Deep Learning meets Nonparametric Regression: Are Weight-Decayed DNNs Locally Adaptive?

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study the theory of neural network (NN) from the lens of classical nonparametric regression problems with a focus on NN's ability to *adaptively* estimate functions with *heterogeneous smoothness* — a property of functions in Besov or Bounded Variation (BV) classes. Existing work on this problem requires tuning the NN architecture based on the function spaces and sample sizes. We consider a "Parallel NN" variant of deep ReLU networks and show that the standard weight decay is equivalent to promoting the ℓ_p-sparsity ($0 < p < 1$) of the coefficient vector of an end-to-end learned function bases, i.e., a dictionary. Using this equivalence, we further establish that by tuning only the weight decay, such Parallel NN achieves an estimation error arbitrarily close to the minimax rates for both the Besov and BV classes. Notably, it gets exponentially closer to minimax optimal as the NN gets deeper. Our research sheds new lights on why depth matters and how NNs are more powerful than kernel methods.

1 Introduction

Why do deep neural networks (DNNs) *work* better? They are universal function approximators [6], but so are splines and kernels. They learn data-driven representations, but so are the shallower and linear counterparts such as matrix factorization. There is surprisingly little theoretical understanding on why DNNs are superior to these classical alternatives.

In this paper, we study DNNs in nonparametric regression problems — a classical branch of statistical theory and methods with more than half a century of associated literature [23, 7, 43, 10, 21, 34, 30]. Nonparametric regression addresses the following fundamental problem:

- Let $y_i = f(x_i) + \text{Noise}$ for $i = 1, ..., n$. How can we estimate a function f using data points $(x_1, y_1), ..., (x_n, y_n)$ in conjunction with the knowledge that f belongs to a function class $\mathcal{F}$?

Function class $\mathcal{F}$ typically imposes only weak regularity assumptions such as smoothness, which makes nonparametric regression widely applicable to real-life applications under weak assumptions.

Local adaptivity. A subset of nonparametric regression techniques were shown to have the property of *local adaptivity* [22] in both theory and practice. These include wavelet smoothing [10], locally adaptive regression splines [22], trend filtering [37, 44] and adaptive local polynomials [2, 3]. We say a nonparametric regression technique is *locally adaptive* if it can cater to local differences in smoothness, hence allowing more accurate estimation of functions with varying smoothness and abrupt changes.

In light of such a distinction, it is natural to consider the following question.

Are NNs *locally adaptive*, i.e., optimal in learning functions with heterogeneous smoothness?

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

This is a timely question to ask, partly because the bulk of recent theory of NN leverages its asymptotic Reproducing Kernel Hilbert Space (RKHS) in the overparameterized regime [19, 5, 1]. RKHS-based approaches, e.g., kernel ridge regression with any fixed kernels are *suboptimal* in estimating functions with heterogeneous smoothness [9]. Therefore, existing deep learning theory based on RKHS does not satisfactorily explain the advantages of neural networks over kernel methods.

We build upon the recent work of Suzuki [36] and Parhi and Nowak [27] who provided encouraging first answers to the question above about the local adaptivity of NNs. Specifically, Parhi and Nowak [27, Theorem 8] showed that a two-layer *truncated power function* activated neural network with a non-standard regularization is equivalent to the *locally adaptive regression splines* (LARS) [22]. This connection implies that such non-standard NNs achieve the minimax rate for the (higher order) bounded variation (BV) classes. We provide a detailed discussion about this work in Section B. Suzuki [36] showed that multilayer ReLU DNNs can achieve minimax rate for the Besov class, but requires the width, depth and an artificially imposed sparsity-level of the DNN weights to be carefully calibrated according to parameters of the Besov class, thus is quite different from how DNNs are typically trained in practice.

In this paper, we aim at addressing the same *locally adaptivity* question for a more commonly used neural network with standard weight decayed training.

Parallel neural networks. We restrict our attention on a special network architecture called *parallel neural network* [16, 13] which learns an ensemble of subnetworks — each being a multilayer ReLU DNNs. Parallel NNs have been shown to be more well-behaved both theoretically [16, 48, 14, 13] and empirically [47, 41]. Moreover, the idea of parallel NNs was used in many successful NN architectures such as SqueezeNet, ResNext and Inception (see [13] and the references therein).

Weight decay. Weight decay is a common method in deep learning to reduce overfitting. Empirically, the regularizer is not necessarily explicit. Many tricks in deep learning, including early stopping [45], quantization [18], and dropout [42] have similar effect as weight decay. In this paper, we make no assumption on the training method thus there is no (implicit) regularizers apart from weight decay.

Summary of results. Our main contributions are:

1. We prove that the (standard) weight decay in training an L-layer *parallel* ReLU-activated neural network is equivalent to a sparse ℓ_p penalty term (where $p = 2/L$) on the linear coefficients of a learned representation.
2. We show that neural networks can approximate B-spline basis functions of any order without the need of choosing the order parameter manually. In other words, neural networks can adapt to functions of different order of smoothness, and even functions with different smoothness in different regions in their domain.
3. We show that the estimation error of weight decayed parallel ReLU neural network decreases polynomially with the number of samples up to a constant error for estimating functions with heterogeneous smoothness in the both BV and Besov classes, and the exponential term in the error rate is close to the minimax rate. Notably, the method requires tuning only the weight decay parameter.
4. We find that deeper models achieve closer to the optimal error rate. This result helps explain why deep neural networks can achieve better performance than shallow ones empirically.

The above results separate NNs with any linear methods such as kernel ridge regression. To the best of our knowledge, we are the first to demonstrate that standard techniques ("weight decay" and ReLU activation) suffice for DNNs in achieving the optimal rates for estimating BV and Besov functions.

2 Preliminary

2.1 Notation and problem setup.

We denote regular font letters as scalars, bold lower case letters as vectors and bold upper case letters as matrices. $a \lesssim b$ means $a \leq Cb$ for some constant C that does not depend on a or b, and $a \eqsim b$ denotes $a \lesssim b$ and $b \lesssim a$. See Table 1 for the full list of symbols used.

Table 1: Symbols used in this paper

symbol	Meaning		
$a/\boldsymbol{a}/\mathbf{A}$	scalars / vectors / matrices.	$[a,b]$	$\{x \in \mathbb{R} : a \le x \le b\}$
$B^{\alpha}_{p,q}$	Besov space.	$[n]$	$\{x \in \mathbb{N} : 1 \le x \le n\}$.
$\lvert \cdot \rvert_{B^{\alpha}_{p,q}}$	Besov quasi-norm .	$\lVert \cdot \rVert_F$	Frobenius norm.
$\lVert \cdot \rVert_{B^{\alpha}_{p,q}}$	Besov norm.	$\lVert \cdot \rVert_p$	ℓ_p-norm.
$M_m(\cdot)$	m^{th} order Cardinal B-spline bases.	d	Dimension of input.
$M_{m,k,\boldsymbol{s}}(\cdot)$	m^{th} order Cardinal B-spline basis function of resolution k at position $\boldsymbol{s}$.	M	# subnetworks in a parallel NN.
		L	# layers in a (parallel) NN.
		w	Width of a subnetwork.
$\sigma(\cdot)$	ReLU activation function.	n	# samples.
$\mathbf{W}^{(\ell)}_j, \boldsymbol{b}^{(\ell)}_j$	Weight and bias in the ℓ-th layer in the j-th subnetwork.	$\mathbb{R}, \mathbb{Z}, \mathbb{N}$	Set of real numbers, integers, and nonnegative integers.

Let f_0 be the target function to be estimated. The training dataset is $\mathcal{D}_n := \{(\boldsymbol{x}_i, y_i), y_i = f_0(\boldsymbol{x}_i) + \epsilon_i, i \in [n]\}$, where x_i are fixed and ϵ_i are zero-mean, independent Gaussian noises with variance σ^2. In the following discussion, we assume $\boldsymbol{x}_i \in [0,1]^d$, $f_0(x_i) \in [-1,1], \forall i$.

We will be comparing estimators under the mean square error (MSE), defined as

$$\mathrm{MSE}(\hat{f}) := \mathbb{E}_{\mathcal{D}_n} \frac{1}{n} \sum_{i=1}^{n} (\hat{f}(\boldsymbol{x}_i) - f_0(\boldsymbol{x}_i))^2.$$

The optimal worst-case MSE is described by $R(\mathcal{F}) := \min_{\hat{f}} \max_{f_0 \in \mathcal{F}} \mathrm{MSE}(\hat{f})$, we say that $\hat{f}$ is optimal if $\mathrm{MSE}(\hat{f}) \eqsim R(\mathcal{F})$. The empirical (square error) loss is defined as $\hat{L}(\hat{f}) := \frac{1}{n}\sum_{i=1}^{n}(\hat{f}(\boldsymbol{x}_i) - y_i)^2$. The corresponding population loss is $L(\hat{f}) := \mathbb{E}[\frac{1}{n}\sum_{i=1}^{n}(\hat{f}(\boldsymbol{x}_i) - y_i')^2 | \hat{f}]$ where y_i' are new data points. It is clear that $\mathbb{E}[L(\hat{f})] = \mathrm{MSE}[\hat{f}] + \sigma^2$.

2.2 Besov Spaces and Bound Variation Space

Besov space, denoted as $B^{\alpha}_{p,q}$, is a flexible function class parameterized by α, p, q whose definition is deferred to Section C.1. Here $\alpha \ge 0$ determines the smoothness of functions, $1 \le p \le \infty$ determines the averaging (quasi-)norm over locations, $1 \le q \le \infty$ determines the averaging (quasi-)norm over scale which plays a relatively minor role. Smaller p is more forgiving to inhomogeneity and loosely speaking, when the function domain is bounded, smaller p induces a larger function space. On the other hand, it is easy to see from definition that $B^{\alpha}_{p,q} \subset B^{\alpha}_{p,q'}$, if $q < q'$. Without loss of generalizability, in the following discussion we will only focus on $B^{\alpha}_{p,\infty}$.

The Besov space is closely connected to other function spaces including the Hölder space ($\mathcal{C}^{\alpha}$) and the Sobolev space (W^{α}_p). Specifically, if the domain of the functions is d-dimensional [36, 31],

- $\forall \alpha \in \mathbb{N}$, $B^{\alpha}_{p,1} \subset W^{\alpha}_p \subset B^{\alpha}_{p,\infty}$, and $B^{\alpha}_{2,2} = W^{\alpha}_2$.
- For $0 < \alpha < \infty$ and $\alpha \in \mathcal{N}, \mathcal{C}^{\alpha} = B^{\alpha}_{\infty,\infty}$.
- If $\alpha > d/p$, $B^{\alpha}_{p,q} \subset \mathcal{C}^0$.

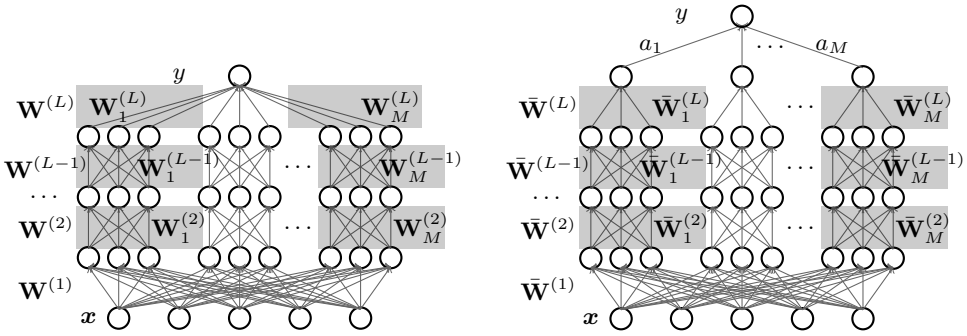
When $p = 1$, the Besov space allows higher inhomogeneity, and it is more general than the Sobolev or Hölder space.

Bounded variation (BV) space is a more interpretable class of functions with spatially heterogeneous smoothness [10]. It is defined through the total variation (TV) of a function. For $(m+1)$th differentiable function $f : [0,1] \to \mathbb{R}$, the mth order total variation is defined as

$$TV^{(m)}(f) := TV(f^{(m+1)}) = \int_{[0,1]} |f^{(m+1)}(x)| dx,$$

and the corresponding mth order Bounded Variation class

$$BV(m) := \{f : TV(f^{(m)}) < \infty\}.$$



$\min_{f_j} L(\sum_j f_j) + \lambda \sum_{\ell=1}^{L} \sum_{j=1}^{M} \|\mathbf{W}_j^{(\ell)}\|_F^2.$ $\quad$ $\min_{\{a_j, \bar{f}_j\}} L(\sum_j a_j \bar{f}_j)\ s.t. \sum_{j=1}^{M} |a_j|^{2/L} \leq P'.$

(a) Parallel NN with Weight Decay $\quad$ (b) Sparse Regression with Learned Representation

Figure 1: Parallel neural network and the equivalent sparse regression model we discovered.

The more general definition is given in Section C.2. Bounded variation class is tightly connected to Besov classes. Specifically [8]:

$$B_{1,1}^{m+1} \subset BV(m) \subset B_{1,\infty}^{m+1} \tag{1}$$

This allows the results derived for the Besov space to be easily applied to BV space.

Minimax MSE It is well known that minimax rate for Besov and 1D BV classes are $O(n^{-\frac{2\alpha}{2\alpha+d}})$ and $O(n^{-(2m+2)/(2m+3)})$ respectively . The minimax rate for *linear estimators* in 1D BV classes is known to be $O(n^{-(2m+1)/(2m+2)})$ [22, 10].

3 Main Results: Parallel ReLU DNNs

Consider a parallel neural network containing M multi layer perceptrons (MLP) with ReLU activation functions called *subnetworks*. Each subnetwork has width w and depth L. The input is fed to all the subnetworks, and the output of the parallel NN is the summation of the output of each subnetwork. The architecture of a parallel neural network is shown in Figure 1a. Let $\mathbf{W}_j^{(\ell)}$ and $\boldsymbol{b}_j^{(\ell)}$ denote the weight and bias in the ℓ-th layer in the j-th subnetwork respectively. Training this model with weight decay returns:

$$\underset{\{\mathbf{W}_j^{(\ell)}, \boldsymbol{b}_j^{(\ell)}\}}{\arg\min}\ \hat{L}(f) + \lambda \sum_{j=1}^{M} \sum_{\ell=1}^{L} \left\|\mathbf{W}_j^{(\ell)}\right\|_F^2, \tag{2}$$

where $f(x) = \sum_{j=1}^{M} f_k(x)$ denotes the parallel neural network, $f_j(\cdot)$ denotes the j-th subnetwork, and $\lambda > 0$ is a fixed scaling factor.

Theorem 1. *For any fixed $\alpha - d/p > 1, r > 0, L \geq 3$, given an L-layer parallel neural network satisfying*

- *The width of each subnetwork is fixed and large enough: $w \gtrsim d$. See Theorem 8 for the detail.*
- *The number of subnetworks is large enough: $M \gtrsim m^d n^{\frac{1-2/L}{2\alpha/d+1-2/(pL)}}$.*

With proper choice of the parameter of weight decay λ, the solution $\hat{f}$ parameterized by (2) satisfies

$$\mathrm{MSE}(\hat{f}) = \tilde{O}\big(n^{-\frac{2\alpha/d(1-2/L)}{2\alpha/d+1-2/(pL)}}\big) + Const. \tag{3}$$

where $\tilde{O}$ shows the scale up to a logarithmic factor, and the trailing constant term decreases exponentially with L.

We explain the proof idea in the next section,but defer the extended form of the theorem and the full proof to Section F. Before that, we comment on a few interesting aspects of the result.

Near optimal rates and the effect of depth. The first term in the MSE bound is the estimation error and the second term is (part of) the approximation error of this NN. Recall that the minimax rate of a Besov class is $O(n^{-\frac{2\alpha}{2\alpha+d}})$ thus as the depth parameter L increases it can get arbitrarily close to the minimax rate. The constant term would be a negligible if we choose $L \gtrsim \log n$. This result says that deeper parallel neural networks achieves lower error and gets closer to the statistical limit.

Overparameterization and sparsity. We also note that the result does not depend on M as long as M is large enough. This means that the neural network can be arbitrarily overparameterized while not overfitting. The underlying reason is *sparsity*. As it will become clearer in the proof sketch, weight decayed training of a parallel L-layer ReLU NNs is equivalent to a sparse regression problem with an ℓ_p penalty assigned to the coefficient vector of a learned dictionary. Here $p = 2/L$ which promotes even sparser solutions than an ℓ_1 penalty.

No architecture Tuning. For any fixed L, the required architecture of the model does not depend on the dataset or the target function (n, α) expect the number of subnetworks M, for which the only requirement is being large enough. As a result, one can design a model using a large guess on M, and achieve the claimed near-optimal error rate by only tuning the weight decay parameter.

Bounded Variation classes. Thanks to the Besov space embedding of the BV class (1), our theorem also implies the result for the BV class in $1D$.

Corollary 2. *If the target function is in bounded variation class $f_0 \in BV(m)$, For any fixed $L \geq 3$, for a neural network satisfying the requirements in Theorem 1 with $\mathrm{d} = 1$ and with proper choice of the parameter of weight decay λ, the NN $\hat{f}$ parameterized by (5) satisfies*

$$\mathrm{MSE}(\hat{f}) = \tilde{O}(n^{-\frac{(2m+2)(1-2/L)}{2m+3-2/L}}) + Const.$$

where $\tilde{O}$ shows the scale up to a logarithmic factor, and the trailing constant term decreases exponentially with L.

It is known that any linear estimators such as kernel smoothing and smoothing splines cannot have an error lower than $O(n^{-(2m+1)/(2m+2)})$ for $BV(m)$ [10]. This partly explains the advantage of DNNs over kernels.

Representation learning and adaptivity. The results also shed a light on the role of representation learning in DNN's ability to adapt. Specifically, different from the two-layer NN in [27], which achieves the minimax rate of $BV(m)$ by choosing appropriate activation functions using each m, each subnetwork of a parallel NN can learn to approximate the spline basis of an arbitrary order, which means that if we choose L to be sufficiently large, such Parallel NN with optimally tuned λ is simultaneously near optimal for $m = 1, 2, 3, \ldots$. In fact, even if different regions of the space has different *orders* of smoothness, the NN will still be able to learn appropriate basis functions in each local region. To the best of our knowledge, this is a property that none of the classical nonparametric regression methods possess.

Synthesis vs Analysis methods. Our result could also inspire new ideas in estimator design. There are two families of methods in non-parametric estimation. One called *synthesis* framework which focuses on constructing appropriate basis functions to encode the contemplated structures and regress the data to such basis, e.g., wavelets [10]. The other is called *analysis* framework which uses analysis regularization on the data directly (see, e.g., RKHS methods [34] or trend filtering [37]). It appears to us that parallel NN is doing both simultaneously. It has a parametric family capable to synthesizing an $O(n)$ subset of an exponentially large family of basis, then *implicitly* use sparsity-inducing analysis regularization to select the relevant basis functions. In this way the estimator does not actually have to explicitly represent that exponentially large set of basis functions, thus computationally more efficient.

4 Proof Overview

We start by first proving that a parallel neural network trained with weight decay is equivalent to an ℓ_p-sparse regression problem with representation learning (Section 4.1); which helps decompose its

MSE into an estimation error and approxmation error. Then we bound the two terms in Section 4.2 and Section 4.3 respectively.

4.1 Equivalence to ℓ_p sparse regression with a learned feature representation

It is widely known that ReLU function is 1-homogeneous:

$$\sigma(ax) = a\sigma(x), \forall a \geq 0, x \in \mathbb{R}.$$

In any consecutive two layers in a neural network (or a subnetwork), one can multiply the weight and bias in one layer with a positive constant, and divide the weight in another layer with the same constant. The neural network after such transformation is equivalent to the original one:

$$\mathbf{W}^{(2)}\sigma(\mathbf{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)} = \frac{1}{c}\mathbf{W}^{(2)}\sigma(c\mathbf{W}^{(1)}\boldsymbol{x} + c\boldsymbol{b}^{(1)}), \quad \forall c > 0, \boldsymbol{x}. \tag{4}$$

This property allows us to reformulate (2) to an ℓ_p sparsity constraint problem:

Proposition 3. *Fix the input dataset $\mathcal{D}_n$ and a constant $c_1 > 0$. For every λ, there exists $P' > 0$ such that (2) is equivalent to the following problem:*

$$\begin{aligned} \underset{\{\bar{\mathbf{W}}_j^{(\ell)}, \bar{\boldsymbol{b}}_j^{(\ell)}, a_j\}}{\arg\min} \quad & \hat{L}\Big(\sum_{j=1}^{M} a_j \bar{f}_j\Big) = \frac{1}{n}\sum_i (y_i - \bar{f}_{1:M}(\boldsymbol{x}_i)^T \boldsymbol{a})^2 \\ s.t. \ & \|\bar{\mathbf{W}}_j^{(1)}\|_F \leq c_1\sqrt{d}, \forall j \in [M], \\ & \|\bar{\mathbf{W}}_j^{(\ell)}\|_F \leq c_1\sqrt{w}, \forall j \in [M], 2 \leq \ell \leq L, \qquad \|\{a_j\}\|_{2/L}^{2/L} \leq P' \end{aligned} \tag{5}$$

where $\bar{f}_j(\cdot)$ is a subnetwork with parameters $\bar{\mathbf{W}}_j^{(\ell)}, \bar{\boldsymbol{b}}_j^{(\ell)}$.

This equivalent model is demonstated in Figure 1b. The proof can be found in Section D.1. The constraint $\|\bar{\mathbf{W}}_j^{(1)}\|_F \lesssim \sqrt{d}, \|\bar{\mathbf{W}}_j^{(\ell)}\|_F \lesssim \sqrt{w}, \forall \ell > 1$ is typical in deep learning for better numerical stability. The equivalent model in Proposition 3 is also a parallel neural network, but it appends one layer with parameters $\{a_k\}$ at the end of the neural network and the constraint on the Frobenius norm is converted to the $2/L$ norm on the factors $\{a_k\}$. Since $L \gg 2$ in a typical application, $2/L \ll 1$ and this constraint can enforce a sparser model than that in Section B.

There are two useful implications of Proposition 3. First, it gives an intuitive explanation on how a weight decayed Parallel NN works. Specifically, it can be viewed as a sparse linear regression with representation learning. Second, the conversion into the constrained form allows us to adapt generic statistical learning machinery (a self-bounding argument) from Suzuki [36, Proposition 4] for studying this constrained ERM problem.

The adaptation is nontrivial because (1) our regression problem has a *fixed design* (so data points are not iid); (2) there is an *unconstrained* subspace with no bounded metric entropy. Specifically, our Proposition 14 shows that the MSE of the regression problem can be bounded by

$$\mathrm{MSE}(\hat{f}) = O\Bigg(\underbrace{\inf_{f \in \mathcal{F}} \mathrm{MSE}(f)}_{\text{approximation error}} + \underbrace{\frac{\log \mathcal{N}(\mathcal{F}_\|, \delta, \|\cdot\|_\infty) + d(\mathcal{F}_\perp)}{n} + \delta}_{\text{estimation error}}\Bigg)$$

in which $\mathcal{F}$ decomposes into $\mathcal{F}_\| \times \mathcal{F}_\perp$, where $\mathcal{F}_\perp$ is an unconstrained subspace with finite dimension, and $\mathcal{F}_\|$ is a compact set in the orthogonal complement with a δ-covering number of $\mathcal{N}(\mathcal{F}_\|, \delta, \|\cdot\|_\infty)$ in $\|\cdot\|_\infty$-norm. This decomposes MSE into an approximation errorand an estimation error. The novel analysis of these two represents the major technical contribution of this paper.

4.2 Estimation Error Analysis

The decomposition above reveals that to bound the estimation error, it suffices to compute the covering number of the constraint set in the sup-norm of the function it represents.

Previous results that bound the covering number of neural networks [46, 36] depends on the width of the neural networks explicitly, which cannot be applied when analysing a potentially infinitely wide neural network. In this section, we leverage the ℓ_p-norm bounded coefficients to avoid the dependence in M in the covering number bound.

Theorem 4. *The covering number of the model defined in (5) apart from the bias in the last layer satisfies*

$$\log \mathcal{N}(\mathcal{F}, \delta) \lesssim w^{2+2/(1-2/L)} L^2 \sqrt{d} P'^{\frac{1}{1-2/L}} \delta^{-\frac{2/L}{1-2/L}} \log(wP'/\delta). \tag{6}$$

The proof can be found in Section D.2. It requires the following lemma:

Lemma 5. $\log \mathcal{N}(\mathcal{G}, \delta) \lesssim k \log(1/\delta)$ *for some finite* c_3*, and for any* $g \in \mathcal{G}, |a| \leq 1$*, we have* $ag \in \mathcal{G}$*. The covering number of* $\mathcal{F} = \left\{ \sum_{i=1}^{M} a_i g_i \middle| g_i \in \mathcal{G}, \|a\|_p^p \leq P, 0 < p < 1 \right\}$ *for any* $P > 0$ *satisfies*

$$\log \mathcal{N}(\mathcal{F}, \epsilon) \lesssim k P^{\frac{1}{1-p}} (\delta/c_3)^{-\frac{p}{1-p}} \log(c_3 P/\delta)$$

up to a double logarithmic factor.

See Section D.3 for the proof of Lemma 5. The covering number in Theorem 4 does not depend on the number of subnetworks M. In other words, it provides a bound of estimation error for an arbitrarily wide parallel neural network as long as the total Frobenius norm is bounded.

4.3 Approximation Error Analysis

The approximation error analysis involves two steps. In Section 4.3.1, we analyse how a subnetwork can approximate a B-spline basis. Then in Section 4.3.2 we show that a sparse linear combination of B-spline bases approximates Besov functions. Both add up to the total error in approximating Besov functions with a parallel neural network (Theorem 8).

4.3.1 Approximation Error of B-spline Basis Function

As is shown in Section C.1, functions in Besov space can be alternatively represented in a sequence space via the coefficients of a cardinal B-spline basis. In this section we study the approximation ability of ReLU neural networks to B-spline basis function.

Proposition 6. *There exists a parallel neural network that has the structure and satisfy the constraint in Proposition 3 for* d*-dimensional input and one output, containing* $M = O(m^d)$ *subnetworks, each of which has width* $w = O(d)$ *and depth* $L = O(\log(c(m, d)/\epsilon))$ *for some constant* w, c *that depends only on* m *and* d*, denoted as* $\tilde{M}_m(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^d$*, such that*

- $|\tilde{M}_{m,k,\boldsymbol{s}}(\boldsymbol{x}) - M_{m,k,\boldsymbol{s}}(\boldsymbol{x})| \leq \epsilon$*, if* $0 \leq 2^k(x_i - s_i) \leq m + 1, \forall i \in [d]$*,*
- $\tilde{M}_{m,k,\boldsymbol{s}}(\boldsymbol{x}) = 0$*, otherwise.*
- *The weights in the last layer satisfy* $\|a\|_{2/L}^{2/L} \lesssim 2^k m^d e^{2md/L}$*.*

The proof can be found in Section E.1. Note that the product of the coefficients among all the layers are proportional to 2^k, instead of 2^{km} when approximating truncated power basis functions. This is because the transformation from M_m to $M_{m,k,\boldsymbol{s}}$ only scales the domain of the function by 2^k, while the codomain of the function is not changed. To apply the transformation to the neural network, one only need to scale weights in the first layer by 2^k, which is equivalent to scaling the weights in each layer bt $2^{k/L}$ and adjusting the bias according.

4.3.2 Approximation Error in Besov Space

With the results given in Section 4.3.1, we can estimate the approximation error of parallel ReLU neural networks to functions in Besov space.

Proposition 7. *Let* $\alpha - d/p > 1, r > 0$*. Let* $M_{m,k,\boldsymbol{s}}$ *be the B-spline of order* m *with scale* 2^{-k} *in each dimension and position* $\boldsymbol{s} \in \mathbb{R}^d$*. For any function in Besov space* $f_0 \in B_{p,q}^{\alpha}$ *and any positive integer* $\bar{M}$*, there is an* $\bar{M}$*-sparse approximation using B-spline basis of order* m *satisfying* $0 < \alpha < \min(m, m - 1 + 1/p)$*:* $\check{f}_{\bar{M}} = \sum_{i=1}^{\bar{M}} a_{k_i,\boldsymbol{s}_i} M_{m,k_i,\boldsymbol{s}_i}$ *for any positive integer* $\bar{M}$ *such that the approximation error is bounded as* $\|\check{f}_{\bar{M}} - f_0\|_r \lesssim \bar{M}^{-\alpha/d} \|f_0\|_{B_{p,q}^{\alpha}}$*, and the coefficients satisfy*

$$\|\{2^{k_i} a_{k_i,\boldsymbol{s}_i}\}_{k_i,\boldsymbol{s}_i}\|_p \lesssim \|f_0\|_{B_{p,q}^{\alpha}}.$$

The proof can be found in Section E.2.

Remark 1. *The requirement in Proposition 7: $\alpha - d/p > 1$ is stronger than the condition typically found in approximation theorem $\alpha - d/p \geq 0$ [11], so-called "Boundary of continuity", or the condition in Suzuki [36] $\alpha > d(1/p - 1/r)_+$. This is because although the functions in $B^{\alpha}_{p,q}$ when $0 \leq \alpha - d/p < 1$ can be approximated by B-spline basis, the sum of weighted coefficients may not converge. One simple example is the step function $f_{step}(x) = \mathbf{1}(x \geq 0.5), f_{step} \in B^1_{1,\infty}$. Although it can be decomposed using first order B-spline basis as in (10), the summation of the coefficients is infinite. Actually one only needs a ReLU neural network with one hidden layer and two neurons to approximate this function to arbitrary precision, but the weight need to go to infinity.*

Theorem 8. *Under the same condition as Proposition 7, for any positive integer $\bar{M}$, any function in Besov space $f_0 \in B^{\alpha}_{p,q}$ can be approximated by a parallel neural network with no less than $O(m^d \bar{M})$ number of subnetworks satisfying:*

1. *Each subnetwork has width $w = O(d)$ and depth L.*
2. *The weights in each layer satisfy $\|\bar{\mathbf{W}}^{(\ell)}_k\|_F \leq O(\sqrt{w})$ except the first layer $\|\bar{\mathbf{W}}^{(1)}_k\|_F \leq O(\sqrt{d})$,*
3. *The scaling factors have bounded $2/L$-norm: $\|\{a_j\}\|^{2/L}_{2/L} \lesssim m^d e^{2md/L} \bar{M}^{1-2/(pL)}$.*
4. *The approximation error is bounded by*
$$\|\tilde{f} - f_0\|_r \leq (c_4 \bar{M}^{-\alpha/d} + c_5 e^{-c_6 L})\|f\|_{B^{\alpha}_{p,q}}$$
where c_4, c_5, c_6 are constants that depend only on m, d and p.

Here $\bar{M}$ is the number of "active" subnetworks, which is not to be confused with the number of subnetworks at initialization. The proof can be found in Section E.3.

Using the estimation error in Theorem 4 and approximation error in Theorem 8, by choosing $\bar{M}$ to minimax the total error, we can conclude the sample complexity of parallel neural networks using weight decay, which is the main result (Theorem 1) of this paper. See Section F for the detail.

5 Experiment

We empirically compare a parallel neural network (PNN) and a vanilla ReLU neural network (NN) with smoothing spline, trend filtering (TF) [37], and wavelet denoising. Trend filtering can be viewed as a more efficient discrete spline version of locally adaptive regression spline and enjoys the same optimal rates for the BV classes. Wavelet denoising is also known to be minimax-optimal for the BV classes. The results are shown in Figure 2. We use two target functions: a Doppler function whose frequency is decreasing(Figure 2(a)-(c)), and a combination of piecewise linear function and piecewise cubic function, or "vary" function (Figure 2(d)-(f)). We repeat each experiment 10 times and take the average. The shallow area in Figure 2(b)(e) shows 95% confidence interval by inverting the Wald's test. The degree of freedom is computed based on Tibshirani [38].

As can be shown in the figure, both TF and wavelet denoising can adapt to the different levels of smoothness in the target function, while smoothing splines tend to be oversmoothed where the target function is less smooth (the left side in (a)(d)). The prediction of PNN is similar to TF and wavelet denoising and shows local adaptivity. Besides, the MSE of PNN almost follows the same trend as TF and wavelet denoising which is consistent with our theoretical understanding that the error rate of neural network is closer to locally adaptive methods. Notably PNN, TF and wavelet denoising achieve lower error at a much smaller degree-of-freedom than smoothing splines.

In a vanilla NN, weight decay is equivalent to ℓ_1 regularizer in any two successive layers, but to the best of our knowledge it does not lead to sparse representation learning unless some specific sparse structure is enforced. While our theory does not apply to vanilla neural networks, the results seem to suggest the NN behaves similar to smoothing spline and is *not* locally adaptive.

There are some mild drops in the best MSE one can achieve with NN vs TF in both examples. We are surprised that the drop is small because NN needs to learn the basis functions that TF essentially hard-coded. The additional price to pay for using a more adaptive and more flexible representation learning method seems not high at all.

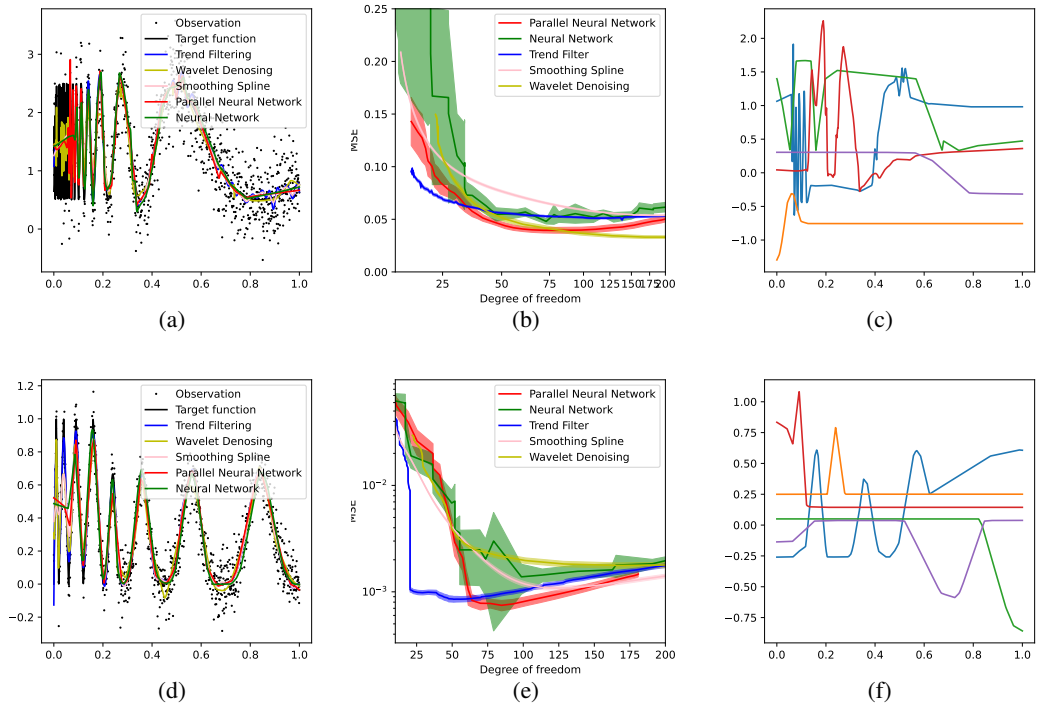


Figure 2: Numerical experiment results of the Doppler function (a)-(c), and "vary" function (d-f). (a)(d): Estimations when the degree of freedom is 30(a) or 50(d). (b)(e) Mean squared error in predicting the true function. Note that the horizontal axis in (b) is not linear. (c)(f): Output of each "active" subnetwork.

In Figure 2(c)(f), we give the output of each "active" subnetwork, i.e. the subnetworks whose output is not a constant. Notice that the number of active subnetworks is much smaller than the initialization. This is because weight decay induces ℓ_p sparsity and the weight in most of the subnetworks reduces towards 0 after training. More details are shown in Section G.

6 Conclusion and discussion

In this paper, we show that a deep parallel neural network can be locally adaptive by tuning only the weight decay parameter. This confirms that neural networks can be nearly optimal in learning functions with heterogeneous smoothness which separates them from kernel methods. We prove that training an L layer parallel neural network with weight decay is equivalent to an $\ell_{2/L}$-penalized regression model with representation learning. Since in typical application $L \gg 2$, this shows that weight decay promotes a sparse linear combination of the learned bases. Using this method, we proved that a parallel neural network can achieve close to the minimax rate in the Besov space and bounded variation (BV) space. Our result reveals that one do not need to specify the smoothness parameter α (or m). Neural networks can adapt to different degree of smoothness, or choose different parameters for different regions of the domain of the target function. This is a new type of adaptivity not possessed by traditional adaptive nonparametric regression methods like locally adaptive regression spline or trend filtering.

On the other hand, as the depth of neural network L increases, $2/L$ tends to 0 and the error rate moves closer to the minimax rate of Besov and BV space. This indicates that when the sample size is large enough, deeper models have smaller error than shallower models, and helps explain why empirically deep neural networks has better performance than shallow neural networks.

It is not known to us whether our result can be generalized to other types of neural networks, eg. vanilla neural networks and convolution neural networks. Future work is needed to answer this question.

References

[1] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8141–8150, 2019.

[2] Dheeraj Baby and Yu-Xiang Wang. Online forecasting of total-variation-bounded sequences. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[3] Dheeraj Baby and Yu-Xiang Wang. Adaptive online estimation of piecewise polynomial trends. *Neural Information Processing Systems (NeurIPS)*, 2020.

[4] Andrew R Barron. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14(1):115–133, 1994.

[5] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. In *International Conference on Machine Learning*, pages 541–549. PMLR, 2018.

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

[7] Carl De Boor, Carl De Boor, Etats-Unis Mathématicien, Carl De Boor, and Carl De Boor. *A practical guide to splines*, volume 27. Springer-Verlag New York, 1978.

[8] Ronald A DeVore and George G Lorentz. *Constructive approximation*, volume 303. Springer Science & Business Media, 1993.

[9] David L Donoho, Richard C Liu, and Brenda MacGibbon. Minimax risk over hyperrectangles, and implications. *The Annals of Statistics*, pages 1416–1437, 1990.

[10] David L Donoho, Iain M Johnstone, et al. Minimax estimation via wavelet shrinkage. *The annals of Statistics*, 26(3):879–921, 1998.

[11] Dinh Dũng. Optimal adaptive sampling recovery. *Advances in Computational Mathematics*, 34(1):1–41, 2011.

[12] Tolga Ergen and Mert Pilanci. Convex geometry and duality of over-parameterized neural networks. *Journal of machine learning research*, 2021.

[13] Tolga Ergen and Mert Pilanci. Path regularization: A convexity and sparsity inducing regularization for parallel relu networks. *arXiv preprint arXiv:2110.09548*, 2021.

[14] Tolga Ergen and Mert Pilanci. Revealing the structure of deep neural networks via convex duality. In *International Conference on Machine Learning*, pages 3004–3014. PMLR, 2021.

[15] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[16] Benjamin D Haeffele and René Vidal. Global optimality in neural network training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7331–7339, 2017.

[17] Daniel Hsu, Sham M Kakade, and Tong Zhang. An analysis of random design linear regression. *arXiv preprint arXiv:1106.2363*, 2011.

[18] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Proceedings of the 30th international conference on neural information processing systems*, pages 4114–4122. Citeseer, 2016.

[19] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8580–8589, 2018.

[20] Seung-Jean Kim, Kwangmoo Koh, Stephen Boyd, and Dimitry Gorinevsky. \ell_1 trend filtering. *SIAM review*, 51(2):339–360, 2009.

[21] Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[22] Enno Mammen and Sara van de Geer. Locally adaptive regression splines. *The Annals of Statistics*, 25(1):387–413, 1997.

[23] Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1): 141–142, 1964.

[24] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[25] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[26] Greg Ongie, Rebecca Willett, Daniel Soudry, and Nathan Srebro. A function space view of bounded norm infinite width relu nets: The multivariate case. In *International Conference on Learning Representations*, 2019.

[27] Rahul Parhi and Robert D Nowak. Banach space representer theorems for neural networks and ridge splines. *J. Mach. Learn. Res.*, 22:43–1, 2021.

[28] Rahul Parhi and Robert D Nowak. What kinds of functions do deep neural networks learn? insights from variational spline theory. *arXiv preprint arXiv:2105.03361*, 2021.

[29] Rahul Parhi and Robert D Nowak. Near-minimax optimal estimation with shallow relu neural networks. *arXiv preprint arXiv:2109.08844*, 2021.

[30] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[31] Veeranjaneyulu Sadhanala, Yu-Xiang Wang, Addison J Hu, and Ryan J Tibshirani. Multivariate trend filtering for lattice data. *arXiv preprint arXiv:2112.14758*, 2021.

[32] Pedro Savarese, Itay Evron, Daniel Soudry, and Nathan Srebro. How do infinite width bounded norm networks look in function space? In *Conference on Learning Theory*, pages 2667–2690. PMLR, 2019.

[33] Johannes Schmidt-Hieber. Nonparametric regression using deep neural networks with relu activation function. *The Annals of Statistics*, 48(4):1875–1897, 2020.

[34] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[35] Nathan Srebro, Jason DM Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *NIPS*, volume 17, pages 1329–1336. Citeseer, 2004.

[36] Taiji Suzuki. Adaptivity of deep relu network for learning in besov and mixed smooth besov spaces: optimal rate and curse of dimensionality. *arXiv preprint arXiv:1810.08033*, 2018.

[37] Ryan J Tibshirani. Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics*, 42(1):285–323, 2014.

[38] Ryan J Tibshirani. Degrees of freedom and model search. *Statistica Sinica*, pages 1265–1296, 2015.

[39] Ryan J Tibshirani. Equivalences between sparse models and neural networks. 2021. URL `http://www.stat.cmu.edu/~ryantibs/papers/sparsitynn.pdf`.

[40] Ryan J Tibshirani. Personal communication, Jan. 24, 2022.

[41] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29:550–558, 2016.

[42] Stefan Wager, Sida Wang, and Percy Liang. Dropout training as adaptive regularization. *arXiv preprint arXiv:1307.1493*, 2013.

[43] Grace Wahba. *Spline models for observational data*, volume 59. Siam, 1990.

[44] Yu-Xiang Wang, Alex Smola, and Ryan Tibshirani. The falling factorial basis and its statistical applications. In *International Conference on Machine Learning*, pages 730–738. PMLR, 2014.

[45] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[46] Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2017.07.002. URL `https://www.sciencedirect.com/science/article/pii/S0893608017301545`.

[47] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[48] Hongyang Zhang, Junru Shao, and Ruslan Salakhutdinov. Deep neural networks with multi-branch architectures are intrinsically less non-convex. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1099–1109. PMLR, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This work does not have potential negative societal impacts to the best of our knowledge.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] All the proofs are in the appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] Although we did not publish the code, the experiments are very simple and can be easily reproduced using the information given in Section G.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section G.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We provided the confidence interval in Figure 2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] The experiments are light-weighted and can be finished in an ordinary PC in a reasonable time.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A] We only used synthetic dataset.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]