde{\mathbf{x}}_{b,0}) &= 0, \\ U_{b,2} - 2U_{b,1} + u(\mathbf{x}_b) &= 0, \\ U_{b,k} - 2U_{b,k-1} + U_{b,k-2} &= 0, \qquad k = 3,\dots K. \end{aligned} \tag{34}$$

These algebraic equations are discrete analogs of matching the first-order derivatives along the estimated normal direction, $\tilde{\boldsymbol{\nu}}$.

4. **Construction of the GPDM estimator:** We now define the GPDM estimator for the differential operator $\mathscr{L}$ in (1). The discrete estimator will be constructed based on the available training data $\{\mathbf{x}_i \in M\}_{i=1}^N$ and the estimated ghost points, $\{\tilde{\mathbf{x}}_{b,k}\}_{b,k=1,0}^{N_b,K}$. In particular, since the interior ghost points, $\{\tilde{\mathbf{x}}_{b,0}\}_{b=1}^{N_b}$ may or may not coincide with any interior points on the manifold, we assume that $X^h := \{\mathbf{x}_1, \ldots, \tilde{\mathbf{x}}_{b,0}, \ldots, \mathbf{x}_N\}$ has N components that include the estimated ghost points in the following discussion. With this notation, we define a non-square matrix,
$$\mathbf{L}^h := (\mathbf{L}^{(1)}, \mathbf{L}^{(2)}) \in \mathbb{R}^{N\times(N+N_bK)}, \tag{35}$$
constructed as in (5), by evaluating the kernel on components of X^h for each row and the components of $X^h \cup \{\tilde{\mathbf{x}}_{b,k}\}_{b,k=1}^{N_b,K}$ for each column. With these definitions and those in (32)-(34), we note that
$$\mathbf{L}^h\mathbf{U} = \mathbf{L}^{(1)}\mathbf{u}_M + \mathbf{L}^{(2)}\mathbf{U}_G = (\mathbf{L}^{(1)} + \mathbf{L}^{(2)}\mathbf{G})\mathbf{u}_M = \tilde{\mathbf{L}}\mathbf{u}_M,$$
where $\mathbf{U} := (\mathbf{u}_M, \mathbf{U}_G) \in \mathbb{R}^{N+N_bK}$ and $\mathbf{G} \in \mathbb{R}^{N_bK\times N}$ is defined as a solution operator to (34), which is given in a compact form as $\mathbf{U}_G = \mathbf{G}\mathbf{u}_M$. Then, the GPDM estimator $\tilde{\mathbf{L}}$ is defined as an $N \times N$ matrix,
$$\tilde{\mathbf{L}} := \mathbf{L}^{(1)} + \mathbf{L}^{(2)}\mathbf{G}. \tag{36}$$
Note that we have the consistency of GPDM estimator for the differential operator defined on functions that take values on the extended $M \cup \Delta M$ (see Lemma 2.3 in [71]).

5. **Combination with the discretization of the boundary conditions:** We take $\mathbf{L}_c \in \mathbb{R}^{(N-N_b)\times N}$ to be a sub-matrix of $\tilde{\mathbf{L}} \in \mathbb{R}^{N\times N}$ in (36), where the $N - N_b$ rows of $\mathbf{L}_c$ correspond to the interior points from X^h. There are $N - N_b$ equations from the linear system $(-\mathbf{a} + \mathbf{L}_c)\mathbf{u}_M = \mathbf{f}$. To close the discretized problem, we use the N_b equations from the Dirichlet boundary condition at the boundary points, $u(\bar{\mathbf{x}}_b) = g(\bar{\mathbf{x}}_b)$ for $\bar{\mathbf{x}}_b \in \{\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_{N_b}\} \subset X^h \cap \partial M$. With the construction of $\mathbf{L}_c$ and the Dirichlet boundary conditions, we alternatively solve the system in (10) using DNN approach.

B Global convergence analysis of neural network optimization

In this section, we will prove several lemmas in preparation for the proof of Theorem 4.2. The proof of Theorem 4.2 will be presented after these lemmas.

The Rademacher complexity is a basic tool for generalization analysis. In our analysis, we will use several important lemmas and theorems related to it. To be self-contained, they are listed as follows.

Definition B.1 (The Rademacher complexity of a function class $\mathscr{F}$)**.** *Given a sample set $S = \{z_1, \ldots, z_N\}$ on a domain $\mathscr{Z}$, and a class $\mathscr{F}$ of real-valued functions defined on $\mathscr{Z}$, the empirical Rademacher complexity of $\mathscr{F}$ on S is defined as*
$$\mathrm{Rad}_S(\mathscr{F}) = \frac{1}{N}\mathbb{E}_{\boldsymbol{\tau}}\left[\sup_{f\in\mathscr{F}}\sum_{i=1}^N \tau_i f(z_i)\right],$$
where $\tau_1, \ldots, \tau_N$ are independent random variables drawn from the Rademacher distribution, i.e., $\mathbb{P}(\tau_i = +1) = \mathbb{P}(\tau_i = -1) = \frac{1}{2}$ for $i = 1, \ldots, N$.

First, we recall a well-known contraction lemma for the Rademacher complexity.

Lemma B.1 (Contraction lemma [62])**.** *Suppose that $\psi_i : \mathbb{R} \to \mathbb{R}$ is a C_{L}-Lipschitz function for each $i \in [N] := \{1, \ldots, N\}$. For any $\boldsymbol{y} \in \mathbb{R}^N$, let $\boldsymbol{\psi}(\boldsymbol{y}) = (\psi_1(y_1), \cdots, \psi_N(y_N))^\intercal$. For an arbitrary set of functions $\mathscr{F}$ on an arbitrary domain $\mathscr{Z}$ and an arbitrary choice of samples $S = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\} \subset \mathscr{Z}$, we have*
$$\mathrm{Rad}_S(\boldsymbol{\psi} \circ \mathscr{F}) \le C_{\mathrm{L}}\mathrm{Rad}_S(\mathscr{F}).$$

Second, the Rademacher complexity of linear predictors can be characterized by the lemma below.

Lemma B.2 (Rademacher complexity for linear predictors [62])**.** *Let $\Theta = \{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_m\} \in \mathbb{R}^n$. Let $\mathscr{G} = \{g(\boldsymbol{w}) = \boldsymbol{w}^\intercal\boldsymbol{x} : \|\boldsymbol{x}\|_1 \le 1\}$ be the linear function class with parameter $\boldsymbol{x}$ whose ℓ^1 norm is bounded by 1. Then*
$$\mathrm{Rad}_\Theta(\mathscr{G}) \le \max_{1\le k\le m}\|\boldsymbol{w}_k\|_\infty\sqrt{\frac{2\log(2n)}{m}}.$$

Finally, let us state a general theorem concerning the Rademacher complexity and generalization gap of an arbitrary set of functions $\mathscr{F}$ on an arbitrary domain $\mathscr{Z}$, which is essentially given in [62].

Theorem B.1 (Rademacher complexity and generalization gap [62])**.** *Suppose that f's in $\mathscr{F}$ are non-negative and uniformly bounded, i.e., for any $f \in \mathscr{F}$ and any $\boldsymbol{z} \in \mathscr{Z}$, $0 \le f(\boldsymbol{z}) \le B$. Then for any $\delta \in (0,1)$, with probability at least $1-\delta$ over the choice of N i.i.d. random samples $S = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\} \subset \mathscr{Z}$, we have*

$$\sup_{f\in\mathscr{F}} \left| \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{z}_i) - \mathbb{E}_{\boldsymbol{z}} f(\boldsymbol{z}) \right| \le 2\mathbb{E}_S \mathrm{Rad}_S(\mathscr{F}) + B\sqrt{\frac{\log(2/\delta)}{2N}},$$
$$\sup_{f\in\mathscr{F}} \left| \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{z}_i) - \mathbb{E}_{\boldsymbol{z}} f(\boldsymbol{z}) \right| \le 2\mathrm{Rad}_S(\mathscr{F}) + 3B\sqrt{\frac{\log(4/\delta)}{2N}}.$$

Here, we should point out that the distribution of $\boldsymbol{z}$ is arbitrary. In our specific application, $\mathbb{E}_{\boldsymbol{z}} = \mathbb{E}_{\pi}$.

Now we are going to prove several lemmas for Theorem 4.2. In the analysis below, we use $\bar{a}_k^t := \bar{a}_k(t) := \gamma^{-1} a_k(t)$ with $0 < \gamma < 1$, e.g., $\gamma = \frac{1}{\sqrt{m}}$ or $\gamma = \frac{1}{m}$, and $\bar{\boldsymbol{\theta}}(t) := \mathrm{vec}\{\bar{a}_k^t, \boldsymbol{w}_k^t\}_{k=1}^m$.

Lemma B.3. *For any $\delta \in (0,1)$ with probability at least $1-\delta$ over the random initialization in* (22)*, we have*

$$\begin{aligned} \max_{k\in[m]} \{|\bar{a}_k^0|, \|\boldsymbol{w}_k^0\|_\infty\} &\le \sqrt{2\log\frac{2m(n+1)}{\delta}}, \\ \max_{k\in[m]} \{|a_k^0|\} &\le \gamma\sqrt{2\log\frac{2m(n+1)}{\delta}}. \end{aligned} \tag{37}$$

Proof. If $\mathrm{X} \sim \mathscr{N}(0,1)$, then $\mathbb{P}(|\mathrm{X}| > \varepsilon) \le 2\mathbb{E}^{-\frac{1}{2}\varepsilon^2}$ for all $\varepsilon > 0$. Since $\bar{a}_k^0 \sim \mathscr{N}(0,1)$, $(\boldsymbol{w}_k^0)_\alpha \sim \mathscr{N}(0,1)$ for $k \in [m], \alpha \in [n]$, and they are all independent, by setting

$$\varepsilon = \sqrt{2\log\frac{2m(n+1)}{\delta}},$$

one can obtain

$$\begin{aligned} \mathbb{P}\left(\max_{k\in[m]} \{|\bar{a}_k^0|, \|\boldsymbol{w}_k^0\|_\infty\} > \varepsilon\right) &= \mathbb{P}\left(\left(\bigcup_{k\in[m]} \{|\bar{a}_k^0| > \varepsilon\}\right) \bigcup \left(\bigcup_{k\in[m],\alpha\in[n]} \{|(\boldsymbol{w}_k^0)_\alpha| > \varepsilon\}\right)\right) \\ &\le \sum_{k=1}^{m} \mathbb{P}\left(|\bar{a}_k^0| > \varepsilon\right) + \sum_{k=1}^{m}\sum_{\alpha=1}^{n} \mathbb{P}\left(|(\boldsymbol{w}_k^0)_\alpha| > \varepsilon\right) \\ &\le 2m\mathrm{e}^{-\frac{1}{2}\varepsilon^2} + 2mn\mathrm{e}^{-\frac{1}{2}\varepsilon^2} \\ &= 2m(n+1)\mathrm{e}^{-\frac{1}{2}\varepsilon^2} \\ &= \delta, \end{aligned}$$

which implies the conclusions of this lemma. □

Lemma B.4. *For any $\delta \in (0,1)$ with probability at least $1-\delta$ over the random initialization in* (22)*, we have*

$$R_S(\boldsymbol{\theta}^0) \le \frac{1}{2}\left(1 + 3\gamma n^r\sqrt{m}\|\boldsymbol{A}\|_2\left(2\log\frac{4m(n+1)}{\delta}\right)^{(r+1)/2}\left(r\sqrt{2\log(2n)} + \sqrt{\log(8/\delta)/2}\right)\right)^2,$$

Proof. From Lemma B.3 we know that with probability at least $1-\delta/2$,

$$|\bar{a}_k^0| \le \sqrt{2\log\frac{4m(n+1)}{\delta}} \quad \text{and} \quad \|\boldsymbol{w}_k^0\|_1 \le n\sqrt{2\log\frac{4m(n+1)}{\delta}}.$$

Let

$$\mathscr{H} = \{h(\bar{a}, \boldsymbol{w}; \boldsymbol{x}) \mid h(\bar{a}, \boldsymbol{w}; \boldsymbol{x}) = \bar{a}\sigma(\boldsymbol{w}^\top \boldsymbol{x}), \boldsymbol{x} \in \Omega\}.$$

Each element in the above set is a function of $\bar{a}$ and $\boldsymbol{w}$ while $\boldsymbol{x} \in [0,1]^n$ is a parameter. Since $\|\boldsymbol{x}\|_\infty \le 1$, we have

$$|h(\bar{a}_k^0, \boldsymbol{w}_k^0; \boldsymbol{x})| \le |\bar{a}_k^0|\|\boldsymbol{w}_k^0\|_1^r \le n^r\left(2\log\frac{4m(n+1)}{\delta}\right)^{(r+1)/2}.$$

Then with probability at least $1-\delta/2$, by the Rademacher-based uniform convergence theorem, we have

$$\begin{aligned}\frac{1}{\gamma m}\sup_{\boldsymbol{x}\in\Omega}|\phi(\boldsymbol{x};\boldsymbol{\theta}^0)| &= \sup_{\boldsymbol{x}\in\Omega}\left|\frac{1}{m}\sum_{k=1}^m h(\bar{a}_k^0,\boldsymbol{w}_k^0;\boldsymbol{x}) - \mathbb{E}_{(\bar{a},\boldsymbol{w})\sim\mathcal{N}(0,\boldsymbol{I}_{n+1})}h(\bar{a},\boldsymbol{w};\boldsymbol{x})\right| \\ &\le 2\mathrm{Rad}_{\bar{\boldsymbol{\theta}}^0}(\mathscr{H}) + 3n^r\left(2\log\frac{4m(n+1)}{\delta}\right)^{(r+1)/2}\sqrt{\frac{\log(8/\delta)}{2m}},\end{aligned}$$

where

$$\mathrm{Rad}_{\bar{\boldsymbol{\theta}}^0}(\mathscr{H}) := \frac{1}{m}\mathbb{E}_{\boldsymbol{\tau}}\left[\sup_{\boldsymbol{x}\in\Omega}\sum_{k=1}^m \tau_k h(\bar{a}_k^0,\boldsymbol{w}_k^0;\boldsymbol{x})\right] = \frac{1}{m}\mathbb{E}_{\boldsymbol{\tau}}\left[\sup_{\boldsymbol{x}\in\Omega}\sum_{k=1}^m \tau_k \bar{a}_k^0\sigma(\boldsymbol{w}_k^{0\intercal}\boldsymbol{x})\right],$$

where $\boldsymbol{\tau}$ is a random vector in $\mathbb{N}^m$ with i.i.d. entries $\{\tau_k\}_{k=1}^m$ following the Rademacher distribution. With probability at least $1-\delta/2$, $\psi_k(y_k) = \bar{a}_k\sigma(y_k)$ for $k\in[m]$ is a Lipschitz continuous function with a Lipschitz constant

$$r n^{r-1}\left(2\log\frac{4m(n+1)}{\delta}\right)^{r/2}$$

when $y_k \in [-n\sqrt{2\log(4m(n+1)/\delta)}, n\sqrt{2\log(4m(n+1)/\delta)}]$. We continuously extend $\psi_k(y_k)$ to the domain $\mathbb{R}$ with the same Lipschitz constant.

Applying Lemma B.1 with $\psi_k(y_k)$, we have

$$\begin{aligned}\frac{1}{m}\mathbb{E}_{\boldsymbol{\tau}}\left[\sup_{\boldsymbol{x}\in\Omega}\sum_{k=1}^m \tau_k \bar{a}_k^0\sigma(\boldsymbol{w}_k^{0\intercal}\boldsymbol{x})\right] &\le \frac{1}{m}r n^{r-1}\left(2\log\frac{4m(n+1)}{\delta}\right)^{r/2}\mathbb{E}_{\boldsymbol{\tau}}\left[\sup_{\boldsymbol{x}\in\Omega}\sum_{k=1}^m \tau_k \boldsymbol{w}_k^{0\intercal}\boldsymbol{x}\right] \\ &\le \frac{r n^r\sqrt{2\log(2n)}}{\sqrt{m}}\left(2\log\frac{4m(n+1)}{\delta}\right)^{(r+1)/2}, \qquad (38)\end{aligned}$$

where the second inequality is by the Rademacher bound for linear predictors in Lemma B.2.

So one can get

$$\begin{aligned}\sup_{\boldsymbol{x}\in\Omega}|\phi(\boldsymbol{x};\boldsymbol{\theta}^0)| &\le \gamma\sqrt{m}n^r\left(2\log\frac{4m(n+1)}{\delta}\right)^{(r+1)/2}\left(2r\sqrt{2\log(2n)}+3\sqrt{\log(8/\delta)/2}\right) \\ &\le 3\gamma\sqrt{m}n^r\left(2\log\frac{4m(n+1)}{\delta}\right)^{(r+1)/2}\left(r\sqrt{2\log(2n)}+\sqrt{\log(8/\delta)/2}\right).\end{aligned}$$

Then

$$\begin{aligned}R_S(\boldsymbol{\theta}^0) &\le \frac{1}{2N}\left(\|\boldsymbol{A}\|_2\|\phi(S;\boldsymbol{\alpha}^0)\|_2+\sqrt{N}\right)^2 \\ &\le \frac{1}{2}\left(1+3\gamma\sqrt{m}n^r\|\boldsymbol{A}\|_2\left(2\log\frac{4m(n+1)}{\delta}\right)^{(r+1)/2}\left(r\sqrt{2\log(2n)}+\sqrt{\log(8/\delta)/2}\right)\right)^2,\end{aligned}$$

where the first inequality comes from the fact that $|f|\le 1$ by our assumption of the target function. □

The following lemma shows the positive definiteness of $\boldsymbol{G}^{(a)}$ at initialization.

Lemma B.5. *For any $\delta\in(0,1)$, if $m \ge \frac{16N^4C_n}{\lambda_S^2\delta}$, then with probability at least $1-\delta$ over the random initialization in* (22)*, we have*

$$\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0)\right) \ge \frac{3}{4}\lambda_S,$$

where $C_n := \mathbb{E}\|\boldsymbol{w}\|_1^{4r} < +\infty$ with $\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I}_n)$.

Proof. We define $\Omega_{ij} := \{\boldsymbol{\theta}^0 \mid |\boldsymbol{G}_{ij}^{(a)}(\boldsymbol{\theta}^0) - \boldsymbol{K}_{ij}^{(a)}| \le \frac{\lambda_S}{4N}\}$. Note that

$$|g^{(a)}(\boldsymbol{w}_k^0;\boldsymbol{x}_i,\boldsymbol{x}_j)| \le \|\boldsymbol{w}_k^0\|_1^{2r}.$$

So

$$\mathrm{Var}\left(g^{(a)}(\boldsymbol{w}_k^0;\boldsymbol{x}_i,\boldsymbol{x}_j)\right) \le \mathbb{E}\left(g^{(a)}(\boldsymbol{w}_k^0;\boldsymbol{x}_i,\boldsymbol{x}_j)\right)^2 \le \mathbb{E}\|\boldsymbol{w}_k^0\|_1^{4r} = C_n,$$

and

$$\mathrm{Var}\left(\boldsymbol{G}_{ij}^{(a)}(\boldsymbol{\theta}^0)\right) = \frac{1}{m^2}\sum_{k=1}^{m}\mathrm{Var}\left(g^{(a)}(\boldsymbol{w}_k^0;\boldsymbol{x}_i,\boldsymbol{x}_j)\right) \le \frac{C_n}{m}.$$

Then the probability of the event Ω_{ij} has the lower bound:

$$\mathbb{P}(\Omega_{ij}) \ge 1 - \frac{\mathrm{Var}\left(\boldsymbol{G}_{ij}^{(a)}(\boldsymbol{\theta}^0)\right)}{[\lambda_S/(4N)]^2} \ge 1 - \frac{16N^2C_n}{\lambda_S^2 m}.$$

Thus, with probability at least $\left(1-\frac{16N^2C_n}{\lambda_S^2 m}\right)^{N^2} \ge 1 - \frac{16N^4C_n}{\lambda_S^2 m}$, we have all events Ω_{ij} for $i,j\in[N]$ to occur. This implies that with probability at least $1-\frac{16N^4C_n}{\lambda_S^2 m}$, we have

$$\|\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0) - \boldsymbol{K}^{(a)}\|_{\mathrm{F}} \le \frac{\lambda_S}{4}.$$

Note that $\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0)$ and $\boldsymbol{K}^{(a)}$ are positive semi-definite normal matrices. Let $\boldsymbol{v}$ be the singular vector of $\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0)$ corresponding to the smallest singular value, then

$$\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0)\right) = \boldsymbol{v}^\intercal \boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0)\boldsymbol{v} = \boldsymbol{v}^\intercal \boldsymbol{K}^{(a)}\boldsymbol{v} + \boldsymbol{v}^\intercal(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0) - \boldsymbol{K}^{(a)})\boldsymbol{v} \ge \lambda_S - \|\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0) - \boldsymbol{K}^{(a)}\|_2 \ge \lambda_S - \|\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0) - \boldsymbol{K}^{(a)}\|_{\mathrm{F}}.$$

So

$$\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0)\right) \ge \lambda_S - \|\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0) - \boldsymbol{K}^{(a)}\|_{\mathrm{F}} \ge \frac{3}{4}\lambda_S.$$

For any $\delta\in(0,1)$, if $m \ge \frac{16N^4C_n}{\lambda_S^2\delta}$, then with probability at least $1-\frac{16N^4C_n}{\lambda_S^2 m} \ge 1-\delta$ over the initialization $\boldsymbol{\theta}^0$, we have $\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0)\right) \ge \frac{3}{4}\lambda_S$. □

The following lemma estimates the empirical loss dynamics before the stopping time t^* in (27).

Lemma B.6. *For any $\delta\in(0,1)$, if $m \ge \frac{16N^4C_n}{\lambda_S^2\delta}$, then with probability at least $1-\delta$ over the random initialization in* (22)*, we have for any $t\in[0,t^*)$*

$$R_S(\boldsymbol{\theta}(t)) \le \exp\left(-\frac{m\lambda_S\lambda_{\boldsymbol{A}}t}{N}\right)R_S(\boldsymbol{\theta}^0).$$

Proof. From Lemma B.5, for any $\delta\in(0,1)$ with probability at least $1-\delta$ over initialization $\boldsymbol{\theta}^0$ and for any $t\in[0,t^*)$ with t^* defined in (27), we have $\boldsymbol{\theta}(t)\in\mathscr{M}(\boldsymbol{\theta}^0)$ defined in (28) and

$$\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta})\right) \ge \lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0)\right) - \|\boldsymbol{G}^{(a)}(\boldsymbol{\theta}) - \boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0)\|_{\mathrm{F}} \ge \frac{3}{4}\lambda_S - \frac{1}{4}\lambda_S = \frac{1}{2}\lambda_S.$$

Note that $\boldsymbol{G}_{ij} = \frac{1}{m}\nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{x}_i;\boldsymbol{\theta})\cdot\nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{x}_j;\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}}R_S = \frac{1}{N}\nabla_{\boldsymbol{\theta}}\phi(S;\boldsymbol{\theta})\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{e}$, so

$$\|\nabla_{\boldsymbol{\theta}}R_S(\boldsymbol{\theta}(t))\|_2^2 = \frac{m}{N^2}\boldsymbol{e}^\intercal\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{G}(\boldsymbol{\theta}(t))\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{e} \ge \frac{m}{N^2}\boldsymbol{e}^\intercal\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(t))\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{e},$$

where the last equation is true by the fact that $G^{(w)}(\boldsymbol{\theta}(t))$ is a Gram matrix and hence positive semi-definite. Together with

$$\begin{aligned}\frac{m}{N^2}\boldsymbol{e}^\intercal\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(t))\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{e} &\ge \frac{m}{N^2}\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(t))\right)\boldsymbol{e}^\intercal\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{e}.\\ &\ge \frac{2m}{N}\lambda_{\min}\left(\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(t))\right)\lambda_{\min}\left(\boldsymbol{A}\boldsymbol{A}^\intercal\right)R_S(\boldsymbol{\theta}(t))\\ &\ge \frac{m}{N}\lambda_S\lambda_{\boldsymbol{A}}R_S(\boldsymbol{\theta}(t)),\end{aligned}$$

then finally we get

$$\frac{\mathrm{d}}{\mathrm{d}t}R_S(\boldsymbol{\theta}(t)) = -\|\nabla_{\boldsymbol{\theta}}R_S(\boldsymbol{\theta}(t))\|_2^2 \le -\frac{m}{N}\lambda_S\lambda_{\boldsymbol{A}}R_S(\boldsymbol{\theta}(t)).$$

Integrating the above equation yields the conclusion in this lemma. □

The following lemma shows that the parameters in the two-layer neural network are uniformly bounded in time during the training before time t^* as defined in (27)-(28).

Lemma B.7. *For any $\delta \in (0,1)$, if*

$$m \geq \max\left\{ \frac{32N^4 C_n}{\lambda_S^2 \delta}, \frac{5\max\{n,r\}N\sqrt{2\lambda_{\boldsymbol{A},N}R_S(\boldsymbol{\theta}^0)}}{\lambda_S \lambda_{\boldsymbol{A}}} \left(2n\sqrt{2\log\frac{4m(n+1)}{\delta}}\right)^{r-1} \right\},$$

then with probability at least $1-\delta$ over the random initialization in (22)*, for any $t \in [0, t^*)$ and any $k \in [m]$,*

$$\begin{aligned} &|a_k(t) - a_k(0)| \leq q, \quad \|\boldsymbol{w}_k(t) - \boldsymbol{w}_k(0)\|_\infty \leq q, \\ &|a_k(0)| \leq \gamma\eta, \qquad\qquad \|\boldsymbol{w}_k(0)\|_\infty \leq \eta, \end{aligned}$$

where

$$q := \left(2n\sqrt{2\log\frac{4m(n+1)}{\delta}}\right)^r \frac{2\max\{n,r\}N\sqrt{2\lambda_{\boldsymbol{A},N}R_S(\boldsymbol{\theta}^0)}}{nm\lambda_S\lambda_{\boldsymbol{A}}}$$

and

$$\eta := \sqrt{2\log\frac{4m(n+1)}{\delta}}.$$

Proof. Let $\xi(t) = \max\limits_{k\in[m], s\in[0,t]} \{|a_k(s)|, \|\boldsymbol{w}_k(s)\|_\infty\}$. Note that

$$\begin{aligned} |\nabla_{a_k} R_S(\boldsymbol{\theta})|^2 &= \left[\frac{1}{N}\sum_{i=1}^N (e^\intercal \boldsymbol{A}^\intercal \boldsymbol{A})_i \sigma(\boldsymbol{w}_k^\intercal \boldsymbol{x}_i)\right]^2 \\ &\leq \|\boldsymbol{w}_k\|_1^{2r} \left[\frac{1}{N}\sum_{i=1}^N (|e^\intercal \boldsymbol{A}^\intercal \boldsymbol{A}|)_i\right]^2 \\ &\leq 2\|\boldsymbol{w}_k\|_1^{2r} \lambda_{\boldsymbol{A},N} R_S(\boldsymbol{\theta}) \\ &\leq 2n^{2r}(\xi(t))^{2r}\lambda_{\boldsymbol{A},N}R_S(\boldsymbol{\theta}), \end{aligned}$$

where $\lambda_{\boldsymbol{A},N}$ denotes the largest eigenvalue of $\boldsymbol{A}$, and

$$\begin{aligned} \|\nabla_{\boldsymbol{w}_k} R_S(\boldsymbol{\theta})\|_\infty^2 &= \left\| \frac{1}{N}\sum_{i=1}^N (e^\intercal \boldsymbol{A}^\intercal \boldsymbol{A})_i a_k \sigma'(\boldsymbol{w}_k^\intercal \boldsymbol{x}_i)\boldsymbol{x}_i \right\|_\infty^2 \\ &\leq |a_k|^2 r^2 \|\boldsymbol{w}_k\|_1^{2(r-1)} \left\| \frac{1}{N}\sum_{i=1}^N (e^\intercal \boldsymbol{A}^\intercal \boldsymbol{A})_i \boldsymbol{x}_i \right\|_\infty^2 \\ &\leq |a_k|^2 r^2 \|\boldsymbol{w}_k\|_1^{2(r-1)} \left| \frac{1}{N}\sum_{i=1}^N (|e^\intercal \boldsymbol{A}^\intercal \boldsymbol{A}|)_i \right|^2 \\ &\leq 2|a_k|^2 r^2 \|\boldsymbol{w}_k\|_1^{2(r-1)} \lambda_{\boldsymbol{A},N} R_S(\boldsymbol{\theta}) \\ &\leq 2r^2 n^{2(r-1)} (\xi(t))^{2r} \lambda_{\boldsymbol{A},N} R_S(\boldsymbol{\theta}). \end{aligned}$$

From Lemma B.6, if $m \geq \frac{32N^4C_n}{\lambda_S^2\delta}$, then with probability at least $1-\delta/2$ over random initialization, one can represent (24) in an integral form and obtain,

$$\begin{aligned} |a_k(t) - a_k(0)| &\leq \int_0^t |\nabla_{a_k} R_S(\boldsymbol{\theta}(s))| \,\mathrm{d}s \\ &\leq \sqrt{2\lambda_{\boldsymbol{A},N}} n^r (\xi(t))^r \int_0^t \sqrt{R_S(\boldsymbol{\theta}(s))}\,\mathrm{d}s \\ &\leq \sqrt{2\lambda_{\boldsymbol{A},N}} n^r (\xi(t))^r \int_0^t \sqrt{R_S(\boldsymbol{\theta}^0)} \exp\left(-\frac{m\lambda_S\lambda_{\boldsymbol{A}} s}{2N}\right)\mathrm{d}s \\ &\leq \frac{2\sqrt{2\lambda_{\boldsymbol{A},N}} n^r N \sqrt{R_S(\boldsymbol{\theta}^0)}}{m\lambda_S\lambda_{\boldsymbol{A}}} (\xi(t))^r \\ &\leq p(\xi(t))^r, \end{aligned}$$

where $p := \frac{2\sqrt{2\lambda_{\boldsymbol{A},N}}\max\{n,r\}n^{r-1}N\sqrt{R_S(\boldsymbol{\theta}^0)}}{m\lambda_S\lambda_{\boldsymbol{A}}}$.

$$\begin{aligned}
\|\boldsymbol{w}_k(t)-\boldsymbol{w}_k(0)\|_\infty &\le \int_0^t \|\nabla_{\boldsymbol{w}_k} R_S(\boldsymbol{\theta}(s))\|_\infty \,\mathrm{d}s \\
&\le \sqrt{2\lambda_{\boldsymbol{A},N}}\, r n^{r-1}(\xi(t))^r \int_0^t \sqrt{R_S(\boldsymbol{\theta}(s))}\,\mathrm{d}s \\
&\le \sqrt{2\lambda_{\boldsymbol{A},N}}\, r n^{r-1}(\xi(t))^r \int_0^t \sqrt{R_S(\boldsymbol{\theta}^0)}\exp\left(-\frac{m\lambda_S\lambda_{\boldsymbol{A}}s}{2N}\right)\mathrm{d}s \\
&\le \frac{2\sqrt{2\lambda_{\boldsymbol{A},N}}Nr n^{r-1}\sqrt{R_S(\boldsymbol{\theta}^0)}}{m\lambda_S\lambda_{\boldsymbol{A}}}(\xi(t))^r \\
&\le p(\xi(t))^r.
\end{aligned}$$

We should point out that the above inequalities hold for all $t \in (0, t^*)$ since the upper bounds are based on Lemma B.6, thus

$$\xi(t) \le \xi(0) + p(\xi(t))^r, \tag{39}$$

for all $t \in (0, t^*)$. From Lemma B.3 with probability at least $1-\delta/2$,

$$\xi(0) = \max_{k\in[m]}\{|a_k(0)|, \|\boldsymbol{w}_k(0)\|_\infty\} \le \max\left\{\gamma\sqrt{2\log\frac{4m(n+1)}{\delta}}, \sqrt{2\log\frac{4m(n+1)}{\delta}}\right\} \le \sqrt{2\log\frac{4m(n+1)}{\delta}} = \eta. \tag{40}$$

Since

$$m \ge \frac{5\max\{n,r\}N\sqrt{2\lambda_{\boldsymbol{A},N}R_S(\boldsymbol{\theta}^0)}}{\lambda_S\lambda_{\boldsymbol{A}}}\left(2n\sqrt{2\log\frac{4m(n+1)}{\delta}}\right)^{r-1} = \frac{5}{2}mp(2\eta)^{r-1},$$

then $p(2\eta)^{r-1} \le \frac{2}{5}$. Let

$$t_0 := \inf\{t \mid \xi(t) > 2\eta\}.$$

Then t_0 is the first time for the magnitude of a NN parameter exceeding 2η. Recall that t^*, introduced in (27), denotes the first time for $\|\boldsymbol{G}^{(a)}(\boldsymbol{\theta}) - \boldsymbol{G}^{(a)}(\boldsymbol{\theta}^0)\|_{\mathrm{F}} > \frac{1}{4}\lambda_S$. We will prove $t_0 \ge t^*$, i.e., we show that, as long as the kernel $\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(t))$ introduced by the gradient descent method is well controlled around its initialization $\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(0))$, all network parameters have a well-controlled magnitude in the sense that there is no parameter with a magnitude larger than 2η. We will prove $t_0 \ge t^*$ by contradiction. Suppose that $t_0 < t^*$. For $t \in [0, t_0)$, by (39), (40), and $\xi(t) \le 2\eta$, we have

$$\xi(t) \le \eta + p(2\eta)^{r-1}\xi(t) \le \eta + \frac{2}{5}\xi(t),$$

then

$$\xi(t) \le \frac{5}{3}\eta.$$

After letting $t \to t_0$, the inequality just above contradicts with the definition of t_0. So $t_0 \ge t^*$ and then $\xi(t) \le 2\eta$ for all $t \in [0, t^*)$. Thus

$$\begin{aligned}
|a_k(t) - a_k(0)| &\le (2\eta)^r p \\
\|\boldsymbol{w}_k(t) - \boldsymbol{w}_k(0)\|_\infty &\le (2\eta)^r p.
\end{aligned}$$

Finally, notice that

$$(2\eta)^r p = \left(2n\sqrt{2\log\frac{4m(n+1)}{\delta}}\right)^r \frac{2\max\{n,r\}N\sqrt{2\lambda_{\boldsymbol{A},N}R_S(\boldsymbol{\theta}^0)}}{nm\lambda_S\lambda_{\boldsymbol{A}}} = q, \tag{41}$$

which ends the proof. □

Now we are ready to prove Theorem 4.2.

Proof of Theorem 4.2. From Lemma B.6, it is sufficient to prove that the stopping time t^* in Lemma B.6 is equal to $+\infty$. We will prove this by contradiction.

Suppose $t^* < +\infty$. Note that

$$|\boldsymbol{G}^{(a)}_{ij}(\boldsymbol{\theta}(t^*)) - \boldsymbol{G}^{(a)}_{ij}(\boldsymbol{\theta}(0))| \le \frac{1}{m}\sum_{k=1}^m |g^{(a)}(\boldsymbol{w}_k(t^*); \boldsymbol{x}_i, \boldsymbol{x}_j) - g^{(a)}(\boldsymbol{w}_k(0); \boldsymbol{x}_i, \boldsymbol{x}_j)|. \tag{42}$$

By the mean value theorem,

$$|g^{(a)}(\boldsymbol{w}_k(t^*);\boldsymbol{x}_i,\boldsymbol{x}_j)-g^{(a)}(\boldsymbol{w}_k(0);\boldsymbol{x}_i,\boldsymbol{x}_j)|\le\|\nabla_{\boldsymbol{w}}g^{(a)}\left(c\boldsymbol{w}_k(t^*)+(1-c)\boldsymbol{w}_k(0);\boldsymbol{x}_i,\boldsymbol{x}_j\right)\|_\infty\|\boldsymbol{w}_k(t^*)-\boldsymbol{w}_k(0)\|_1$$

for some $c\in(0,1)$. Further computation yields

$$\nabla_{\boldsymbol{w}}g^{(a)}(\boldsymbol{w};\boldsymbol{x}_i,\boldsymbol{x}_j)=\left[\sigma'(\boldsymbol{w}^\intercal\boldsymbol{x}_i)\boldsymbol{x}_i\right]\times\left[\sigma(\boldsymbol{w}^\intercal\boldsymbol{x}_j)\right]+\left[\sigma'(\boldsymbol{w}^\intercal\boldsymbol{x}_j)\boldsymbol{x}_j\right]\times\left[\sigma(\boldsymbol{w}^\intercal\boldsymbol{x}_i)\right]$$

for all $\boldsymbol{w}$. Hence, it holds for all $\boldsymbol{w}$ that $\|\nabla_{\boldsymbol{w}}g^{(a)}(\boldsymbol{w};\boldsymbol{x}_i,\boldsymbol{x}_j)\|_\infty\le 2r\|\boldsymbol{w}\|_1^{2r-1}$. Therefore, the bound in (42) becomes

$$|\boldsymbol{G}_{ij}^{(a)}(\boldsymbol{\theta}(t^*))-\boldsymbol{G}_{ij}^{(a)}(\boldsymbol{\theta}(0))|\le\frac{2r}{m}\sum_{k=1}^m\|c\boldsymbol{w}_k(t^*)+(1-c)\boldsymbol{w}_k(0)\|_1^{2r-1}\|\boldsymbol{w}_k(t^*)-\boldsymbol{w}_k(0)\|_1. \tag{43}$$

By Lemma B.7,

$$\|c\boldsymbol{w}_k(t^*)+(1-c)\boldsymbol{w}_k(0)\|_1\le\|\boldsymbol{w}_k(0)\|_1+\|\boldsymbol{w}_k(t^*)-\boldsymbol{w}_k(0)\|_1\le n(\eta+q)\le 2n\eta,$$

where η and q are defined in Lemma B.7. So, (43) and the above inequalities indicate

$$|\boldsymbol{G}_{ij}^{(a)}(\boldsymbol{\theta}(t^*))-\boldsymbol{G}_{ij}^{(a)}(\boldsymbol{\theta}(0))|\le r(2n)^{2r}\eta^{2r-1}q,$$

and

$$\begin{aligned}\|\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(t^*))-\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(0))\|_{\mathrm{F}} &\le nr(2n)^{2r}\eta^{2r-1}q\\ &= nr(2n)^{2r}(2\log\frac{4m(n+1)}{\delta})^{(3r-1)/2}(2n)^r\frac{2\max\{n,r\}N\sqrt{2\lambda_{\boldsymbol{A},N}R_S(\boldsymbol{\theta}^0)}}{nm\lambda_S\lambda_{\boldsymbol{A}}}\\ &= 2^{9r/2+1}n^{3r-1}N^2r\left(\log\frac{4m(n+1)}{\delta}\right)^{(3r-1)/2}\frac{\max\{n,r\}\sqrt{\lambda_{\boldsymbol{A},N}R_S(\boldsymbol{\theta}^0)}}{m\lambda_S\lambda_{\boldsymbol{A}}}\\ &\le \frac{1}{4}\lambda_S,\end{aligned}$$

if we choose

$$m\ge 2^{9r/2+3}n^{3r-1}N^2r\left(\log\frac{4m(n+1)}{\delta}\right)^{(3r-1)/2}\frac{\max\{n,r\}\sqrt{\lambda_{\boldsymbol{A},N}R_S(\boldsymbol{\theta}^0)}}{\lambda_S^2\lambda_{\boldsymbol{A}}}.$$

The fact that $\|\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(t^*))-\boldsymbol{G}^{(a)}(\boldsymbol{\theta}(0))\|_{\mathrm{F}}\le\frac{1}{4}\lambda_S$ above contradicts with the definition of t^* in (27).

Let us summarize the conclusion in the above discussion. Let $C_n:=\mathbb{E}\|\boldsymbol{w}\|_1^{4r}<+\infty$ with $\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I}_n)$. The largest eigenvalue and the condition number of $\boldsymbol{A}\boldsymbol{A}^\intercal$ are denoted as $\lambda_{\boldsymbol{A},N}$ and κ_A, respectively. For any $\delta\in(0,1)$, define

$$m_1=\frac{32N^4C_n}{\lambda_S^2\delta},$$

$$m_2=\frac{5\max\{n,r\}N\sqrt{2\lambda_{\boldsymbol{A},N}R_S(\boldsymbol{\theta}^0)}}{\lambda_S\lambda_{\boldsymbol{A}}}\left(2n\sqrt{2\log\frac{4m(n+1)}{\delta}}\right)^{r-1},$$

and

$$m_3=2^{9r/2+3}n^{3r-1}N^2r\left(\log\frac{4m(n+1)}{\delta}\right)^{(3r-1)/2}\frac{\max\{n,r\}\sqrt{\lambda_{\boldsymbol{A},N}R_S(\boldsymbol{\theta}^0)}}{\lambda_S^2\lambda_{\boldsymbol{A}}}. \tag{44}$$

Then when $m\ge\max\{m_1,m_2,m_3\}$, with probability at least $1-\delta$ over the random initialization $\boldsymbol{\theta}^0$, we have, for all $t\ge 0$,

$$R_S(\boldsymbol{\theta}(t))\le\exp\left(-\frac{m\lambda_S\lambda_{\boldsymbol{A}}t}{N}\right)R_S(\boldsymbol{\theta}^0).$$

Note that $\mathcal{O}(\kappa_A\mathrm{poly}(N,r,n,\frac{1}{\delta},\frac{1}{\lambda_S}))\ge\max\{m_1,m_2,m_3\}$. Hence, we have completed the proof. □

C More details on the numerical experiments

In this Appendix, we report the detailed hyper-parameter setting for the three examples presented in the main text. We also report the numerical values of the errors (up to four decimals) corresponding to the result in Figures 1-3 in the main text. For convenience, we report the notations in the following table.

Notation	Explanation
k	k-nearest neighbors in DM
ϵ	the bandwidth parameter for local integral in DM
m	the width of hidden layer in FNN
T	the number of iterations for training a FNN
N	the number of training points
N_b	the number of training points on the boundary
γ	the regularization coefficient for $\frac{1}{2}\|\boldsymbol{\phi}_{\boldsymbol{\theta}}\|_2^2$ introduced in Section 3
λ	the penalty coefficient to enforce the boundary condition in the loss function (10)

Table 3: The summary of hyperparameter notations in the algorithms.

	N	625	1225	2500	5041	10000	19881	40000	80089
DM	ϵ	0.1166	0.0508	0.0237	0.0118	0.0059	0.0029	0.0014	0.0008
	k	128	128	128	256	256	256	512	768
NN	T	2000	3000	4000	4000	4000	4000	4000	8000
	m	50	71	100	141	200	282	400	583
	γ	0.001	0.001	0.001	0.001	0.001	0.002	0.005	0.01

Table 4: The hyperparameter setting for **Example 1**: 2D torus embedded in $\mathbb{R}^3$.

	N	625	1225	2500	5041	10000	19881	40000	80089
DM	forward error	3.7837	1.8985	1.0231	0.4871	0.2519	0.1251	0.0630	0.0326
	inverse error	0.7606	0.3175	0.1511	0.0680	0.0336	0.0159	N/A	N/A
NN	training error	0.7563	0.3183	0.1508	0.0680	0.0336	0.0158	0.0082	0.0036
	testing error	0.7764	0.3213	0.1516	0.0682	0.0336	0.0159	0.0082	0.0036

Table 5: The errors for DM and NN corresponding to **Example 1**: 2D torus embedded in $\mathbb{R}^3$. N/A indicates that the result is not computable.

	N	512	1331	4096	12167	24389
DM	ϵ	0.43	0.23	0.12	0.073	0.051
	k	128	128	256	256	256
NN	T	1000	2000	2000	3000	3000
	m	100	150	250	400	500
	γ	0.001	0.001	0.005	0.005	0.005

Table 6: The hyperparameter setting for **Example 2**: 3D manifold embedded in $\mathbb{R}^{12}$.

	N	512	1331	4096	12167	24389
DM	forward error	0.4241	0.1498	0.0403	0.0109	0.0039
	inverse error	0.2614	0.1113	0.0347	0.0092	N/A
NN	training error	0.2665	0.1148	0.0297	0.0066	0.0023
	testing error	0.2715	0.1346	0.0302	0.0069	0.0024

Table 7: The errors for DM and NN corresponding to **Example 2**: 3D manifold embedded in $\mathbb{R}^{12}$. N/A indicates that the result is not computable.

	N	1024	2025	4096	16384	32400
DM	ϵ	0.0221	0.0096	0.0048	0.0013	0.00064
	k	128	128	128	256	256
NN	T	2000	3000	4000	10000	12000
	m	100	100	150	250	250
	λ	5.0	5.0	5.0	5.0	5.0

Table 8: The hyperparameter setting for **Example 3**: 2D semi-torus with Dirichlet condition.

	N	1024	2025	4096	16384	32400
DM	forward error	6.7976	6.8733	7.0843	8.1603	8.9551
	inverse error	0.0627	0.0286	0.0149	0.0047	0.0026
NN	training error	0.0613	0.0285	0.0150	0.0048	0.0029
	testing error	0.0634	0.0293	0.0154	0.0048	0.0029

Table 9: The errors for DM and NN corresponding to **Example 3**: 2D semi-torus with Dirichlet condition.

References

[1] On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes. *Communications in Computational Physics*, 28(5):2042–2074, 2020.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, 09–15 Jun 2019.

[3] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *36th International Conference on Machine Learning, ICML 2019*, 36th International Conference on Machine Learning, ICML 2019, pages 477–502. International Machine Learning Society (IMLS), January 2019. 36th International Conference on Machine Learning, ICML 2019 ; Conference date: 09-06-2019 Through 15-06-2019.

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

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

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

[7] Julius Berner, Philipp Grohs, and Arnulf Jentzen. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of black–scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, 2(3):631–657, 2020.

[8] T. Berry and J. Harlim. Variable bandwidth diffusion kernels. *Appl. Comput. Harmon. Anal.*, 40:68–96, 2016.

[9] Marcelo Bertalmıo, Li-Tien Cheng, Stanley Osher, and Guillermo Sapiro. Variational problems and partial differential equations on implicit surfaces. *Journal of Computational Physics*, 174(2):759–780, 2001.

[10] Andrea Bonito, J Manuel Cascón, Khamron Mekchay, Pedro Morin, and Ricardo H Nochetto. High-order afem for the laplace–beltrami operator: Convergence rates. *Foundations of Computational Mathematics*, 16(6):1473–1539, 2016.

[11] Fernando Camacho and Alan Demlow. L2 and pointwise a posteriori error estimates for fem for elliptic pdes on surfaces. *IMA Journal of Numerical Analysis*, 35(3):1199–1227, 2015.

[12] R. Coifman and S. Lafon. Diffusion maps. *Appl. Comput. Harmon. Anal.*, 21:5–30, 2006.

[13] Qiang Du, Yiqi Gu, Haizhao Yang, and Chao Zhou. The discovery of dynamics via linear multistep methods and deep learning: Error estimation. *arxiv:2103.11488*, 2021.

[14] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1675–1685. PMLR, 09–15 Jun 2019.

[15] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[16] Chenguang Duan, Yuling Jiao, Yanming Lai, Xiliang Lu, and Zhijian Yang. Convergence rate analysis for deep ritz method. *arxiv:2103.13330*, 2021.

[17] D Dunson, Hau-Tieng Wu, and Nan Wu. Spectral convergence of graph Laplacian and Heat kernel reconstruction in L^∞ from random samples. *arXiv preprint arXiv:1912.05680*, 2019.

[18] Gerhard Dziuk and Charles M Elliott. Finite element methods for surface pdes. *Acta Numerica*, 22:289–396, 2013.

[19] 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, Dec 2017.

[20] Weinan E, Chao Ma, and Lei Wu. A priori estimates of the population risk for two-layer neural networks. *Communications in Mathematical Sciences*, 17(5):1407 – 1425, 2019.

[21] Weinan E, Chao Ma, and Lei Wu. Barron Spaces and the Compositional Function Spaces for Neural Network Models. *Constructive Approximation*, 2020.

[22] Weinan E and Qingcan Wang. Exponential convergence of the deep neural network approximation for analytic functions. *Science China Mathematics*, 61(10):1733–1740, 10 2018.

[23] Charles M Elliott and Björn Stinner. Modeling and computation of two phase geometric biomembranes using surface finite elements. *Journal of Computational Physics*, 229(18):6585–6612, 2010.

[24] Z. Fang and J. Zhan. A physics-informed neural network framework for pdes on 3d surfaces: Time independent problems. *IEEE Access*, 8:26328–26335, 2020.

[25] Edward J Fuselier and Grady B Wright. A high-order kernel method for diffusion and reaction-diffusion equations on surfaces. *Journal of Scientific Computing*, 56(3):535–565, 2013.

[26] F. Gilani and J. Harlim. Approximating solutions of linear elliptic PDE's on smooth manifold using local kernels. *J. Comput. Phys.*, 395:563–582, 2019.

[27] David Gilbarg and Neil S Trudinger. *Elliptic partial differential equations of second order*. springer, 2015.

[28] Ingo Gühring, Gitta Kutyniok, and Philipp Petersen. Error bounds for approximations with deep relu neural networks in ws,p norms. *Analysis and Applications*, 18(05):803–859, 2020.

[29] 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.

[30] Jiequn Han and Jihao Long. Convergence of the deep bsde method for coupled fbsdes. *Probability, Uncertainty and Quantitative Risk*, 5(1):5, 2020.

[31] Qing Han and Fanghua Lin. *Elliptic partial differential equations*, volume 1. American Mathematical Soc., 2011.

[32] J. Harlim. *Data-Driven Computational Methods: Parameter and Operator Estimations*. Cambridge University Press, 2018.

[33] John Harlim, Daniel Sanz-Alonso, and Ruiyi Yang. Kernel methods for bayesian elliptic inverse problems on manifolds. *SIAM/ASA Journal on Uncertainty Quantification*, 8(4):1414–1445, 2020.

[34] Qingguo Hong, Jonathan W. Siegel, and Jinchao Xu. A priori analysis of stable neural network solutions to numerical pdes. *arxiv:2104.02903*, 2021.

[35] 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. *SN Partial Differential Equations and Applications*, 1(10), 2020.

[36] Martin Hutzenthaler, Arnulf Jentzen, and von Wurstemberger Wurstemberger. Overcoming the curse of dimensionality in the approximative pricing of financial derivatives with default risks. *Electron. J. Probab.*, 25:73 pp., 2020.

[37] Martin Hutzenthaler, Arnulf Jentzen, and von Wurstemberger Wurstemberger. Overcoming the curse of dimensionality in the approximative pricing of financial derivatives with default risks. *Electron. J. Probab.*, 25:73 pp., 2020.

[38] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8580–8589, Red Hook, NY, USA, 2018. Curran Associates Inc.

[39] Shixiao W. Jiang and John Harlim. Ghost point diffusion maps for solving elliptic pde's on manifolds with classical boundary conditions. *Comm. Pure Appl. Math. (in press), arXiv preprint arXiv:2006.04002*, 2020.

[40] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. *European Journal of Applied Mathematics*, page 1–15, 2020.

[41] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[42] John M Lee. *Introduction to Smooth Manifolds*. Springer, 2013.

[43] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, volume 98. Siam, 2007.

[44] Ke Li, Kejun Tang, Tianfan Wu, and Qifeng Liao. D3M: A Deep Domain Decomposition Method for Partial Differential Equations. *IEEE Access*, 8:5283–5294, 2020.

[45] Senwei Liang, Liyao Lyu, Chunmei Wang, and Haizhao Yang. Reproducing activation function for deep learning. *arxiv:2101.04844*, 2021.

[46] Jianfeng Lu and Yulong Lu. A priori generalization error analysis of two-layer neural networks for solving high dimensional schrödinger eigenvalue problems. *arxiv:2105.01228*, 2021.

[47] Jianfeng Lu, Yulong Lu, and Min Wang. A priori generalization analysis of the deep ritz method for solving high dimensional elliptic equations. *arxiv:2101.01708*, 2021.

[48] Jianfeng Lu, Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep Network Approximation for Smooth Functions. *arXiv e-prints*, page arXiv:2001.03040, January 2020.

[49] Yiping Lu, Chao Ma, Yulong Lu, Jianfeng Lu, and Lexing Ying. A mean field analysis of deep ResNet and beyond: Towards provably optimization via overparameterization from depth. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6426–6436. PMLR, 13–18 Jul 2020.

[50] Tao Luo and Haizhao Yang. Two-layer neural networks for partial differential equations: Optimization and generalization theory. *ArXiv*, abs/2006.15733, 2020.

[51] Colin B Macdonald and Steven J Ruuth. The implicit closest point method for the numerical solution of partial differential equations on surfaces. *SIAM Journal on Scientific Computing*, 31(6):4330–4350, 2010.

[52] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[53] Facundo Mémoli, Guillermo Sapiro, and Paul Thompson. Implicit brain imaging. *NeuroImage*, 23:S179–S188, 2004.

[54] 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.

[55] Hadrien Montanelli and Haizhao Yang. Error bounds for deep ReLU networks using the Kolmogorov–Arnold superposition theorem. *Neural Networks*, 129:1–6, 2020.

[56] Hadrien Montanelli, Haizhao Yang, and Qiang Du. Deep ReLU networks overcome the curse of dimensionality for bandlimited functions. *Journal of Computational Mathematics*, To appear.

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

[58] Cécile Piret. The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces. *Journal of Computational Physics*, 231(14):4662–4675, 2012.

[59] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686 – 707, 2019.

[60] Matthias Rauter and Željko Tuković. A finite area scheme for shallow granular flows on three-dimensional surfaces. *Computers & Fluids*, 166:184–199, 2018.

[61] Steven J Ruuth and Barry Merriman. A simple embedding method for solving partial differential equations on surfaces. *Journal of Computational Physics*, 227(3):1943–1961, 2008.

[62] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[63] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation characterized by number of neurons. *Communications in Computational Physics*, 28(5):1768–1811, 2020.

[64] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network with approximation error being reciprocal of width to power of square root of depth. *Neural Computation*, 33(4):1005–1036, 2021.

[65] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Neural network approximation: Three hidden layers are enough. *Neural Networks*, 141:160–173, 2021.

[66] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Optimal approximation rate of ReLU networks in terms of width and depth. *Journal de Mathématiques Pures et Appliquées*, to appear.

[67] Jonathan W. Siegel and Jinchao Xu. Approximation rates for neural networks with general activation functions. *Neural Networks*, 128:313 – 321, 2020.

[68] Amit Singer. From graph to manifold Laplacian: The convergence rate. *Appl. Comp. Harmonic Anal.*, 21:128–134, 2006.

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

[70] Epifanio G Virga. *Variational theories for liquid crystals*. CRC Press, 2018.

[71] Q Yan, S.W. Jiang, and J Harlim. Kernel-based methods for Solving Time-Dependent Advection-Diffusion Equations on Manifolds. *arXiv preprint arXiv:2105.13835*, 2021.

[72] Yunfei Yang and Yang Wang. Approximation in shift-invariant spaces with deep ReLU neural networks. *arXiv e-prints*, page arXiv:2005.11949, May 2020.

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

[74] 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.

[75] Dmitry Yarotsky and Anton Zhevnerchuk. The phase diagram of approximation rates for deep neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13005–13015. Curran Associates, Inc., 2020.

[76] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, 2020.