Scalable Subsampling Inference for Deep Neural Networks

KEJIN WU, Department of Mathematics, University of California San Diego, USA
DIMITRIS N. POLITIS, Department of Mathematics and Halicioğlu Data Science Institute, University of California San Diego, USA

Deep neural networks (DNN) has received increasing attention in machine learning applications in the last several years. Recently, a non-asymptotic error bound has been developed to measure the performance of the fully connected DNN estimator with ReLU activation functions for estimating regression models. The paper at hand gives a small improvement on the current error bound based on the latest results on the approximation ability of (forward) DNN. More importantly, however, a non-random subsampling technique–scalable subsampling–is applied to construct a 'subagged' DNN estimator. Under regularity conditions, it is shown that the subagged DNN estimator is computationally efficient without sacrificing accuracy for either estimation or prediction tasks. Beyond point estimation/prediction, we propose different approaches to build confidence and prediction intervals based on the subagged DNN estimator. In addition to being asymptotically valid, the proposed confidence/prediction intervals appear to work well in finite samples. All in all, the scalable subsampling DNN estimator offers the complete package in terms of statistical inference, i.e., (a) computational efficiency; (b) point estimation/prediction accuracy; and (c) allowing for the construction of practically useful confidence and prediction intervals.

1 INTRODUCTION

In the last several years, machine learning (ML) methods have been developed rapidly fueled by ever-increasing amounts of data and computational power. Among different ML methods, a popular and widely-used technique is Neural Networks (NN) that models the relationship between inputs and outputs through layers of connected computational neurons. The idea of applying such a biology-analogous framework can be traced to the work of [17].

At the end of the 20th century, people focused on the feed-forward Shallow Neural Networks (SNN) with sigmoid-type activation functions. An SNN has only one hidden layer but is shown to possess the universal approximation property, i.e., it can be used to approximate any Borel measurable function from one finite dimensional space to another with any desired degree of accuracy—see [6, 12] and references within. However, the SNN practical performance left much to be desired. In the last ten or so years, Deep Neural Networks (DNN) received increased attention due to their great empirical performance.

Although DNN have become a state-of-the-art model, their theoretical foundation is still in development. Notably, [27, 28] explored the approximation ability of DNN[1] for any function f that belongs to an Hölder Banach space; here, the sigmoid-type activation functions are now replaced by ReLU-type functions to avoid the gradient vanishing problem. The aforementioned work showed that the optimal error of the DNN estimator f_{DNN} can be uniformly bounded, i.e.,

$$||f - f_{\text{DNN}}||_\infty = O\left(W^{-2\xi/d}\right); \quad (1)$$

here, ξ is some smoothness measurement of the target function $f : \mathbf{R}^d \to \mathbf{R}$ —see Section 4 for a formal definition; W is the size of a neural network f_{DNN}, i.e., the total number of parameters; and d is the dimension of the function inputs.

However, the bound (1) is not useful in practice. The reason is three-fold: (a) it requires a discontinuous weight assignment to build the desired DNN, so it is not feasible to train such

[1]All DNNs considered in this paper have the forward property, which implies that the input, hidden neurons and output are connected in an acyclic relationship.

Manuscript accepted by ACM/IMS Journal of Data Science.

DNN with usual gradient-based methods; (b) the structure of the DNN might not be the standard fully connected form so finding the satisfied specific structure becomes another difficult; most importantly, (c) this error bound is on the *optimal* estimation we can achieve from a finely designed DNN. It fails to tell us any story about the situation of applying the DNN estimator to solving real-world problems.

For example, what is the performance of the DNN to estimate a regression function with n independent samples $\{(Y, \boldsymbol{X}_i)\}_{i=1}^n$ generated from an underlying true model f? It is easy to see that the error ε of f_{DNN} in sup-norm can be arbitrarily small if we allow W to be arbitrarily large based on Eq. (1). However, this optimal performance is hardly achievable and only represents the theoretically best estimation. What we attempt to do in this paper is to determine an empirically optimal $\widehat{f}_{\text{DNN}}$ with samples $\{(Y, \boldsymbol{X}_i)\}_{i=1}^n$ and then explore its estimation and prediction inference. Guided by this spirit, people usually think $\widehat{f}_{\text{DNN}}$ as an M-estimator and set different loss functions for various purposes:

$$\widehat{f}_{\text{DNN}} \in \arg \min_{f_\theta \in \mathcal{F}_{\text{DNN}}} \frac{1}{n} \sum_{i=1}^{n} L(f_\theta(\boldsymbol{x}_i), y_i); \tag{2}$$

here $\mathcal{F}_{\text{DNN}}$ is a user-chosen space that contains all DNN candidates; $L(\cdot,\cdot)$ is the loss function, e.g., Mean Squared Errors loss for the regression problem with real-valued output, i.e., $L(f_\theta(\boldsymbol{x}_i), y_i) = (f_\theta(\boldsymbol{x}_i) - y_i)^2/2$; $\{(y, \boldsymbol{x}_i)\}_{i=1}^n$ are realizations of $\{(Y, \boldsymbol{X}_i)\}_{i=1}^n$.

In the paper at hand, we consider DNN-based estimation and prediction inference in the data-generating model: $Y_i = f(\boldsymbol{X}_i) + \epsilon_i$; here, the ϵ_i are independent, identically distributed (i.i.d.) from a distribution F_ϵ that has mean 0 and variance σ^2—we will use the shorthand $\epsilon_i \sim$ i.i.d. $(0, \sigma^2)$. Consequently, $f(\boldsymbol{x}_i) = \mathbb{E}(Y_i | \boldsymbol{X}_i = \boldsymbol{x}_i)$. Furthermore, the regression function $f(\cdot)$ will be assumed to satisfy some smoothing condition which will be specified later. Note that the additive model with heteroscedastic error: $Y_i = f(\boldsymbol{X}_i) + g(\boldsymbol{X}_i) \cdot \epsilon_i$ can be analyzed similarly by applying two DNNs, one to estimate $f(\cdot)$ and one for $g(\cdot)$.

From a nonparametric regression view, it is well-known that the optimal convergence rate of the estimation for a p-times continuously differentiable regression function of a d-dimensional argument is $n^{2p/(2p+d)}$—see [22]. If we assume the regression function belongs to a more general Hölder Banach space, we can define a non-integer $\xi = p + s$ to represent the smoothness order of f; here $0 < s \leq 1$ is the Hölder coefficient. The optimal rate of non-parametric estimation can also be extended to such non-integer smoothness order; see Condition $3'$ and Definition 2 of [16]. Focusing on DNN estimation, the optimal and achievable error bound on the L_2 norm of $\widehat{f}_{\text{DNN}}$ is $O(n^{-\xi/(\xi+d)} \cdot \log^8(n))$ with a *high probability*; this bound is due to [9] but the rate appears slower than the optimal rate that we can attain. Besides, although $\widehat{f}_{\text{DNN}}$ will become more accurate as the sample size increases, training DNN becomes very time-consuming. Moreover, it is infeasible to load massive data into a PC or even a supercomputer since its node memory is also limited in the computation process as pointed out by [30].

In this paper, we first give a small improvement on the bound of [9] using the latest results on the DNN approximation ability. Then, we resolve the computational issue involving huge data by applying the Scalable Subsampling technique of [20] to create a set of subsamples and then build a so-called subagging DNN estimator $\overline{f}_{\text{DNN}}$. Under regularity conditions, we can show that the subagging DNN estimator $\overline{f}_{\text{DNN}}$ could possess a faster convergence rate than a single DNN estimator $\widehat{f}_{\text{DNN}}$ trained on the whole sample. Lastly, using the same set of subsamples, we can build a Confidence Interval (CI) for f based on $\overline{f}_{\text{DNN}}$. Due to the prevalent undercoverage phenomenon of CIs with finite samples, we propose two ideas to improve the empirical coverage rate: (1) we enlarge the CI by replacing the standard deviation estimation in the margin of errors with a term which is close to an estimation of mean square error; the explicit analysis on the effects of this replacement is given in Section 4.3.1; (2) we take an iterated subsampling method to build a specifically designed CI which is a combination of pivot-CI and quantile-CI; see the concrete steps in Algorithm 2. Beyond estimation inference, we also perform predictive inference (with both point and interval predictions).

Outline: The paper is organized as follows. In Section 2, we give a short introduction to the structure of DNN. In Section 3, we describe the methodology of scalable subsampling. Subsequently, the performance of the subagging DNN estimator and its associated confidence/prediction intervals are analyzed in Section 4 and Section 5. Simulation and empirical studies are given in Section 6 and Section 7, respectively. We conclude this paper in Section 8. Proofs are given in Appendix: A. Other additional materials are put in Appendix: B.

Contributions: We summarize our main contributions as follows:

1 Taking advantage of the latest approximation result about DNN, we refine the high probability non-asymptotic error bound of one specific type of DNN. Its extension to a general DNN estimator is straightforward.
2 Under simple and mild conditions, we show that the non-asymptotic error bound can be further improved with the help of the recently proposed scalable subsampling technique. Moreover, our subagging DNN-based estimator is more computationally efficient than training DNNs with various sizes on a whole dataset.
3 Beyond the refinement of the error bound of the DNN estimator in the mean square sense, we propose a scaling-down bias order estimation method, which is of independent interest and may be useful in other problems.
4 We give a comprehensive discussion on how to make a practically useful Confidence Interval (CI) with a DNN estimator, especially for finite sample cases. In addition, we distinguish the difference between CI and Prediction interval (PI) and show our PI is asymptotically valid under mild conditions.

Related Work: The comparisons of our method to mostly related work will be drawn throughout this paper. Here, we give a summary. The fundamental idea of scalable subsampling shares a similar spirit with the divide-and-conquer approach originally applied in the algorithm to decrease the computational complexity; see [5, 8, 13]. In short, the initial divide-and-conquer approach consists of three steps: (1) Divide the problem into a number of subproblems; (2) Conquer the subproblems; (3) Combine all solutions of subproblems. In the machine learning community, the divide-and-conquer idea can be integrated with DNNs for various purposes. For example, [10] applied a Recurrent neural network (RNN) which is made up of some small RNNs to improve the accuracy of language identification; [7] decomposed a challenging problem of determining the number and locations of acoustic point sources into several subproblems which DNNs can solve. However, these applications are mostly concerned with the algorithm or DNN structure design and lack theoretical validation. With careful discussion regarding theory and practical implementation, we show that our scalable subsampling approach can improve the error bound of DNN on the estimation of target functions in the regression setting. In this perspective, our method is also related to the bagging and subagging ideas that were proposed by [3] and [4], respectively. Along with the bagging idea, the prediction or estimation accuracy with DNN model can be improved in solving real-world problems; see [11, 14] for example. However, even the classical subagging estimator could be computationally infeasible with a massive dataset. On the other hand, our scalable subsampling estimator is more efficient since it is based on non-random subsamples and can control the overlapping level of different subsamples.

Notations: We will use the following norms: $\|g\|_{L_2(X)} := \mathbb{E}[g(X)^2]^{1/2}$; $\|g\|_\infty := \sup_{\boldsymbol{x}} |g(\boldsymbol{x})|$. Also, we employ the notation $a_n = \Theta(d_n)$ to denote "exact order", i.e., that there exist two constants c_1, c_2 satisfying $c_1 \cdot c_2 > 0$, and $c_1 d_n \leq a_n \leq c_2 d_n$. We also use $\mathbb{E}_n[\cdot]$ to represent the sample average operator.

2 STANDARD FULLY CONNECTED DEEP NEURAL NETWORK

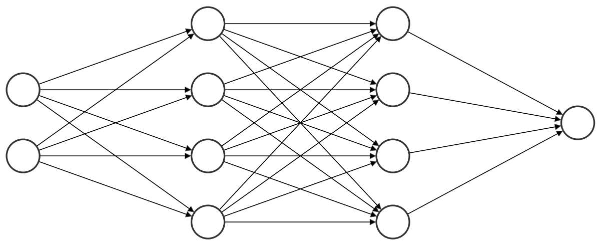
For completeness, we now briefly introduce the fully connected forward DNN, and each layer has a number of hidden units that are of the same order of magnitude. This type of DNN is the so-called Multi-layer Perceptron (MLP). Since an MLP has more structure restrictions than a general forward DNN, its estimation error bound will be larger than the variant with a general DNN. Besides, with the fact a DNN can be embedded into an MLP, the proof of the error bound with an MLP is one step further than the proof of the error bound with a general DNN. Thus, we will give new theoretical results regarding the error bound of MLP estimators on some target functions. Our theory can be extended to general forward DNN estimators straightforwardly. To simplify notations, we refer to the DNN as the MLP with the ReLU activation function; see more discussions in Remark 4.1. In short, the DNN can be viewed as a parameterized family of functions. Its structure mainly depends on the input dimension d, depth $L \in \mathbb{N}$, width $\boldsymbol{H} \in \mathbb{N}^L$ and the output dimension. The depth L describes how many hidden layers a DNN possesses; the width $\boldsymbol{H} = (H_1, \ldots, H_L)$ represents the number of neurons in each hidden layer. The fully connected property indicates that each hidden neuron receives information from all hidden neurons at the previously hidden layer in a functional way.

Formally, if we let $\boldsymbol{u}_l = (u_{l,1}, \ldots, u_{l,H_l})^T$ to represent all number of neurons at the l-th hidden layer for $l = 0, \ldots, L+1$; here, $\boldsymbol{u}_0$ represents the input vector $(x_1, \ldots, x_d)^T$ and $\boldsymbol{u}_{L+1}$ is the output. Therefore, we can pretend that the input layer and the output layer are the 0-th and $(L+1)$-th hidden layers, respectively. Then, $u_{l,i} = \sigma(\boldsymbol{u}_{l-1}^T \boldsymbol{w}_{l-1,i} + b_{l-1,i})$ for $l = 1, \ldots, L$ and $i = 1, \ldots, H_l$; here $\boldsymbol{w}_{l-1,i} \in \mathbb{R}^{H_{l-1}}$ is the weight vector which connects the $(l-1)$-th hidden layer and the neuron $u_{l,i}$; $b_{l-1,i} \in \mathbb{R}$ is the corresponding intercept term; $\sigma(\cdot)$ is the so-called activation function and we take the ReLU function in this paper. To get the output layer, we just take $u_{L+1,i} = \boldsymbol{u}_L^T \boldsymbol{w}_{L,i} + b_{L,i}$ for $i = 1, \ldots, H_{L+1}$; here H_{L+1} is equal to the output dimension. To express the functionality of the DNN in a more concise way, we can stack $\{\boldsymbol{w}_{l-1,i}^T\}_{i=1}^{H_l}$ by row to get $\boldsymbol{W}_{l-1} \in \mathbb{R}^{H_l} \times \mathbb{R}^{H_{l-1}}$ and collect $\{b_{l-1,i}\}_{i=1}^{H_l}$ to be a vector $\boldsymbol{b}_{l-1}$ for $l = 1, \ldots, L+1$. Subsequently, we can treat the DNN as a function that takes the input $\boldsymbol{x}$ and returns output in the below way:

$$f_{\mathrm{DNN}}(\boldsymbol{x}) = \boldsymbol{W}_L(\sigma(\boldsymbol{W}_{L-1}(\cdots \sigma(\boldsymbol{W}_2 \sigma(\boldsymbol{W}_1 \sigma(\boldsymbol{W}_0 \boldsymbol{x} + \boldsymbol{b}_0) + \boldsymbol{b}_1) + \boldsymbol{b}_2) \cdots) + \boldsymbol{b}_{L-1}) + \boldsymbol{b}_L.$$

We can understand that the function $f_{\mathrm{DNN}}(\boldsymbol{x})$ maps $\boldsymbol{x}$ to the 1-st hidden layer and then map the 1-st hidden to the 2-nd hidden layer and so on iteratively with weights $\{\boldsymbol{W}_l\}_{l=0}^L$, $\{\boldsymbol{b}_l\}_{l=0}^L$ and the activation function $\sigma(\cdot)$. We can then compute the total number of parameters in a DNN by the formula $W = \sum_{i=0}^L (H_i \cdot H_{i+1} + H_{i+1})$. A simple DNN is presented in Fig. 1. It has a constant width of 4 and a depth of 2.

Fig. 1. The illustration of a fully connected DNN with $L = 2$, $H = 4$ and $W = 37$, and input dimension $d = 2$ and output dimension 1.



3 SCALABLE SUBSAMPLING

Scalable subsampling is one type of non-stochastic *subsampling* technique proposed by [20]. Assume that we observe the sample $\{Z_1, \ldots, Z_n\}$; then, scalable subsampling relies on $q = \lfloor (n - b)/h \rfloor + 1$ number of subsamples $B_1, \ldots, B_q$ where $B_j = \{Z_{(j-1)h+1}, \ldots, Z_{(j-1)h+b}\}$; here, $\lfloor \cdot \rfloor$ denotes the floor function, and h controls the amount of overlap (or separation) between B_j and B_{j+1}. In general, the subsample size b and the overlap h are functions of n, but these dependencies will not be explicitly denoted, e.g.,

$$b = \Theta(n^{\beta}) \ ; \ h = a \cdot b,$$

where $0 < \beta < 1$ and $a > 0$. More importantly, tuning b and h can make scalable subsampling samples have different properties. For example, if $h = 1$, the overlap is the maximum possible; if $h = 0.2b$, there is 80% overlap between B_j and B_{j+1}; if $h = b$, there is no overlap between B_j and B_{j+1} but these two blocks are adjacent; if $h = 1.2b$, there is a block of about $0.2b$ data points that separate the blocks B_j and B_{j+1}.

The *bagging* idea was initially proposed by [3], where the subsample is bootstrapped (sampling with replacement) with the same size as the original sample. As revealed by that work, the main benefit of taking this technique is that the mean-squared error (MSE) of the bagging estimator can decrease, especially for unstable estimators that may change a lot with different samples, e.g., neural networks and regression trees. There are ample works about combining the neural networks with the bagging technique to improve its generalization performance; e.g., see applications in the work of [11, 14] for references. However, the drawback of the original bagging method is that the estimation process needs to be performed with n-size bootstrap resamples many times which is infeasible with massive data. [4] proposed the *subagging* idea which is based on all subsamples as opposed to bootstrap resamples. However, even choosing a single random subsample could be computationally challenging when n is large. As pointed out in [23], drawing a random sample of size b from n items using the Sparse Fisher-Yates Sampler takes $O(b)$ time and space which corresponds to optimal time and space complexity for this problem.

Facing such computational dilemmas, scalable subsampling and subagging as proposed by [20] can be seen as an extension of the Divide-and-Conquer principle—see e.g. [13]. Moreover, in addition to the computational savings, scalable subagging may yield an estimator that is not less (and sometimes more) accurate than the original; the following example illustrates such a case.

Example 3.1 (Kernel-smoothed function estimation). A remarkable example from the work of [20] is the scalable subagging kernel estimator. Suppose our goal is estimating the value of function g at a specific point; here, the function g can be a probability density, spectral density, or other function that is estimated in a nonparametric setting. Denote the estimand θ and its corresponding kernel-smoothed estimator $\hat{\theta}_n$ based on the whole sample, and assume that $\hat{\theta}_n$ satisfies the following conditions:

(1) $\mathbb{E}(\hat{\theta}_n^2) < \infty$ for all n;
(2) $n^\gamma(\mathbb{E}(\hat{\theta}_n) - \theta) \longrightarrow C$ and $\mathrm{Var}(n^\alpha \hat{\theta}_n) \longrightarrow \sigma^2$ as $n \longrightarrow \infty$, where C is a non-zero constant, $\sigma^2 > 0$ and $\gamma > \alpha > 0$.

Define the scalable subagging estimator as:

$$\bar{\theta}_{b,n,SS} = q^{-1} \sum_{i=1}^{q} \hat{\theta}_{b,i},$$

here q is the total number of subsamples and $\hat{\theta}_{b,i}$ is the non-parametric estimator based on the i-th subsample B_i. To achieve the fastest convergence rate of $\bar{\theta}_{b,n,SS}$ we may let $\beta = \frac{1}{1+2(\gamma-\alpha)}$. As a result, the Mean Squared Error (MSE)of the scalable subagging estimator $\bar{\theta}_{b,n,SS}$ is $\Theta(n^{-2\gamma/(1+2(\gamma-\alpha))})$; see Lemma 4.1 of [20] for a detailed discussion. To achieve such a convergence rate in the context of nonparametric estimation, the crucial point is using an undersmoothed bandwidth on the subsample statistics. To elaborate, suppose we are employing a non-negative (second-order) kernel for smoothing in which case the MSE-optimal bandwidth is $\Theta\left(n^{-1/5}\right)$. To conduct efficient subagging, however, the $\hat{\theta}_{b,i}$ should be computed using an undersmoothed bandwidth of order $o\left(b^{-1/5}\right)$. For example, if we choose the bandwidth for $\hat{\theta}_{b,i}$ to be $\Theta\left(b^{-1/4}\right)$ instead, then the choices $\alpha = 3/8$, $\gamma = 1/2$, $h = O(b)$, and $b = \Theta\left(n^\beta\right)$ with $\beta = 0.8$ implies that the rate of convergence of $\bar{\theta}_{b,n,SS}$ is $n^{2/5}$. This rate is not only faster than the rate of $\hat{\theta}_n$ that used the sub-optimal bandwidth $\Theta\left(n^{-1/4}\right)$; it is actually the fastest rate achievable by any estimator that uses a non-negative kernel with its associated MSE-optimal bandwidth. Nevertheless, $\bar{\theta}_{b,n,SS}$ can be computed faster than $\hat{\theta}_n$, and may thus be preferable. In addition to the asymptotic results, the simulation study of [20] reveals that the error of the scalable subsampling estimator can actually be smaller than the full-sample nonparametric estimator with its own optimal bandwidth choice.

In the next section, we will introduce how to compute the scalable subsampling DNN estimator. Then, we will show that our aggregated DNN estimator could possess a smaller MSE than the optimal DNN estimator trained on the whole sample, under some conditions. We also discuss some specifically designed confidence intervals to measure the estimation accuracy via the approaches mentioned in Section 1.

4 ESTIMATION INFERENCE WITH DNN

Although the DNN has captured much attention in practice, its theoretical validation is still in development. Recently, [9] gave a high-probability non-asymptotic error bound to measure the performance of the DNN estimator under two regularity assumptions. In short, the error of using $\widehat{f}_{\mathrm{DNN}}$ to estimate f comes from two sources: (1) the stochastic error, which measures the difference of $\widehat{f}_{\mathrm{DNN}}$ and the best one in a DNN class $\mathcal{F}_{\mathrm{DNN}}$; (2) the approximation error, which measures how well the target function f can be approximated by a DNN which comes from $\mathcal{F}_{\mathrm{DNN}}$ concerning some specific loss, i.e., the approximation ability of $\mathcal{F}_{\mathrm{DNN}}$. The work of [9] hinges on using a DNN to estimate functions belonging to Sobolev space w.r.t. L_∞ norm. To sync with the latest results on the approximation ability of DNNs, we consider estimating functions in an Hölder space. This change brings benefits two-fold: (1) In practice, the Hölder space is a more appropriate and direct space for our target function if it possesses some smoothness property. In other words, the smoothness of our target function in Hölder space can be described thoroughly; (2) we can decrease the approximation error by applying the latest results on the approximation ability of DNNs for function in Hölder space from [28]. It turns out that we can finally get a precise $\widehat{f}_{\text{DNN}}$; see [8] for a detailed discussion of Hölder and Sobolev spaces. We present our assumptions below:

- A1: The regression data are i.i.d. copies of $\boldsymbol{Z} = (Y, \boldsymbol{X}) \in \mathcal{Y} \times [-1,1]^d$, where $\boldsymbol{X}$ has a continuous distribution, and $\mathcal{Y} \subset [-M, M]$ for some positive constant M. Correspondingly, we set the space of all DNN candidate functions to be $\mathcal{F}_{\text{DNN}} = \{f_\theta : ||f_\theta||_\infty \leq 2M\}$;
- A2: The target regression function f lies in the Hölder space $C^{k,\alpha}\left([-1,1]^d\right)$ which is the space of k times continuously differentiable functions on $[-1,1]^d$ having a finite norm defined by

$$||f||_{C^{k,\alpha}\left([-1,1]^d\right)} = \max\left\{\max_{\mathbf{k}:|\mathbf{k}|\leq k}\max_{\mathbf{x}\in[-1,1]^d}\left|D^{\mathbf{k}}f(\mathbf{x})\right|, \max_{\mathbf{k}:|\mathbf{k}|=k}\sup_{\substack{\mathbf{x},\mathbf{y}\in[-1,1]^d\\ \mathbf{x}\neq\mathbf{y}}}\frac{\left|D^{\mathbf{k}}f(\mathbf{x}) - D^{\mathbf{k}}f(\mathbf{y})\right|}{||\mathbf{x}-\mathbf{y}||^\alpha}\right\},$$

  where the smoothness index is $\xi = k + \alpha$ with an integer $k \geq 0$ and $0 < \alpha \leq 1$.
- A3: The sample size n is larger than $(2eM)^2 \vee \text{Pdim}(f_{\text{DNN}})$ where $\text{Pdim}(f_{\text{DNN}})$ is the pseudo-dimension of f_{DNN} which satisfies:

$$c \cdot WL\log(W/L) \leq \text{Pdim}(f_{\text{DNN}}) \leq C \cdot WL\log W,$$

  with some universal constants $c, C > 0$ and Euler's number e; see [1] for details.

Remark. *We can weaken the assumption on the domain of $\boldsymbol{X}$ to $[-C_x, C_x]^d$ for some constant C_x, i.e., we can work on a compact domain of $\boldsymbol{X}$; see also the proof of [28].*

As shown in [9], with $H_1 = H_2 = \cdots = H_L = \Theta(n^{\frac{d}{2(\xi+d)}}\log^2 n)$, $L = \Theta(\log n)$, the L_2 norm loss and empirical mean squared error of the deep fully connected ReLU network estimator from Eq. (2) can be bounded with probability at least $1 - \exp\left(-n^{\frac{d}{\xi+d}}\log^8 n\right)$, i.e.,

$$\begin{aligned} \left\|\widehat{f}_{\text{DNN}} - f\right\|^2_{L_2(X)} &\leq C_1 \cdot \left\{n^{-\frac{\xi}{\xi+d}}\log^8 n + \frac{\log\log n}{n}\right\}, \\ \text{and } \mathbb{E}_n\left[\left(\widehat{f}_{\text{DNN}} - f\right)^2\right] &= \Theta\left(\left\|\widehat{f}_{\text{DNN}} - f\right\|^2_{L_2(X)}\right); \end{aligned} \tag{3}$$

here $C_1 > 0$ is a constant which is independent of n and may depend on d, M, and other fixed constants.

Obviously, the L_2 norm error bound in (3) is sub-optimal compared to the fastest convergence rate we can achieve for nonparametric function estimation. With the latest approximation theory on DNNs, we can improve the error bound in Eq. (3) by decreasing the power of the $\log(n)$ term. Meanwhile, this faster rate is satisfied with a narrower DNN. We give our first theorem about the convergence rate of $\widehat{f}_{\text{DNN}}$ below.

Theorem 4.1. *Under assumptions A1 to A3, width $H = \Theta(n^{\frac{d}{2(\xi+d)}}\log n)$, and depth $L = \Theta(\log n)$. Then, the L_2 norm loss of the deep fully connected ReLU network estimator Eq. (2) can be bounded with probability at least $1 - \exp\left(-n^{\frac{d}{\xi+d}}\log^6 n\right)$, i.e.,*

$$\left\|\widehat{f}_{DNN} - f\right\|^2_{L_2(X)} \leq C_2 \cdot \left\{n^{-\frac{\xi}{\xi+d}}\log^6 n + \frac{\log\log n}{n}\right\}; \tag{4}$$

here $C_2 > 0$ is another constant.

It appears that the above gives the fastest rate obtainable based on the current literature. Later, we will show how this error bound can be further improved by applying the scalable subagging technique under some mild conditions.

Remark 4.1. *The improvement implied by Theorem 4.1 can also be applied to Corollaries 1 and 2 of [9] to improve the corresponding error bounds. Corollaries 1 and 2 of [9] discuss the error bound of general forward DNN.*

4.1 Scalable subagging DNN estimator

We first review the idea of scalable subagging and explain how this technique can be combined with DNN estimation. We focus on the regression problem and assume we observe sample $\{(Y_1, \boldsymbol{X}_1), \ldots, (Y_n, \boldsymbol{X}_n)\}$.

Analogously to the subagging kernel-smoothed estimator of Example 3.1, we can define the subagging DNN estimator as:

$$\overline{f}_{\mathrm{DNN}}(\boldsymbol{X}) = \frac{1}{q}\sum_{j=1}^{q} \widehat{f}_{\mathrm{DNN},b,j}(\boldsymbol{X}); \tag{5}$$

here, $q = \lfloor (n-b)/h \rfloor + 1$, and $\widehat{f}_{\mathrm{DNN},b,j}(\cdot)$ is the minimizer of the empirical loss function in Eq. (2) just using the data in the j-th subsample namely $B_j = \{(Y_{(j-1)h+1}, \boldsymbol{X}_{(j-1)h+1}), \ldots, (Y_{(j-1)h+b}, \boldsymbol{X}_{(j-1)h+b})\}$. In this subsection, we consider $h = b$.

In nonparametric function estimation where the estimation is performed through the kernel technique, the bandwidth can control the bias order of the kernel-smoothed estimator. As shown in Example 3.1, the optimal convergence rate can be recovered by combining scalable subagging trick and undersmoothing bandwidth. Similarly, in the context of neural network estimation, the whole architecture of a DNN controls its smoothness similar to the role of the shape (order) of a kernel. The depth and width of a DNN play the role of tuning parameters similar to the bandwidth of a kernel. Moreover, according to the prevailing wisdom, a deeper DNN may possess a lower bias; this conjecture was confirmed by [26] with ResNet on some image datasets.

However, as far as we know, there is no theoretical result that explains the relationship between bias and the width/depth of a DNN. Here, we make the below assumptions to restrict the order of the bias of $\widehat{f}_{\mathrm{DNN}}$:

- A4: $\mathbb{E}(\widehat{f}_{\mathrm{DNN}}(\boldsymbol{x}) - f(\boldsymbol{x})) = O(n^{-\Lambda/2})$ uniformly in $\boldsymbol{x}$ for some constant $\Lambda > 0$.

To boost the scalable subagging method, a fundamental preliminary condition is that the bias of the estimator is comparatively negligible to its standard deviation—see [20] for details. Thus, we further impose an additional assumption on the order of bias:

- A5: The bias exponent in Assumption A4 satisfies the inequality: $\Lambda > \frac{\xi}{\xi+d}$.

We claim that assumptions A4 and A5 can be achievable due to the fact that as revealed in [28], the approximation ability in the uniform sup-norm of a DNN can be as fast as $W^{-2\xi/d}$. Although this rate is not instructive in practice, the existence of a DNN that satisfies the bias order requirements A4 and A5 is possible. To see its feasibility, let $n = W^{\kappa} L \log W$ such that A3 is satisfied. Let's assume that $\widehat{f}_{\mathrm{DNN}}$ can be trained as the optimal one indicated by sup-norm. This assumption is solely about an optimization problem whose difficulty is beyond the scope of this paper. Ignoring the slowly changing term, it is easy to see A4 and A5 are satisfied when $1 < \kappa < \frac{4(\xi+d)}{d}$.

We should also notice that practitioners tend to build a large DNN whose size is larger than the sample size. i.e., the DNN interpolates the sample in the modern machine-learning practice.

Interestingly, such an over-parameterized estimator breaks the classical understanding of the bias-variance trade-off since its generalization performance can even be better than a DNN which lies in the under-parameterized regime. Actually, this phenomenon is described as the double-descent of the risk by [2]. Thus, A4 and A5 are achievable when we consider DNNs with an overwhelming number of parameters, i.e., $W > n$ so that the bias in A4 can be as low as $O(W^{-2\xi/d})$; however, assumption A3 may fail which means the consistency property of the DNN estimator may be lost. It is interesting to explore whether the scalable subsampling can work for DNN estimators in an over-parameterized regime; we leave this to future work.

Remark 4.2. *In this paper, we focus on applying the scalable subagging technique to DNNs whose size is less than the sample size but it is straightforward to extend our methodology to a large DNN. However, as a sacrifice, the consistency property of $\widehat{f}_{\text{DNN}}$ to f may not be held. We leave this extension to future work. We just give a preliminary analysis from the computability aspect at this moment. As we can expect, the saving of computational cost from applying scalable subagging will be more significant for executing estimation with a large DNN. To see this fact, let's assume that we consider a DNN with size $W = \Theta(n^{\phi})$, $\phi > 1$. Then, the computational complexity will be mainly determined by how many manipulations (e.g., forward calculation and backward updating) we carry out to train the DNN. The total number of manipulations is also affected by the batch size and the number of epochs. Thus, we summarize that the total number of manipulations is $O(n \cdot W \cdot E)$; here E represents the number of epochs, i.e., the number of complete passes of the training through the algorithm. It is fair to assume that the complexity is in the order of $n^{\phi+1} := n^{\varphi}$. When the size of the DNN is larger than the sample size, $\varphi > 2$. Thus, for the subagging estimator, the computational complexity is approximately to be $O(n^{\beta\varphi}q) = O(n^{1+\beta(\varphi-1)})$. The ratio of $n^{1+\beta(\varphi-1)}$ over n^{φ} is $n^{-(\varphi-1)(1-\beta)}$. Thus, for a fixed β, the larger φ to be, the more computation can be saved by deploying the subagging technique.*

Aggregating all the above, the following theorem quantifies the error bound of the scalable subagging DNN estimator of Eq. (5):

Theorem 4.2. *Assume A1 to A5, and let $\beta = \frac{1}{1+\Lambda-\frac{\xi}{\xi+d}}$. Then, with probability at least $(1 - \exp(-n^{\frac{d}{\xi+d}}\log^6 n))^q$ the error bound of the subagging estimator Eq. (5) in L_2 norm is:*

$$\left\|\bar{f}_{DNN} - f\right\|^2_{L_2(X)} \leq n^{\frac{-\Lambda}{\Lambda+\frac{d}{\xi+d}}}\mathcal{L}(n),$$

where $\mathcal{L}(n)$ is a slowly varying function involving a constant and all $\log(n)$ terms.

Remark 4.3. *Choosing $\beta = \frac{1}{1+\Lambda-\frac{\xi}{\xi+d}}$ in Theorem 4.2 ensures that the square bias term will be always relatively negligible compared to the variance which is important for the success of scalable subsampling; see related discussion in Remark 4.4.*

Note that the final accuracy of DNN heavily depends on many other factors in practice, e.g., which optimizer we choose in the training stage, which parameter initialization strategy we take, and how large the batch size should be. Thus, a solely theoretical rate is insufficient to verify the superiority of the scalable subsampling DNN estimator. We then deploy simulation studies in Section 6 to provide supplementary evidence.

4.2 Estimation of the bias order of DNN estimator

Although Theorem 4.2 shows the possibility of getting a smaller error bound, it depends on the bias exponent Λ which is typically unknown. In this subsection, we propose two approaches to estimate the value of Λ via subsampling. As far as we know, it is the first attempt to quantify the bias of the DNN estimator.

First note that A4 implies that, for any i, $\mathbb{E}(\widehat{f}_{\text{DNN},b,i}(\boldsymbol{x}) - f(\boldsymbol{x})) = O(n^{-\beta\Lambda/2})$. Since $\overline{f}_{\text{DNN}}(\boldsymbol{X}) = \frac{1}{q}\sum_{j=1}^{q}\widehat{f}_{\text{DNN},b,j}(\boldsymbol{X})$, it follows that the bias of $\overline{f}_{\text{DNN}}(\boldsymbol{x})$ is $O(n^{-\beta\Lambda/2})$, so we can write

$$\left|\mathbb{E}(\overline{f}_{\text{DNN}}(\boldsymbol{x}) - f(\boldsymbol{x}))\right| = c_b \cdot b^{-\Lambda/2} + o(b^{-\Lambda/2}). \tag{6}$$

Recall that $\overline{f}_{\text{DNN}}(\boldsymbol{X})$ was built based on subsamples of size b. If we have another DNN estimator $\widehat{f}_{\text{DNN},b_0}(\boldsymbol{x})$ trained on sample of size b_0, then its bias will be $c_b \cdot b_0^{-\Lambda/2} + o(b_0^{-\Lambda/2})$. Then,

$$\begin{aligned}\left|\mathbb{E}\left(\overline{f}_{\text{DNN}}(\boldsymbol{x}) - f(\boldsymbol{x})\right)\right| &= \left|\mathbb{E}\left(\overline{f}_{\text{DNN}}(\boldsymbol{x}) - \widehat{f}_{\text{DNN},b_0}(\boldsymbol{x}) + \widehat{f}_{\text{DNN},b_0}(\boldsymbol{x}) - f(\boldsymbol{x})\right)\right| \\ &= \left|\mathbb{E}\left(\overline{f}_{\text{DNN}}(\boldsymbol{x}) - \widehat{f}_{\text{DNN},b_0}(\boldsymbol{x})\right) + \mathbb{E}\left(\widehat{f}_{\text{DNN},b_0}(\boldsymbol{x}) - f(\boldsymbol{x})\right)\right|.\end{aligned} \tag{7}$$

If $b \longrightarrow \infty$ and $b/b_0 \longrightarrow 0$, the bias of $\overline{f}_{\text{DNN}}(\boldsymbol{x})$ is asymptotically determined by the first term on the r.h.s. of Eq. (7). So we can try to estimate $\left|\mathbb{E}\left(\overline{f}_{\text{DNN}}(\boldsymbol{x}) - \widehat{f}_{\text{DNN},b_0}(\boldsymbol{x})\right)\right|$ to approximate the l.h.s. of Eq. (7).

Ideally, if we have a large enough sample, we can carve out M non-overlapping (or partially overlapping) b_0-size subsamples and compute $\{\widehat{f}^{(i)}_{\text{DNN},b_0}(\boldsymbol{x})\}_{i=1}^{M}$. If we further separate each b_0-size subsample into multiple non-overlapping (or partially overlapping) b-size subsamples, $\{\overline{f}^{(i)}_{\text{DNN}}(\boldsymbol{x})\}_{i=1}^{M}$ can be built and each $\overline{f}^{(i)}_{\text{DNN}}(\boldsymbol{x})$ possesses the same bias order as our desired DNN estimator. Subsequently, the bias of $\overline{f}_{\text{DNN}}(\boldsymbol{x})$ can be estimated by the sample mean of $\{\overline{f}^{(i)}_{\text{DNN}}(\boldsymbol{x}) - \widehat{f}^{(i)}_{\text{DNN},b_0}(\boldsymbol{x})\}_{i=1}^{M}$. We can then use this information to estimate the value of Λ. By the law of large numbers, we can get accurate bias estimation as $M \longrightarrow \infty$. However, as we can easily see, this approach is computationally heavy and requires a large dataset.

Consequently, we propose another way to perform the bias estimation; we will call it *scaling-down* estimation method. To elaborate, recall that our goal is estimating the bias of $\overline{f}_{\text{DNN}}(\boldsymbol{x})$ that was built based on subsamples of size b. Consider different DNN estimators $\widehat{f}_{\text{DNN},b_1}(\boldsymbol{x})$ and $\widehat{f}_{\text{DNN},b_2}(\boldsymbol{x})$ which are trained on samples of size b_1 and b_2 respectively; here $b_1 \ll b$ and $b_2 \ll b_1$. As before, A4 implies that the bias of $\widehat{f}_{\text{DNN},b_i}(\boldsymbol{x})$ is $c_b \cdot b_i^{-\Lambda/2} + o(b_i^{-\Lambda/2})$ for $i = 1, 2$. Then, a key observation is that:

$$\begin{aligned}\left|\mathbb{E}\left(\widehat{f}_{\text{DNN},b_i}(\boldsymbol{x}) - f(\boldsymbol{x})\right)\right| &= \left|\mathbb{E}\left(\widehat{f}_{\text{DNN},b_i}(\boldsymbol{x}) - \overline{f}_{\text{DNN}}(\boldsymbol{x}) + \overline{f}_{\text{DNN}}(\boldsymbol{x}) - f(\boldsymbol{x})\right)\right| \\ &= \left|\mathbb{E}\left(\widehat{f}_{\text{DNN},b_i}(\boldsymbol{x}) - \overline{f}_{\text{DNN}}(\boldsymbol{x})\right) + \mathbb{E}\left(\overline{f}_{\text{DNN}}(\boldsymbol{x}) - f(\boldsymbol{x})\right)\right|, \text{ for } i = 1, 2.\end{aligned} \tag{8}$$

Due to the relationship between b, b_1, b_2, the bias of $\widehat{f}_{\text{DNN},b_i}(\boldsymbol{x})$ is dominated by the first term on the r.h.s. of Eq. (8). We then have two different estimates of the bias of $\widehat{f}_{\text{DNN},b_i}(\boldsymbol{x})$, namely:

$$\widehat{B}_i = \left|\frac{1}{q_i}\sum_{j=1}^{q_i}\left(\widehat{f}^{(j)}_{\text{DNN},b_i}(\boldsymbol{x}) - \overline{f}_{\text{DNN}}(\boldsymbol{x})\right)\right|, \text{ for } i = 1, 2.$$

Fixing the value of i, $\{\widehat{f}^{(j)}_{\text{DNN},b_i}(\boldsymbol{x})\}_{j=1}^{q_i}$ is value of $\widehat{f}_{\text{DNN},b_i}(\boldsymbol{x})$ computed from the jth subsample of size b_i carved out the whole sample; as before, these subsamples can be non-overlapping or partially overlapping and their number is denoted by q_i. Ignoring the $o(\cdot)$ term in Eq. (6), we can solve the following system of equations to approximate both c_b and Λ:

$$\begin{cases}\widehat{B}_1 &= c_b \cdot b_1^{-\Lambda/2} \\ \widehat{B}_2 &= c_b \cdot b_2^{-\Lambda/2}.\end{cases} \tag{9}$$

Taking logarithms in Eq. (9) turns it into a linear system in c_b and Λ. Finally, we can estimate the bias of $\overline{f}_{\text{DNN}}(\boldsymbol{x})$ by *scaling down* $\widehat{B}_1$ by a factor $(b/b_1)^{-\Lambda/2}$, i.e., the bias of $\overline{f}_{\text{DNN}}(\boldsymbol{x})$ is approximately $\widehat{B}_1 \cdot (b/b_1)^{-\Lambda/2}$. We summarize this procedure in Algorithm 1.

Algorithm 1 *Scaling-down* bias estimation of DNN estimator

```
Step 1  Fix a subsample size $b$, and compute $\overline{f}_{\text{DNN}}(\boldsymbol{x})$ at point $\boldsymbol{x}$.
Step 2  Fix two subsample sizes $b_1 \ll b$ and $b_2 \ll b_1$, and separate the whole sample into $q_1$ and
        $q_2$ number of $b_1$-size and $b_2$-size subsamples, respectively. Compute $\{\widehat{f}^{(j)}_{\text{DNN},b_i}(\boldsymbol{x})\}_{j=1}^{q_i}$ at
        $\boldsymbol{x}$ for $i = 1, 2$.
Step 3  Solve Eq. (9) to get $c_b$ and $\Lambda$.
Step 4  Estimate the bias of $\overline{f}_{\text{DNN}}(\boldsymbol{x})$ by $\widehat{B}_1 \cdot (b/b_1)^{-\Lambda/2}$.
```

4.3 Confidence intervals

Beyond point estimation, it is important to quantify DNN estimation accuracy; this can be done via a standard error or –even better– via a Confidence Interval (CI). More specifically, for a point of interest $\boldsymbol{X} = \boldsymbol{x}$, we hope to find a CI which satisfies:

$$\mathbb{P}(B_l \leq f(\boldsymbol{x}) \leq B_u) = 1 - \delta;$$

here $\mathbb{P}$ should be understood as the conditional probability given $\boldsymbol{X} = \boldsymbol{x}$; B_l and B_u are lower and upper bound for $f(\boldsymbol{x})$ that are functions of the DNN estimator; δ is the significance level. Since we can have different CI constructions having the same δ, we are also interested in the CI length (CIL) which is defined as CIL $= B_u - B_l$. We aim for a (conditional) CI that is the most accurate (in terms of its coverage being close to $1 - \delta$) but with the shortest length.

Analogously to Example 3.1, we assume the variance term of the DNN estimator trained with sample size n and evaluated at $\boldsymbol{x}$:

B1 $\text{Var}(n^{\alpha}\widehat{f}_{\text{DNN}}(\boldsymbol{x})) \longrightarrow \sigma^2 > 0$ as $n \longrightarrow \infty$.

Generally speaking, we have two choices to build CI for $f(\boldsymbol{x})$: (1) Pivot-CI (PCI), the type of CI obtained by estimating the sampling distribution of a pivotal quantity, e.g. the estimator centered at its expectation; (2) Quantile-CI (QPI), the type of CI based on quantiles of the estimated sampling distribution of the (uncentered) estimator of interest. More details are given in the example below.

Example 4.1 (Types of CI). For any unknown quantity θ estimated by $\hat{\theta}_n$, we may build a scalable subagging estimator $\bar{\theta}_{b,n,SS} = q^{-1}\sum_{i=1}^{q}\hat{\theta}_{b,i}$ to approximate it. To construct a CI for θ based on $\bar{\theta}_{b,n,SS}$, we are aided by the CLT of [20], i.e.,

$$\kappa_n(\bar{\theta}_{b,n,SS} - \theta) \xrightarrow{d} N(C_{\mu}, C_{\sigma}^2), \text{ as } n \longrightarrow \infty, \tag{10}$$

under mild conditions; here C_{μ} and C_{σ}^2 are the mean and variance of limiting distribution, respectively, and $\kappa_n = n^{\frac{1-\beta+2\alpha\beta}{2}}$.[2]

The form of the PCI based on CLT (10) depends crucially on whether $C_{\mu} = 0$ or not; see the next two subsections for details. On the other hand, the QCI is easier to build but it has its own deficiencies. In the context of this example, it is tempting to create a QCI for θ by taking the $\delta/2$ and $1 - \delta/2$ quantile values of the empirical distribution of the points $\{\hat{\theta}_{b,1}, \ldots, \hat{\theta}_{b,q}\}$. However, the resulting CI will be too conservative, i.e., its coverage will be (much) bigger than $1 - \delta$. The reason

[2]We note a typo in [20] where κ_n was incorrectly written as $n^{-\frac{1-\beta+2\alpha\beta}{2}}$.

is that the empirical distribution of $\{\hat{\theta}_{b,i}, \ldots, \hat{\theta}_{b,q}\}$ is approximating the sampling distribution of estimator $\hat{\theta}_b$ which has bigger variance than that of the target $\hat{\theta}_n$. We could try to re-scale the empirical distribution of $\{\hat{\theta}_{b,1}, \ldots, \hat{\theta}_{b,q}\}$ as in classical subsampling—see [21]. We still consider the QCI in the simulation studies. As expected, this QCI is the most conservative one; see details in Section 6.

4.3.1 PCI in the case where $C_\mu = 0$. If $C_\mu = 0$, i.e., when the square bias is relatively negligible compared to the variance in estimation, we can rely on Eq. (10) to build a PCI for the true function f at a point $\boldsymbol{x}$. All we need is a consistent estimator of C_σ^2, e.g., $\widehat{C}_\sigma^2 = b^{2\alpha} q^{-1} \sum_{i=1}^{q} \left(\hat{\theta}_{b,i} - \bar{\theta}_{b,n,SS}\right)^2$. In that case, a PCI for θ based on the CLT can be written as:

$$\bar{\theta}_{b,n,SS} \pm z_{1-\delta/2} \cdot \widehat{C}_\sigma \cdot \kappa_n^{-1}, \tag{11}$$

where $z_{1-\delta/2}$ is the $1-\delta/2$ quantile of the standard normal distribution.

Observing that there is a common term $n^{\beta\alpha}$ in κ_n and $\widehat{C}_\sigma$, we can estimate $\widehat{C}_\sigma \cdot \kappa_n^{-1}$ as a whole rather than computing κ_n and $\widehat{C}_\sigma^2$ separately. As a result, we can get a simplified PCI based on Eq. (11) as follows:

$$\overline{f}_{\text{DNN}}(\boldsymbol{x}) \pm z_{1-\delta/2} \cdot M_\sigma; \tag{12}$$

here $M_\sigma = \widehat{C}_\sigma \cdot \kappa_n^{-1}$ which can be approximated by $\sqrt{q^{-1} \sum_{i=1}^{q} \left(\widehat{f}_{\text{DNN},b,i}(\boldsymbol{x}) - \overline{f}_{\text{DNN}}(\boldsymbol{x})\right)^2 / n^{\frac{1-\beta}{2}}}$. Note that the building of the CI does not require the knowledge of α which is the order of the variance term in B1. However, the estimation $\widehat{C}_\sigma$ may not be accurate when q is small since it is only an average of q terms. As a result, the PCI according to Eq. (12) may undercover the true model values. Thus, we may relax the desired property of CI. Instead of requiring the exact coverage rate of a CI to be $1-\delta$, we seek a CI such that:

$$\mathbb{P}(B_l \leq f(\boldsymbol{x}) \leq B_u) \geq 1 - \delta. \tag{13}$$

Thus, the optimal candidate will be the CI which has the shortest length and guarantees the lowest coverage rate larger than $1-\delta$. To satisfy Eq. (13), we may enlarge the CI appropriately by replacing $\widehat{C}_\sigma^2$ with $\widetilde{C}_\sigma^2 = \widehat{C}_\sigma^2 + (\overline{f}_{\text{DNN}}(\boldsymbol{x}) - y)^2$; here $y = f(\boldsymbol{x}) + \epsilon$.

It is appealing to think that $\widetilde{C}_\sigma^2$ is close to the MSE of $\overline{f}_{\text{DNN}}(\boldsymbol{x})$. However,

$$(\overline{f}_{\text{DNN}}(\boldsymbol{x}) - y)^2 = \left(\frac{1}{q}\sum_{i=1}^{q}(\widehat{f}_{\text{DNN},b,i}(\boldsymbol{x}) - y)\right)^2 = \left(\frac{1}{q}\sum_{i=1}^{q}\left(\widehat{f}_{\text{DNN},b,i}(\boldsymbol{x}) - f(\boldsymbol{x})\right) - \epsilon\right)^2.$$

When q is large, $(\overline{f}_{\text{DNN}}(\boldsymbol{x}) - y)^2 \to (C_\mu - \epsilon)^2$ where C_μ is the bias of $\overline{f}_{\text{DNN}}(\boldsymbol{x})$. Therefore, $\widetilde{C}_\sigma^2$ is not exactly the MSE, but it can still be used to enlarge the CI to some extent. We can then define another PCI as:

$$\overline{f}_{\text{DNN}}(\boldsymbol{x}) \pm z_{1-\delta/2} \cdot \widetilde{M}_\sigma, \tag{14}$$

where

$$\widetilde{M}_\sigma = \sqrt{q^{-1}\sum_{i=1}^{q}\left(\widehat{f}_{\text{DNN},b,i}(\boldsymbol{x}) - \overline{f}_{\text{DNN}}(\boldsymbol{x})\right)^2 / n^{1-\beta} + (\overline{f}_{\text{DNN}}(\boldsymbol{x}) - y)^2 / n^{1-\beta+2\alpha\beta}}. \tag{15}$$

Since the order of the variance term α is involved in the above terms, we consider two extreme situations in the simulation sections: (1) We take $2\alpha = 0$ which is a most enlarged case; or (2) take $2\alpha = 1$ which is a mildly enlarged case.

Remark 4.4 (The condition to guarantee $C_\mu = 0$). *According to Eq.* (10), $C_\mu = 0$ *is satisfied as long as* $\beta > \frac{1}{1+\Lambda-2\alpha}$ *under A5. If we take* $\beta = \frac{1}{1+\Lambda-\frac{\xi}{\xi+d}}$ *in Theorem 4.2, we can find that the condition for* $C_\mu = 0$ *is always satisfied. This is not surprising due to A5 imposing the requirement on the convergence rate of the bias term. However, as explained in Remark 4.3, this* β *is not the optimal one to generate the smallest error bound. Thus, we could arrive at a stage where the orders of the squared bias and variance are the same once we know* α*. Due to the high variability of training a DNN in practice, we introduce a method in Section 4.3.2 to build CI appropriately under the situation that* $C_\mu \neq 0$*, which serves for cases where the bias is not relatively negligible.*

4.3.2 PCI in the case where $C_\mu \neq 0$. It is worthwhile to discuss how can we build a PCI for scalable subsampling DNN estimator when $C_\mu \neq 0$. Note that [20] proposed an *iterated* scalable subsampling technique that is applicable in the case $C_\mu \neq 0$. While this technique is also applicable in the case $C_\mu = 0$, we may prefer the construction of Section 4.3.1 since it is less computer-intensive. However, we should notice that the additional computational burden brought by *iterated* subsampling is negligible when $n \longrightarrow \infty$; see analysis in the below Remark.

Remark (Complexity analysis of iterated subsampling). *For the computational issue of the iterated subsampling stage, the total time of training all DNN estimators* $\widehat{f}^{(j)}_{DNN,b,i}$ *for* $i \in \{1, \ldots, q\}$ *and* $j \in \{1, \ldots, q'\}$ *(iterated subsampling stage) is less than the time of training all DNN estimators in the first subsampling stage, i.e.,* $\widehat{f}_{DNN,b,i}$ *for* $i \in \{1, \ldots, q\}$*. We can see the reason by analyzing the computational complexity of the iterated subsampling stage. In total, we need to train* $q \cdot q' = O(n^{1-\beta^2})$ *number of models with sample size* n^{β^2}*. As the assumption we made in Remark 4.2, the complexity of training a DNN is mainly determined by its size, sample size and the number of epochs, so the training time of the iterated stage is around* $q \cdot q' \cdot O(n^{\beta^2} \cdot n^{\beta^2}) = O(n^{1+\beta^2})$ *when the sample size is close to the size of DNN. Similarly, we can analyze that the complexity of training DNNs in the first subsampling stage is around* $O(n^{1+\beta})$*. Since* $\beta < 1$*, the complexity of the first subsampling stage will dominate the iterated stage when n is large enough. In other words, the complexity cost of applying the iterated subsampling technique is negligible when we are dealing with a huge dataset.*

For completeness, we present the *iterated* subsampling here in the remark below.

Remark 4.5 (Iterated subsampling). *With the same notations in Example 4.1, we can perform the iterated subsampling in three steps: (1) Let* $b = \lfloor n^\beta \rfloor$*, then apply the scalable subsampling technique to sample* $X_1, \ldots, X_n$ *and get q subsets* $\{B_i\}_{i=1}^q$*. Compute* $\bar{\theta}_{b,n,SS}$*; we call it "first stage subsampling"; (2) Take another subsample size* $b' = \lfloor b^\beta \rfloor$ *and apply scalable subagging method again to all* $\{B_i\}_{i=1}^q$*, i.e., as if* B_i *where the only data at hand and make subagging estimator for each* B_i *subsamples; such subagging estimator* $\bar{\theta}_{b',b,SS,i}$ *is computed by averaging* q' *estimators* $\{\hat{\theta}^{(j)}_{b',b,SS,i}\}_{j=1}^{q'}$*; here* $q' = \lfloor (b-b')/h' \rfloor + 1$*. As a result, we can get q number of* $\{\bar{\theta}_{b',b,SS,i}\}_{i=1}^q$*; we call it "iterated stage subsampling"; (3) Find the subsampling distribution* $L_{b',b,SS}(z) = q^{-1}\sum_{i=1}^q 1\left\{\kappa_b\left(\bar{\theta}_{b',b,SS,i} - \bar{\theta}_{b,n,SS}\right) \leq z\right\}$*;* κ_b *is a function of b. In the context of DNN estimation, we use* $\widehat{f}^{(j)}_{DNN,b,i}$ *to represent the DNN estimator in the iterated subsampling stage on the j-th subsamples from the i-th subsample in the first stage subsampling.*

Denote $J_n(z) = \mathbb{P}(\kappa_n(\bar{\theta}_{b,n,SS} - \theta) \leq z)$, and $J(z)$ is the limit of $J_n(z)$ as $n \longrightarrow \infty$; recall that (10) implied that $J(z)$ is Gaussian. Proposition 2.1 of [20] shows that $L_{b',b,SS}(z)$ converges to $J(z)$ in probability for all points of continuity of $J(z)$. Due to Eq. (10), $J(z)$ is continuous everywhere, and therefore the convergence is uniform. Thus, both $L_{b',b,SS}(z)$ and $J_n(z)$ converge in a uniform fashion to $J(z)$ in probability which implies that:

$$\sup_z \left|L_{b',b,SS}(z) - J_n(z)\right| \xrightarrow{p} 0, \text{ as } n \longrightarrow \infty. \tag{16}$$

Thus, iterated subsampling can be used to estimate the distribution J_n. We can build the CI in a pivotal style without explicitly referring to the form of J that involves the two unknown parameters. A further issue is that normality might not be well represented in J_n since it is based on an average of q quantities; having a large q requires a huge n. To compensate for the data size requirement, we take a specific approach to build CI which can be considered as a combination of PCI and QCI to some extent. Algorithm 2 describes all the steps to construct the CI for f at a point $\boldsymbol{x}$ based on the subagging DNN estimator and iterated subsampling method.

Algorithm 2 PCI of $f(\boldsymbol{x})$ based on iterated subsampling

```
Step 1  Fix the subsample size $b$, compute $\overline{f}_{\mathrm{DNN}}(\boldsymbol{x})$ at point $\boldsymbol{x}$.
Step 2  Fix the subsample size $b'$ of iterated subsampling, perform necessary steps in Remark 4.5
        to find
            $L_{b',b,SS}(z) = q^{-1}\sum_{i=1}^{q} 1\left\{\kappa_b\left(\overline{f}_{\mathrm{DNN},i}(\boldsymbol{x}) - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x})\right) \leq z\right\};$
        here $\overline{f}_{\mathrm{DNN},i}(\boldsymbol{x}) = \frac{1}{q'}\sum_{j=1}^{q'}\widehat{f}^{(j)}_{\mathrm{DNN},b,i}$ is the subagging DNN estimator on the $i$-th subsamples
        in Step 1 at the point $\boldsymbol{x}$.
Step 3  Denote the $\delta/2$ and $1-\delta/2$ quantile values of the distribution $L_{b',b,SS}(z)$ as $b_l$ and $b_u$.
Step 4  Determine the PCI of $f(\boldsymbol{x})$ by:
            $[\overline{f}_{\mathrm{DNN}}(\boldsymbol{x}) - b_u/\kappa_n\,,\ \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}) - b_l/\kappa_n].$    (17)
        In other words, we take $B_l = \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}) - b_u/\kappa_n$ and $B_u = \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}) - b_l/\kappa_n$.
```

Note that to construct the PCI (17) above, the values of κ_n and κ_b are required. Recall that $\kappa_n = n^{\frac{1-\beta+2\alpha\beta}{2}}$ and $\kappa_b = n^{\beta\frac{1-\beta+2\alpha\beta}{2}}$. Although β is the practitioner's choice, α is typically unknown. Remark 4.6 explains how upper and lower bounds for α can be used in the PCI construction.

Remark 4.6. *In constructing the PCI (17) we can replace κ_b by a larger value (say $\bar{\kappa}_b$) and replace κ_n by a smaller value (say $\underline{\kappa}_n$) and the coverage bound of Eq.* (13) *would still be met. From Theorem 4.1, the fastest rate of the variance decrease is of order $O(n^{-1})$; so α could be as large as $1/2$ in which $\bar{\kappa}_b = n^{\frac{\beta}{2}}$. On the other hand, the slowest rate is influenced by $n^{-\frac{\xi}{\xi+d}}$; if we pretend the smoothness of the true model is equal to the input dimension (although it is actually smoother), we can take $\alpha = 1/4$ to compute $\underline{\kappa}_n = n^{\frac{1-\beta/2}{2}}$.*

5 PREDICTIVE INFERENCE WITH THE DNN ESTIMATOR

Most of the work in DNN estimation has applications in prediction although this is typically a point prediction. However, as in the estimation case, it is important to be able to quantify the accuracy of the point predictors which can be done via the construction of Prediction Intervals (PI); see related work of [18, 24, 25, 29] on predictive inference with dependent or independent data.

Consider the problem of predicting a response Y_0 that is associated with a regressor value of interest denoted by $\boldsymbol{x}_0$ and its corresponding prediction interval. The L_2 optimal point predictor of Y_0 is $f(\boldsymbol{x}_0)$ which is well approximated by $\bar{f}_{DNN}(\boldsymbol{x}_0)$ as Theorem 4.2 shows. To construct a PI for Y_0, we need to take the variability of the errors into account since, conditionally on $\boldsymbol{X}_0 = \boldsymbol{x}_0$, we have $Y_0 = f(\boldsymbol{x}_0) + \epsilon_0$.

If the model f and the error distribution F_ϵ were both known, we could construct a PI which covers Y_0 with $1-\delta$ confidence level as follows:

$$\left[f(\boldsymbol{x}_0)+z_{\epsilon,\delta/2}, f(\boldsymbol{x}_0)+z_{\epsilon,1-\delta/2}\right]; \tag{18}$$

here $z_{\epsilon,1-\delta/2}$ and $z_{\epsilon,\delta/2}$ are the $1-\delta/2$ and $\delta/2$ quantile values of F_ϵ, respectively. Of course, we do not know the true model f but we may replace it with our scalable subsampling DNN estimator $\overline{f}_{\text{DNN}}$. In addition, F_ϵ is also unknown and must be estimated; a typical estimator is $\widehat{F}_\epsilon$ which is the empirical distribution of residuals. To elaborate, we define $\widehat{F}_\epsilon$ as follows:

$$\begin{aligned}&\widehat{F}_\epsilon(z) := \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{\hat{\epsilon}_i\leq z};\ \mathbb{1}_{(\cdot)}\text{is the indicator function.}\\ &\hat{\epsilon}_i = f(\boldsymbol{x}_i)-\overline{f}_{\text{DNN}}(\boldsymbol{x}_i),\ \text{for } i=1,\ldots,n.\end{aligned} \tag{19}$$

To consistently estimate the error distribution F_ϵ, we need to make some mild assumptions on F_ϵ, namely:

- B2: The error distribution F_ϵ has zero mean and is differentiable on the real line and $\sup_z p_\epsilon(z)<\infty$ where $p_\epsilon(z)$ is the density function of error ϵ.

The following Lemma can be proved analogously to the proof of Lemma 4.1 in [25].

Lemma 5.1. *Under A1-A5 and B1, we have* $\sup_z|\widehat{F}_\epsilon(z)-F_\epsilon(z)|\xrightarrow{p}0$.

We can then apply the PI below to approximate the 'oracle' PI of Eq. (18):

$$\left[\overline{f}_{\text{DNN}}(\boldsymbol{x}_0)+\hat{z}_{\epsilon,\delta/2}, \overline{f}_{\text{DNN}}(\boldsymbol{x}_0)+\hat{z}_{\epsilon,1-\delta/2}\right]; \tag{20}$$

here $\hat{z}_{\epsilon,1-\delta/2}$ and $\hat{z}_{\epsilon,\delta/2}$ are the $1-\delta/2$ and $\delta/2$ quantile values of $\widehat{F}_\epsilon$, respectively. To construct this PI in practice, we can rely on Algorithm 3 below:

Algorithm 3 PI of Y_0 conditional on $\boldsymbol{x}_0$

```
Step 1   Train the subagging DNN estimator $\overline{f}_{\text{DNN}}(\cdot)$ and find the empirical distribution of
         residuals $\widehat{F}_\epsilon$ as Eq. (19).
Step 2   Evaluate the subagging DNN estimator at $\boldsymbol{x}_0$ to get $\overline{f}_{\text{DNN}}(\boldsymbol{x}_0)$.
Step 3   Determine $\hat{z}_{\epsilon,\delta/2}$ and $\hat{z}_{\epsilon,1-\delta/2}$ by taking lower $\delta/2$ and $1-\delta/2$ quantiles of $\widehat{F}_\epsilon$.
Step 4   Construct PI as Eq. (20).
```

We claim that the PI in Eq. (20) is asymptotically valid (conditionally on $\boldsymbol{X}_0=\boldsymbol{x}_0$), i.e., it satisfies

$$\mathbb{P}\left(Y_0\in\left[\overline{f}_{\text{DNN}}(\boldsymbol{x}_0)+\hat{z}_{\epsilon,\delta/2}, \overline{f}_{\text{DNN}}(\boldsymbol{x}_0)+\hat{z}_{\epsilon,1-\delta/2}\right]\right)\xrightarrow{p}1-\delta, \tag{21}$$

where the above probability is conditional on $\boldsymbol{X}_0=\boldsymbol{x}_0$. This statement is guaranteed by Theorem 5.1. To describe it, denote $Y_0^*=\overline{f}_{\text{DNN}}(\boldsymbol{x}_0)+\epsilon_0^*$ where ϵ_0^* has the distribution $\widehat{F}_\epsilon$.

Theorem 5.1. *Under A1-A5 and B1-B2, the distribution of Y_0^* converges to the distribution of Y_0 uniformly (in probability), i.e.,*

$$\sup_z\left|F_{Y_0^*|\boldsymbol{X}_0=\boldsymbol{x}_0}(z)-F_{Y_0|\boldsymbol{X}_0=\boldsymbol{x}_0}(z)\right|\xrightarrow{p}0,\ \textit{as } n\to\infty. \tag{22}$$

Although the PI in Eq. (20) is asymptotically valid, it may undercover Y_0 in the finite sample case. This problem is mainly due to two reasons: (1) PI in Eq. (20) does not take the variability of model estimation into account; and (2) the scale of the error distribution is typically underestimated by the residual distribution with finite samples. For issue (1), we can rely on a so-called pertinent PI which is able to capture the model estimation variability; this pertinence property is crucial, especially for the prediction inference of time series data in which multiple-step ahead forecasting is usually required. For issue (2), we can "enlarge" the residual distribution by basing it on the so-called predictive (as opposed to fitted) residuals. Although the predictive residuals are asymptotically equivalent to the fitted residuals, i.e., $\hat{\epsilon}$ in Eq. (19), the corresponding PI could have a better coverage rate; see [19] for the formal definition of pertinent PI and predictive residuals.

In this paper, due to the computational issues in fitting DNN models, we only build the PI in Eq. (20). Taking a fairly large enough sample size in Section 6, this PI works well, and its empirical coverage rate is only slightly lower than that of the oracle.

6 SIMULATIONS

In this section, we attempt to check the performance of the scalable subagging DNN estimator with simulation examples. More specifically, we consider two aspects of one estimator: (1) Time-complexity, we take the running time of the training stage to measure its complexity for a fixed hyperparameter setting, e.g., fixed number of epochs and batch size; (2) Estimation accuracy, we take empirical MSE (mean square error)/MSPE (mean square prediction error) and empirical coverage rate to measure the accuracy of point estimations/predictions and confidence/prediction intervals. We deployed the simulation studies on the CentOS Linux 7 (Core) system. All simulations run parallelly with 40 CPUs (Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz).

6.1 Simulations on point estimations

As shown in Section 4, the scalable subagging DNN estimator is more computationally efficient but also more accurate meantime compared to the DNN estimator trained with the whole sample size under some mild conditions. Here, we hope to verify such dominating performance with simulated data. To perform simulations, we consider below models:

- Model-1: $Y = \sum_{i=1}^{10} X_i + \epsilon$, where $(X_1, \ldots, X_{10}) \sim N(0, \boldsymbol{I})$.
- Model-2: $Y = \sum_{i=1}^{10} i \cdot X_i + \epsilon$, where $(X_1, \ldots, X_{10}) \sim N(0, \boldsymbol{I})$.
- Model-3: $Y = X_1^2 + \sin(X_2 + X_3) + \epsilon$, where $(X_1, X_2, X_3) \sim N(0, \boldsymbol{I})$.
- Model-4: $Y = X_1^2 + \sin(X_2 + X_3) + \exp(-|X_4 + X_5|) + \epsilon$, where $(X_1, X_2, X_3, X_4, X_5) \sim N(0, \boldsymbol{I})$;

here $\boldsymbol{I}$ is an identity matrix with the correct dimension for each model; ϵ is the standard normal error. To be consistent with folk wisdom, we build $\widehat{f}_{\text{DNN},b,i}$ with a relatively large depth to decrease the bias. Meanwhile, we take the width as large as possible to make its size close to the sample size so that A3 could be satisfied and we are in the under-parameterized region. In order to make a comprehensive comparison between the scalable subsampling DNN (SS-DNN) estimator $\overline{f}_{\text{DNN}}$ and classical DNN estimators, we consider 5 DNN estimators which are trained with the whole sample but with different structures:

(1) A DNN possesses the same depth and width as $\widehat{f}_{\text{DNN},b,i}$. We denote it "S-DNN".
(2) A DNN possesses the same depth as $\widehat{f}_{\text{DNN},b,i}$, but a larger width so that its size is close to the sample size. We denote it "DNN-deep-1".
(3) A DNN possesses the same depth as $\widehat{f}_{\text{DNN},b,i}$, but a larger width so that its size is close to half of the sample size. We denote it "DNN-deep-2".

(4) A DNN possesses only one hidden layer, but a larger width so that its size is close to the sample size. We denote it "DNN-wide-1".
(5) A DNN possesses only one hidden layer, but a larger width so that its size is close to half of the sample size. We denote it "DNN-wide-2".

We deploy DNN (1) to check the performance of a DNN with the same structure as $\widehat{f}_{\text{DNN},b,i}$, but it is trained with the whole dataset. We deploy DNNs (2) - (5) to challenge the scalable subsampling DNN estimator with various wide or deep DNNs. We build the DNN estimator with *PyTorch* in *Python*. To train all different DNNs, we use the stochastic gradient descent algorithm *Adam* developed by [15] with a learning rate 0.01. In addition, we take the number of epochs and batch size to be 200 and 10 to make the DNN fully trained for the first and iterated subsampling stages, respectively. The choice of batch size, the number of epochs and the base sample size is intended to make sure all five DNN estimators can achieve a great estimation performance (this is revealed by Table 1, all errors are small). Based on this fact, we then check the running time and the accuracy of different DNNs. We use the function *time.time()* in *Python* to compute the running time of the training procedure, namely Training Time.

To evaluate the point estimation performance, we apply two empirical MSE criteria:

$$\text{MSE-1:}\frac{1}{n}\sum_{i=1}^{n}(\widehat{f}_{\text{DNN}}(\boldsymbol{x}_i)-y_i)^2 \ ; \ \text{MSE-2:}\frac{1}{n}\sum_{i=1}^{n}(\widehat{f}_{\text{DNN}}(\boldsymbol{x}_i)-f(\boldsymbol{x}_i))^2;$$

here $\widehat{f}_{\text{DNN}}(\cdot)$ represents different DNN estimators and $f(\cdot)$ is the true regression function; $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ are realizations of samples; we call it training data.

An estimator is optimal in MSE-1 criterion if its MSE-1 is closest to the sample variance of errors, namely $\hat{\sigma}_\epsilon^2 = \frac{1}{n}\sum_{i=1}^n \epsilon_i^2$; here $\{\epsilon_i^2\}_{i=1}^n$ are observed error values. An estimator is optimal in the MSE-2 criterion if its MSE-2 is closest to 0. We present MSE-1 and MSE-2 of different estimators in Table 1. In addition, we also present $\hat{\sigma}_\epsilon^2$ of the corresponding simulated sample as the benchmark to compare the performance of different estimators according to the MSE-1 criterion.

Beyond the point estimation measured on training data, we are also interested in the performance of difference DNN estimators on test data. Thus, we generate new samples: $\{\boldsymbol{x}_{0,i}, y_{0,i}\}_{i=1}^N$; here we take $N = 2\cdot 10^5$ to evaluate the prediction performance. Similarly, we consider two MSPEs and we denote them MSPE-1 and MSPE-2 following:

$$\text{MSPE-1:}\frac{1}{N}\sum_{i=1}^{N}(\widehat{f}_{\text{DNN}}(\boldsymbol{x}_{0,i})-y_{0,i})^2 \ ; \ \text{MSPE-2:}\frac{1}{N}\sum_{i=1}^{N}(\widehat{f}_{\text{DNN}}(\boldsymbol{x}_{0,i})-f(\boldsymbol{x}_{0,i}))^2;$$

we expect that the best estimator on prediction tasks should have the smallest MSPE-2 and the MSPE-1 which is closest to $\hat{\sigma}_{\epsilon,0}^2 = \frac{1}{N}\sum_{i=1}^N(\epsilon_{0,i})^2$; here $\{\epsilon_{0,i}\}_{i=1}^N$ are observed error values for the test data. We present all simulation results in Table 1; here empirical MSE/MSPE and Training Time (in seconds) were computed as averages of 200 replications.

We can summarize several notable findings from the simulation results:

- $\overline{f}_{\text{DNN}}$ is always the most computationally efficient one, it is even faster than applying a single DNN estimator with the same structure as $\widehat{f}_{\text{DNN},b,i}$ but trained on the whole sample. Notably, our scalable subsampling procedure can save more than 50% running time of applying DNN-deep-1 or DNN-wide-1 for Model-4 data with size $2\cdot 10^4$.
- According to the MSE-1, $\overline{f}_{\text{DNN}}$ is the most accurate one for all simulations. For example, the MSE-1 of $\overline{f}_{\text{DNN}}$ is around 40% closer[3] to $\hat{\sigma}_\epsilon^2$ than the MSE-1 of DNN-deep-1 and DNN-deep-2

[3]Saying that the error E_1 is $a\%$ closer to $\hat{\sigma}_\epsilon^2$ than the error E_2 means that $|(E_1-\hat{\sigma}_\epsilon^2)|/|(E_2-\hat{\sigma}_\epsilon^2)|$ is $a\%$.

Table 1. MSE/MSPE and Training Time (in seconds) of different DNN models on various simulation datasets with error terms.

Estimator:	SS-DNN	S-DNN	DNN-deep-1	DNN-deep-2	DNN-wide-1	DNN-wide-2
Model-1, $n = 10^4$, $\hat{\sigma}_\epsilon^2 = 1.0011$, $\hat{\sigma}_{\epsilon,0}^2 = 1.0003$						
Width	[20,20]	[20,20]	[90,90]	[60,60]	[800]	[400]
MSE-1	1.0034	1.0168	0.9975	1.0036	1.0136	1.0151
MSE-2	0.1011	0.0579	0.1039	0.0894	0.0466	0.0433
MSPE-1	1.1020	1.0678	1.1299	1.1059	1.0543	1.0487
MSPE-2	0.1019	0.0675	0.1296	0.1057	0.0540	0.0484
Training Time	209	225	403	303	373	274
Model-2, $n = 10^4$, $\hat{\sigma}_\epsilon^2 = 1.0012$, $\hat{\sigma}_{\epsilon,0}^2 = 1.0011$						
Width	[20,20]	[20,20]	[90,90]	[60,60]	[800]	[400]
MSE-1	1.0506	1.1355	1.1314	1.1350	1.0768	1.0745
MSE-2	0.1232	0.1625	0.1889	0.1839	0.1249	0.1194
MSPE-1	1.1339	1.1469	1.1841	1.1737	1.1254	1.1237
MSPE-2	0.1338	0.1468	0.1841	0.1736	0.1253	0.1238
Training Time	224	240	417	320	376	280
Model-3, $n = 10^4$, $\hat{\sigma}_\epsilon^2 = 0.9997$,$\hat{\sigma}_{\epsilon,0}^2 = 1.0001$						
Width	[15,15,15]	[15,15,15]	[65,65,65]	[45,45,45]	[2000]	[1000]
MSE-1	1.0014	1.0361	1.0299	1.0308	1.0286	1.0290
MSE-2	0.0296	0.0536	0.0533	0.0522	0.0426	0.0431
MSPE-1	1.0310	1.0565	1.0572	1.0571	1.0453	1.0449
MSPE-2	0.0310	0.0564	0.0572	0.0570	0.0453	0.0449
Training Time	353	379	561	468	483	363
Model-4, $n = 10^4$, $\hat{\sigma}_\epsilon^2 = 1.0014$,$\hat{\sigma}_{\epsilon,0}^2 = 1.0003$						
Width	[15,15,15]	[15,15,15]	[65,65,65]	[45,45,45]	[2000]	[1000]
MSE-1	1.0243	1.0488	1.0318	1.0350	1.0457	1.0460
MSE-2	0.0757	0.0830	0.1076	0.0980	0.0729	0.0728
MSPE-1	1.0792	1.0878	1.1117	1.1048	1.0756	1.0752
MSPE-2	0.0790	0.0875	0.1114	0.1045	0.0754	0.0749
Training Time	359	376	560	471	551	394
Model-4, $n = 2 \cdot 10^4$, $\hat{\sigma}_\epsilon^2 = 0.9991$,$\hat{\sigma}_{\epsilon,0}^2 = 0.9999$						
Width	[20,20,20]	[20,20,20]	[95,95,95]	[65,65,65]	[2800]	[1400]
MSE-1	1.0093	1.0483	1.0419	1.0438	1.0508	1.0508
MSE-2	0.0490	0.0653	0.0686	0.0675	0.0635	0.0635
MSPE-1	1.0501	1.0669	1.0692	1.0689	1.0622	1.0625
MSPE-2	0.0502	0.0670	0.0692	0.0689	0.0623	0.0626
Training Time	748	775	1684	1198	1549	998

Note: "width" represents the number of neurons of each hidden layer, e.g., [20, 20] means that there are two hidden layers within the DNN and each has 20 neurons.

for Model-2 data with 10^4 size, which is remarkable noticing that $\overline{f}_{\text{DNN}}$ is trained with less time.

- According to the MSE-2, $\overline{f}_{\text{DNN}}$ can work best when the data is large enough for Models 3-4 which are non-linear. In particular, MSE-2 of $\overline{f}_{\text{DNN}}$ is more than 20% less[4] than the MSE-2 of all other models for Model-4 data with size $2 \cdot 10^4$.
  For Model-2, the performance of $\overline{f}_{\text{DNN}}$ is just slightly worse than the optimal estimator. For Model-1, the performance of $\overline{f}_{\text{DNN}}$ is still worse than the optimal estimator. We guess the reason may be that the Model-1 and Model-2 are linear models. In this case, a wide DNN is sufficient to mimic such a linear relationship.
- For MSPEs, $\overline{f}_{\text{DNN}}$ works slightly worse than the optimal model for Model-1 and Model-2 cases, but it turns out to be the optimal one for Model-3 and Model-4 cases. This phenomenon is consistent with the behavior of MSEs. More specifically, the MSPE-2 of $\overline{f}_{\text{DNN}}$ is more than 20% less than the MSPE-2 of all other models for Model-4 data with size $2 \cdot 10^4$; the MSPE-1 of $\overline{f}_{\text{DNN}}$ is around 50% closer to $\hat{\sigma}^2_{\epsilon,0}$ than the MSPE-1 of all other methods for Model-3 data with 10^4 size.
- The model-selection step for "wide" or "deep" type DNN estimators is necessary but it is obscure meanwhile; see DNN-wide-2 works better than DNN-wide-1 for the Model-2 MSE case; however, the situation reverses for the Model-3 MSE case. This phenomenon occurs for "Deep" type DNN estimators also; see the performance of S-DNN, DNN-deep-1 and DNN-wide-2; there is no single one that beats the others uniformly. For MSPE, we can also find such a reverse phenomenon. On the other hand, by applying the scalable subagging estimator, we can avoid the model-selection difficulty and just make $\widehat{f}_{\text{DNN},b,i}$ deep and large enough.

We also considered evaluating the ability of various DNN estimators to estimate regression models solely, i.e., removing the error terms in the four simulation models above. Due to this change, the MSE-1 error is equivalent to the MSE-2 error. We found that the SS-DNN is still the most time-efficient estimator. It even runs faster than training S-DNN with the whole sample size. Applying the scalable subagging method can gain more computational savings for training with a larger sample size or a larger model. The SS-DNN is also the most accurate estimator except in the case with 10^4 Model-4 simulated data. For this case, the accuracy of SS-DNN is slightly worse than the estimator DNN-deep-1. We conjecture the reason is that Model-4 is relatively complicated so a DNN with 3 depths and constant width 15 has a high bias. After increasing the sample size to 20000, the subagging estimator beats other models.

6.2 Simulations for CI and PI

We continue using the four models in Section 6.1 to test the accuracy of multiple confidence and prediction intervals defined in previous sections with scalable subagging DNN estimators. To make sure we have enough subsamples to do iterated subsampling for CI, we take the sample size to be $2 \cdot 10^5$, which implies $q = 38$ when $\beta = 0.7$. It further implies that the number of subsamples for the iterated subagging stage is $q = \lfloor n^{\beta(1-\beta)} \rfloor = 12$. For developing the prediction interval, we take the sample size to be 10^4 or $2 \cdot 10^4$. To determine the structure of the subagging DNN estimator, we keep the strategy summarized in the previous subsection, i.e., we make its size as close to the sample size as possible no matter in the first or the iterated subsampling stage. We take the same training setting with *PyTorch* to find $\overline{f}_{\text{DNN}}(\boldsymbol{x}_0)$ as we have done in Section 6.1.

[4]Saying that the error E_1 is $a\%$ less than the error E_2 means that $(E_1 - E_2)/E_2$ is negative and $|(E_1 - E_2)/E_2| = a\%$.

We call the naive QCI which is determined by the equal-tail quantile of estimations $\{\widehat{f}_{\mathrm{DNN},b,1}(\boldsymbol{x}), \ldots, \widehat{f}_{\mathrm{DNN},b,q}(\boldsymbol{x})\}$ QCI-1; we should notice that this QCI may be too conservative as we explained in Example 4.1; we call the QCI based on Eq. (17) QCI-2; we call the PCI based on Eq. (12) PCI-1; we call the PCI based on Eq. (15) with taking $2\alpha = 0$, PCI-2; we call the PCI based on Eq. (15) with taking $2\alpha = 1$, PCI-3; the PI represents the prediction interval defined in Eq. (20). For all CIs and PI defined in previous sections, they have asymptotically validity conditional on the observation $\boldsymbol{X} = \boldsymbol{x}$. We attempt to check the conditional coverage rate with simulations for finite sample cases. To achieve this purpose, we fix 10 unchanged test points $\{(y_{t,1}, \boldsymbol{x}_{t,1}), \ldots, (y_{t,10}, \boldsymbol{x}_{t,10})\}$ which are different from training points for each simulation model; these 10 points can be recovered by setting *numpy.random.seed(0)* and generate sample according to the model.

To evaluate the performance of (conditional) CI for each test point, we repeat the simulation process $K = 500$ times and apply the below formulas to compute the empirical coverage rate (ECR) and empirical length (EL) of different CIs for each test point:

$$\mathrm{ECR}_j = \frac{1}{K}\sum_{i=1}^{K} \mathbb{1}_{f(\boldsymbol{x}_{t,j})\in[B_{l,i,j},B_{u,i,j}]}~,~\mathrm{EL}_j = \frac{1}{K}\sum_{i=1}^{K}(B_{u,i,j} - B_{l,i,j}), \text{for } j = 1, \ldots, 10;$$

here $f(\boldsymbol{x}_{t,j})$ is the true model value evaluated at the j-th test data point; $B_{u,i,j}$ and $B_{l,i,j}$ are the corresponding upper and lower bounds of different CIs at the i-th replication for the j-th test point, respectively. We take the nominal significance level $\delta = 0.05$. Simulation results are tabularized in Table 2.

To evaluate the performance of (conditional) PI for each test point, the procedure is slightly complicated and we summarize it in below four steps:

Step 1 Take the sample size n to be 10^4 or $2 \cdot 10^4$; simulate $K = 500$ sample sets: $\{(y_i^{(k)}, \boldsymbol{x}_i^{(k)})_{i=1}^n\}_{k=1}^K$ based on one of four simulation models.

Step 2 For each sample set, train the subsampling DNN estimator and build the prediction interval for 10 test points by:

$$[\overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_{t,j}) + \hat{z}_{\delta/2}, \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_{t,j}) + \hat{z}_{1-\delta/2}],~\text{for } j = 1, \ldots, 10,$$

where $\hat{z}_{\epsilon,1-\delta/2}$ and $\hat{z}_{\epsilon,\delta/2}$ are the $1-\delta/2$ and $\delta/2$ quantile values of the empirical distribution of the residuals, respectively.

Step 3 To check the performance of PIs for test points based on each sample set, simulate $\{y_{s,j}\}_{s=1}^M$ conditional on $\boldsymbol{x}_{t,j}$ for $j = 1, \ldots, 10$ pretending the true data-generating model is known and check the empirical coverage rate and empirical length by below formulas:

$$\mathrm{ECR}_{i,j} = \frac{1}{M}\sum_{s=1}^{M} \mathbb{1}_{y_{s,j}\in[B_{l,i,j},B_{u,i,j}]}~,~\mathrm{EL}_{i,j} = B_{u,i,j} - B_{l,i,j}, \text{for } j = 1, \ldots, 10; i = 1, \ldots, 500;$$

$B_{l,i,j}$ and $B_{u,i,j}$ are the corresponding upper and lower bounds of PI for the j-th test point based on i-th sample set defined in Step 2; $M = 3000$.

Step 4 For $j = 1, \ldots, 10$, estimate $\mathbb{P}(Y_0 \in PI|\boldsymbol{X}_0 = \boldsymbol{x}_{t,j})$ by the average of empirical coverage rate of corresponding (conditional) PI on K sample sets, i.e., Average$(\mathrm{ECR}_{i,j})$ w.r.t. i; estimate length of (conditional) PI for j-th test point by Average$(\mathrm{EL}_{i,j})$ w.r.t. i.

We take the nominal significance level $\delta = 0.05$. Simulation results are tabularized in Table 3.

Remark (Different levels of conditioning). *As explained in the work of [24], we have several conditioning levels to measure the performance of PI or CI. What we consider in this paper is* $\mathbb{P}_1 := \mathbb{P}(\cdot|\boldsymbol{X}_0 = \boldsymbol{x}_0)$ *which shall be interpreted as the conditional probability on* $\boldsymbol{X}_0 = \boldsymbol{x}_0$*. If we consider the empirical coverage rate of* $ECR_{i,j}$*, it approximates another conditioning level, i.e.,*

$\mathbb{P}_2 := \mathbb{P}(\cdot|\boldsymbol{X}_0 = \boldsymbol{x}_0, \{(Y_j, \boldsymbol{X}_j)\}_{j=1}^n)$; here $\{(Y_j, \boldsymbol{X}_j)\}_{j=1}^n$ represents the whole sample. By Lemma 4 of [24], if $A \in \sigma\left(\{\mathbf{X}_j\}_{j=1}^n, \{Y_j\}_{j=1}^n, X_f, Y_0\right)$ is an arbitrary measurable event, then $\mathbb{E}_{\{(Y_j,\boldsymbol{X}_j)\}_{j=1}^n}\mathbb{P}_2(A) = \mathbb{P}_1(A)$. Besides, $1-\delta$ conditional coverage under $\mathbb{P}_1$ will imply the marginal coverage $\mathbb{P}_0 := \mathbb{E}_{\boldsymbol{X}}\mathbb{P}_1$. This unconditional coverage is implied by the popular Conformal Prediction method in the machine learning community. Simulation studies show that our CIs and PIs also have great unconditional coverage.

We can summarize several findings based on simulation results:

- For the empirical coverage rate of quantile-type CIs, the naive QCI-1 over-covers true model values as we expect. Also, the corresponding CI length is always larger than the length of QCI-2 and it is actually the largest one among 5 different CIs. On the other hand, the specifically designed QCI-2 returns ECRs that are closer to the specified confidence level than QCI-1. Meanwhile, ECR of QCI-2 is larger than the nominal confidence level for almost all test points since we take κ_n and κ_b according to the strategy in Remark 4.6 to enlarge the CI.
- For the empirical coverage rate of pivot-type CIs, although the length of PCI-1 is the shortest one, the ECR of PCI-1 is less than the nominal confidence level for almost all test points since C_σ^2 may be underestimated and we may have the bias issue in practice. For the PCI-3 whose margin of error is enlarged in a mild way, although its ECR is always larger than PCI-1, it still undercover true model value mostly. For the PCI-2 in which the margin of error is enlarged in a most extreme way, it has a much better performance according to the coverage rate but with a larger CI length as a sacrifice.
- We claim that the PCI-2 is the optimal CI candidate according to the overall performance based on length and coverage rate. For example, considering the comparison between PCI-2 and QCI-2, we can find some cases in which both CIs have a close coverage rate but the length of PCI-2 is less than 50% length of QCI-2. This phenomenon sustains all four models. For the QCI-2, we conjecture it will be a good alternative if we have more samples so that $L_{b',b,SS}(x)$ can approximate $J_n(x)$ well in the iterated subsampling stage.
- For the prediction task, all PIs for four models and all test points have almost the same coverage rate and length. Most ECRs are slightly less than the nominal confidence level which is not a surprise since we omit the variability in the model estimation and the residual distribution may underestimate the true error distribution for a finite sample case. For the length of PI, all PIs' lengths are close to $2 \cdot z_{0.975}$ since the true error distribution is assumed to be standard normal in simulations and we took equal-tail PI.

Table 2. Empirical Coverage Rate and Empirical Length of different (conditional) CIs with various simulation models.

Test point:	1	2	3	4	5	6	7	8	9	10
Model-1, $n = 2 \cdot 10^5$										
ECR										
QCI-1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
QCI-2	0.988	0.984	0.998	1.000	0.984	0.932	0.998	0.984	0.994	0.976
PCI-1	0.938	0.918	0.954	0.940	0.836	0.610	0.942	0.938	0.890	0.918
PCI-2	0.946	0.996	0.998	1.000	0.882	0.984	1.000	0.954	0.938	1.000
PCI-3	0.938	0.922	0.958	0.960	0.836	0.616	0.948	0.938	0.890	0.928
EL										
QCI-1	2.94	3.05	2.49	2.46	1.85	1.30	1.94	2.65	1.81	4.19
QCI-2	1.73	1.50	1.54	1.54	1.20	0.97	1.16	1.48	1.24	1.81
PCI-1	0.46	0.45	0.38	0.37	0.29	0.20	0.28	0.38	0.28	0.61
PCI-2	0.47	0.63	0.67	1.28	0.30	0.37	0.61	0.41	0.31	1.15
PCI-3	0.46	0.45	0.38	0.41	0.29	0.21	0.28	0.39	0.28	0.63
Model-2, $n = 2 \cdot 10^5$										
ECR										
QCI-1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
QCI-2	0.952	0.980	0.980	0.962	0.960	0.746	0.984	0.974	0.970	0.958
PCI-1	0.932	0.952	0.954	0.926	0.916	0.894	0.944	0.940	0.936	0.958
PCI-2	0.946	0.984	0.996	1.000	0.940	0.990	0.984	0.960	0.952	1.000
PCI-3	0.932	0.956	0.958	0.948	0.916	0.898	0.950	0.940	0.936	0.964
EL										
QCI-1	3.31	2.60	3.04	3.35	2.58	2.35	2.75	3.04	3.09	4.21
QCI-2	1.44	1.30	1.39	1.52	1.25	1.10	1.39	1.43	1.44	1.60
PCI-1	0.51	0.40	0.47	0.52	0.41	0.37	0.40	0.48	0.49	0.61
PCI-2	0.52	0.57	0.73	1.33	0.41	0.47	0.67	0.50	0.50	1.14
PCI-3	0.51	0.40	0.48	0.55	0.41	0.37	0.41	0.48	0.49	0.63
Model-3, $n = 2 \cdot 10^5$										
ECR										
QCI-1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
QCI-2	0.998	1.000	1.000	0.998	0.996	0.974	0.996	0.998	1.000	1.000
PCI-1	0.858	0.920	0.728	0.920	0.804	0.880	0.930	0.918	0.914	0.920
PCI-2	1.000	0.938	1.000	1.000	1.000	0.948	0.964	0.998	0.932	0.988
PCI-3	0.920	0.920	0.842	0.924	0.820	0.882	0.930	0.922	0.914	0.922
EL										
QCI-1	1.51	1.47	0.51	0.57	0.93	2.94	2.00	1.32	1.12	0.94
QCI-2	1.03	1.16	0.47	0.52	0.75	1.83	1.57	0.97	0.90	0.76
PCI-1	0.24	0.23	0.08	0.09	0.15	0.46	0.31	0.21	0.17	0.15
PCI-2	1.46	0.25	0.90	0.23	0.53	0.63	0.35	0.33	0.20	0.24
PCI-3	0.28	0.23	0.10	0.09	0.15	0.47	0.31	0.21	0.17	0.15
Model-4, $n = 2 \cdot 10^5$										
ECR										
QCI-1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
QCI-2	0.852	1.000	0.998	0.966	0.998	0.968	0.994	1.000	0.996	0.940
PCI-1	0.932	0.902	0.940	0.912	0.928	0.938	0.910	0.932	0.590	0.776
PCI-2	0.980	1.000	0.986	0.996	0.998	0.948	1.000	0.998	1.000	1.000
PCI-3	0.936	0.948	0.942	0.920	0.938	0.938	0.936	0.940	0.614	0.780
EL										
QCI-1	3.71	0.79	1.00	1.91	2.92	1.48	1.35	2.48	1.49	0.89
QCI-2	2.10	0.74	0.84	1.38	2.02	0.97	1.03	1.68	1.08	0.70
PCI-1	0.56	0.12	0.15	0.29	0.45	0.22	0.21	0.39	0.23	0.13
PCI-2	0.82	0.93	0.20	0.54	1.01	0.24	0.90	0.64	0.72	0.29
PCI-3	0.57	0.14	0.15	0.29	0.47	0.22	0.23	0.39	0.24	0.14

Table 3. Empirical Coverage Rate and Empirical Length of (conditional) PIs with various simulation models.

Test point:	1	2	3	4	5	6	7	8	9	10
$n = 10^4$										
Model-1:										
EL = 3.91										
ECR:	0.929	0.934	0.934	0.931	0.939	0.944	0.940	0.935	0.940	0.925
Model-2:										
EL = 4.00										
ECR:	0.936	0.938	0.938	0.935	0.942	0.945	0.942	0.937	0.939	0.928
Model-3:										
EL = 3.93										
ECR:	0.948	0.946	0.949	0.949	0.948	0.936	0.943	0.947	0.947	0.949
Model-4:										
EL = 3.96										
ECR:	0.901	0.951	0.950	0.943	0.938	0.949	0.949	0.943	0.947	0.950
$n = 2 \cdot 10^4$										
Model-1:										
EL = 3.91										
ECR:	0.939	0.941	0.941	0.940	0.945	0.946	0.945	0.941	0.943	0.936
Model-2:										
EL = 3.95										
ECR:	0.941	0.944	0.942	0.943	0.945	0.947	0.946	0.943	0.944	0.938
Model-3:										
EL = 3.92										
ECR:	0.948	0.947	0.950	0.949	0.949	0.943	0.945	0.948	0.948	0.949
Model-4:										
EL = 3.94										
ECR:	0.921	0.950	0.950	0.945	0.941	0.948	0.949	0.944	0.947	0.949

7 EMPIRICAL STUDIES

In this section, we deploy an empirical study to verify the efficiency of the scalable subsampling technique with real-world data. We take the Combined Cycle Power Plant (CCPP) dataset from the UCI machine learning repository (https://archive.ics.uci.edu/dataset/294/combined+cycle+power+plant). This dataset includes 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011). There are four predictors, hourly average ambient variables Temperature (T), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V). The response variable is the net hourly electrical energy output (EP) of the plant.

We split the whole dataset into a training set with 6000 data points and a test dataset with 3568 data points by the function *train_test_split* from *sklearn.model_selection*. We set *random_state=1* to make our results reproducible. We let the subagging DNN estimators have the structure $[20, 15]$ which means there are two hidden layers; one layer has 20 neurons and the other has 15 neurons. Similarly, we let the S-DNN, DNN-deep-1, DNN-deep-2, DNN-wide-1 and DNN-wide-2 have structure $[20, 15]$, $[60, 50, 50]$, $[40, 30, 30]$, $[1000]$ and $[500]$, respectively. We intend to set non-uniform hidden layers, i.e., the number of neurons of all hidden layers is unequal. For the hyperparameter setting, we set $\beta = 0.6$; number of epochs is 200; the minibatch size is 10; the learning rate is 0.005. To evaluate the performance of different DNN estimators, we consider the standard MSE-1 and MSPE-1 on the test dataset. The MSE-2 and MSPE-2 criterion applied in Section 6 is not available since we do not know the true underlying regression model for real-world data. Beyond the point predictions, we also consider the 95% nominal level conditional PI with the scalable subsampling estimator $\overline{f}_{\mathrm{DNN}}(\boldsymbol{X})$ being the pivot.

The MSE-1 and MSPE-1 results of different DNN estimators are presented in Table 4. From there, the SS-DNN estimator shows the optimal performance according to the perspectives in computational time and estimations/predictions accuracy. To evaluate the performance of the conditional PI with the scalable subsampling estimator $\overline{f}_{\mathrm{DNN}}(\boldsymbol{X})$ being the pivot, we consider conditioning $\mathbb{P}_2$ for each test point, i.e., the probability to cover the true test point conditional on each test predictors vector and the whole 6000 training data points. To better show the coverage results, we sort all test data points in increasing order w.r.t. EP values. Then, we plot the sorted EP values and their order indices with their corresponding PIs in Fig. 3; see the plot from Appendix: B. To present the results more easily, we randomly sample 200 data points from the test dataset and plot sorted EP values and their associated PIs in Fig. 2. We further consider the empirically average coverage rate for all test points, i.e., $\sum_{i=1}^{3568}\mathbb{1}(y_i \in \mathrm{PI}_i)/3568$; $\mathbb{1}$ is the indicator function; y_i is the i-th test point and PI_i is its associated PI. We should notice that this average coverage rate is dedicated to estimating the conditioning probability $\mathbb{P}(\cdot|\{(Y_j,\boldsymbol{X}_j)\}_{j=1}^n)$; $\{(Y_j,\boldsymbol{X}_j)\}_{j=1}^n$ represents the whole sample; $n = 6000$ in this empirical study. It turns out that the overall average coverage rate for all 3568 test points is 0.952 and the average PI length is 18.523.

Table 4. MSE/MSPE and Training Time (in seconds) of different DNN models on the EECP dataset.

Estimator:	SS-DNN	S-DNN	DNN-deep-1	DNN-deep-2	DNN-wide-1	DNN-wide-2
Width	[20,15]	[20,15]	[60,50,50]	[40,30,30]	[1000]	[500]
MSE-1	24.153	46.044	30.511	29.427	39.877	32.006
MSPE-1	25.240	47.982	31.965	30.892	41.736	33.656
Training Time	94	95	132	121	99	88

Note: "width" represents the number of neurons of each hidden layer, e.g., [20, 15] means that there are two hidden layers within the DNN, and one has 20 neurons and the other one has 15 neurons.

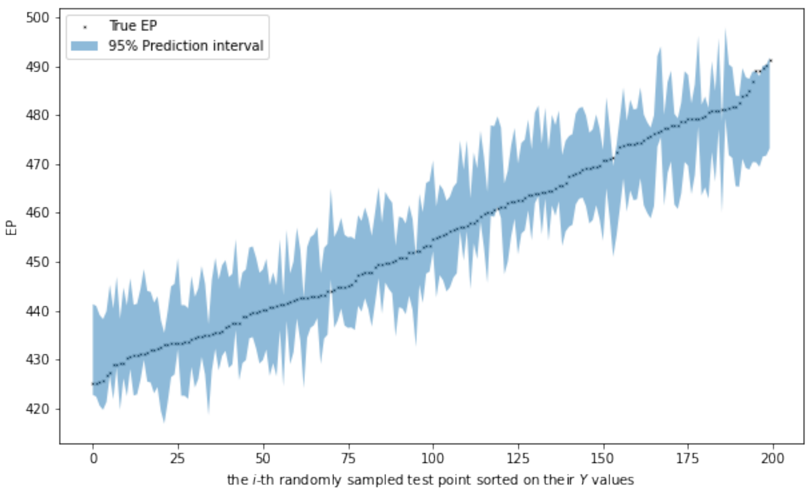


Fig. 2. Sorted EP values of 200 randomly sampled test points in increasing order and their corresponding conditional PIs with $\overline{f}_{\text{DNN}}(X)$ being the pivot.

8 CONCLUSIONS

In this paper, we revisit the error bound of fully connected DNN with the ReLU activation function on estimating regression models. By taking into account the latest DNN approximation results, we improve the current error bound. Under some mild conditions, we show that the error bound of the DNN estimator may be further improved by applying the scalable subsampling technique. As a result, the scalable subsampling DNN estimator is computationally efficient without sacrificing accuracy. The theoretical result is verified by simulation with various linear or non-linear regression models and empirical studies.

Beyond the error analysis for point estimations and point predictions, we propose different approaches to build asymptotically valid confidence and prediction intervals. More specifically, to overcome the undercoverage issue of CIs with finite samples, we consider several methods to enlarge the CI. As shown by simulations, our point estimations/predictions and confidence/prediction intervals based on scalable subsampling work well in practice. All in all, the scalable subsampling DNN estimator offers the complete package in terms of statistical inference, i.e., (a) computational efficiency; (b) point estimation/prediction accuracy; and (c) allowing for the construction of practically useful confidence and prediction intervals.

APPENDIX A: PROOFS

Proof of Theorem 4.1. This result can be easily shown based on the proof of Theorem 1 in the work of [9]. We take the intermediate result from the final step of their proof: With probability at least $1-\exp(-\gamma)$,

$$\left\|\widehat{f}_{\text{DNN}}-f\right\|_{L_2(X)} \leq C\left(\sqrt{\frac{H^2L^2\log\left(H^2L\right)}{n}\log n}+\sqrt{\frac{\log\log n+\gamma}{n}}+\epsilon_n\right), \tag{23}$$

where C is an appropriate constant; in this proof, C represents appropriate constants and its meaning may change according to the context; $\epsilon_n = \|f_{\mathrm{DNN}} - f\|_\infty$; $f_{\mathrm{DNN}} = \arg\min_{f_\theta \in \mathcal{F}_{\mathrm{DNN}}} \|f_\theta - f\|_\infty$. By Theorem 3.1 of [28] and Lemma 1 of [9], we can conclude that there is a standard fully connected DNN whose depth and width satisfy below inequalities:

$$\begin{aligned} H &\le C\epsilon_n^{-\frac{d}{\xi}} \log\left(1/\epsilon_n\right), \\ L &\le C \cdot \log\left(1/\epsilon_n\right), \end{aligned} \tag{24}$$

for any ϵ_n; Furthermore, we can find the upper bound of $H^2L^2\log\left(H^2L\right)$ based on Eq. (24):

$$H^2L^2\log\left(H^2L\right) \le C \cdot \epsilon_n^{-\frac{2d}{\xi}} \left(\log\left(1/\epsilon_n\right)\right)^5 .$$

Subsequently, we rewrite the Eq. (23) as below:

$$\left\|\widehat{f}_{\mathrm{DNN}} - f\right\|_{L_2(X)} \le C\left(\sqrt{\frac{\epsilon_n^{-\frac{2d}{\xi}} \left(\log\left(1/\epsilon_n\right)\right)^5}{n}\log n} + \sqrt{\frac{\log\log n + \gamma}{n}} + \epsilon_n\right). \tag{25}$$

To optimize the bound, we can choose $\epsilon_n = n^{-\frac{\xi}{2(\xi+d)}}, H = \Theta(n^{\frac{d}{2(\xi+d)}}\log n), L = \Theta(\log n)$. This gives:

$$\left\|\widehat{f}_{\mathrm{DNN}} - f\right\|_{L_2(X)} \le C\left(n^{-\frac{\xi}{2(\xi+d)}}\log^3 n + \sqrt{\frac{\log\log n + \gamma}{n}}\right). \tag{26}$$

As a result, we get:

$$\left\|\widehat{f}_{\mathrm{DNN}} - f\right\|^2_{L_2(X)} \le C\left(n^{-\frac{\xi}{(\xi+d)}}\log^6 n + \frac{\log\log n + \gamma}{n}\right). \tag{27}$$

Finally, we take $\gamma = n^{\frac{d}{d+\xi}}\log^6(n)$, which implies Theorem 4.1.

□

Proof of Theorem 4.2. Under A1-A5, we can analyze the expected square error for the subagging DNN estimator as below:

$$\begin{aligned} &\mathbb{E}(\overline{f}_{\mathrm{DNN}}(\boldsymbol{X}) - f(\boldsymbol{X}))^2 \\ &= \mathbb{E}\left[\frac{1}{q}\sum_{i=1}^{q}\widehat{f}_{\mathrm{DNN},b,i}(\boldsymbol{X}) - f(\boldsymbol{X})\right]^2 \\ &= \frac{1}{q^2}\mathbb{E}\left[\sum_{i=1}^{q}\left(\widehat{f}_{\mathrm{DNN},b,i}(\boldsymbol{X}) - f(\boldsymbol{X})\right)\right]^2 \\ &= \frac{1}{q^2}\mathbb{E}\left[\sum_{i=1}^{q}\left(\widehat{f}_{\mathrm{DNN},b,i}(\boldsymbol{X}) - f(\boldsymbol{X})\right)^2\right] + \frac{1}{q^2}\mathbb{E}\left[\sum_{i,j,i\neq j}\left(\widehat{f}_{\mathrm{DNN},b,i}(\boldsymbol{X}) - f(\boldsymbol{X})\right)\cdot\left(\widehat{f}_{\mathrm{DNN},b,j}(\boldsymbol{X}) - f(\boldsymbol{X})\right)\right]. \end{aligned} \tag{28}$$

For the first term on the r.h.s. of Eq. (28), by the error bound ignoring the slowly varying term, we can get:

$$\begin{aligned}\frac{1}{q^2}\mathbb{E}\left[\sum_{i=1}^{q}\left(\widehat{f}_{\mathrm{DNN},b,i}(\boldsymbol{X})-f(\boldsymbol{X})\right)^2\right] &\leq \frac{1}{q^2}\cdot q\cdot O\left(n^{-\frac{\beta\xi}{\xi+d}}\right)\\ &=\frac{1}{q}O\left(\frac{1}{n^{\frac{\beta\xi}{\xi+d}}}\right)\\ &=O\left(\frac{1}{n^{\frac{\beta\xi}{\xi+d}+1-\beta}}\right);\end{aligned}\tag{29}$$

this is satisfied with at least probability $(1-\exp(-n^{\frac{d}{\xi+d}}\log^6 n))^q$.

Ideally, we hope β can take a small value to improve the error bound for Eq. (29). However, it is restricted to do this since the bias of the subagging estimator will get increased once we take β smaller and smaller. Thus, we need to consider the second term on the r.h.s. of Eq. (28). Start by considering on specific pair:

$$\begin{aligned}&\mathbb{E}\left[\left(\widehat{f}_{\mathrm{DNN},b,i}(\boldsymbol{X})-f(\boldsymbol{X})\right)\cdot\left(\widehat{f}_{\mathrm{DNN},b,j}(\boldsymbol{X})-f(\boldsymbol{X})\right)\right]\\ &=\mathbb{E}\left[\mathbb{E}\left[\left(\widehat{f}_{\mathrm{DNN},b,i}(\boldsymbol{X})-f(\boldsymbol{X})\right)\cdot\left(\widehat{f}_{\mathrm{DNN},b,j}(\boldsymbol{X})-f(\boldsymbol{X})\right)\middle|\boldsymbol{X}\right]\right]\\ &=\mathbb{E}\left[\mathbb{E}\left[\left(\widehat{f}_{\mathrm{DNN},b,i}(\boldsymbol{X})-f(\boldsymbol{X})\right)\middle|\boldsymbol{X}\right]\cdot\mathbb{E}\left[\left(\widehat{f}_{\mathrm{DNN},b,j}(\boldsymbol{X})-f(\boldsymbol{X})\right)\middle|\boldsymbol{X}\right]\right].\end{aligned}\tag{30}$$

The last equality is due to the independence between subsample B_i and B_j. As we mentioned in the main text, we face difficulty in determining the rate of the bias of the subagging estimator. Thus, A4 and A5 are used to make additional assumptions on the bias term. We present A4 as below:

$$\mathbb{E}(\widehat{f}_{\mathrm{DNN}}(\boldsymbol{x})-f(\boldsymbol{x}))=O(n^{-\Lambda/2})\ ;\ \mathbb{E}(\widehat{f}_{\mathrm{DNN},b,i}(\boldsymbol{x})-f(\boldsymbol{x}))=O(n^{-\beta\Lambda/2}).$$

A5 then requires the bias order of $\widehat{f}_{\mathrm{DNN}}$ satisfies the inequality: $\Lambda>\frac{\xi}{\xi+d}$.

Then, we can find the order of Eq. (30) is:

$$\mathbb{E}\left[\left(\widehat{f}_{\mathrm{DNN},b,i}(\boldsymbol{X})-f(\boldsymbol{X})\right)\cdot\left(\widehat{f}_{\mathrm{DNN},b,j}(\boldsymbol{X})-f(\boldsymbol{X})\right)\right]=O(n^{-\beta\Lambda}).$$

Combine these two pieces, we can analyze Eq. (28):

$$\begin{aligned}&\mathbb{E}(\bar{f}_{\mathrm{DNN}}(\boldsymbol{X})-f(\boldsymbol{X}))^2\\ &\leq O\left(\frac{1}{n^{\frac{\beta\xi}{\xi+d}+1-\beta}}\right)+2\cdot\frac{1}{q^2}\cdot\binom{q}{2}\cdot O\left(\frac{1}{n^{\beta\Lambda}}\right)\\ &=O\left(\frac{1}{n^{\frac{\beta\xi}{\xi+d}+1-\beta}}\right)+O\left(\frac{1}{n^{\beta\Lambda}}\right).\end{aligned}\tag{31}$$

If the bias term is more negligible than the other term, i.e.,

$$\beta\Lambda\geq\frac{\beta\xi}{\xi+d}+1-\beta,\ i.e., \beta\geq\frac{1}{1+\Lambda-\frac{\xi}{\xi+d}}.$$

The above lower bound satisfies the requirement of β being positive. Then, Λ needs to be larger than $\frac{\xi}{\xi+d}$ to make sure the lower bound of β is less than 1 which is satisfied due to A5. Meanwhile, we want to take β as small as possible, i.e., $\beta = \frac{1}{1+\Lambda-\frac{\xi}{\xi+d}}$. This results in the error bound below:

$$\mathbb{E}(\overline{f}_{\mathrm{DNN}}(\boldsymbol{X}) - f(\boldsymbol{X}))^2 \leq O\left(n^{\frac{-\Lambda}{\Lambda+\frac{d}{\xi+d}}}\right).$$

The fact that $\frac{\Lambda}{\Lambda+\frac{d}{\xi+d}}$ is larger than $\frac{\xi}{\xi+d}$ is guaranteed by the requirement that $\Lambda > \frac{\xi}{\xi+d}$, i.e., A5 again.

□

Proof of Theorem 5.1. Since error ϵ_0 and ϵ_0^* are independent to $\boldsymbol{x}_0$, we actually have $\sup_z |F_{\epsilon_0^*|X_0=\boldsymbol{x}_0}(z) - F_{\epsilon_0|X_0=\boldsymbol{x}_0}(z)| \xrightarrow{p} 0$ based on Lemma 5.1. Thus, we can write:

$$\sup_z |\mathbb{P}(Y_0^* - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0) \leq z) - \mathbb{P}(Y_0 - f(\boldsymbol{x}_0) \leq z)| \xrightarrow{p} 0, \tag{32}$$

where $\mathbb{P}(\cdot)$ represents $\mathbb{P}(\cdot|\boldsymbol{X}_0 = \boldsymbol{x}_0)$. We can start by considering the below expression:

$$\begin{aligned}
&\sup_z |\mathbb{P}(Y_0^* - f(\boldsymbol{x}_0) \leq z) - \mathbb{P}(Y_0 - f(\boldsymbol{x}_0) \leq z)| \\
&= \sup_z |\mathbb{P}(Y_0^* - f(\boldsymbol{x}_0) \leq z) - \mathbb{P}(Y_0^* - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0) \leq z) + \mathbb{P}(Y_0^* - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0) \leq z) - \mathbb{P}(Y_0 - f(\boldsymbol{x}_0) \leq z)| \\
&\leq \sup_z |\mathbb{P}(Y_0^* - f(\boldsymbol{x}_0) \leq z) - \mathbb{P}(Y_0^* - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0) \leq z)| + \sup_z |\mathbb{P}(Y_0^* - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0) \leq z) - \mathbb{P}(Y_0 - f(\boldsymbol{x}_0) \leq z)|.
\end{aligned} \tag{33}$$

For the first term on the r.h.s. of the above inequality, we have:

$$\begin{aligned}
&\sup_z |\mathbb{P}(Y_0^* - f(\boldsymbol{x}_0) \leq z) - \mathbb{P}(Y_0^* - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0) \leq z)| \\
&= \sup_z |\mathbb{P}(Y_0^* - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0) + \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0) - f(\boldsymbol{x}_0) \leq z) - \mathbb{P}(Y_0^* - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0) \leq z) \\
&= \sup_z |F_{\epsilon_0^*}(z + f(\boldsymbol{x}_0) - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0)) - F_{\epsilon_0^*}(z)| \\
&= \sup_z |F_{\epsilon_0^*}(z + f(\boldsymbol{x}_0) - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0)) - F_{\epsilon_0}(z + f(\boldsymbol{x}_0) - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0)) \\
&+ F_{\epsilon_0}(z + f(\boldsymbol{x}_0) - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0)) - F_{\epsilon_0}(z) + F_{\epsilon_0}(z) - F_{\epsilon_0^*}(z)| \\
&\leq \sup_z |F_{\epsilon_0^*}(z + f(\boldsymbol{x}_0) - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0)) - F_{\epsilon_0}(z + f(\boldsymbol{x}_0) - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0))| \\
&+ \sup_z |F_{\epsilon_0}(z + f(\boldsymbol{x}_0) - \overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0)) - F_{\epsilon_0}(z)| + \sup_z |F_{\epsilon_0}(z) - F_{\epsilon_0^*}(z)|.
\end{aligned} \tag{34}$$

We should notice that the first and third terms of the r.h.s. of Eq. (34) converge to 0 in probability. For the middle term, since $\overline{f}_{\mathrm{DNN}}(\boldsymbol{x}_0)$ converges to $f(\boldsymbol{x}_0)$ in probability and $\sup_z |p_{\epsilon_0}(z)|$ is assumed to be bounded as B2, this term also converges to 0 in probability by applying the Taylor expansion. Combining all the pieces, we have:

$$\sup_z \left| F_{Y_0^*|X_0=\boldsymbol{x}_0}(z) - F_{Y_0|X_0=\boldsymbol{x}_0}(z) \right| \xrightarrow{p} 0. \tag{35}$$

□

APPENDIX B: ADDITIONAL PLOTS

We present the sorted 3658 test points and corresponding conditional PIs in Fig. 3. The overall average coverage rate for all 3568 test points is 0.952 and the average length is 18.523.

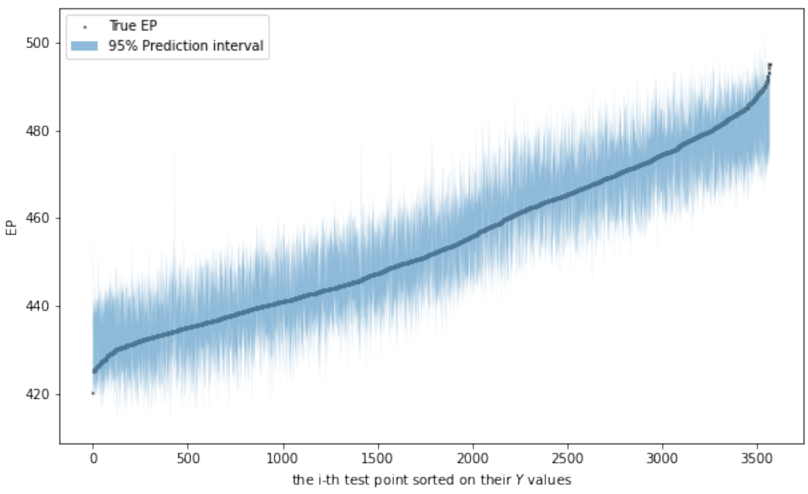


Fig. 3. All sorted EP values in increasing order and their corresponding conditional PIs. The overall average coverage rate for all 3568 test points is 0.952 and the average PI length is 18.523.

REFERENCES

[1] Bartlett, P. L., Harvey, N., Liaw, C., and Mehrabian, A. (2019). Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. The Journal of Machine Learning Research, 20(1):2285–2301.

[2] Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. Proceedings of the National Academy of Sciences, 116(32):15849–15854.

[3] Breiman, L. (1996). Bagging predictors. Machine learning, 24:123–140.

[4] Bühlmann, P. and Yu, B. (2002). Analyzing bagging. The annals of Statistics, 30(4):927–961.

[5] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (1989). Introduction to algorithms. MIT press.

[6] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4):303–314.

[7] Du, H., Li, Z., Liu, J., Liu, Y., and Sun, J. (2023). Divide-and-conquer dnn approach for the inverse point source problem using a few single frequency measurements. Inverse Problems, 39(11):115006.

[8] Evans, L. C. (1998). Partial differential equations. American Mathematical Society.

[9] Farrell, M. H., Liang, T., and Misra, S. (2021). Deep neural networks for estimation and inference. Econometrica, 89(1):181–213.

[10] Gelly, G., Gauvain, J.-L., Le, V. B., and Messaoudi, A. (2016). A divide-and-conquer approach for language identification based on recurrent neural networks. In Interspeech, pages 3231–3235. San Francisco, CA.

[11] Ha, K., Cho, S., and MacLachlan, D. (2005). Response models based on bagging neural networks. Journal of Interactive Marketing, 19(1):17–30.

[12] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359–366.

[13] Jordan, M. I. (2013). On statistics, computation and scalability. Bernoulli, 19(4):1378–1390.

[14] Khwaja, A., Naeem, M., Anpalagan, A., Venetsanopoulos, A., and Venkatesh, B. (2015). Improved short-term load forecasting using bagged neural networks. Electric Power Systems Research, 125:109–115.

[15] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[16] Kohler, M., Langer, S., and Reif, U. (2023). Estimation of a regression function on a manifold by fully connected deep neural networks. Journal of Statistical Planning and Inference, 222:160–181.

[17] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4):115–133.

[18] Pan, L. and Politis, D. N. (2016). Bootstrap prediction intervals for linear, nonlinear and nonparametric autoregressions. Journal of Statistical Planning and Inference, 177:1–27.

[19] Politis, D. N. (2015). Model-free prediction and regression: a transformation-based approach to inference. Springer.

[20] Politis, D. N. (2024). Scalable subsampling: computation, aggregation and inference. Biometrika, 111(1):347–354.

[21] Politis, D. N., Romano, J. P., and Wolf, M. (1999). Subsampling. Springer Science & Business Media.

[22] Stone, C. J. (1982). Optimal global rates of convergence for nonparametric regression. The annals of statistics, 10(4):1040–1053.

[23] Ting, D. (2021). Simple, optimal algorithms for random sampling without replacement. arXiv preprint arXiv:2104.05091.

[24] Wang, Y. and Politis, D. N. (2021). Model-free bootstrap and conformal prediction in regression: Conditionality, conjecture testing, and pertinent prediction intervals. arXiv preprint arXiv:2109.12156.

[25] Wu, K. and Politis, D. N. (2024). Bootstrap prediction inference of nonlinear autoregressive models. Journal of Time Series Analysis.

[26] Yang, Z., Yu, Y., You, C., Steinhardt, J., and Ma, Y. (2020). Rethinking bias-variance trade-off for generalization of neural networks. In International Conference on Machine Learning, pages 10767–10777. PMLR.

[27] Yarotsky, D. (2018). Optimal approximation of continuous functions by very deep relu networks. In Conference on learning theory, pages 639–649. PMLR.

[28] Yarotsky, D. and Zhevnerchuk, A. (2020). The phase diagram of approximation rates for deep neural networks. Advances in neural information processing systems, 33:13005–13015.

[29] Zhang, Y. and Politis, D. N. (2023). Bootstrap prediction intervals with asymptotic conditional validity and unconditional guarantees. Information and Inference: A Journal of the IMA, 12(1):157–209.

[30] Zou, T., Li, X., Liang, X., and Wang, H. (2021). On the subbagging estimation for massive data. arXiv preprint arXiv:2103.00631.