# Random Sketching for Neural Networks With ReLU

Di Wang, Jinshan Zeng, and Shao-Bo Lin

***Abstract*— Training neural networks is recently a hot topic in machine learning due to its great success in many applications. Since the neural networks' training usually involves a highly nonconvex optimization problem, it is difficult to design optimization algorithms with perfect convergence guarantees to derive a neural network estimator of high quality. In this article, we borrow the well-known random sketching strategy from kernel methods to transform the training of shallow rectified linear unit (ReLU) nets into a linear least-squares problem. Using the localized approximation property of shallow ReLU nets and a recently developed dimensionality-leveraging scheme, we succeed in equipping shallow ReLU nets with a specific random sketching scheme. The efficiency of the suggested random sketching strategy is guaranteed by theoretical analysis and also verified via a series of numerical experiments. Theoretically, we show that the proposed random sketching is almost optimal in terms of both approximation capability and learning performance. This implies that random sketching does not degenerate the performance of shallow ReLU nets. Numerically, we show that random sketching can significantly reduce the computational burden of numerous backpropagation (BP) algorithms while maintaining their learning performance.**

***Index Terms*— Generalization error, neural networks, random sketching, rectified linear unit (ReLU).**

## I. Introduction

DEEP learning [1] has made some significant breakthroughs in a wide range of applications, including but not limited to the computer vision [2], speech recognition [3], and go game [4]. The great success of deep learning in practice stimulates the theoretical studies on its outperformance and rationality. In particular, advantages of deep neural networks (deep nets for short) over shallow neural networks (shallow nets for short) have been rigorously verified in terms of the localized approximation [5], sparse approximation [6], [7], hierarchical structures grasping [8], [9], manifold learning [10], [11], and rotation invariance protection [12]. We refer readers to a fruitful review [13] for details of deep learning in the application and a recent work [14] for the necessity of depth in theory.

Due to the highly nonconvex nature of the deep learning models, the training of deep nets generally involves the existence of local minima, saddle points, plateau, and even some flat regions [15]. Thus, training deep nets to realize their theoretically optimal performance is usually very difficult. Although there are numerous optimization algorithms, including the popular stochastic gradient descent (SGD) [16], conjugate gradient (CG) [17], alternating direction method of multipliers (ADMM) [18], and block coordinate descent (BCD) methods [19], have been proposed to equip deep learning models, their convergence issues are still open.

Under the well-known overparameterization assumption, the convergence issue of SGD for some shallow nets model was recently settled in [20] and [21] from the optimization viewpoint, by showing that SGD finds near-globally optimal solutions to the model. Here and hereafter, overparameterization means that the number of free parameters is larger than the size of data. However, overparameterization makes the neural networks model so flexible that its globally optimal solution suffers from the well-known overfitting phenomenon [22] in the sense that it fits the training data perfectly but fails to predict new query points.

Two popular strategies to avoid overfitting is either to restrict the range of parameters [18] or to early stop the algorithm before converging [23]. Due to the overparameterization setting and model selection, the former requires huge computations, inevitably hindering its wide use in tackling massive data. The latter, however, results in an inconsistency between optimization and learning; it requires a complex neural networks model and numerous iterations to guarantee the convergence in the perspective of optimization, while the convergence of the iterative algorithm might be not so important in machine learning if the model is too complex. Therefore, it is highly desired to develop training strategies for neural networks, even for shallow nets, with theoretical guarantees and low computational burden.

### A. Motivations

In this article, we aim at borrowing the random sketching scheme [24] from kernel learning to neural networks' training, in order to reduce the computational burden and avoid the aforementioned inconsistency between optimization and learning. Random sketching, originally proposed for kernel methods, is a computation-reduction technique to sketch a few columns from the kernel matrix to reduce the computational burden of kernel methods without sacrificing their learning performances. We refer readers to a nice work [25] for recent developments of random sketching. The main idea of our study can be illustrated in Fig. 1.

Intuitively, designing random sketching strategies for neural networks is much more difficult than that for kernel methods

Manuscript received May 26, 2019; revised September 23, 2019, November 25, 2019, and January 23, 2020; accepted March 4, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61772374, Grant 61876133, Grant 11771012, Grant 61977038, Grant 61603162, and Grant 61876074, in part by the Zhejiang Provincial Natural Science Foundation under Grant LY17F030004, and in part by the Two Thousand Talents Plan of Jiangxi Province. *(Corresponding author: Shao-Bo Lin.)*

Di Wang and Shao-Bo Lin are with the Center of Intelligent Decision-Making and Machine Learning, School of Management, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: wangdi@amss.ac.cn; sblin1983@gmail.com).

Jinshan Zeng is with the School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China (e-mail: jsh.zeng@gmail.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.2979228

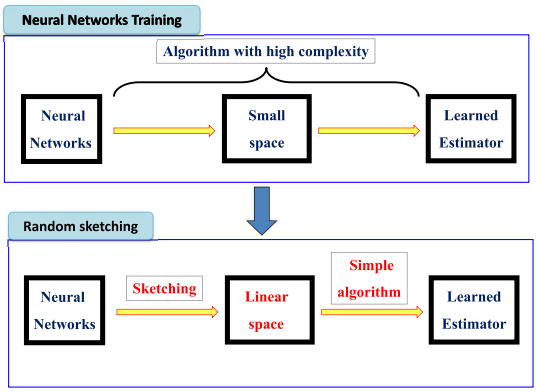


Fig. 1. Deep learning with random sketching.

since the latter refers to selecting representative elements from a large but finite-dimensional linear space, while the former focuses on choosing representative elements from a nonlinear space. A crucial way is to derive a representation theorem for neural networks, just as kernel methods did in [22]. Using the localized approximation property [11] of shallow nets with rectifier linear units (shallow ReLU nets) and a recently developed dimension leveraging approach [26], we succeed in deriving a representation theorem for shallow ReLU nets. The procedures can be described as follows.

*Step 1:* Constructing a trapezoid-shaped function via ReLU to embody the localized approximation property.

*Step 2:* Sketching a few points randomly in some interval to constitute thresholds of shallow ReLU nets to guarantee the approximation property of shallow ReLU nets.

*Step 3:* Generating a series of inner weights via random sampling on the unit sphere for the purpose of dimension leveraging.

Since the inner weights and thresholds are randomly assigned before the learning process, we obtain a set of basis functions constructed by shallow ReLU nets with random sketching. The only thing left is then to determine the outer weights that can be obtained by solving a linear least-squares problem.

### B. Main Contributions

The developed shallow ReLU nets with random sketching provide a feasible way to reduce the computational burden of neural networks' training and avoid the optimization-learning inconsistency. Our main contributions can be summarized as follows.

*1) Methodology Novelty:* We develop a novel neural networks' training strategy via combining shallow ReLU nets with random sketching. Due to the localized approximation property of shallow ReLU nets, we suggest that the inner weights of the shallow ReLU nets can be randomly sketched from the unit sphere, while thresholds of the shallow ReLU nets can be randomly sketched from an interval. Once the inner weights and thresholds are determined via random sketching, the nonlinear and nonconvex neural networks' training problem [15] can be transformed into a linear least-squares problem, significantly reducing the computational burden.

*2) Theoretical Novelty:* Using a dimension-leveraging technique developed in [26] and [27], we prove that sketching inner weights and thresholds in the aforementioned manner does not degenerate the representation performance of shallow ReLU nets. Furthermore, we provide almost optimal learning rate estimates in the framework of learning theory [22] and rigorously prove that the generalization ability of shallow ReLU nets with random sketching is comparable to the state-of-the-art results.

*3) Numerical Novelty:* We compare the proposed random sketching scheme with 12 popular backpropagation (BP)-type algorithms. The experimental results show that as far as the test accuracy is concerned, random sketching is comparable with these BP-type algorithms. However, the training time of random sketching is much less than these algorithms, showing the advantage of implementing a random sketching scheme on the neural networks' training. All the numerical results verify our theoretical assertions and show that combining neural networks' training with random sketching is feasible and efficient.

The rest of this article is organized as follows. In Section II, we introduce the step-stone for our construction and detailed implementation of the random sketching scheme. In Section III, we study the theoretical behaviors of random sketching. Section IV provides numerical experiments to show the efficiency of the proposed method. In Section V, we discuss extensions of the proposed random sketching scheme for deep ReLU nets. We conclude our study shortly in Section VII.

## II. Random Sketching for Shallow ReLU Nets

In this section, we first introduce the main tools of our study and then propose the random sketching scheme for shallow ReLU nets.

### A. Representation Theorem for Shallow ReLU Nets

Random sketching [24], [25], [28], originally developed for kernel methods, aims to sketch a few representative functions in some appropriately random way so that the generalization power of kernel methods is not degenerated too much. The representation theorem of kernel methods [22] and random linear algebraic [29] enable the feasibility of random sketching via providing numerous feasible sketching strategies [25].

The lack of similar representation theorem for deep learning [30] makes it difficult to design the random sketching strategy for neural networks. The core of our approach is to build a representation theorem for shallow ReLU nets by using their localized approximation property and a recently developed dimension-leveraging technique [26]. Let $\mathbb{B}^d$ and $\mathbb{S}^{d-1}$ be the unit ball and unit sphere in $\mathbb{R}^d$, respectively. Recalling [26], [31], any $f \in L^2(\mathbb{B}^d)$ has a ridge function representation

$$f(x) = \sum_{j=1}^{\infty} a_j \int_{\mathbb{S}^{d-1}} g_j(\xi)\phi_j(x \cdot \xi) d\omega_{d-1}(\xi) \tag{1}$$

where $g_j$ and $\phi_j$ are polynomials of degrees at most $j$ defined on $\mathbb{S}^{d-1}$ and $[-1, 1]$, respectively, and $d\omega_{d-1}$ denotes the area element of $\mathbb{S}^d$. For arbitrarily fixed $x$, $g_j(\xi)\phi_j(x \cdot \xi)$ is then a spherical polynomial. Hence, for any $j \in \mathbb{N}$, spherical quadrature formulas established in [32] and [33] show that there exists a set of positive numbers $\{w_i\}_{i=1}^{\tilde{n}_j}$ such that

$$\int_{\mathbb{S}^{d-1}} g_j(\xi)\phi_j(x \cdot \xi) d\omega_{d-1}(\xi) = \sum_{i=1}^{\tilde{n}_j} w_i \phi_j(\xi_i \cdot x) \tag{2}$$

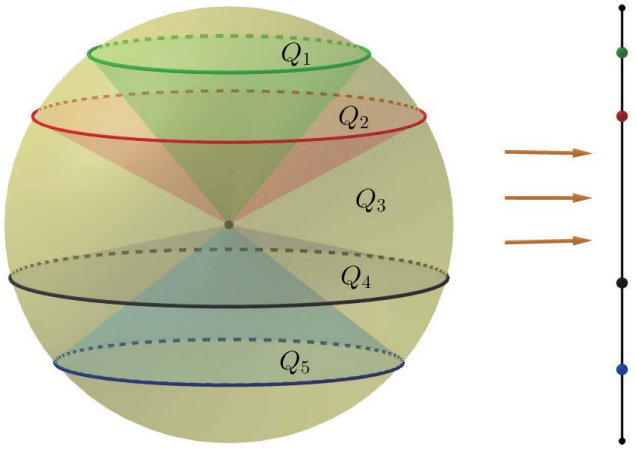


Fig. 2. Division of the ball via localized approximation.

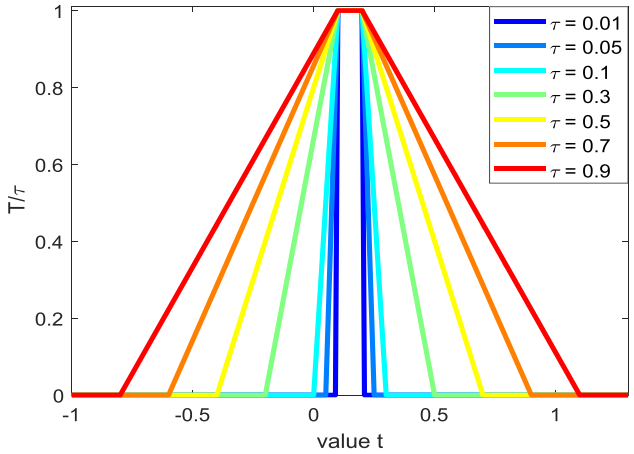


Fig. 3. Localized approximation for ReLU nets.

holds with high probability, provided $\{\xi_i\}_{i=1}^{\tilde{n}_j}$ are drawn independently according the uniform distribution on the sphere for some $\tilde{n}_j \in \mathbb{N}$ depending on $j$. Under this circumstance

$$\big\{\phi_j(\xi_i \cdot x)\colon 1 \le i \le \tilde{n}_j,\, j \in \mathbb{N}\big\}$$

consists a basis for $L^2(\mathbb{B}^d)$.

For arbitrary $\xi_i \in \mathbb{S}^{d-1}$, we divide $\mathbb{B}^d$ into $\ell$ domains according to $t_k \le \xi_i \cdot x \le t_{k+1}$ for $\{t_k\}_{k=1}^{\ell} \subset (-1, 1)$, $t_0 = -1$, and $t_{\ell+1} = 1$. Fig. 2 presents an illustration for the division with $\ell = 4$, showing that such a division corresponds to a partition of $[-1, 1]$, which can be reflected by shallow ReLU nets as follows. Let $\sigma(t) := \max\{0, t\}$ be the ReLU function. Define a trapezoid-shaped function $T_{\tau,a,b}$ with a parameter $0 < \tau \le 1$ as

$$T_{\tau,a,b}(t) := \sigma(t-a+\tau) - \sigma(t-a) - \sigma(t-b) + \sigma(t-b-\tau). \tag{3}$$

For $\ell \in \mathbb{N}$ and $\{t_k\}_{k=1}^{\ell} \subset \mathbb{R}$ with $t_1 < t_2 < \cdots < t_\ell$, it is easy to check

$$\frac{1}{\tau} T_{\tau,t_k,t_{k+1}}(t) = \begin{cases} 1, & \text{if } t_k \le t \le t_{k+1} \\ 0, & \text{if } t \ge t_{k+1}+\tau, \text{ or } t \le t_k - \tau \\ \dfrac{t_{k+1}+\tau-t}{\tau}, & \text{if } t_{k+1} < t < t_{k+1}+\tau \\ \dfrac{t-t_k+\tau}{\tau}, & \text{if } t_k - \tau < t < t_{k+1}. \end{cases} \tag{4}$$

This means, as shown in Fig. 3, that for sufficiently small $\tau$, the ReLU nets defined in (3) can realize the indicator function and is capable of determining where $\xi_i \cdot x$ is located. Such a localized approximation property shows that if the target function is modified only on a small region ($Q1$ in Fig. 2, for example), only a few neurons, rather than the entire network, need to be retrained. Furthermore, if the number $\ell$ of domains is large enough, the position of $\xi_i \cdot x$ will be precisely captured by the ReLU nets (3). Thus

$$\operatorname{span}\big\{T_{\tau,t_k,t_{k+1}}\colon k = 0, \ldots, \ell\big\}$$

is dense in the univariate continuous function spaces for $\ell \to \infty$. With this, the set

$$\mathcal{R} := \operatorname{span}\big\{T_{\tau,t_k,t_{k+1}}(\xi_i \cdot x)\colon 1 \le i \le n, k = 0, \ldots, \ell\big\} \tag{5}$$

is a good approximation of $L^2(\mathbb{B}^d)$ for appropriate $n$ and $\ell$. The abovementioned procedure, similar to the representation theorem of kernel methods, makes it possible to sketch a few of functions in $\mathcal{R}$ to build up the hypothesis space for the learning purpose.

### *B. Random Sketching for Shallow ReLU Nets*

Based on the representation formula (5), we present a random sketching strategy for shallow ReLU nets. Our sketching strategy can be described as the following four steps. In the first step, we construct a shallow ReLU net like (3) to capture the position information for the dot product $\xi \cdot x$. In this construction, a partition $\{t_k\}_{k=0}^{\ell+1}$ of an interval is required. For the sake of brevity in theory analysis, we restrict our analysis on $\mathbb{B}_{1/2}^d$, i.e., the ball centered on the origin with radius $1/2$. Our results can be easily extended to balls with arbitrary finite radius by a scaling strategy. In the second step, we draw $t_0 = -1/2$, $t_{\ell+1} = 1/2$ and $\{t_k\}_{k=1}^{\ell}$ independently according to the uniform distribution from $(-1/2, 1/2)$. In the third step, we focus on searching $\{\xi_i\}_{i=1}^{n}$ such that (2) holds for some degree of polynomials depending on $\ell$. Motivated by the spherical quadrature rule established in [33] and [34], it is feasible to independently sample $n$ points on the unit sphere according to the uniform distribution to generate $\{\xi_i\}_{i=1}^{n}$. With these, we obtain a hypothesis space based on shallow ReLU nets and random sketching

$$\mathcal{H}_{\ell,n,\tau} := \left\{ \sum_{j=1}^{n} \sum_{k=1}^{\ell} a_{jk} T_{\tau,t_{k-1},t_k}\big(\xi_j \cdot x\big)\colon a_{jk} \in \mathbb{R} \right\}. \tag{6}$$

Once $\{\xi_j\}_{j=1}^{n}$ and $\{t_k\}_{k=1}^{\ell}$ are specified, $\mathcal{H}_{\ell,n,\tau}$ is then a linear space, whose structure can be found in Fig. 4. The final step focuses on designing the learning algorithms based on the hypothesis space (6) for different learning tasks. In this article, we are concerned with the standard least-squares regression problem [22] and, thus, employ the empirical risk minimization (ERM) with the square loss on $\mathcal{H}_{\ell,n,\tau}$ as

$$f^*_{D,\ell,n,\tau} := \arg\min_{f \in \mathcal{H}_{\ell,n,\tau}} \frac{1}{m} \sum_{(x_i,y_i) \in D} |f(x_i) - y_i|^2 \tag{7}$$

where $D = \{(x_i, y_i)\}_{i=1}^{m}$ with $x_i \in \mathcal{X} = \mathbb{B}_{1/2}^d$ and $y_i \in \mathcal{Y} \subseteq [-M, M]$ for some $M > 0$ is the sample set. Since

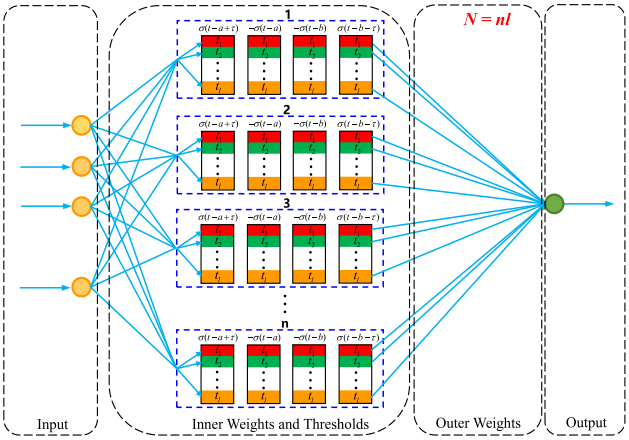


Fig. 4. Topological structures of ReLU nets.

**Algorithm 1** Random Sketching for Shallow ReLU Nets

**Input:** Training data set $D = \{(x_i, y_i) : i = 1, \ldots, m\}$, parameters $n, \ell \in \mathbb{N}$ and $\tau \in (0, 1)$.

Step 1 (Initialization): Normalize $D_x = \{x_i\}_{i=1}^m$ to $\mathbb{B}_{1/2}^d$ and compute $M = \max_{1\le i\le m} |y_i|$.

Step 2 (Sketching thresholds): Let $t_0 = -1/2$, and randomly sketch $\ell$ points in $(-1/2, 1/2)$ according to the uniform distribution and sort them increasingly as $t_1 < t_2 < \cdots < t_\ell$.

Step 3 (Sketching inner weights): Randomly sketch $n$ inner weights $\{\xi_j\}_{j=1}^n$ according to the uniform distribution on the unit sphere $\mathbb{S}^{d-1}$.

Step 4 (Generating hypothesis space): Generate the hypothesis space (6).

Step 5 (Deriving outer weights): Define the estimator by

$$f_{D,\ell,n,\tau}(x) := \pi_M f^*_{D,l,n,\tau}(x), \tag{8}$$

where $f^*_{D,\ell,n,\tau}$ is given by (7).

**Output:** Thresholds $\{t_k\}_{k=1}^\ell$, inner weights $\{\xi_j\}_{j=1}^n$, outer weights $\{a_{j,k}\}_{j,k=1}^{n,\ell}$ and the estimator $f_{D,\ell,n,\tau}(x)$.

$|y_i| \le M$, it is natural to project the final output $f^*_{D,\ell,n,\tau}$ to the interval $[-M, M]$ by the standard truncation operator [7], [35], [36], and the final estimator is $\pi_M f^*_{D,\ell,n,\tau}(x) := \text{sign}(f^*_{D,\ell,n,\tau}(x)) \min\{|f^*_{D,\ell,n,\tau}(x)|, M\}$.

We summarize the proposed scheme in Algorithm 1. There are three parameters in total, that is, $\ell, n, \tau$ in our algorithm. The parameter $\ell$ denotes the number of different thresholds, and it controls the expressivity and approximation capability of the hypothesis space (6). Thus, $\ell$ is the key parameter in Algorithm 1. The parameter $n$ denotes the number of different inner weights and contributes to dimensionality leveraging. In fact, the only requirement on $n$ is to guarantee the spherical quadrature rule (2) for some $j$. Thus, large $n$ permits the quadrature rule but requires more neurons resulting in large variance for (7), while small $n$ fails to guarantee the approximation performance of $\mathcal{H}_{\ell,n,\tau}$ in approximating multivariate functions. Based on our previous study [33], $n = \tilde{c}\ell^{d-1}$ for some $\tilde{c}$ is suitable for a moderately large $d$. The parameter $\tau$ is to describe the power of ReLU nets to capture the position information of $\xi_j \cdot x$. Too small $\tau$ leads to perfect position grasping performance of $T_{\tau,a,b}/\tau$ but makes the inner weights be extremely large, which also affects the generalization performance of $f_{D,\ell,n,\tau}$. Since the capacity of (6), measured by the covering number [14], behaves logarithmically with respect to $\tau$, the performance of $f_{D,\ell,n,\tau}$, however, behaves relatively stable with respect to $\tau$, which can be found in both our theoretical analysis and numerical results shown latter. Since inner weights and thresholds are determined before the learning process, Algorithm 1 requires $\mathcal{O}(mn^2\ell^2)$ float computations, which is even smaller than $\mathcal{O}(m^3)$, i.e., the computational complexity for the widely used kernel ridge regression [7], provided $\ell$ and $n$ are relatively small. Indeed, our theoretical results in Section III show that $\ell \sim m^{1/(2r+d)}$ and $n \sim \ell^{d-1}$, where $r$ describes the regularity of the data. Under this circumstance, $n\ell \sim m^{d/(2r+d)} \ll m$, provided $r$ is large enough. This phenomenon can also be found in our simulation results in Section IV. We end this section with two important remarks.

*Remark 1:* Due to (3), the parameter $\tau$ should be smaller than $t_{k+1} - t_k$ for any $k$. Noting that $\{t_k\}_{k=1}^\ell$ are randomly drawn according to the uniform distribution on $(-1/2, 1/2)$, and $\tau$ is extremely small for some $\{t_k\}_{k=1}^\ell$. Especially, when $\ell$ is large, $\tau$ is then an extremely small random value at the first glance. We highlight that $\tau$ in our approach is set to be a parameter in a deterministic manner. The reasons are two folds. On one hand, it can be found in [37, Problem 2.4] that for $\tau \le \ell^{-5}$, $\tau \le t_{k+1} - t_k$ holds for any $k$ with high probability. On the other hand, our theoretical results as well as latter numerical studies show that $\ell$ is not very large, i.e., $\ell \le m^{1/d}$. Therefore, $\tau \le m^{-5/d}$ is small enough to derive a good estimator with high probability.

*Remark 2:* As shown in [38], there are some limitations for shallow ReLU nets in approximation and learning, which are also the limitations for shallow ReLU nets with random sketching. In particular, as shown in Theorems 1 and 2, the shallow ReLU net with random sketching suffers from the well-known saturation phenomenon [14] in the sense that their approximation and learning performances cannot be improved further when the regularity of target functions achieves a certain level. Of course, it can be avoided by deepening the neural networks [14]. The problem is that our present random sketching scheme is only available for shallow ReLU nets. It remains open to design a random sketching strategy to equip deep nets. In this article, we are only concerned with regression problems. How to develop a random sketching scheme for neural networks classification, which is available to real tasks such as image classification, is also an interesting issue. We will keep in this study in future work.

### C. Related Work

Training neural networks is a classical research topic in machine learning. Numerous algorithms, such as the BP [39] and SGD [40], have been proposed to solve neural networks related optimization problems. For example, the convergence issue of SGD has been studied in [41] for shallow nets, in [42] for shallow ReLU nets with small weights, in [43] for shallow ReLU nets with normalized weights, in [44] for shallow ReLU nets under the overparameterized assumption, and in [45] for deep ReLU nets under the overparameterized assumption. It should be noted that the overparameterized assumption plays a central role in the existing literature. However, too many parameters imply ultralarge capacity of the hypothesis space, which inevitably leads to the instability of the derived

estimator and the overfitting from the learning theory [22]. This consequently follows an inconsistency between optimization and learning; the convergence of optimization algorithms requires numerous free parameters and then the large capacity of the neural networks model, while the learnability needs a small capacity of the hypothesis space to reduce the variance.

To avoid the aforementioned inconsistency between optimization and learning, we borrow the idea of random sketching from kernel methods to reduce the capacity of the neural networks model in an off-line manner. The basic idea of our study is to randomly sketch neural networks to build a linear hypothesis space and then use a simple learning algorithm to derive the estimator. The most related work in this article is [26], where some deterministic assignment of inner weights and thresholds of neural networks is provided. Compared with [26], the novelty of our analysis is shown as the following twofold. On one hand, we use random sketching rather than the deterministic assignment, which enhances the stability [25]. On the other hand, our construction is based on the localized approximation property of the ReLU nets, while the construction in [26] relies on the sigmoid activation function. The localized approximation property of ReLU nets, as shown in Fig. 3, connects small regions of inputs with a small number of neurons and, thus, reduces the computational burden when the target function varies in a small region.

## III. Theoretical Behaviors

In this section, we study theoretical behaviors for the proposed random sketching scheme.

### A. Approximation Capability

Shallow ReLU nets generate their estimators conventionally through solving the following nonlinear optimization problem:

$$\min_{(a_j,w_j,b_j)\in\mathbb{R}\times\mathbb{R}^d\times\mathbb{R}} \sum_{i=1}^{m}\left|\sum_{j=1}^{N} a_j\sigma\left(w_j\cdot x_i+b_j\right)-y_i\right|^2 \tag{9}$$

where $w_j$ is the inner weight which connects the input layer to the $j$th hidden neuron, $b_j$ is the threshold of the $j$th hidden neuron, and $a_j$ is the outer weight that connects the $j$th hidden layer to the output layer. As shown in Algorithm 1, the proposed random sketching scheme builds a hypothesis space by linear combinations of shallow ReLU nets and transforms the nonlinear neural networks' learning problem (9) to a linear least-squares problem (7). In this section, we aim to show that such a transformation does not degenerate the approximation ability of shallow ReLU nets too much.

For this purpose, we introduce the well-known Sobolev space $W^r(L^2(\mathbb{B}^d_{1/2}))$ with $r>0$ as follows. When $r=k$ is an integer, a function $f\in L^2(\mathbb{B}^d_{1/2})$ is in $W^k(L^2(\mathbb{B}^d_{1/2}))$ if and only if its distributional derivatives $D^v f$ of order $k$ are in $L^2(\mathbb{B}^d_{1/2})$, and

$$|f|^2_{W^k(L^2(\mathbb{B}^d_{1/2}))}=\sum_{|\mathbf{v}|=k}\|D^{\mathbf{v}}f\|^2_{L^2(\mathbb{B}^d_{1/2})}$$

presents the seminorm for $W^k(L^2(\mathbb{B}^d_{1/2}))$, where $|\mathbf{v}|=v_1+\ldots,v_d$ for $\mathbf{v}=(v_1,\ldots,v_d)$. The norm for $W^k(L^2(\mathbb{B}^d_{1/2}))$ is obtained by adding $\|f\|_{L^2(\mathbb{B}^d_{1/2})}$ to $|f|_{W^k(L^2(\mathbb{B}^d_{1/2}))}$. For other values of $r$, we obtain $W^r(L^2(\mathbb{B}^d_{1/2}))$ as the interpolation space [27]

$$W^r\left(L^2\left(\mathbb{B}^d_{1/2}\right)\right)=\left(L^2\left(\mathbb{B}^d_{1/2}\right),W^k\left(L^2\left(\mathbb{B}^d_{1/2}\right)\right)\right)_{\theta,2}$$
$$\theta=\frac{r}{k},\quad 0<r<k.$$

Throughout this article, we assume $f\in W^r(L^2(\mathbb{B}^d_{1/2}))$, which is a standard assumption in approximation theory and has been widely used to quantify the approximation capability for numerous hypothesis spaces [26], [27], [31], [37], [46], [47]. The following theorem is our first main result, whose proof is provided in Section VI.

*Theorem 1:* Let $d\geq 2$, $\ell,n\in\mathbb{N}$, and $A\geq 1$. If $f\in W^r(L^2(\mathbb{B}^d_{1/2}))$ with $0\leq r\leq(d+1/2)$, $n\sim\ell^{d-1}$, and $0<\tau\leq\ell^{-5}$, then with confidence $1-C_1\ell^{-A}$, there holds

$$\inf_{g\in\mathcal{H}_{\ell,n,\tau}}\|f-g\|_{L^2(\mathbb{B}^d_{1/2})}\leq C_2(A\log\ell)^{\frac{d+1}{2}}\ell^{-r}\|f\|_{W^r(L^2(\mathbb{B}^d_{1/2}))}$$

where $C_1$ and $C_2$ are constants depending only on $r$ and $d$.

Theorem 1 presents the expressivity of $\mathcal{H}_{\ell,n,\tau}$ constructed in (6). Neglecting the logarithmic term, the approximation rate of shallow ReLU nets with random sketching can achieve $\mathcal{O}(\ell^{-r})$. Noting that $n\sim\ell^{d-1}$, there are totally $n\ell\sim\ell^d$ neurons in $\mathcal{H}_{\ell,n,\tau}$. Thus, the approximation rate reaches the linear width [48], and $\mathcal{H}_{\ell,n,\tau}$ is one of the best linear approximants for approximating functions in $W^r(L^2(\mathbb{B}^d_{1/2}))$.

Compared with shallow nets with other popular activation functions [49], [50], the established approximation rate is also almost optimal. In particular, it can be found in [49] that to approximate functions in $W^r(L^2(\mathbb{B}^d_{1/2}))$, the best approximation rate of shallow nets with some widely used activation functions is $\mathcal{O}(N^{-r/d})$, where $N$ denotes the number of neurons. With these, we find that random sketching does not degenerate the approximation capability of shallow nets and, thus, is feasible and efficient.

### B. Learning Performances

We analyze the learning performance of Algorithm 1 in the framework of statistical learning theory [22], where $D=\{x_i,y_i\}_{i=1}^m$ are assumed to be drawn independently according to an unknown joint distribution $\rho:=\rho(x,y)=\rho_X(x)\rho(y|x)$ with $\rho_X$ the marginal distribution and $\rho(y|x)$ the conditional distribution on $\mathcal{Z}:=\mathcal{X}\times\mathcal{Y}$. The performance of $f_{D,\ell,n,\tau}$ is then measured by the generalization error $\mathcal{E}(f_{D,\ell,n,\tau}):=\int_{\mathcal{Z}}(f_{D,\ell,n,\tau}(x)-y)^2d\rho$. Since the regression function defined by $f_\rho(x):=\int_{\mathcal{Y}}yd\rho(y|x)$ minimizes the generalization error, the performance of $f_{D,\ell,n,\tau}$ can be reflected by

$$\mathcal{E}\left(f_{D,\ell,n,\tau}\right)-\mathcal{E}\left(f_\rho\right)=\|f_{D,\ell,n,\tau}-f_\rho\|^2_\rho. \tag{10}$$

In particular, if $y_i=f(x_i)+\epsilon_i$ with the Gaussian noise $\epsilon_i$, we have $f_\rho(x)=f(x)$. Thus, the widely used Gaussian noise assumption is a special case for the aforementioned learning theory framework.

Besides the smoothness assumption shown in the previous subsection, we need another distortion assumption for the marginal distribution. Let $J$ be the identity mapping

$$L^2\left(\mathbb{B}^d_{1/2}\right)\xrightarrow{J}L^2_{\rho_X}$$

and $D_{\rho_X}=\|J\|$. $D_{\rho_X}$ is called the distortion of $\rho_X$, which measures how much $\rho_X$ distorts the Lebesgue measure. The distortion assumption, i.e., $D_{\rho_X}<\infty$, has been widely used

in learning theory [7], [35], [51]. It is obvious that the uniform distribution satisfies the distortion assumption. Based on the smoothness assumption for the regression function and distortion assumption for the marginal distribution, we present our second main result on generalization error analysis for $f_{D,\ell,n,\tau}$ in the following theorem.

*Theorem 2:* Let $d \geq 2$, $\beta \geq 1$, and $0 < r \leq \frac{d+1}{2}$. If $n \sim \ell^{d-1}$, $\ell \sim m^{\frac{1}{d+2r}}$, and $0 < \tau \leq \ell^{-5}$, then there exist constants $C_3$, $C_4$, and $C_5$ depending only on $d, r, M$, and $f_\rho$ such that with confidence $1 - C_3\, m^{-\beta}$, there holds

$$C_4 m^{-\frac{2r}{2r+d}} \leq \sup_{f_\rho \in W^r\left(L^2\left(\mathbb{B}_{1/2}^d\right)\right)} E_{\rho^m}\left(\| f_\rho - f_{D,l,n,\tau} \|_\rho^2\right) \leq C_5 D_{\rho_X}^2 (\beta \log m)^{d+1} m^{-\frac{2r}{2r+d}}. \quad (11)$$

The proof of Theorem 2 is presented in Section VI. Theorem 2 involves two types of randomness: the randomness of data and randomness of the sketching strategy. Since we are interested in the role of random sketching, the randomness of data is reflected by the expectation with high probability. Neglecting the logarithmic term, the derived learning rates for random sketching are optimal. Furthermore, it can be found in [37, Ch. 3] that these optimal learning rates cannot be further improved by using any other learning algorithms, showing that Algorithm 1 is one of the most powerful learning algorithms in embodying the smoothness of $f_\rho$ and distortion of $\rho_X$.

## IV. Experimental Results

In this section, we present both toy simulations and real-world experiments to show the efficiency and effectiveness of the proposed method compared with 12 BP-type algorithms [56], including the gradient descent (GD) BP, GD with momentum (GDM) BP, GD with adaptive learning rate BP (GDA), GDM and adaptive learning rate BP (GDX), resilient BP (RP), CG-BP with Fletcher–Reeves (CGF) updates, CG-BP with Polak–Ribiére (CGP) updates, CG-BP with Powell–Beale (CGB) restarts, scaled CG (SCG) BP, BFGS quasi-Newton BP (BFG), one-step secant (OSS) BP, and Levenberg–Marquardt (LM) BP. The first five BP algorithms are based on GD, and the last seven BP algorithms are based on CG descent. All the simulations and experiments are implemented on a Desktop workstation with Intel Core i7-8700K 3.70-GHz CPU, 32-GB RAM, and Windows 10 operating system. The results are recorded via averaging results from 20 individual trails.

### A. Toy Simulations

In this section, we carry out four simulations to verify our theoretical statements. The regression function $f$ is assumed to be known with the form $f(x) = g(r(x))$, where $x \in \mathbb{B}_{1/2}^4$

$$g(r) = \begin{cases} (1-r)^6\left(35r^2 + 18r + 3\right), & \text{if } r \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

where $r(x) = 2.2\|x\|_2$ and $\|x\|_2$ denotes the Euclidean norm of vector $x$. We generate the training set $D = \{x_i, y_i\}_{i=1}^m$, where $\{x_i\}_{i=1}^m$ is independently drawn according to the uniform distribution on the ball with radius of 0.5 and $y_i = f(x_i) + \epsilon$ with the Gaussian noise $\epsilon \sim N(0, \sigma^2)$. The generalization abilities are evaluated by applying resultant estimators to the testing set $D_{\text{test}} = \{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^p$, which is generated similar to $D$ but with a promise that $y_i^{(t)} = f(x_i^{(t)})$.

The simulations are done with four purposes. In the first simulation, we investigate the role of parameters $n$, $\ell$, and $\tau$ for the generalization ability of Algorithm 1. In the second simulation, we aim at presenting comparisons of generalization abilities and training time for all mentioned methods with different sizes of training samples. In the third simulation, we conduct experiments for training samples with different levels of noise. Finally, we study the differences between the ways of parameter selection via cross validation and testing set.

*Simulation 1* In the first simulation, the sizes of the training set and testing set are both set as 3000, and the variance of Gaussian noise is set as $1/10$. We select $n \in \{1, 2, 3, \ldots, 100\}$, $\ell \in \{1, 2, 3, \ldots, 20\}$, and $\tau \in \{0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 15, 20, 25, 30\}$. We try three groups of simulations by fixing one of $n$, $\ell$, and $\tau$ and varying the other two. The simulation results are reported in Fig. 5, from which we can conclude the following three assertions.

1) Numerically speaking, for the parameter $\ell$ in the range $[2, 6]$ and the parameter $\tau$ in the range $[0.05, 0.5]$, there exists a lower bound for the parameter $n$, denoted by $n_B$ ($n_B \approx 20$), larger than which the parameter $n$ is not very sensitive to the generalization ability. From the theoretical side, $n$ is chosen to guarantee the quadratic formula on the sphere for spherical polynomials of order $\ell$. Thus, once $n$ is larger than a certain level, the learning performance of the proposed method is not very sensitive to $n$.
2) The parameter $\tau$ is also not sensitive to the generalization ability for small $n$ and $\ell$.
3) Large $n$ and $\ell$ give the risk of overfitting for all values $\tau$. To further illustrate the relationship between parameters and the generalization ability, we fix two of $n$, $\ell$, and $\tau$ as the optimal values and vary the rest one. The results are shown in Fig. 6. It is clear to see that the optimal $\tau$ is 0.1, and the relation between optimal $n$ and $\ell$ can be approximated as $n = \ell^3$, which verifies the correctness of our theoretical assertions in Theorem 1.

*Simulation 2:* In this simulation, we compare the proposed method with the 12 BP-type algorithms in terms of presenting RMSE and training time for varying sizes of the training set. Based on the numerical results provided in Simulation 1, we select $\ell \in \{1, 2, 3, \ldots, 7\}$, $\tau \in \{0.1, 0.2, 0.3, \ldots, 1\}$ and fix $n = \ell^3$ for the proposed method. For GD-based BP algorithms, numbers neurons and epochs are selected from $\{10, 20, 30, \ldots, 200\}$ and $\{1000, 3000, 5000, 7000\}$, respectively. For CG descent-based BP algorithms, numbers neurons and epochs are selected from $\{10, 20, 30, \ldots, 200\}$ and $\{500, 1000, 1500, 2000\}$, respectively. We record RMSEs and training time by varying the training set size $m \in \{1000, 1500, 2000, \ldots, 10\,000\}$ and fixing the testing set size $p = 3000$. The simulation results are presented in Figs. 7 and 8, where the left figures are comparisons between the proposed method and GD-based BP algorithms and the right figures are comparisons between the proposed method and CG descent-based BP algorithms. From the abovementioned results, we have the following observations.

1) For all methods, RMSE decreases as the size of the training set increases, while training time increases with the size of training set increasing.

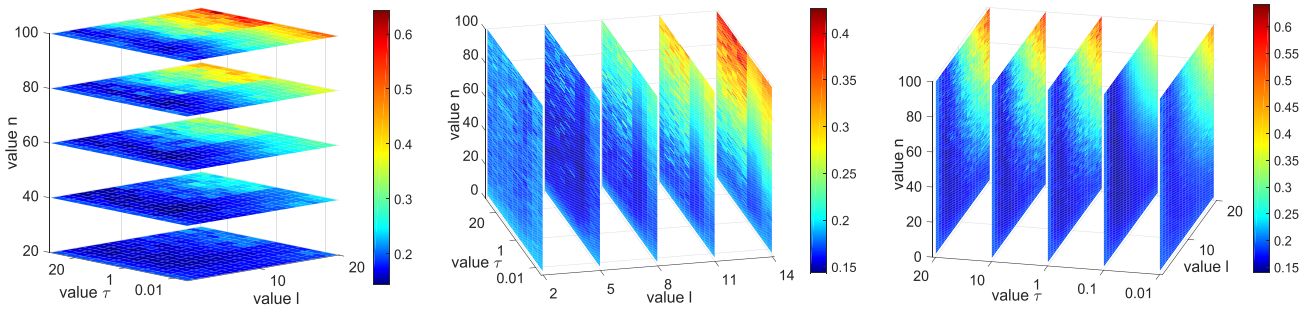


Fig. 5. Relation between RMSE and parameters $n$, $\ell$, and $\tau$.

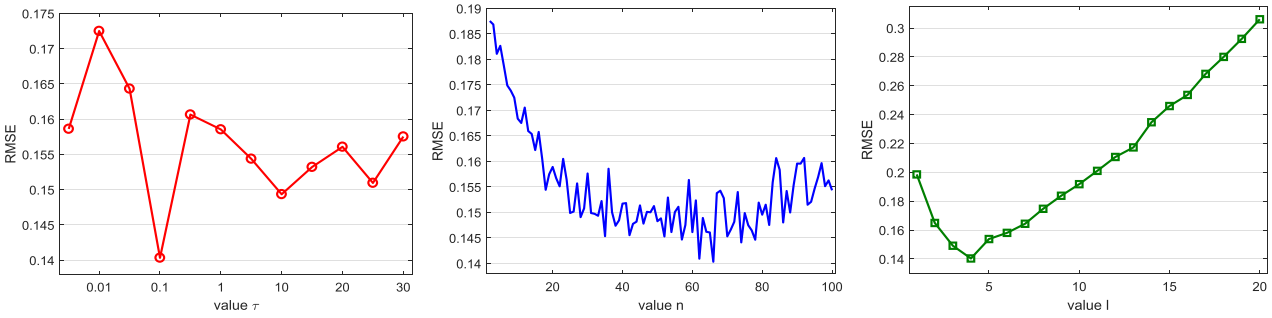


Fig. 6. Relation between RMSE and one parameter by fixing the other two as optimal values. (a) Relation between RMSE and $\tau$. (b) Relation between RMSE and $n$. (c) Relation between RMSE and $\ell$.

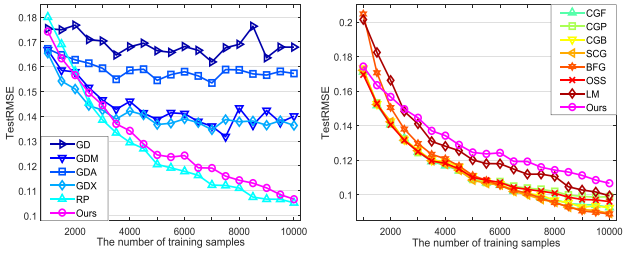


Fig. 7. Relation between RMSE and the number of training samples with the Gaussian noise $\epsilon \sim N(0, 1/10)$.

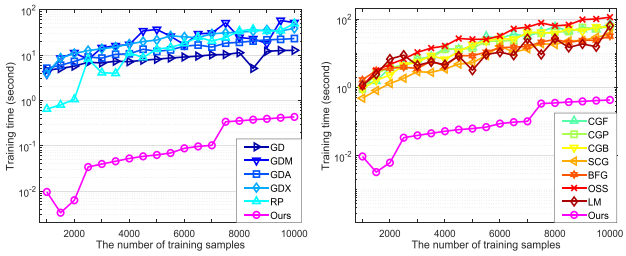


Fig. 8. Relation between the training time and the number of training samples with the Gaussian noise $\epsilon \sim N(0, 1/10)$.

2) RMSEs of CG descent-based BP algorithms are smaller than those of GD-based BP algorithms, especially for the large training set.
3) The proposed method outperforms most of GD-based BP algorithms, but its generalization ability is a little worse than that of CG descent-based BP algorithms.
4) The training time in Fig. 8 shows evidence that the proposed method is much faster than the BP algorithms.

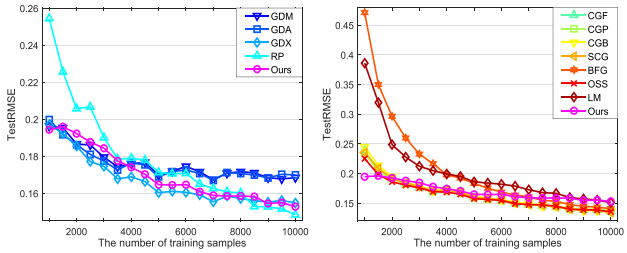


Fig. 9. Relation between RMSE and the number of training samples with the Gaussian noise $\epsilon \sim N(0, 0.49)$.

This is the main focus of this article, which achieves high efficiency in the training stage, while the generalization ability is comparable with the BP algorithms.

*Simulation 3:* In the third simulation, we study the learning performance of the compared methods as the variance of Gaussian noise increases to 0.49. The setting of this simulation is the same as that of Simulation 2. Because the GD algorithm has the exploding gradient issue, it has no record for RMSE and training time. The experimental results are reported in Figs. 9 and 10. It can be observed that CG descent-based BP algorithms do not perform very well when the size of the training set is small. The proposed method also outperforms most of the BP algorithms, and it is much faster than BP algorithms in the training stage.

*Simulation 4:* The optimal parameters in Simulations 2 and 3 are selected with the minimal RMSE on the testing set, which is denoted by TestOpt in the following. In this simulation, we investigate the differences between TestOpt and the commonly used cross validation for the proposed method. The simulation setting is the same as in Simulation 2, and fivefold cross validation is used to select optimal parameters.

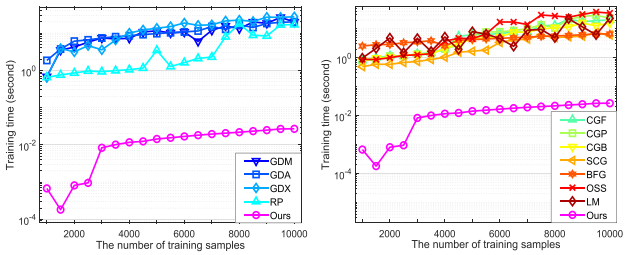


Fig. 10. Relation between training time and the number of training samples with the Gaussian noise $\epsilon \sim N(0, 0.49)$.

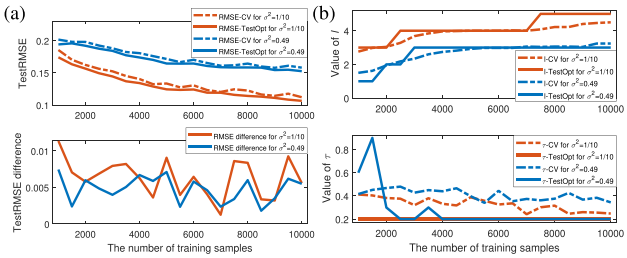


Fig. 11. Comparison between the ways of parameter selection via cross validation and testing set. (a) Comparison of the RMSE on the testing set. (b) Comparison of the optimal values of tunable parameters.

The comparisons of the two parameter selection methods are shown in Fig. 11, where the dashed and solid curves represent the results of cross validation and TestOpt, respectively. The RMSEs of the two parameter selection methods and the differences of RMSEs between the two methods with different sizes of training set and different levels of Gaussian noise are, respectively, reported in the top and bottom rows of Fig. 11(a). It can be seen that the generalization ability of cross validation is very close to that of TestOpt, and the differences of RMSEs are not sensitive to the size of training set and the level of noise. The mean values of the optimal parameters are shown in Fig. 11(b), where the top figure is for parameter $\ell$ and the bottom figure is for parameter $\tau$. It can be found that the optimal $\ell$ curves of cross validation increase smoothly with the size of training set increasing, and they are close to the optimal $\ell$ curves of TestOpt. The optimal $\tau$ of TestOpt stabilizes at 0.2 for both levels of noise when the training set size is larger than 4000, and the optimal $\tau$ curves of cross validation are getting closer and closer to the optimal $\tau$ curves of TestOpt with the increasing size of training set. The reason for the small differences between the two parameter-selection methods is that the inputs of training and testing samples are independently drawn from the same distribution, and the outputs of training samples have zero-mean Gaussian noise.

All the aforementioned simulations verify the correctness of our theoretical assertions in Section III.

### *B. Experiments on Real-World Benchmark Data Sets*

In this section, we carry out experiments on a family of real-world benchmark data sets to show the effectiveness of the proposed method. They include the wide applications in the economic and social fields, such as bank, stock, manufacturing, and service. For example, the data sets `WineRed` and `WineWhite` are related to the red and white variants of Vinho Verde from Portugal. They, respectively, have 1599 and 4898 samples. The inputs include 11 most common physicochemical laboratory tests (such as fixed acidity, volatile acidity, citric acid, pH, and alcohol), and the outputs are the sensory

TABLE I
OVERALL DESCRIPTION OF THE DATA SETS

| Datasets | Attributes | Training # | Testing # |
|---|---|---|---|
| Bank8FM[1] | 8 | 3599 | 900 |
| Delta_elevators[1] | 6 | 7614 | 1903 |
| Delta_ailerons[1] | 5 | 5703 | 1426 |
| WineRed[2] | 11 | 1279 | 320 |
| WineWhite[2] | 11 | 3918 | 980 |
| Stock[1] | 9 | 760 | 190 |
| Boston[3] | 13 | 405 | 101 |
| Abalone[1] | 8 | 3342 | 835 |
| CCPP1[2] | 4 | 7654 | 1914 |
| Bike[2] | 14 | 13903 | 3476 |

tests obtained by the median of at least three evaluations of wine experts. Each expert evaluates the wine quality in the range from 0 (very bad) to 10 (very excellent). These two data sets are used to investigate the relationship between physicochemical laboratory tests and sensory tests. The data set `CCPP1` contains 9568 samples that are collected from a combined cycle power plant over the years 2006–2011. The goal is to predict the net hourly electrical energy of the plant from the input attributes that are consisted of hourly average ambient variables temperature, exhaust vacuum, ambient pressure, and relative humidity. The data set `Bike` is collected from a bike-sharing system in the years 2011 and 2012. The inputs include the environmental and seasonal settings, such as weather conditions, precipitation, day of the week, and season. The outputs are the bike-sharing counts. We use the data set based on an hourly basis, which contains 17 379 samples, to predict the bike rental count hourly for different environmental and seasonal settings. The overall description of these data sets is shown in Table I, in which # stands for the number of samples. All data samples are normalized in the ball centered on the region with radius 1/2. For each data set, 80% data samples are randomly selected as the training samples, and the remaining samples are used as the testing samples for evaluating the performance of the compared methods.

For BP-type algorithms, there are generally two important parameters: numbers neurons and epochs. We use fivefold cross validation on the training set to select the optimal parameters. The number of neurons is selected from $\{10, 20, 30, \ldots, 100\}$, and the number of epochs is, respectively, selected from $\{1000, 3000, 5000, 7000\}$ and $\{500, 1000, 1500, 2000\}$ for GD-based BP algorithms and CG descent-based BP algorithms. In the proposed method, there are three parameters: $n$, $\ell$, and $\tau$. First, for each fixed $\ell \in \{2, 3, \ldots, 10\}$, we perform fivefold cross validation on the training set with $n \in \{10, 20, 30, \ldots, 100\}$ and $\tau \in \{0.1, 0.2, 0.3, \ldots, 1\}$. Second, for each pair $(n, \tau)$, we average the RMSE over $\ell$ and the cross validation and select the optimal $n$ and $\tau$ with the minimal average RMSE. Finally, we average the RMSE over the cross validation for each $\ell$ with the optimal pair $(n, \tau)$ and then select the optimal $\ell$ with the minimal average RMSE. For all the mentioned algorithms, we train the model with the training set by using the optimal parameters and evaluate the performance of the trained model on the testing set. Following the common evaluation procedure, 20 independent sets of training and testing

[1] https://www.dcc.fc.up.pt/%7eltorgo/Regression/
[2] http://archive.ics.uci.edu/ml/datasets.php
[3] https://www.cs.toronto.edu/%7edelve/data/boston/bostonDetail.html

TABLE II
COMPARISONS OF RMSE (std.) AMONG THE 12 BP ALGORITHMS AND THE PROPOSED METHOD

| Datasets | GD | GDM | GDA | GDX | RP | CGF | CGP | CGB | SCG | BFG | OSS | LM | Ours |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bank8FM | 0.054 (0.009) | 0.057 (0.020) | 0.032 (0.001) | 0.032 (0.001) | 0.031 (0.001) | 0.030 (0.001) | 0.030 (0.001) | 0.030 (0.001) | 0.030 (0.001) | 0.030 (0.001) | 0.031 (0.001) | 0.064 (0.107) | 0.035 (0.002) |
| Delta_e | 0.010 (0.005) | 0.010 (0.004) | 0.003 (0.000) | 0.003 (0.000) | 0.003 (0.000) | 0.003 (0.000) | 0.003 (0.000) | 0.003 (0.000) | 0.003 (0.000) | 0.003 (0.000) | 0.003 (0.000) | 0.002 (0.001) | 0.001 (0.000) |
| Delta_a | 0.003 (0.002) | 0.003 (0.001) | 0.002 (0.001) | 0.002 (0.001) | 0.002 (0.001) | 0.002 (0.001) | 0.002 (0.001) | 0.001 (0.001) | 0.002 (0.001) | 0.002 (0.001) | 0.002 (0.001) | 0.002 (0.001) | 0.0002 (0.000) |
| WineRed | —— | 0.823 (0.541) | 0.644 (0.025) | 0.648 (0.027) | 0.662 (0.033) | 0.672 (0.047) | 0.666 (0.038) | 0.675 (0.045) | 0.663 (0.042) | 1.110 (0.652) | 0.657 (0.037) | 1.087 (1.055) | 0.690 (0.119) |
| WineWhite | —— | 0.803 (0.202) | 0.729 (0.024) | 0.720 (0.029) | 0.708 (0.028) | 0.725 (0.059) | 0.712 (0.032) | 0.754 (0.123) | 0.719 (0.040) | 0.766 (0.090) | 0.719 (0.041) | 0.716 (0.026) | 0.790 (0.054) |
| Stock | —— | —— | 1.060 (0.074) | 0.926 (0.060) | 0.802 (0.061) | 0.763 (0.056) | 0.775 (0.059) | 0.772 (0.056) | 0.763 (0.057) | 0.832 (0.066) | 0.784 (0.057) | 0.908 (0.107) | 0.954 (0.165) |
| Boston | —— | —— | 3.606 (0.590) | 3.575 (0.584) | 3.777 (0.644) | 3.795 (0.776) | 3.766 (0.697) | 4.050 (0.818) | 3.941 (0.895) | 4.485 (1.901) | 3.662 (0.595) | 5.060 (2.157) | 5.356 (0.830) |
| Abalone | —— | —— | 2.184 (0.083) | 2.159 (0.082) | 2.134 (0.067) | 2.140 (0.079) | 2.128 (0.070) | 2.143 (0.084) | 2.130 (0.063) | 2.533 (0.807) | 2.145 (0.074) | 3.278 (2.036) | 2.235 (0.175) |
| CCPP1 | —— | —— | 4.250 (0.143) | 4.189 (0.134) | 3.952 (0.139) | 3.965 (0.127) | 3.982 (0.149) | 4.064 (0.309) | 3.911 (0.121) | 4.372 (0.105) | 3.983 (0.134) | 3.876 (0.184) | 4.073 (0.137) |
| Bike | —— | —— | 9.338 (1.641) | 4.989 (2.193) | 1.633 (0.515) | 1.663 (0.428) | 2.137 (0.918) | 1.265 (0.269) | 0.344 (0.102) | 12.18 (3.806) | 1.245 (0.620) | 0.004 (0.001) | 0.015 (0.034) |

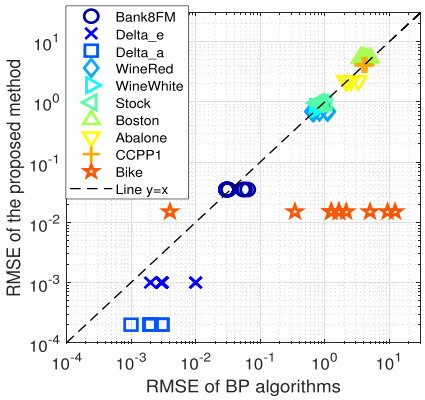


Fig. 12. RMSE of the proposed method versus RMSE of the 12 BP algorithms.

samples are generated by running 20 random divisions for each data set. The average RMSE together with standard deviation is reported in Table II. Table III records the corresponding average training time as well as standard deviation. It should be noted that the training time recorded in Table III is the time cost of training a model with the optimal parameters, and it does not include the parameter-selection procedure. Algorithms GD and GDM have no records of RMSE and training time for some data sets because the exploding gradients problem happens.

To more clearly demonstrate the advantages of the proposed method, the RMSE of the proposed method versus the RMSE of the 12 BP algorithms on the ten data sets is shown graphically in Fig. 12, where the $x$- and $y$-axes, respectively, represent the RMSE of BP algorithms and the RMSE of the proposed method, and different colors represent the results of different data sets. The dashed line with the equation $y = x$ is a baseline to give an intuitive comparison between the BP algorithms and the proposed method. From Table II and Fig. 12, we have the following observations.

1) For large data sets, such as `Delta_elevators`, `Delta_ailerons`, and `Bike`, the proposed method outperforms almost all mentioned BP algorithms.
2) For medium data sets, such as `Bank8FM`, `WineWhite`, and `Abalone`, the proposed method exhibits comparable (or a little superior) generalization performance to BP algorithms.
3) For small data sets, such as `Boston`, `Stock`, and `WineRed`, the generalization ability of the proposed method is a little weaker than BP algorithms.

Hence, we can conclude that the data set size has an important role in the learning results for the proposed method. Large data set is more conductive to good generalization performance for the proposed method compared with BP algorithms. This is because large data set usually requires large $\ell$ (as shown in Theorems 1 and 2), which implies good approximation capability of shallow ReLU nets. As far as the training time is concerned, it also can be seen clearly that the proposed method shows high efficiency in the training stage from Table III.

All these simulations and real-world experiments illustrate that the proposed method significantly reduces the computational complexity of BP algorithms while maintaining comparable generalization performance to BP algorithms.

## V. POTENTIAL EXTENSION FOR DEEP RELU NETS

The random sketching scheme described in Section II presents a computation-reduction technique for shallow ReLU nets. The following Algorithm 2 is an extension of Algorithm 1 for deep ReLU nets.

Although it is difficult to derive similar theoretical guarantees for Algorithm 2, we conduct a simple simulation to show the usability of the random sketching scheme for deep

[4]The generalization performance is not stable on the small-size data sets `Boston`, `Stock`, and `WineRed`; hence, we repeat the experiments 100 times with random divisions on these three data sets for fairness.

TABLE III
COMPARISONS OF TRAINING TIME (std.) AMONG THE 12 BP ALGORITHMS AND THE PROPOSED METHOD

| Datasets | GD | GDM | GDA | GDX | RP | CGF | CGP | CGB | SCG | BFG | OSS | LM | Ours |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bank8FM | 6.965 (1.184) | 6.821 (0.859) | 6.960 (0.953) | 6.831 (0.017) | 5.670 (1.364) | 3.736 (1.805) | 3.615 (0.921) | 3.343 (1.133) | 2.509 (0.704) | 4.532 (2.079) | 6.187 (2.292) | 3.393 (2.207) | 0.011 (0.004) |
| Delta_e | 8.885 (3.952) | 10.541 (0.877) | 0.997 (1.323) | 0.536 (0.146) | 0.058 (0.004) | 0.071 (0.006) | 0.074 (0.012) | 0.069 (0.006) | 0.059 (0.004) | 0.121 (0.044) | 0.121 (0.044) | 0.156 (0.027) | 0.015 (0.011) |
| Delta_a | 2.818 (4.192) | 1.159 (2.344) | 0.088 (0.083) | 0.075 (0.077) | 0.052 (0.003) | 0.054 (0.002) | 0.055 (0.002) | 0.054 (0.003) | 0.050 (0.001) | 0.058 (0.014) | 0.060 (0.019) | 0.058 (0.007) | 0.018 (0.008) |
| WineRed | —— | 2.946 (1.426) | 4.248 (1.065) | 2.656 (1.626) | 1.236 (0.984) | 1.160 (0.370) | 1.172 (0.451) | 1.015 (0.742) | 0.601 (0.196) | 6.340 (4.261) | 1.166 (0.415) | 4.463 (1.971) | 0.008 (0.020) |
| WineWhite | —— | 6.520 (1.901) | 10.210 (2.489) | 9.3799 (2.931) | 5.636 (1.229) | 3.107 (1.633) | 3.140 (1.400) | 1.767 (1.618) | 1.9210 (0.717) | 7.962 (7.581) | 5.302 (3.003) | 5.829 (4.121) | 0.010 (0.004) |
| Stock | —— | —— | 4.594 (0.441) | 5.076 (0.687) | 4.310 (0.913) | 3.503 (1.182) | 3.306 (1.113) | 3.021 (1.252) | 1.730 (0.622) | 2.431 (1.366) | 6.076 (1.588) | 3.172 (1.602) | 0.009 (0.004) |
| Boston | ——- | —— | 3.448 (0.804) | 2.604 (1.197) | 1.154 (0.974) | 0.844 (0.299) | 0.828 (0.235) | 0.876 (0.358) | 0.528 (0.204) | 3.893 (2.892) | 0.986 (0.275) | 3.416 (1.454) | 0.002 (0.001) |
| Abalone | —— | —— | 6.993 (1.199) | 6.093 (2.405) | 3.817 (1.819) | 2.174 (0.970) | 1.960 (1.564) | 2.618 (1.651) | 1.549 (0.765) | 7.255 (6.579) | 4.020 (1.248) | 4.513 (1.655) | 0.011 (0.004) |
| CCPP1 | —— | —— | 10.48 (1.309) | 15.59 (4.491) | 26.23 (4.793) | 40.98 (18.68) | 38.51 (12.73) | 34.35 (22.08) | 26.49 (4.97) | 18.83 (2.25) | 89.91 (20.93) | 23.26 (15.96) | 0.215 (0.082) |
| Bike | —— | —— | 16.56 (2.144) | 17.71 (0.203) | 16.17 (3.451) | 5.513 (2.065) | 7.931 (5.317) | 4.875 (1.109) | 11.38 (2.009) | 30.63 (2.598) | 30.97 (11.70) | 35.36 (30.56) | 0.076 (0.057) |

ReLU nets. This simulation investigates the relation between RMSE and the number of hidden layers for all the mentioned methods. The sizes of training and testing sets are both set as 3000, and the variance of Gaussian noise is set as 0.49. According to the experimental results for parameter selection in Simulation 3, we have the following ranges of the parameter for selection. In the proposed method, the parameters $n$, $\ell$, and $\tau$ of hidden layers are equally selected from $\{2, 4, 6, \ldots, 50\}$, $\{1, 2, 3, \ldots, 5\}$, and $\{0.1, 0.2, 0.3, \ldots, 1\}$, respectively. In GD-based BP algorithms, numbers neurons and epochs of hidden layers are equally selected from $\{2, 4, 6, \ldots, 50\}$ and $\{1000, 3000, 5000, 7000\}$, respectively. In CG descent-based BP algorithms, numbers neurons and epochs of hidden layers are equally selected from $\{2, 4, 6, \ldots, 50\}$ and $\{500, 1000, 1500, 2000\}$, respectively. The average RMSEs and training time with optimal parameters for different numbers $L \in \{1, 2, \ldots, 10\}$ of hidden layers are presented in Figs. 13 and 14, where the left figures are comparisons between Algorithm 2 and GD-based BP algorithms, and the right figures are comparisons between Algorithm 2 and CG descent-based BP algorithms. From these results, it can be seen that Algorithm 2 performs similarly with most of BP algorithms. Although some BP algorithms (e.g., GDA, GDX, CGF, and OSS) outperform Algorithm 2 in terms of RMSE, they are much slower in the training stage. This is the main focus of this article, on a balance between computational burden and generalization ability. Moreover, RMSEs for most of BP algorithms are more sensitive to the number of hidden layers than that of the proposed method.

It should be highlighted that in our construction, deep ReLU nets perform similarly as shallow ReLU nets. This is mainly due to that the random sketching strategy in this article is designed for shallow ReLU nets only. Applying such a sketching strategy directly to deep ReLU nets may not help to take advantage of the benefits of depth. We consider that there are other random sketching schemes suitable to deep nets and will keep studying this topic in the future.

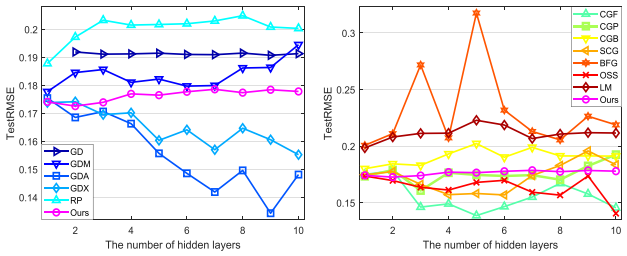


Fig. 13. Relation between RMSE and the number of hidden layers on data with the Gaussian noise $\epsilon \sim N(0, 0.49)$.

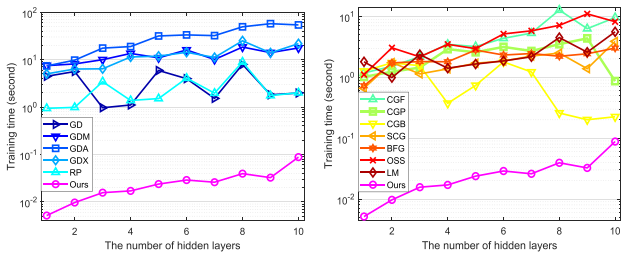


Fig. 14. Relation between training time and the number of hidden layers on data with the Gaussian noise $\epsilon \sim N(0, 0.49)$.

## VI. PROOFS

### A. Approximating Polynomials by ReLU Nets on $[-/2, 1/2]$

Let $G_s^{\nu}(t)$ be the Gegenbauer polynomial [52] with index $\nu$. It is known that the family of polynomials $\{G_s^{\nu}\}_{s=0}^{\infty}$ is a complete orthogonal system in the weighted space $L^2(\mathbb{I}, w_{\nu})$ with $\mathbb{I} := [-1, 1]$ and $w_{\nu}(t) := (1 - t^2)^{\nu - \frac{1}{2}}$, that is

$$\int_{\mathbb{I}} G_{s'}^{\nu}(t) G_s^{\nu}(t) w_{\nu}(t) dt = \begin{cases} 0, & s' \neq s \\ h_{s,\nu}, & s' = s \end{cases}$$

**Algorithm 2** Random Sketching for Deep ReLU Nets

**Input:** Training data set $D = \{(x_i, y_i) : i = 1, \dots, m\}$, number of hidden layers $L$, parameters $n_\kappa, \ell_\kappa \in \mathbb{N}$ and $\tau_\kappa \in (0, 1)$ in the $\kappa$-th ($\kappa = 1, 2, \dots, L$) hidden layer.

Step 1 (Initialization): Normalize $D_x = \{x_i\}_{i=1}^m$ to $\mathbb{B}_{1/2}^d$ and compute $M = \max_{1\le i\le m} |y_i|$. Initialize $z_i^{(0)} = x_i$ ($i = 1, \dots, m$) and $\iota = d$.

**for** $\kappa = 1, 2, \dots, L$ **do**

// Step 2-Step 4 is for the $\kappa$-th hidden layer

Step 2 (Sketching thresholds): Let $t_0^{(\kappa)} = -1/2$, and randomly sketch $\ell_\kappa$ points in $(-1/2, 1/2)$ according to the uniform distribution and rank them as $t_1^{(\kappa)} < t_2^{(\kappa)} < \cdots < t_{\ell_\kappa}^{(\kappa)}$.

Step 3 (Sketching inner weights): Randomly sketch $n_\kappa$ inner weights $\{\alpha_j^{(\kappa)}\}_{j=1}^{n_\kappa}$ according to the uniform distribution on the unit sphere $\mathbb{S}^{\iota-1}$.

Step 4 (Calculating outputs): Compute vector $z_i^{(\kappa)}$ ($i = 1, 2, \dots, m$) with the $p$-th element $[z_i^{(\kappa)}]_p = T_{\tau_\kappa, t_{k-1}^{(\kappa)}, t_k^{(\kappa)}}(\alpha_j^{(\kappa)} \cdot z_i^{(\kappa-1)})$ where $p = (k-1)n_\kappa + j$ for $k = 1, 2, \dots, \ell_\kappa$ and $j = 1, 2, \dots, n_\kappa$. Let $\iota = \ell_\kappa n_\kappa$.

**end for**

Step 5 (Generating hypothesis space): Build up the hypothesis space

$$\mathcal{H} := \left\{a \cdot z^{(L)} : a \in \mathbb{R}^{\ell_L n_L \times 1}\right\}. \tag{12}$$

Step 6 (Least squares): Denote $\bar{D} = \{(z_i^{(L)}, y_i) : i = 1, \dots, m\}$. Define the estimator by

$$f_{\bar{D}}^* := \arg\min_{f\in\mathcal{H}} \frac{1}{m} \sum_{\left(z_i^{(L)}, y_i\right)\in\bar{D}} |f\left(z_i^{(L)}\right) - y_i|^2. \tag{13}$$

and

$$f_{\bar{D}}\left(z^{(L)}\right) := \pi_M f_{\bar{D}}^*\left(z^{(L)}\right). \tag{14}$$

**Output:** Thresholds $\{t_k^{(\kappa)}\}_{k=0}^{\ell_\kappa}$ and inner weights $\{\alpha_j^{(\kappa)}\}_{j=1}^{n_\kappa}$ for the $\kappa$th hidden layer ($\kappa = 1, 2, \dots, L$), and the estimator $f_{\bar{D}}(z^{(L)})$.

where

$$h_{s,\nu} = \frac{\pi^{1/2}(2\nu)_s \Gamma\left(\nu + \frac{1}{2}\right)}{(s+\nu)s!\Gamma(\nu)}$$

and

$$(a)_0 := 0, (a)_s := a(a+1)\cdots(a+s-1) = \frac{\Gamma(a+s)}{\Gamma(a)}.$$

Define

$$U_s := \left(h_{s,d/2}\right)^{-1/2} G_s^{d/2},\ s = 0, 1, \dots \tag{15}$$

Then, $\{U_s\}_{s=0}^\infty$ is a complete orthonormal system for the weighted space $L^2(\mathbb{I}, w)$ with $w(t) := (1-t^2)^{\frac{d-1}{2}}$. Introduce the univariate Sobolev spaces

$$W^\alpha\left(L^2(\mathbb{I}, w)\right) := \left\{g\colon \|g\|_{W^\alpha\left(L^2(\mathbb{I},w)\right)}^2 = \sum_{k=0}^\infty \left[(k+1)^\alpha \hat{g}_{w,k}\right]^2 < \infty\right\}$$

where

$$\hat{g}_{w,k} := \int_{\mathbb{I}} g(t)U_k(t)w(t)dt.$$

It is easy to see that

$$\|p\|_{W^\alpha\left(L^2(\mathbb{I},w)\right)} \le (s+1)^\alpha \|p\|_{L^2(\mathbb{I},w)} \quad \forall\ p \in \mathcal{P}_s(\mathbb{I}) \tag{16}$$

where $\mathcal{P}_s(\mathbb{A})$ denotes the algebraic polynomials defined on the set $\mathbb{A}$ of degrees at most $s$.

Denote by $\mathbb{J} := [-1/2, 1/2]$. For arbitrary $p \in \mathcal{P}_s(\mathbb{J})$, define

$$N_\ell(t) := \sum_{k=0}^{\ell} p(t_k) T_{\tau, t_k, t_{k+1}}(t)/\tau \tag{17}$$

where $t_0 = -1/2$ and $t_{\ell+1} = 1/2$. We can get the following error estimate.

*Lemma 1:* Let $\ell, s \in \mathbb{N}$, and $\tau > 0$. Then

$$\|p - n_\ell\|_{L_2(\mathbb{J})}^2 \le \tilde{c}\left(\varepsilon^2 \|p\|_{W^1(L_2(\mathbb{I},w))}^2 + s^2 \ell\tau \|p\|_{L^2(\mathbb{I},w)}^2\right)$$

holds with confidence at least $1 - \varepsilon^{-1}e^{-\ell\varepsilon}$.

*Proof:* Denote $A_k = [t_k, t_{k+1})$ for $k = 0, \dots, \ell-1$, $A_\ell = [t_\ell, t_{\ell+1}]$, and $h = \max_{0\le k\le \ell} |t_{k+1} - t_k|$. Then

$$\begin{aligned}
\|p - n_\ell\|_{L_2(\mathbb{J})}^2 &= \int_{\mathbb{J}} |p(t) - \sum_{k=0}^{\ell} p(t_k) T_{\tau,t_k,t_{k+1}}(t)/\tau|^2 dt \\
&\le 2\sum_{j=0}^{\ell} \int_{A_j} |p(t) - p\left(t_j\right) T_{\tau,t_j,t_{j+1}}(t)/\tau|^2 dt \\
&\quad + 2\sum_{j=0}^{\ell} \int_{A_j} \left|\sum_{k\ne j} p(t_k) T_{\tau,t_k,t_{k+1}}(t)/\tau\right|^2 dt \\
&=: \mathcal{B}_1 + \mathcal{B}_2.
\end{aligned} \tag{18}$$

It follows from (4) and Hölder's inequality that

$$\begin{aligned}
\mathcal{B}_1 &= 2\sum_{j=0}^{\ell} \int_{A_j} \left|p(t) - p\left(t_j\right)\right|^2 dt = 2\sum_{j=0}^{\ell} \int_{A_j} \left|\int_{t_j}^t p'(u)du\right|^2 dt \\
&\le 2\sum_{j=0}^{\ell} \int_{A_j} \left(t - t_j\right) \int_{t_j}^t \left|p'(u)\right|^2 du dt \\
&\le 2h\sum_{j=0}^{\ell} \int_{A_j} \int_{t_j}^{t_{j+1}} \left|p'(u)\right|^2 du dt \\
&= 2h^2 \sum_{j=0}^{\ell} \int_{t_j}^{t_{j+1}} \left|p'(u)\right|^2 du = 2h^2 \|p'\|_{L_2(\mathbb{J})}^2.
\end{aligned}$$

From [27, p.184], there exists a constant $c_1$ depending only on $d$ such that

$$\|p'_\ell\|_{L_2(\mathbb{J})}^2 \le c_1 \|p\|_{W^1(L_2(\mathbb{I},w))}^2.$$

Thus, we have

$$\mathcal{B}_1 \le 2c_1 h^2 \|p\|_{W^1(L_2(\mathbb{I},w))}^2. \tag{19}$$

Furthermore, (4) yields

$$\mathcal{B}_2 \le 4\sum_{j=1}^{\ell-1} \int_{A_j} \left|p\left(t_{j-1}\right) T_{\tau,t_{j-1},t_j}(t)/\tau\right|^2 dt \tag{20}$$

$$
\begin{aligned}
&+4\sum_{j=1}^{\ell-1}\int_{A_j}\left|p\left(t_{j+1}\right)T_{\tau,t_{j+1},t_{j+2}}(t)/\tau\right|^2dt\\
&+2\int_{A_0}\left|p(t_1)T_{\tau,t_1,t_2}(t)/\tau\right|^2\,dt\\
&+2\int_{A_\ell}\left|p(t_{\ell-1})T_{\tau,t_{\ell-1},t_\ell}(t)/\tau\right|^2dt\\
&\le 4\sum_{j=1}^{\ell-1}\int_{t_j}^{t_j+\tau}\left|p\left(t_{j-1}\right)\right|^2dt+4\sum_{j=1}^{\ell-1}\int_{t_{j+1}-\tau}^{t_{j+1}}\left|p\left(t_{j+1}\right)\right|^2dt\\
&+2\int_{t_1-\tau}^{t_1}|p(t_1)|^2\,dt+2\int_{t_\ell}^{t_\ell+\tau}|p(t_{\ell-1})|^2dt\\
&\le 8\ell\tau\|p\|^2_{L^\infty(\mathbb{J})}. \qquad (21)
\end{aligned}
$$

Due to the well-known Nikolskii inequality for algebraic polynomials [53], we obtain

$$\|p\|^2_{L^\infty(\mathbb{J})}\le c_2s^2\|p\|_{L^2(\mathbb{J})}\le c_2s^22^{(d-1)/2}\|p\|^2_{L^2(\mathbb{I},w)}.$$

Then

$$\mathcal{B}_2\le c_22^{(d+5)/2}s^2\ell\tau\|p\|^2_{L^2(\mathbb{I},w)} \qquad (22)$$

where $c_2$ is an absolute constant. Inserting (19) and (22) into (18), we then have

$$\|p-N_\ell\|^2_{L_2(\mathbb{J})}\le 2c_1h^2\|p\|^2_{W^1(L_2(\mathbb{I},w))}+c_22^{(d+5)/2}s^2\ell\tau\|p\|^2_{L^2(\mathbb{I},w)}. \qquad (23)$$

Finally, as $\{t_i\}_{i=1}^{\ell}$ are drawn independently according to the uniform distribution, it follows from [54, Th. 4.2] that with confidence at least $1-\varepsilon^{-1}e^{-\ell\varepsilon}$, there holds

$$h\le\varepsilon.$$

Plugging this into (23), we have

$$\|p-N_\ell\|^2_{L_2(\mathbb{J})}\le 2c_1\varepsilon^2\|p\|^2_{W^1(L_2(\mathbb{I},w))}+c_22^{(d+5)/2}s^2\ell\tau\|p\|^2_{L^2(\mathbb{I},w)}$$

with confidence $1-\varepsilon^{-1}e^{-\ell\varepsilon}$. This completes the proof of lemma 1 with $\tilde{c}=\max\{2c_1,c_22^{(d+5)/2}\}$. ■

### *B. Preliminaries for Construction*

Denote by $\mathbb{H}_j^{d-1}$ and $\Pi_s^{d-1}$ the class of all spherical harmonics of degree $j$ and the class of all spherical polynomials with total degrees $j\le s$, respectively. It can be found in [52] that $\Pi_s^{d-1}=\bigoplus_{j=0}^s\mathbb{H}_j^{d-1}$. Since the dimension of $\mathbb{H}_j^{d-1}$ is given by

$$D_j^{d-1}:=\dim\,\mathbb{H}_j^{d-1}=\begin{cases}\dfrac{2j+d-2}{j+d-2}\dbinom{j+d-2}{j}, & j\ge 1\\ 1, & j=0\end{cases}$$

the dimension of $\Pi_s^{d-1}$ is $\sum_{j=0}^s D_j^{d-1}=D_s^d\sim s^{d-1}$. Let $\{Y_{j,i}\colon i=1,\ldots,D_j^{d-1}\}$ be an arbitrary orthonormal system of $\mathbb{H}_j^{d-1}$. For $x\in\mathbb{B}^d$, define

$$P_{k,j,i}(x)=v_k\int_{\mathbb{S}^{d-1}}Y_{j,i}(\xi)U_k(x\cdot\xi)d\omega_{d-1}(\xi) \qquad (24)$$

where

$$v_k:=\left(\frac{(k+1)_{d-1}}{2(2\pi)^{d-1}}\right)^{\frac{1}{2}}.$$

Then, it follows from [31] that

$$\left\{P_{k,j,i}\colon k=0,1,\ldots,s,\ j=k,\ k-2,\ldots,\varepsilon_k,\ i=1,2,\ldots,D_j^{d-1}\right\}$$

is an orthonormal basis for $\mathcal{P}_s(\mathbb{B}^d)$ with

$$\varepsilon_k:=\begin{cases}0, & k\ \text{even}\\ 1, & k\ \text{odd}.\end{cases}$$

Based on the orthonormal system, we define the Sobolev space on $\mathbb{B}^d$ by

$$H^r\left(\mathbb{B}^d\right):=\left\{f\colon\|f\|^2_{H^r}=\sum_{k=0}^\infty\sum_{j\in\Xi_k}\sum_{i=1}^{D_j^{d-1}}\left[(k+1)^r\hat{f}_{k,j,i}\right]^2<\infty\right\}$$

where $\hat{f}_{k,j,i}:=\int_{\mathbb{B}^d}f(x)P_{k,j,i}(x)dx$ and $\Xi_k:=\{k,k-2,\ldots,\varepsilon_k\}$.

For $s\ge 0$ and $n\ge 1$, let a discretization quadrature rule

$$\mathcal{Q}(s,n):=\left\{\left(\lambda_j,\xi_j\right):j=1,\ldots,n\right\},\quad\lambda_j>0 \qquad (25)$$

holds exact for $\Pi_s^{d-1}$, that is

$$\int_{\mathbb{S}^{d-1}}P(\xi)d\omega_{d-1}(\xi)=\sum_{j=1}^n\lambda_jP\left(\xi_j\right)\quad\forall P\in\Pi_s^{d-1}.$$

We then present the first tool for the construction, which can be founded in [26].

*Lemma 2:* Let $s\in\mathbb{N}$. If $\mathcal{Q}(2s,n)=\{\lambda_j,\xi_j\}_{j=1}^n$ is a discretization quadrature rule for $\Pi_{2s}^{d-1}$, then for arbitrary $P\in\mathcal{P}_s(\mathbb{B}^d)$, there holds

$$P(x)=\sum_{j=1}^n\lambda_jp_j\left(\xi_j\cdot x\right) \qquad (26)$$

and

$$\sum_{j=1}^n\lambda_j\|p_j\|^2_{W^1\left(L^2(\mathbb{I},w)\right)}\le c_3\|P\|^2_{H^{\frac{d+1}{2}}\left(L^2\left(\mathbb{B}^d\right)\right)} \qquad (27)$$

where

$$p_j(t):=\sum_{k=0}^s v_k^2\int_{\mathbb{B}^d}U_k\left(\xi_j\cdot y\right)P(y)dyU_k(t) \qquad (28)$$

and $c_3$ is a constant depending only on $d$.

Our second tool is a cubature formula on the sphere with respect to random sampling, which can be easily derived by combining [34] with [54, Th. 4.2].

*Lemma 3:* If $\Lambda_n:=\{\xi_j\}_{j=1}^n$ is a sequence of i.i.d. random variables uniformly distributed from $\mathbf{S}^{d-1}$, then for every $n$, with confidence $1-c_4s^{d-1}e^{-c_4ns^{1-d}}$, there exists a set of positive numbers $\{\lambda_j\}_{j=1}^n$ satisfying

$$\sum_{j=1}^n\lambda_j^2\le\frac{c_5}{n}$$

such that

$$\int_{\mathbb{S}^{d-1}}P(\xi)d\omega(\xi)=\sum_{j=1}^N\lambda_jP\left(\xi_j\right)\quad\forall\ P\in\Pi_s^{d-1}.$$

Our final tool is the following lemma, which can be found in [27].

*Lemma 4:* Let $H$ be a Hilbert space with norm $\|\cdot\|$ and let $A, B \subset H$ be finite-dimensional linear subspaces of $H$ with $\dim A \le \dim B$. If there exists a $\delta \in (0, 1/2)$ such that

$$\sup_{x\in A,\ \|x\|\le 1} \inf_{y\in B} \|x - y\| \le \delta$$

then there is a constant $c$ depending only on $\delta$ and a linear operator $L: A \to B$ such that for every $x \in A$

$$\|Lx - x\| \le c \inf_{y\in B} \|x - y\|$$

and

$$Lx - x \perp A \ (Lx - x \text{ is orthogonal to } A).$$

### C. Constructing ReLU Nets

For arbitrary $f \in W^r(L^2(\mathbb{B}^d_{1/2}))$, we, at first, extend $f$ to a function $f^*$ defined on $\mathbb{B}^d$ [55, Ch. IV] (see also [27]) such that $f^*$ vanishes outside of $\mathbb{B}^d_{3/4}$ and

$$\|f^*\|_{H^r(L^2(\mathbb{B}^d))} \le c' \|f\|_{W^r(L^2(\mathbb{B}^d_{1/2}))} \tag{29}$$

where $c' > 0$ is a constant depending only on $d$. Define

$$P_s(x) = \sum_{k=0}^{s} \sum_{j\in\Xi_k} \sum_{i=1}^{D_j^{d-1}} \hat{f}^*_{k,j,i} P_{k,j,i}(x). \tag{30}$$

Then

$$\begin{aligned}
&\|f^* - P_s\|^2_{L^2(\mathbb{B}^d)} \\
&\quad= \sum_{k=s+1}^{\infty} \sum_{j\in\Xi_k} \sum_{i=1}^{D_j^{d-1}} |\hat{f}^*_{k,j,i}|^2 \\
&\quad\le s^{-2r} \sum_{k=s+1}^{\infty} \sum_{j\in\Xi_k} \sum_{i=1}^{D_j^{d-1}} \left[(k+1)^r |\hat{f}^*_{k,j,i}|\right]^2 \\
&\quad\le s^{-2r} \|f^*\|^2_{W^r(L^2(\mathbb{B}^d))} \le \left(c'\right)^2 s^{-2r} \|f\|^2_{W^r(L^2(\mathbb{B}^d_{1/2}))}.
\end{aligned} \tag{31}$$

Due to Lemma 2, we get

$$P_s(x) = \sum_{j=1}^{n} \lambda_j p_j(\xi_j \cdot x) \tag{32}$$

where $p_j$ is defined by (28). Define

$$\psi_{k,j}(t) := \begin{cases} T_{\tau,t_k,t_{k+1}}(t), & t \in \mathbb{J} \\ t^j, & t \in \mathbb{I}\backslash\mathbb{J}. \end{cases}$$

Once $\{t_j\}_{j=1}^{\ell}$ is randomly sampled, the set

$$\Phi_\ell := \left\{ \sum_{k=0}^{\ell-1} \sum_{j=0}^{\ell-1} c_{k,j} \psi_{k,j}(t) \colon c_{k,j} \in \mathbb{R} \right\} \tag{33}$$

is an $\ell^2$-dimensional linear space. Set $H = L^2(\mathbb{I}, w)$, $A = \mathcal{P}_s(\mathbb{I})$, and $B = \Phi_\ell$ in Lemma 4. It follows from Lemma 1 that with confidence $1 - \varepsilon^{-1}e^{-\ell\varepsilon}$, there holds:

$$\begin{aligned}
&\sup_{p\in A} \inf_{g\in B} \|p - g\|^2_{L_2(\mathbb{I},w)} \\
&\quad\le \sup_{p\in A} \|p - n_\ell\|^2_{L_2(\mathbb{I},w)} \le \sup_{p\in A} \|p - n_\ell\|^2_{L_2(\mathbb{J})} \\
&\quad\le \tilde{c}\left(\varepsilon^2 \|p\|^2_{W^1(L_2(\mathbb{I},w))} + s^2\ell\tau \|p\|^2_{L^2(\mathbb{I},w)}\right)
\end{aligned} \tag{34}$$

which together with the Bernstein inequality (16) shows that

$$\begin{aligned}
&\sup_{p\in A} \inf_{g\in B} \|p - g\|^2_{L_2(\mathbb{I},w)} \\
&\quad\le \tilde{c}\left((s+1)^2\varepsilon^2 \|p\|^2_{L_2(\mathbb{I},w)} + s^2\ell\tau \|p\|^2_{L^2(\mathbb{I},w)}\right)
\end{aligned}$$

holds with confidence $1 - \varepsilon^{-1}e^{-\ell\varepsilon}$. Hence, for $\ell \ge s$, $\tau \le s^{-2}\varepsilon^2\ell^{-1}$, and $\varepsilon \le (\sqrt{2\tilde{c}}/4(s+1))$, with confidence $1 - \varepsilon^{-1}e^{-\ell\varepsilon}$, there holds

$$\sup_{p\in A, \|p\|_{L_2(\mathbb{I},w)}\le 1} \inf_{g\in B} \|p - g\|_{L_2(\mathbb{I},w)} \le \frac{1}{4}.$$

This verifies the condition of Lemma 4 with $\delta = 1/4$. Then, it follows from Lemma 4 and (34) that for arbitrary $j = 1, 2, \ldots, N$, there is a

$$q_j(t) = \begin{cases} \displaystyle\sum_{k=0}^{\ell-1} a_{k,j} T_{\tau,t_k,t_{k+1}}(t), & t \in \mathbb{J}, \\ \displaystyle\sum_{j=0}^{\ell-1} a'_{j,j'} t^j, & t \in \mathbb{I}\backslash\mathbb{J}, \end{cases} \in \Phi_\ell \tag{35}$$

for some sequences $\{a'_{j,j'}\}_{j=0}^{\ell-1}$ and $\{a_{k,j}\}_{k=1}^{\ell-1}$ such that

$$\|q_j - p_j\|^2_{L_2(\mathbb{I},w)} \le c\tilde{c}\left(\varepsilon^2 \|p\|^2_{W^1(L_2(\mathbb{I},w))} + s^2\ell\tau \|p\|^2_{L^2(\mathbb{I},w)}\right)$$

and

$$q_j(t) - p_j(t) = \sum_{k=s+1}^{\infty} \hat{q}_j(k, w) U_k(t) \tag{36}$$

holds with confidence $1 - \varepsilon^{-1}e^{-\ell\varepsilon}$. The abovementioned two estimates together with $\tau \le \varepsilon^2 s^{-2}\ell^{-1}$ show that for $\varepsilon \le (\sqrt{2\tilde{c}}/4(s+1))$ and $s \le \ell$, with confidence $1 - \varepsilon^{-1}e^{-\ell\varepsilon}$, there holds

$$\sum_{k=s+1}^{\infty} |\hat{q}_\ell(k, w)|^2 \le c_6 \varepsilon^2 \|p_\ell\|^2_{W^1(L_2(\mathbb{I},w))} \tag{37}$$

where $c_6 := 2c\tilde{c}$.

With these helps, we define

$$\begin{aligned}
Q(x) &:= \sum_{j=1}^{n} \lambda_j q_j(\xi_j \cdot x) \\
&= \sum_{j=1}^{n} \lambda_j \begin{cases} \displaystyle\sum_{k=0}^{\ell-1} a_{k,j} T_{\tau,t_k,t_{k+1}}(\xi_j \cdot x), & \xi_j \cdot x \in \mathbb{J} \\ \displaystyle\sum_{j'=0}^{\ell-1} a'_{j',j} (\xi_j \cdot x)^{j'}, & \xi_j \cdot x \in \mathbb{I}\backslash\mathbb{J}. \end{cases}
\end{aligned} \tag{38}$$

Since $\xi_j \in \mathbb{S}^{d-1}$ for $j = 1, \ldots, n$, $x \in \mathbb{B}^d_{1/2}$ implies $\xi_j \cdot x \in \mathbb{J}$, and thus

$$Q^*(x) = \sum_{j=1}^{n} \lambda_j \sum_{k=0}^{\ell-1} a_{k,j} T_{\tau,t_k,t_{k+1}}(\xi_j \cdot x) \in \mathcal{H}_{l,n,\tau} \quad \forall\ x \in \mathbb{B}^d_{1/2}. \tag{39}$$

We then have from (32) and (36) that

$$\begin{aligned}
Q(x) - P_s(x) &= \sum_{j=1}^{n} \lambda_j \left(q_j(\xi_j \cdot x) - p_j(\xi_j \cdot x)\right) \\
&= \sum_{j=1}^{n} \lambda_j \sum_{k=s+1}^{\infty} \hat{q}_j(k, w) U_k(\xi_j \cdot x).
\end{aligned}$$

Denote

$$Q_k(x) := \sum_{\ell=1}^{N} \lambda_\ell \hat{q}_\ell(k, w) U_k(\xi_\ell \cdot x).$$

The following lemma derives the bound of $\|Q_k\|_{L^2(\mathbb{B}^d)}$, the proof of which is the same as that in [26] by taking place of the cubature formula based on fixed sampling by Lemma 3.

*Lemma 5:* Let $Q_k$ be defined as earlier. Then, with confidence at least $1 - c_4 s^{d-1} e^{-c_4 n s^{1-d}}$, there holds

$$\|Q_k\|^2_{L^2(\mathbb{B}^d)} \le \upsilon_k^{-2} \sum_{j=1}^{n} \lambda_j |\hat{q}_j(k, w)|^2. \tag{40}$$

### D. Proof of Theorem 1

In this section, we prove Theorem 1 as follows.

*Proof:* It can be found in [26, eq. (46)] that

$$\|Q - P_s\|^2_{L^2(\mathbb{B}^d)} = \sum_{k=s+1}^{\infty} \|Q_k\|^2_{L^2(\mathbb{B}^d)}. \tag{41}$$

Plugging (40) into (41), with confidence $1 - c_4 s^{d-1} e^{-c_4 n s^{1-d}}$, there holds

$$\begin{aligned}\|Q - P_s\|^2_{L^2(\mathbb{B}^d)} &\le \sum_{k=s+1}^{\infty} \upsilon_k^{-2} \sum_{j=1}^{n} \lambda_j |\hat{q}_j(k, w)|^2 \\ &\le s^{1-d} \sum_{k=s+1}^{\infty} \sum_{j=1}^{n} \lambda_j |\hat{q}_j(k, w)|^2.\end{aligned}$$

This together with (37) shows that for $\varepsilon \le ((2\tilde{c})^{1/2}/4(s+1))$ and $s \le \ell$, with confidence

$$1 - c_4 s^{d-1} e^{-c_4 n s^{1-d}} - \varepsilon^{-1} e^{-\ell\varepsilon}$$

there holds

$$\|Q - P_s\|^2_{L^2(\mathbb{B}^d)} \le c_6 s^{1-d} \varepsilon^2 \sum_{j=1}^{n} \lambda_j \|p_j\|^2_{W^1(L_2(\mathbb{I}, w))}.$$

Hence, we obtain from (27) that

$$\|Q - P_s\|^2_{L^2(\mathbb{B}^d)} \le c_3 c_6 s^{1-d} \varepsilon^2 \|P_s\|^2_{W^{\frac{d+1}{2}}(L^2(\mathbb{B}^d))}.$$

For some $A \ge 1$, set $s = (\ell/(A + d + c_4)\log \ell)$, $n = \ell^{d-1}$, and $\varepsilon = (\sqrt{\tilde{c}}/4s)$. With confidence $1 - c_7 \ell^{-A}$, there holds

$$\|Q - P_s\|_{L^2(\mathbb{B}^d)} \le c_8 A^{\frac{d-1}{2}} \ell^{-\frac{d+1}{2}} (\log \ell)^{\frac{d+1}{2}} \|P_s\|_{W^{\frac{d+1}{2}}(L^2(\mathbb{B}^d))}$$

where $c_7$ and $c_8$ are constants depending only on $d$ and $r$. Then, (30) and (29) yield

$$\begin{aligned}&\|Q - P_s\|_{L^2(\mathbb{B}^d)} \\ &\quad \le c_8 A^{\frac{d-1}{2}} \ell^{-\frac{d+1}{2}} (\log \ell)^{\frac{d+1}{2}} \|P_s\|_{W^{\frac{d+1}{2}}(L^2(\mathbb{B}^d))} \\ &\quad \le c' c_8 A^{\frac{d-1}{2}} \ell^{-\frac{d+1}{2}} (\log \ell)^{\frac{d+1}{2}} \|f\|_{W^{\frac{d+1}{2}}(L^2(\mathbb{B}^d_{1/2}))}.\end{aligned}$$

Noting further that (31) and $0 < r \le (d + 1/2)$, we have

$$\begin{aligned}\|f^* - Q\|_{L^2(\mathbb{B}^d)} &\le \|f^* - P_s\|_{L^2(\mathbb{B}^d)} + \|P_s - Q\|_{L^2(\mathbb{B}^d)} \\ &\le c_9 (A \log \ell)^{\frac{d+1}{2}} \ell^{-r} \|f\|_{W^r(L^2(\mathbb{B}^d_{1/2}))}\end{aligned}$$

with $c_9 = c'(c_8 + c_8 + d + 1)$. Hence, with confidence $1 - c_7 \ell^{-A}$, there holds

$$\begin{aligned}\|f - Q^*\|_{L^2(\mathbb{B}^d_{1/2})} &\le \|f^* - Q\|_{L^2(\mathbb{B}^d)} \\ &\le c_9 (A \log \ell)^{\frac{d+1}{2}} \ell^{-r} \|f\|_{W^r(L^2(\mathbb{B}^d_{1/2}))}.\end{aligned}$$

This together with (39) completes the proof of Theorem 1. ■

### E. Proof of Theorem 2

We provide the Proof of Theorem 2 in this section. We need Lemma 6, which was proven in [37, Th. 11.3].

*Lemma 6:* Let $H_m$ be a $u$-dimensional linear space. Define the estimate $f_m$ by

$$f_m := \pi_M \widetilde{f}_m \quad \text{where} \quad \widetilde{f}_m = \arg\min_{f \in H_m} \frac{1}{m} \sum_{i=1}^{m} |f(x_i) - y_i|^2.$$

Then

$$E_{\rho^m}\{\|f_m - f_\rho\|^2_\rho\} \le CM^2 \frac{u \log m}{m} + 8 \inf_{f \in H_m} \|f_\rho - f\|^2_\rho$$

for some universal constant $c$.

Due to Algorithm 1, our algorithm can be regarded as a two-stage learning scheme. The first stage aims at random sketching inner weights and thresholds. Once the sketching process is finished, the second stage focuses on a linear least squares problem. Thus, the randomness only affects the approximation error, as exhibited in Theorem 1. To deduce the upper bound of (2), we combine Theorem 1 with Lemma 6. It can be found from the definition that $\mathcal{H}_{\ell,n,\tau}$ is an $\ell n$-dimensional linear space. Therefore, Lemma 6 implies that

$$E_{\rho^m}\{\|f_{D,\ell,n,\tau} - f_\rho\|^2_\rho\} \le CM^2 \frac{\ell n \log m}{m} + 8 \inf_{g \in \mathcal{H}_{\ell,n,\tau}} \|f_\rho - g\|^2_\rho.$$

Furthermore, it follows from Theorem 1 that with confidence $1 - C_1 \ell^{-A}$, there holds

$$\inf_{g \in \mathcal{H}_{\ell,n,\tau}} \|f_\rho - g\|^2_\rho \le C_2^2 (A \log \ell)^{d+1} \ell^{-2r} \|f_\rho\|^2_{W^r(L^2(\mathbb{B}^d_{1/2}))}.$$

Since $\ell = m^{(1/2r+d)}$, with confidence at least $1 - C_3 m^{-\beta}$ with $\beta = A/(2r + d)$, there holds

$$E_{\rho^m}\{\|f_{D,\ell,n,\tau} - f_\rho\|^2_\rho\} \le C_5 (\beta \log m)^{d+1} m^{-2r/(2r+d)}.$$

Therefore, the upper bound of (11) is deduced. The lower bound of (11) can be found from [37, Th. 3.2]. This finishes the Proof of Theorem 2.

## VII. Conclusion

In this article, we developed a novel random sketching strategy for shallow ReLU nets learning to reduce the computational complexities while maintaining the learning performance of BP algorithms. Our analysis showed that with the help of the trapezoid-shaped function constructed by ReLU, the inner weights of shallow ReLU nets can be drawn independently according to the uniform distribution on the unit sphere, and the thresholds can be drawn independently according to the uniform distribution on an interval. The feasibility of the proposed random sketching scheme was verified by both theoretical assessments and numerical experiments.

## REFERENCES


[1] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] H. Lee, P. T. Pham, L. Yan, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 1096–1104.

[4] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[5] C. K. Chui, X. Li, and H. N. Mhaskar, "Neural networks for localized approximation," *Math. Comput.*, vol. 63, no. 208, p. 607, 1994.

[6] C. Schwab and J. Zech, "Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ," *Anal. Appl.*, vol. 17, no. 1, pp. 19–55, Jan. 2019.

[7] S.-B. Lin, "Generalization and expressivity for deep nets," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1392–1406, May 2019.

[8] H. N. Mhaskar and T. Poggio, "Deep vs. Shallow networks: An approximation theory perspective," *Anal. Appl.*, vol. 14, no. 6, pp. 829–848, Nov. 2016.

[9] M. Kohler and A. Krzyzak, "Nonparametric regression based on hierarchical interaction models," *IEEE Trans. Inf. Theory*, vol. 63, no. 3, pp. 1620–1630, Mar. 2017.

[10] C. K. Chui, S.-B. Lin, and D.-X. Zhou, "Construction of neural networks for realization of localized deep learning," *Frontiers Appl. Math. Statist.*, vol. 4, pp. 14–35, May 2018.

[11] U. Shaham, A. Cloninger, and R. R. Coifman, "Provable approximation properties for deep neural networks," *Appl. Comput. Harmon. Anal.*, vol. 44, no. 3, pp. 537–557, May 2018.

[12] C. K. Chui, S.-B. Lin, and D.-X. Zhou, "Deep neural networks for rotation-invariance approximation and learning," *Anal. Appl.*, vol. 17, no. 05, pp. 737–772, Sep. 2019.

[13] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[14] Z.-C. Guo, L. Shi, and S.-B. Lin, "Realizing data features by deep nets," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 5, 2019, doi: 10.1109/TNNLS.2019.2951788.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[16] D. Davis, D. Drusvyatskiy, S. Kakade, and J. D. Lee, "Stochastic subgradient method converges on tame functions," *Found. Comput. Math.*, vol. 20, no. 1, pp. 119–154, Feb. 2020, doi: 10.1007/s10208-018-09409-5.

[17] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 265–272.

[18] J. Zeng, S.-B. Lin, and Y. Yao, "A convergence analysis of nonlinearly constrained ADMM in deep learning," 2019, *arXiv:1902.02060*. [Online]. Available: http://arxiv.org/abs/1902.02060

[19] J. Zeng, T. T. K. Lau, S. B. Lin, and Y. Yao, "Global convergence of block coordinate descent in deep learning," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 1–27.

[20] I. Safran and O. Shamir, "Spurious local minima are common in two-layer ReLU neural networks," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1–29.

[21] Y. Li and Y. Liang, "Learning overparameterized neural networks via stochastic gradient descent on structured data," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8157–8166.

[22] F. Cucker and D. X. Zhou, *Learning Theory: An Approximation Theory Viewpoint*. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[23] G. Montavon, G. B. Orr, and K. R. Müller. *Neural Networks: Tricks of The Trade*. vol. 7700. Berlin, Germany: Springer, 2012.

[24] C. K. I. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 13, 2000, pp. 682–688.

[25] A. Gittens and M. W. Mahoney, "Revisiting the Nyström method for improved large-scale machine learning," *J. Mach. Learn. Res.*, vol. 17, pp. 1–65, 2016.

[26] J. Fang, S. Lin, and Z. Xu, "Learning through deterministic assignment of hidden parameters," *IEEE Trans. Cybern.*, early access, Dec. 20, 2018, doi: 10.1109/TCYB.2018.2885029.

[27] P. P. Petrushev, "Approximation by ridge functions and neural networks," *SIAM J. Math. Anal.*, vol. 30, no. 1, pp. 155–189, Jan. 1998.

[28] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1177–1184.

[29] M. W. Mahoney, "Lecture notes on randomized linear algebra," 2016, *arXiv:1608.04481*. [Online]. Available: http://arxiv.org/abs/1608.04481

[30] M. Unser, "A representer theorem for deep neural networks," 2018, *arXiv:1802.09210*. [Online]. Available: http://arxiv.org/abs/1802.09210

[31] V. E. Maiorov, "Representation of polynomials by linear combinations of radial basis functions," *Constructive Approx.*, vol. 37, no. 2, pp. 283–293, Apr. 2013.

[32] Q. T. L. Gia and H. N. Mhaskar, "Localized linear polynomial operators and quadrature formulas on the sphere," *SIAM J. Numer. Anal.*, vol. 47, no. 1, pp. 440–466, Jan. 2009.

[33] S.-B. Lin, "Nonparametric regression using needlet kernels for spherical data," *J. Complex.*, vol. 50, pp. 66–83, Feb. 2019.

[34] H. N. Mhaskar, F. J. Narcowich, and J. D. Ward, "Spherical Marcinkiewicz–Zygmund inequalities and positive quadrature," *Math. Comput.*, vol. 70, no. 235, pp. 1113–1131, 2001.

[35] D.-X. Zhou and K. Jetter, "Approximation with polynomial kernels and SVM classifiers," *Adv. Comput. Math.*, vol. 25, nos. 1–3, pp. 323–344, Jul. 2006.

[36] I. Steinwart and A. Christmann, *Support Vector Machines*. New York, NY, USA: Springer, 2008.

[37] L. Györfy, M. Kohler, A. Krzyzak, and H. Walk, *A Distribution-Free Theory of Nonparametric Regression*. Berlin, Germany: Springer, 2002.

[38] S.-B. Lin, "Limitations of shallow nets approximation," *Neural Netw.*, vol. 94, pp. 96–102, Oct. 2017.

[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[40] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 1951.

[41] Y. Tian, "An analytical formula of population gradient for two-layered ReLU network and its applications in convergence and critical point analysis," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3404–3413.

[42] Y. Li and Y. Yuan, "Convergence analysis of two-layer neural networks with ReLU activation," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017, pp. 597–607.

[43] S. Du, J. Lee, Y. Tian, B. Poczós, and A. Singh, "Gradient descent learns one-hiddenlayer CNN: Don't be afraid of spurious local minima," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1339–1348.

[44] S. Du, X. Zhai, B. Poczós, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," in *Proc. 7th Int. Conf. Learn. Represent.*, 2019, pp. 1–19.

[45] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 1–53.

[46] P. Petersen and F. Voigtlaender, "Optimal approximation of piecewise smooth functions using deep ReLU neural networks," *Neural Netw.*, vol. 108, pp. 296–330, Dec. 2018.

[47] S.-B. Lin and D.-X. Zhou, "Distributed kernel-based gradient descent algorithms," *Constructive Approx.*, vol. 47, no. 2, pp. 249–276, Apr. 2018.

[48] A. Pinkus, *N-Widths in Approximation Theory*, vol. 7. Berlin, Germany: Springer, 2012.

[49] V. E. Maiorov and R. Meir, "On the near optimality of the stochastic approximation of smooth functions by neural networks," *Adv. Comput. Math.*, vol. 13, pp. 79–103, Apr. 2000.

[50] S. Lin, J. Zeng, and X. Zhang, "Constructive neural network learning," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 221–232, Jan. 2019.

[51] L. Shi, Y. L. Feng, and D. X. Zhou, "Concentration estimates for learning with $\ell_1$-regularizer and data dependent hypothesis spaces," *Appl. Comput. Harmon. Anal.*, vol. 31, no. 2, pp. 286–302, 2011.

[52] K. Wang and L. Li, *Harmonic Analysis and Approximation on the Unit Sphere*. Beijing, China: Science Press, 2000.

[53] P. Borwein and T. E. Erdélyi, *Polynomials and Polynomial Inequalities*, vol. 161. Berlin, Germany: Springer, 2012.

[54] R. F. Bass and K. Gröchenig, "Random sampling of multivariate trigonometric polynomials," *SIAM J. Math. Anal.*, vol. 36, no. 3, pp. 773–795, Jan. 2005.

[55] R. Adams, *Sobolev Spaces*. New York, NY, USA: Academic, 1975.

[56] H. Demuth, M. Beale, and M. Hagan. Neural network toolbox TM 6. MathWorks, Natick, MA, USA. Accessed: 2010. [Online]. Available: https://kashanu.ac.ir/Files/Content/neural_network_toolbox_6.pdf