

NEW APPROXIMATION RESULTS AND OPTIMAL ESTIMATION FOR FULLY CONNECTED DEEP NEURAL NETWORKS

Zhaoji Tang

Department of Economics, University College London *

December 11, 2025

Abstract

Farrell et al. (2021) establish non-asymptotic high-probability bounds for general deep feedforward neural network (with rectified linear unit activation function) estimators, with Farrell et al. (2021, Theorem 1) achieving a suboptimal convergence rate for fully connected feedforward networks. The authors suggest that improved approximation of fully connected networks could yield sharper versions of Farrell et al. (2021, Theorem 1) without altering the theoretical framework. By deriving approximation bounds specifically for a narrower fully connected deep neural network, this note demonstrates that Farrell et al. (2021, Theorem 1) can be improved to achieve an optimal rate (up to a logarithmic factor). Furthermore, this note briefly shows that deep neural network estimators can mitigate the curse of dimensionality for functions with compositional structure and functions defined on manifolds.

Keywords: Deep Neural Networks, Approximation, Rectified Linear Unit, Optimal Estimation.

*Email: zhaoji.tang.23@ucl.ac.uk. I thank Dennis Kristensen and Andrei Zeleneev for their thoughtful suggestions and patient supervision. All errors are my own.

1 Introduction

Focusing on the rectified linear unit (ReLU) activation function, Farrell et al. (2021, Theorem 2) establish non-asymptotic high probability bounds for general deep feedforward neural network estimators. When restricting to fully connected neural networks (multilayer perceptrons), Farrell et al. (2021, Theorem 1) achieve a suboptimal convergence rate in the sense of Stone (1982).¹

This work demonstrates that the convergence rate for fully connected networks is in fact optimal (up to a logarithmic factor). This is achieved by solely modifying the approximation results in the proof of Farrell et al. (2021, Theorem 1) while leaving the other components of the proof unchanged. The proof of Farrell et al. (2021, Theorem 1) unnecessarily increases the complexity of the neural network class by bounding the number of weights in terms of the squared width. By maintaining their core strategy while constructing a narrower approximation network with square root width, we resolve this issue. Using our approximation idea, other works that do not achieve the optimal convergence rate can also attain optimality (e.g., Feng et al. (2023), Brown (2024), Zhang and Bradic (2024), Colangelo and Lee (2025), Zhang et al. (2024), Chronopoulos et al. (2023), Jiao et al. (2025)).

Our proof adapts the approach of Yarotsky (2017, Theorem 1) and translates it from a wide neural network to a narrower (fully connected) architecture. This narrow architecture is deliberately constructed to resolve the squared width problem, representing a novel contribution to the literature. The key to this construction is a property of ReLU activation functions identified in Schmidt-Hieber (2020)—their ability to preserve information flow unchanged. Following Liu et al. (2022), we convert the results to a fully connected network without substantially altering the depth or width.

Unlike the other works that derive convergence rates specifically for fully connected neural networks (e.g., Liu et al. (2022), Schmidt-Hieber (2020), Shen et al. (2021)), the approach in Farrell et al. (2021) starts from bounds for general neural network classes and then applies fully connected network approximation results to these general bounds. As noted in Farrell et al. (2021), this strategy offers several advantages: it establishes high-probability bounds,

¹While they achieve optimal convergence rates under specific conditions (see Farrell et al. (2021, Corollary 1)), this result applies only to a restricted neural network architecture.

accommodates general loss functions, and permits unbounded weights.

Regarding convergence rates, our results align with most of the existing works that contain an extra logarithmic term (e.g., Bauer and Kohler (2019) and Schmidt-Hieber (2020)), except for Liu et al. (2022).² Their work demonstrates that ReLU feedforward networks can achieve the optimal rate without logarithmic factor terms. While the proof technique from Farrell et al. (2021) inherently retains these logarithmic terms, this approach provides a trade-off: although yielding a slightly slower convergence rate, it avoids the additional assumptions about independent variables' density function that underlie the log-free results.

An important topic in the theoretical deep learning literature, which is not addressed in Farrell et al. (2021), is why deep neural networks often outperform shallow neural networks in practice. Since deep networks are typically employed in high-dimensional settings, a well-known explanation is that, under certain conditions, they can overcome the curse of dimensionality, whereas shallow networks cannot.

The literature in this area can be broadly divided into two strands. First, several works show that deep networks can circumvent the curse of dimensionality for functions with a compositional structure (e.g., Poggio et al. (2017), Schmidt-Hieber (2020)). Second, other studies demonstrate that for functions defined on a low-dimensional manifold, the curse of dimensionality can also be mitigated (e.g., Shaham et al. (2018), Kohler et al. (2022)). For detailed discussions, we refer the reader to the cited works.

Petersen and Zech (2024, Chapter 8) provides concise approximation results for functions with a compositional structure and for functions on manifolds, with proofs based on Yarotsky (2017, Theorem 1). An advantage of Farrell et al. (2021, Theorem 2) is that these approximation results can be applied directly to derive convergence rates. However, achieving optimal rates for fully connected networks requires modifying the approximation using our results. For this reason, we briefly present our findings on mitigating the curse of dimensionality in this note.

The rest of the paper is organized as follows. Section 2 introduces the settings and results in Farrell et al. (2021) and then presents our theoretical results. Section 3 provides results for mitigating the curse of dimensionality. Section 4 concludes, and the proofs are provided in the

²Bauer and Kohler (2019) focused on smooth activation functions rather than ReLU, whereas the results in Schmidt-Hieber (2020) depend on network sparsity.

Appendix.

We use the following norms. Let $\mathbf{X} \in \mathbb{R}^d$ be a random vector with sample realizations \mathbf{x}_i , and let \mathbf{x} denote a generic realization. For a function $g(\mathbf{x})$, define $\|g\|_\infty := \sup_{\mathbf{x}} |g(\mathbf{x})|$, $\|g\|_{L_2(X)} := (\mathbb{E}[g(\mathbf{X})^2])^{1/2}$, and $\|g\|_n := (\mathbb{E}_n[g(\mathbf{x}_i)^2])^{1/2}$, where $\mathbb{E}_n[\cdot]$ denotes the sample mean.

2 Deep neural networks

This section proceeds in two steps. First, we recap the framework and the approach of Farrell et al. (2021). Second, we present our alternative method, clarifying both the distinctions between these approaches and why Farrell et al. (2021, Theorem 1) do not achieve the optimal convergence rate.

2.1 Settings and results in Farrell et al. (2021)

For random covariates $\mathbf{X} \in \mathbb{R}^d$ and a scalar outcome Y , let $\mathbf{Z} = (Y, \mathbf{X}')' \in \mathbb{R}^{d+1}$, with a realization denoted as $\mathbf{z} = (y, \mathbf{x}')'$.

Following Farrell et al. (2021), for a loss function $\ell(f, \mathbf{z})$ and a function space \mathcal{F} , the function of interest f_* is defined as:

$$f_* := \arg \min_{f \in \mathcal{F}} \mathbb{E}[\ell(f, \mathbf{Z})].$$

We wish to estimate f_* using ReLU feedforward neural networks. To this end, define the ReLU activation function as:

$$\sigma_R(x) := \max(x, 0).$$

For covariates $\mathbf{X} \in \mathbb{R}^d$, a general feedforward neural network consists of d input units, a number of hidden computation units, and one output unit. Each hidden unit computes a linear combination of its inputs and applies an activation function. The output unit is also a computation unit, but it does not apply an activation function. A key feature of feedforward neural networks is that the hidden units are organized into ordered groups, called layers. Units in a given layer receive input only from the preceding layers and send output only to the subsequent

layers, but not vice versa.³

We denote the class of general feedforward neural networks by \mathcal{F}_{DNN} , and the loss function satisfies the following assumption.

Assumption 1. *For constants $c_1, c_2, C_\ell \in \mathbb{R}_+$, and for any $f \in \mathcal{F} \cup \mathcal{F}_{\text{DNN}}$ and $g \in \mathcal{F} \cup \mathcal{F}_{\text{DNN}}$, assume the loss function satisfies*

$$|\ell(f, \mathbf{z}) - \ell(g, \mathbf{z})| \leq C_l |f(\mathbf{x}) - g(\mathbf{x})|.$$

For $f_* \in \mathcal{F}$ and for any $h \in \mathcal{F}_{\text{DNN}}$, assume the curvature condition

$$c_1 \mathbb{E}[(h - f_*)^2] \leq \mathbb{E}[\ell(h, \mathbf{Z})] - \mathbb{E}[\ell(f_*, \mathbf{Z})] \leq c_2 \mathbb{E}[(h - f_*)^2].$$

The estimator of the general feedforward neural network is defined as:

$$\hat{f}_{\text{DNN}} \in \arg \min_{\substack{f \in \mathcal{F}_{\text{DNN}} \\ \|f\|_\infty \leq 2M}} \sum_{i=1}^n \ell(f, \mathbf{z}_i). \quad (1)$$

We will focus on a special type of feedforward neural network, known as a fully connected neural network (MLP). A fully connected feedforward neural network can be represented as follows,

$$f_{\text{MLP}}(\mathbf{x}) = \mathbf{W}_L \sigma (\cdots \sigma (\mathbf{W}_3 \sigma (\mathbf{W}_2 \sigma (\mathbf{W}_1 \sigma (\mathbf{W}_0 \mathbf{x} + \mathbf{b}_0) + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3) + \cdots) + \mathbf{b}_L,$$

where L is a positive integer, $l = 0, \dots, L$. \mathbf{W}_l is defined as the $H_{l+1} \times H_l$ matrix, where $H_0 = d$, $H_{L+1} = 1$, \mathbf{b}_l is the H_l -vector and $\sigma : \mathbb{R}^{H_l} \rightarrow \mathbb{R}^{H_l}$ which applies $\sigma_R(\cdot)$ component-wise.

Denote this class by \mathcal{F}_{MLP} . Following the notation in Farrell et al. (2021), let

$$W := \sum_{l=0}^L \{H_l H_{l+1} + H_{l+1}\} \quad (2)$$

denote the total number of weights (also called the size of the network), where L represents the

³For further details on the general case, we refer readers to Farrell et al. (2021).

depth and H the largest number of activation functions (referred to as computational nodes or units) H_l for $l = 1, \dots, l$. In practice, to reduce the number of tuning parameters, we choose H_l to be the same for each l . We refer to H as the width of the neural network.

Based on equation (2), we know that for each MLP, $\exists C \in \mathbb{R}_+$, such that

$$W \leq C \cdot H^2 L. \quad (3)$$

Farrell et al. (2021) make the following sampling assumption.

Assumption 2. Assume that $\mathbf{z}_i = (y_i, \mathbf{x}'_i)', 1 \leq i \leq n$ are i.i.d. copies of $\mathbf{Z} = (Y, \mathbf{X}) \in \mathcal{Y} \times [-1, 1]^d$, where \mathbf{X} is continuously distributed. For an absolute constant $M > 0$, assume $\|f_*\|_\infty \leq M$ and $\mathcal{Y} \subset [-M, M]$.

Given the settings mentioned above, Farrell et al. (2021) obtain the general results.

Theorem 1 (General Feedforward Architecture). Suppose f_* lies in a class \mathcal{F} . Suppose Assumption 2 holds and define

$$\epsilon_{\text{DNN}} := \sup_{\tilde{f} \in \mathcal{F}} \inf_{\substack{f \in \mathcal{F}_{\text{DNN}} \\ \|f\|_\infty \leq 2M}} \|f - \tilde{f}\|_\infty.$$

Let \hat{f}_{DNN} be the deep ReLU network estimator defined by equation (1), for a loss function obeying Assumption 1. Then with probability at least $1 - e^{-\gamma}$, for n large enough,

$$(a) \|\hat{f}_{\text{DNN}} - f_*\|_{L_2(X)}^2 \leq C \left(\frac{WL \log W}{n} \log n + \frac{\log \log n + \gamma}{n} + \epsilon_{\text{DNN}}^2 \right)$$

$$(b) E_n[(\hat{f}_{\text{DNN}} - f_*)^2] \leq C \left(\frac{WL \log W}{n} \log n + \frac{\log \log n + \gamma}{n} + \epsilon_{\text{DNN}}^2 \right)$$

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

One can treat $\frac{WL \log W}{n} \log n$ as the *variance term* and ϵ_{DNN}^2 as the *bias term*. This exhibits the classical *bias-variance tradeoff*. Furthermore, since these are non-asymptotic bounds, there is an additional term $\frac{\gamma}{n}$ that corresponds to the *confidence level*.

The function space for f_* needed to be specified to derive ϵ_{DNN} . The assumption for the function space \mathcal{F} of f_* and the smoothness assumption is as follows.

Assumption 3. Assume f_* lies in \mathcal{F} which is the Sobolev ball $\mathcal{W}^{\beta,\infty}([-1,1]^d)$, with smoothness $\beta \in \mathbb{N}_+$,

$$\mathcal{W}^{\beta,\infty}([-1,1]^d) := \{f : \max_{\alpha, |\alpha| \leq \beta} \operatorname{ess\,sup}_{\mathbf{x} \in [-1,1]^d} |D^\alpha f(\mathbf{x})| \leq 1\},$$

where $\alpha = (\alpha_1, \dots, \alpha_d)$, $|\alpha| = \alpha_1 + \dots + \alpha_d$ and $D^\alpha f$ is the weak derivative.⁴

Yarotsky (2017, Theorem 1) provides an approximation result for a specific neural network architecture. Farrell et al. (2021, Corollary 1) applies this result to Theorem 1 and obtains a nearly optimal convergence rate. However, the result applies only to the particular architecture, and this architecture is difficult to construct in practice.

Directly converting an arbitrary architecture into a fully connected one yields a total number of weights of at least $W \leq C \cdot H^2 L$. Unfortunately, this transformation generally increases W , thereby breaking the variance-bias tradeoff. To address this issue, one needs to explicitly construct a narrower neural network.

Nevertheless, in Farrell et al. (2021, Theorem 1), the authors convert the specific network constructed by Yarotsky (2017) into a fully connected one using a conservative method (i.e. Farrell et al. (2021, Lemma 1)), which allows conversion to *MLP* for any feedforward neural network. From this method, they derive the following results.

Theorem 2 (Multilayer Perceptron). *Suppose Assumptions 2 and 3 hold. Let \hat{f}_{MLP} be the MLP ReLU network estimator defined by equation (1), restricted to \mathcal{F}_{MLP} , for a loss function obeying Assumption 1, with width $H \asymp n^{\frac{d}{2(\beta+d)}} \log^2 n$ and depth $L \asymp \log n$. Then with probability at least $1 - \exp(-n^{\frac{d}{\beta+d}} \log^8 n)$, for n large enough,*

$$(a) \quad \|\hat{f}_{\text{MLP}} - f_*\|_{L_2(X)}^2 \leq C \cdot \{n^{-\frac{\beta}{\beta+d}} \log^8 n + \frac{\log \log n}{n}\} \text{ and}$$

$$(b) \quad \mathbb{E}_n[(\hat{f}_{\text{MLP}} - f_*)^2] \leq C \cdot \{n^{-\frac{\beta}{\beta+d}} \log^8 n + \frac{\log \log n}{n}\},$$

for a constant $C > 0$ independent of n , which may depend on d, β, M , and other fixed constants.

The choice of H is larger than necessary such that the convergence rate for the MLP estimator is $O(n^{-\frac{\beta}{\beta+d}} \log^8 n)$, which is slower than the optimal convergence rate $O(n^{-\frac{2\beta}{2\beta+d}})$ implied by Stone (1982).

⁴Assumption 3 requires $f_* \in [-1, 1]$, which is not compatible with Assumption 2, where $f_* \in [-M, M]$. However, since $[-M, M]$ is bounded, the results extend directly. To remain consistent with Farrell et al. (2021), we follow their assumptions.

2.2 Our approach

The optimality of Farrell et al. (2021, Corollary 1) indicates that bounds in Theorem 1 and Yarotsky (2017, Theorem 1) are sharp. As mentioned above, the main issue arises from the steps taken to convert the approximation neural network in Yarotsky (2017, Theorem 1) into a fully connected architecture.

On the one hand, the network in Yarotsky (2017, Theorem 1) is wide that converting it into a fully connected network inevitably and unnecessarily squares the total number of weights. On the other hand, the conversion method used in Farrell et al. (2021) is conservative. They apply Farrell et al. (2021, Lemma 1) to convert the approximation network into a fully connected one. Their method unnecessarily increases H_{MLP} to be asymptotically equivalent to W of the approximation network.

To address this issue, we modify Yarotsky (2017, Theorem 1) into the following theorem. In the first part, we reconstruct the network in Yarotsky (2017, Theorem 1) into a narrower architecture; in the second part, we apply Lemma 8 in the Appendix to convert it into a fully connected network.

Theorem 3. *For any $\beta, d \in \mathbb{N}_+$ and $\epsilon_{\text{DNN}} \in (0, 1)$,*

1. *there is a feedforward ReLU network architecture that it can approximate any function in $\mathcal{W}^{\beta, \infty}([-1, 1]^d)$ with approximation error less than ϵ_{DNN} , with depth $L \leq c(\ln(1/\epsilon_{\text{DNN}}) + 1)$, and width $H \leq c\epsilon_{\text{DNN}}^{-\frac{d}{2\beta}}$.*
2. *This approximation network can be converted into a fully connected network without significantly change H, L and W . The only change is the constant c . Using the fact that $W \leq C \cdot H^2 L$ for fully connected neural networks, the total number of parameters satisfies $W_{\text{MLP}} \leq c\epsilon_{\text{DNN}}^{-\frac{d}{\beta}}(\ln(1/\epsilon_{\text{DNN}}) + 1)$, with some constant $c = c(d, \beta)$.*

Remark 1. In Yarotsky (2017, Theorem 1), the neural network was constructed via a linear combination of $O(\epsilon_{\text{DNN}}^{-\frac{d}{\beta}})$ subnetworks, requiring a width of $O(\epsilon_{\text{DNN}}^{-\frac{d}{\beta}})$. A direct fully connected implementation (where $W \leq C \cdot H^2 L$) would result in the number of weights at least $O(\epsilon_{\text{DNN}}^{-\frac{2d}{\beta}} L)$. This is the first reason why the rate established in Theorem 2 is suboptimal. The first point solves this by explicitly constructing a narrower network with equivalent performance.

Remark 2. A narrower construction alone is not sufficient to solve the problem. Specifically, Farrell et al. (2021, Lemma 1) says $H_{\text{MLP}} \asymp WL$, where H_{MLP} is the width of the converted MLP and W, L are the total number of weights and depth of the original approximation network.⁵ Since we maintain the same number of weights as in Yarotsky (2017, Theorem 1), even with our narrower construction, applying Farrell et al. (2021, Lemma 1) would increase the width $H_{\text{MLP}} \asymp WL$, effectively squaring it. This is the second reason that Theorem 2 is suboptimal.

The proof is provided in Appendix B and proceeds in two steps. The first step is the same as in the original proof from Yarotsky (2017), while the second step is a reconstruction of the approximating neural network. Although the overall strategy retains the partition of unity and Taylor expansion approach, we introduce significant structural refinements to reduce the network width while maintaining the same order of the coefficients.

Compared with Yarotsky (2017, Theorem 1), Theorem 3 attains the same order of the weights and depth. The only difference is that our construction achieves width $O(\epsilon_{\text{DNN}}^{-d/(2\beta)})$ instead of $O(\epsilon_{\text{DNN}}^{-d/\beta})$. In addition, we provide an explicit conversion to fully connected neural networks.

Choosing $\epsilon_{\text{MLP}} = n^{-\frac{\beta}{2\beta+d}}$, $\gamma = n^{\frac{d}{2\beta+d}} \log^4 n$, and combining Theorem 3 with Theorem 1, we obtain the following results for MLP estimators.

Theorem 4 (Optimal Multilayer Perceptron). *Suppose Assumptions 2 and 3 hold. Let \hat{f}_{MLP} be the MLP ReLU network estimator defined by equation (1), for a loss function obeying Assumption 1, with width $H \asymp n^{\frac{d}{4\beta+2d}}$ and depth $L \asymp \log n$. Then with probability at least $1 - \exp(-n^{\frac{d}{2\beta+d}} \log^4 n)$, for n large enough,⁶*

$$(a) \quad \|\hat{f}_{\text{MLP}} - f_*\|_{L_2(X)}^2 \leq C \cdot \{n^{-\frac{2\beta}{2\beta+d}} \log^4 n + \frac{\log \log n}{n}\} \text{ and}$$

$$(b) \quad \mathbb{E}_n[(\hat{f}_{\text{MLP}} - f_*)^2] \leq C \cdot \{n^{-\frac{2\beta}{2\beta+d}} \log^4 n + \frac{\log \log n}{n}\},$$

for a constant $C > 0$ independent of n , which may depend on d, β, M , and other fixed constants.

Here, we obtain a convergence rate (up to logarithmic factors) of $n^{-\frac{2\beta}{2\beta+d}}$, rather than $n^{-\frac{\beta}{\beta+d}}$.

This is the optimal rate in the sense of Stone (1982).

⁵In fact, Farrell et al. (2021, Lemma 1) can be strengthened so that the width of the resulting fully connected network is bounded above by HL , where H and L are the width and depth of the original architecture. However, this refinement still causes an undesirable expansion of H by an additional factor of L .

⁶The logarithmic term could potentially be improved, but for simplicity we maintain the same log term as in Farrell et al. (2021, Corollary 1).

This result contributes to the strand of literature building on the framework of Farrell et al. (2021) (e.g., Brown (2024), Zhang and Bradic (2024), Zhang et al. (2024), Colangelo and Lee (2025)). Moreover, by leveraging our approximation result, other studies that build on Yarotsky (2017) can also attain the optimal convergence rate for statistical inference (e.g., Feng et al. (2023), Jiao et al. (2025)). Our approximation construction may also be useful for approximating other deep neural network architectures.

3 Mitigating the curse of dimensionality

This section presents concise results illustrating scenarios in which deep neural networks can mitigate the curse of dimensionality. The results are obtained by directly applying the approximation results to Theorem 1. The approximation results used here are adapted from Petersen and Zech (2024, Chapter 8).⁷ The key difference is that we convert the approximation networks into narrower and fully connected architectures using Theorem 3. Subsection 3.1 addresses functions with compositional structures, while Subsection 3.2 focuses on functions defined on manifolds.

3.1 Functions with compositional structure

We modify Assumption 3 to the following compositional function assumption, based on Petersen and Zech (2024, Chapter 8.2).

Let \mathcal{A} be a directed acyclic graph with T vertices η_1, \dots, η_T satisfying the following: the first d vertices η_1, \dots, η_d have no incoming edges; each vertex has at most $d_* \in \mathbb{N}$ incoming edges; and the final vertex η_T has no outgoing edges.

For every vertex η_j with $j > d$, assign a function $f_j : \mathbb{R}^{d_j} \rightarrow \mathbb{R}$, where d_j is the number of elements in the set

$$S_j := \{ i : \text{there is an edge from } \eta_i \text{ to } \eta_j \}.$$

⁷Although Petersen and Zech (2024, Chapter 8) derive approximation results on Hölder spaces, their arguments rely on a variant of Yarotsky (2017). Hence, the same reasoning directly extends to Sobolev spaces. Consequently, the results in this note also apply to Hölder spaces.

Assume $1 \leq d_j \leq d_*$ for all $j > d$, and define

$$\begin{aligned} F_j &:= x_j, \quad \text{for all } j \leq d, \\ F_j &:= f_j((F_i)_{i \in S_j}), \quad \text{for all } j > d. \end{aligned} \tag{4}$$

Then $F_T(x_1, \dots, x_d)$ is a function from \mathbb{R}^d to \mathbb{R} . Assuming

$$\max_{\alpha, |\alpha| \leq \beta} \operatorname{ess\,sup}_{\mathbf{x} \in \mathbb{R}^{d_j}} |D^\alpha f_j(\mathbf{x})| \leq 1, \quad \text{for all } j = d+1, \dots, T, \tag{5}$$

we denote the set of all such functions F_T by $\mathcal{W}_{d_*, T}^{\beta, \infty}([-1, 1]^d)$.

Assumption 4. Assume $f_* \in \mathcal{W}_{d_*, T}^{\beta, \infty}([-1, 1]^d)$.

This model of compositional functions can accommodate a broad class of compositional structures. To illustrate this idea, we present the interaction models from Stone (1994), noting that we allow for more general models than this simple summation case.

Example. For some $d_* \in \{1, \dots, d\}$, denote $\mathbf{x}_I = (x_{i_1}, \dots, x_{i_{d_*}})'$, where $I = \{i_1, \dots, i_{d_*}\}$ and $1 \leq i_1 \leq \dots \leq i_{d_*} \leq d$. And assume the structure

$$f_0(\mathbf{x}) = \sum_{I \subseteq \{1, \dots, d\}, |I|=d_*} f_I(\mathbf{x}_I),$$

where each function $f_I(\cdot)$ satisfies equation (5). Denote the set of indices I as \mathcal{I} , and let its cardinality be k . Order the indices as (I_1, \dots, I_k) . According to the definition in equation (4), we have

$$F_{j+d} = f_{I_j} \quad \text{for } j = 1, \dots, k.$$

Furthermore, define

$$F_{d+k+1} = f_{I_1} + f_{I_2},$$

$$F_{d+k+j} = F_{d+k+1} + f_{I_{j+1}}, \quad \text{for } j = 2, \dots, k-1,$$

$$F_{d+2k-1} = f_0(\mathbf{x}).$$

Thus, we conclude that $f_0 \in \mathcal{W}_{d_*, d+2k-1}^{\beta, \infty}([-1, 1]^d)$.

Applying the approximation result for functions with compositional structure (see Appendix A.2), we obtain the following theorem.

Theorem 5. *Suppose Assumptions 2 and 4 hold. Let \hat{f}_{MLP} be the MLP ReLU network estimator defined by equation (1), restricted to \mathcal{F}_{MLP} , for a loss function obeying Assumption 1, with width $H \asymp n^{\frac{d_*}{2\beta+d_*}} \log^2 n$ and depth $L \asymp \log n$. Then with probability at least $1 - \exp(-n^{\frac{d_*}{2\beta+d_*}} \log^4 n)$, for n large enough,*

$$(a) \quad \|\hat{f}_{\text{MLP}} - f_*\|_{L_2(X)}^2 \leq C \cdot \{n^{-\frac{2\beta}{2\beta+d_*}} \log^4 n + \frac{\log \log n}{n}\} \text{ and}$$

$$(b) \quad \mathbb{E}_n[(\hat{f}_{\text{MLP}} - f_*)^2] \leq C \cdot \{n^{-\frac{2\beta}{2\beta+d_*}} \log^4 n + \frac{\log \log n}{n}\},$$

for a constant $C > 0$ independent of n , which may depend on d_*, M, β, T , and other fixed constants, but it doesn't depend on d .

Here, the convergence rate depends on the largest dimension d_* of the inputs within the compositional structure, rather than the original input dimension d .

It is worth noting that the constant C does not depend on d (although T may potentially depend on d), and this result may be particularly useful in high-dimensional settings. Under this setting, deep neural networks are superior to other machine learning methods in two respects. First, they can capture nonlinear compositional structures. Second, deep neural networks can automatically mitigate the curse of dimensionality without requiring variable selection or dimension reduction, thereby avoiding the bias that such procedures may introduce.

3.2 Functions on manifolds

The following definition of a compact smooth manifold is adapted from Milnor and Weaver (1997, Chapter 1). For a comprehensive treatment of manifolds, we refer the reader to that work.

Definition 1 (Compact Smooth Manifold). For $k, l \in \mathbb{N}_+$, let $U \subset \mathbb{R}^k$ and $V \subset \mathbb{R}^k$. A mapping $f : U \rightarrow V$ is called **smooth** if for each $\mathbf{x} \in U$ there exists an open set $U' \subset \mathbb{R}^k$ such that $\mathbf{x} \in U'$. And there exists a smooth mapping $F : U' \rightarrow \mathbb{R}^l$ such that $\forall \mathbf{x}^1 \in U \cap U', F(\mathbf{x}^1) = f(\mathbf{x}^1)$.

A map $f : U \rightarrow V$ is called a **diffeomorphism** if f maps U homeomorphically onto V and if both f and its inverse f^{-1} are smooth.

For $d, m \in \mathbb{N}_+$, a compact subset $\mathcal{M} \subset \mathbb{R}^d$ is called a **compact smooth manifold** of dimension m if for every $\mathbf{x} \in \mathcal{M}$, there exists an open set $W \subset \mathbb{R}^k$ such that $\mathbf{x} \in W$ and $W \cap \mathcal{M}$ is diffeomorphic to an open subset $U \subset \mathbb{R}^m$.

We adjust Assumption 3 to the following assumption.

Assumption 5. *We further assume that there exists a compact, smooth m -dimensional manifold $\mathcal{M} \subset [-1, 1]^d$ such that $\mathbf{X} \in \mathcal{M}$. And we assume f_* that lies in the Sobolev ball $\mathcal{W}^{\beta, \infty}(\mathcal{M})$, with smoothness $\beta \in \mathbb{N}_+$,*

$$\mathcal{W}^{\beta, \infty}(\mathcal{M}) := \{f : \max_{\alpha, |\alpha| \leq \beta} \text{ess sup}_{\mathbf{x} \in \mathcal{M}} |D^\alpha f(\mathbf{x})| \leq 1\},$$

where $\alpha = (\alpha_1, \dots, \alpha_d)$, $|\alpha| = \alpha_1 + \dots + \alpha_d$ and $D^\alpha f$ is the weak derivative.

Applying the approximation result for functions on manifolds (see Appendix A.2), we obtain the following theorem.

Theorem 6. *Suppose Assumptions 2 and 5 hold. Let \hat{f}_{MLP} be the MLP ReLU network estimator defined by equation (1), restricted to \mathcal{F}_{MLP} , for a loss function obeying Assumption 1, with width $H \asymp n^{\frac{m}{2\beta+m}} \log^2 n$ and depth $L \asymp \log n$. Then with probability at least $1 - \exp(-n^{\frac{m}{2\beta+m}} \log^4 n)$, for n large enough,*

- (a) $\|\hat{f}_{\text{MLP}} - f_*\|_{L_2(X)}^2 \leq C \cdot \{n^{-\frac{2\beta}{2\beta+m}} \log^4 n + \frac{\log \log n}{n}\}$ and
- (b) $\mathbb{E}_n[(\hat{f}_{\text{MLP}} - f_*)^2] \leq C \cdot \{n^{-\frac{2\beta}{2\beta+m}} \log^4 n + \frac{\log \log n}{n}\},$

for a constant $C > 0$ independent of n , which may depend on \mathcal{M}, β, m , and other fixed constants.

In this case, the convergence rate depends on the intrinsic dimension m rather than the ambient dimension d . Notice that the constant term C implicitly depends on d since \mathcal{M} may implicitly depend on d .

In empirical economics applications, it is often difficult to justify that the function of interest possesses a compositional structure. In contrast, it is usually more feasible to assess whether

the data lie on a low-dimensional manifold. Consequently, this result may be more useful in practice.

4 Conclusion

By modifying only the approximation component of the proof in Farrell et al. (2021, Theorem 1), this note establishes that the convergence rate of MLP estimators is indeed optimal (up to a logarithmic factor) in the sense of Stone (1982). To this end, this note extends the results of Yarotsky (2017). Instead of focusing on approximation results tied to network size, we demonstrate that a narrower network of comparable size can achieve the same approximation rate. This result can be of independent interest. One can apply our narrower network construction to convert other neural architectures into fully connected networks without substantially increasing the total number of parameters.

In addition, we establish convergence rates for functions with compositional structures and for functions defined on low-dimensional manifolds. In both cases, consistent with previous literature, we show that deep neural networks can effectively circumvent the curse of dimensionality. We encourage future empirical work to justify the use of deep learning based on these two reasons.

A Lemma

A.1 Lemmas used in Appendix B

We will first introduce the lemma from Yarotsky (2017), which combines results from Liu et al. (2022). We then introduce a lemma used to convert approximation neural networks into fully connected networks. These results are used to show the main results in Appendix B.

The following lemma is based on Yarotsky (2017, Proposition 3), while its final part follows from Liu et al. (2022, Proposition 2).

Lemma 7. *The function $f(x) = x^2$ on the segment $[0, 1]$ can be approximated with any error $\delta > 0$ by a ReLU network $\tilde{f}_{sq,\delta}$ such that $\tilde{f}_{sq,\delta}(0) = 0$ and $|\tilde{f}_{sq,\delta} - x^2| < \delta$ for $x \in [-1, 1]$. Given*

$Q > 0$, and $\epsilon \in (0, 1)$, there is a ReLU network η with two input units that implements a function $\tilde{x} : \mathbb{R}^2 \rightarrow \mathbb{R}$.

Assume without loss of generality that $Q \geq 1$ and set

$$\tilde{x}(x, y) = \frac{Q^2}{8} (\tilde{f}_{\text{sq},\delta}(\frac{|x+y|}{2Q}) - \tilde{f}_{\text{sq},\delta}(\frac{|x|}{2Q}) - \tilde{f}_{\text{sq},\delta}(\frac{|y|}{2Q})),$$

where $\delta = \frac{8\epsilon}{3Q^2}$. Then, the following hold:

1. for any inputs x, y , if $|x| \leq Q$ and $|y| \leq Q$, then $|\tilde{x}(x, y) - xy| \leq \epsilon$;
2. $\tilde{x}(x, 0) = \tilde{x}(0, y) = 0$;
3. the depth and the number of weights and hidden units in η are not greater than $c_1 \ln(1/\epsilon) + c_2$ with an universal constant c_1 and a constant $c_2 = c_2(Q)$,
4. the neural network can be made fully connected with a width of at most 12.

The next lemma enables us to convert the approximating function used in Proof B into fully connected architectures.

Lemma 8. With $\tilde{x}(\cdot)$ defined in Lemma 7, and a fully connected feedforward ReLU neural network is denoted by f_{MLP}^m or f_{MLP}^α , indicating dependence on the parameter m or α , respectively. We assume that, for parameters $m, \alpha \in \mathbb{R}$, m takes values in a finite set S_m and α takes values in a finite set S_α . Denote $N_{\max} := \max(|S_m|, |S_\alpha|)$. Suppose each of the neural networks has a depth at most L , a width at most H , and $a_{m,\alpha} \in \mathbb{R}$. Then, we can convert the following neural networks $\sum_{\alpha \in S_\alpha} \tilde{x}(\sum_{m \in S_m} a_{m,\alpha} f_{\text{MLP}}^m, f_{\text{MLP}}^\alpha)$ into fully connected architectures with depth at most $L + 1 + c_1 \ln(1/\epsilon) + c_2$ and width at most $\max(12, 2H)N_{\max}$.⁸

Remark 3. A key observation for our proof is that the input can always be passed through a ReLU activation function without altering its value, which is not possible with other activation functions (e.g., the sigmoid function).

Proof. For each $f_{\text{MLP}}^m, m \in S_m$, if the depth is less than L , we can use the fact that

$$x = \sigma_R(x) - \sigma_R(-x)$$

⁸For simplicity, we assume the functions satisfies the definition of $\tilde{x}(\cdot)$.

to extend it so that every f_{MLP}^m has the same depth L . Since each f_{MLP}^m is fully connected and has the same depth, combining them into a single fully connected network is achieved by simply connecting each node to the nodes in the adjacent layers. Thus, each value of f_{MLP}^m for $m \in S_m$ can be represented by the fully connected neural network with depth at most L and width at most $2HN_{\max}$ with $|S_m|$ output units. The same arguments apply to f_{MLP}^α .

For each α , the only difference of $\sum_{m \in S_m} a_{m,\alpha} f_{\text{MLP}}^m$ is $a_{m,\alpha}$. They are merely linear combinations of f_{MLP}^m for $m \in S_m$. Therefore, with one more layer, we can use the same fully connected neural network but with at most N_{\max} output units to represent each value of $\sum_{m \in S_m} a_{m,\alpha} f_{\text{MLP}}^m$. Each output unit corresponds to a different value of α . That is,

$$\sum_{m \in S_m} a_{m,\alpha} f_{\text{MLP}}^m = \sigma_R(\sum_{m \in S_m} a_{m,\alpha} f_{\text{MLP}}^m) - \sigma_R(-\sum_{m \in S_m} a_{m,\alpha} f_{\text{MLP}}^m)$$

and the width needed for this operation is at most $2N_{\max}$.

Thus, with a single fully connected neural network, we can represent each value of $\sum_{m \in S_m} a_{m,\alpha} f_{\text{MLP}}^m$ and f_{MLP}^α for each $\alpha \in S_\alpha$. The neural network has at most $2N_{\max}$ output units, depth at most $L + 1$, and width at most $2HN_{\max}$. Each output unit corresponds either $\sum_{m \in S_m} a_{m,\alpha} f_{\text{MLP}}^m$ or f_{MLP}^α for each $\alpha \in S_\alpha$.

For each $\alpha \in S_\alpha$, each $\sum_{m \in S_m} a_{m,\alpha} f_{\text{MLP}}^m$ corresponds to a single f_{MLP}^α . Thus, by Lemma 7, we need at most $L + 1 + c_1 \ln(1/\epsilon) + c_2$ more layers, whose width are at most $\max(12, 2H)N_{\max}$, to represent $\sum_{\alpha \in S_\alpha} \tilde{\times}(\sum_{m \in S_m} a_{m,\alpha} f_{\text{MLP}}^m, f_{\text{MLP}}^\alpha)$. \square

A.2 Lemmas used in Section 3

The lemmas in this subsection are adapted from Petersen and Zech (2024, Chapter 8), with the condition on the number of weights replaced by a condition on width. Based on the proofs in Petersen and Zech (2024, Chapter 8), one can directly establish the following two lemmas. The only change is that, in determining the width, we apply our Theorem 3 rather than Yarotsky (2017, Theorem 1), which is used there to determine the network size. For brevity, we state the two lemmas without proof.

The following lemma is modified from Petersen and Zech (2024, Proposition 8.5) and used

to show Theorem 5.

Lemma 9. *For any $\beta, d, d_*, T \in \mathbb{N}_+$ and $\epsilon \in (0, 1)$, there is a fully connected ReLU network architecture that can represent any function in $\mathcal{W}_{d_*, T}^{\beta, \infty}([-1, 1]^d)$, with the depth $L \leq c(\ln(1/\epsilon) + 1)$, the width $H \leq c\epsilon^{-\frac{d_*}{2\beta}}$. Moreover, using the fact that $W \leq C \cdot H^2 L$, the weights $W \leq c\epsilon^{-\frac{d_*}{\beta}}(\ln(1/\epsilon) + 1)$, with some constant $c = c(d_*, T, \beta)$.*

The next lemma is modified from Petersen and Zech (2024, Proposition 8.7), and used to show Theorem 6.

Lemma 10. *For any $\beta, d, m \in \mathbb{N}_+$ and $\epsilon \in (0, 1)$, suppose \mathcal{M} is a smooth, compact m dimensional manifold s.t. $\mathcal{M} \subseteq [-1, 1]^d$. Then, there is a fully connected ReLU network architecture that can represent any function in $\mathcal{W}^{\beta, \infty}(\mathcal{M})$, with the depth $L \leq c(\ln(1/\epsilon) + 1)$, the width $H \leq c\epsilon^{-\frac{m}{2\beta}}$. Moreover, using the fact that $W \leq C \cdot H^2 L$, the weights $W \leq c\epsilon^{-\frac{m}{\beta}}(\ln(1/\epsilon) + 1)$, with some constant $c = c(d, m, \mathcal{M}, \beta)$.*

B Proof

For completeness, we provide a self-contained presentation below, with our modifications clearly highlighted. Appendix B.1 reproduces Yarotsky (2017, Theorem 1), while Appendix B.2 presents our modified proof based on Appendix B.1. We also include the proof of Yarotsky (2017, Theorem 1) so that readers can clearly see the connections and differences with our results.

B.1 Yarotsky (2017, Theorem 1)

The following lemma is Yarotsky (2017, Theorem 1), while Theorem 3 is a modified version of it.

Lemma 11. *For any $\beta, d \in \mathbb{N}_+$ and $\epsilon \in (0, 1)$, there is a ReLU network architecture that it can approximate any function in $\mathcal{W}^{\beta, \infty}([-1, 1]^d)$ with error ϵ , with the depth $L \leq c(\ln(1/\epsilon) + 1)$, the number of weights $W \leq c\epsilon^{-\frac{d}{\beta}}\ln(1/\epsilon + 1)$, with some constant $c = c(d, \beta)$.*

Proof.

Given a $N \in \mathbb{N}_+$. Consider a partition of unity:

$$\sum_{\mathbf{m}} \phi_{\mathbf{m}}(\mathbf{x}) \equiv 1, \quad \mathbf{x} \in [-1, 1]^d.$$

which is constructed by a grid of $(2N + 1)^d$ function $\phi_{\mathbf{m}}$ on the domain $[-1, 1]^d$

Here $\mathbf{m} = (m_1, \dots, m_d)$, $m_i \in \{-N, \dots, -1, 0, 1, \dots, N\}$, $i = 1, \dots, d$, and define the function the function $\phi_{\mathbf{m}}$ as

$$\phi_{\mathbf{m}}(\mathbf{x}) = \prod_{k=1}^d \psi(3N(x_k - \frac{m_k}{N})),$$

where

$$\psi(x) = \begin{cases} 1, & |x| < 1, \\ 0, & 2 < |x|, \\ 2 - |x|, & 1 \leq |x| \leq 2. \end{cases}$$

Note that

$$\|\psi\|_{\infty} = 1 \quad \text{and} \quad \|\phi_{\mathbf{m}}\|_{\infty} = 1, \forall \mathbf{m}$$

and

$$\text{supp } \phi_{\mathbf{m}} \subset \{\mathbf{x} : \left|x_k - \frac{m_k}{N}\right| < \frac{1}{N}, \forall k\}.$$

For any $\mathbf{m} \in \{-N, \dots, -1, 0, 1, \dots, N\}^d$, consider the degree- $(\beta - 1)$ Taylor polynomial for the function f at $\mathbf{x} = \frac{\mathbf{m}}{N}$:

$$P_{\mathbf{m}}(\mathbf{x}) = \sum_{\alpha: |\alpha| < \beta} \frac{D^{\alpha} f}{\alpha!} \Big|_{\mathbf{x} = \frac{\mathbf{m}}{N}} \left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{\alpha}, \quad (6)$$

with $\alpha! = \prod_{k=1}^d \alpha_k!$ and $(\mathbf{x} - \frac{\mathbf{m}}{N})^{\alpha} = \prod_{k=1}^d (x_k - \frac{m_k}{N})^{\alpha_k}$. Define an approximation to f by

$$f_1 = \sum_{\mathbf{m} \in \{-N, \dots, -1, 0, 1, \dots, N\}^d} \phi_{\mathbf{m}} P_{\mathbf{m}}. \quad (7)$$

Using the Taylor expansion of f , the approximation error is bounded by:

$$|f(\mathbf{x}) - f_1(\mathbf{x})| \leq \frac{2^d d^\beta}{\beta!} \left(\frac{1}{N}\right)^\beta.$$

$\forall x \in \mathbb{R}$, a ceiling function maps x to the least integer greater than or equal to x , denoted $\lceil x \rceil$. Choose

$$N = \lceil \left(\frac{\beta!}{2^d d^\beta} \frac{\epsilon}{2} \right)^{-1/\beta} \rceil, \quad (8)$$

then

$$\|f - f_1\|_\infty \leq \frac{\epsilon}{2}.$$

By equation (6) the coefficients of the polynomials $P_{\mathbf{m}}$ are uniformly bounded $\forall f \in \mathcal{W}^{\beta, \infty}([-1, 1]^d)$:

$$P_{\mathbf{m}}(\mathbf{x}) = \sum_{\alpha: |\alpha| < \beta} a_{\mathbf{m}, \alpha} (\mathbf{x} - \frac{\mathbf{m}}{N})^\alpha, \quad |a_{\mathbf{m}, \alpha}| \leq 1. \quad (9)$$

The second step is to construct a network architecture which can approximate any function of the form (7) with uniform error $\frac{\epsilon}{2}$, assuming that N is given by (8) and the polynomials $P_{\mathbf{m}}$ are of the form (9).

Expand f_1 as

$$f_1(\mathbf{x}) = \sum_{\mathbf{m} \in \{-N, \dots, -1, 0, 1, \dots, N\}^d} \sum_{\alpha: |\alpha| < \beta} a_{\mathbf{m}, \alpha} \phi_{\mathbf{m}}(\mathbf{x}) (\mathbf{x} - \frac{\mathbf{m}}{N})^\alpha.$$

The expansion is a linear combination of $\phi_{\mathbf{m}}(\mathbf{x})(\mathbf{x} - \frac{\mathbf{m}}{N})^\alpha$, and the total number of such terms is less than $d^\beta \prod_{i=1}^d (2N + 1)$. Each of them is a product of at most $d + \beta - 1$ piece-wise linear univariate factors: total number of d for $\psi(3Nx_k - 3m_k)$ functions and at most $\beta - 1$ linear expressions of $x_k - \frac{m_k}{N}$. A neural network can approximate such a product according to Lemma 7.

Choose $Q = d + \beta$ and δ (to be chosen later), and use $\tilde{\times}(\cdot)$ (defined in Lemma 7) to denote the approximate multiplication and iteratively apply $\tilde{\times}$ to approximate

the product $\phi_{\mathbf{m}}(\mathbf{x})(\mathbf{x} - \frac{\mathbf{m}}{N})^{\mathbf{n}}$. That is,

$$\tilde{f}_{\mathbf{m},\alpha}(\mathbf{x}) = \tilde{\times}(\psi(3Nx_1 - 3m_1), \tilde{\times}(\psi(3Nx_2 - 3m_2), \dots, \tilde{\times}(\psi(3Nx_k - 3m_k, \dots))). \quad (10)$$

Since,

$$\left| \tilde{f}_{\mathbf{m},\alpha}(\mathbf{x}) - \phi_{\mathbf{m}}(\mathbf{x})(\mathbf{x} - \frac{\mathbf{m}}{N})^{\alpha} \right| \leq (d + \beta - 1)\delta \quad (11)$$

The full approximation is:

$$\tilde{f} = \sum_{\mathbf{m} \in \{-N, \dots, -1, 0, 1, \dots, N\}^d} \sum_{\alpha: |\alpha| < \beta} a_{\mathbf{m},\alpha} \tilde{f}_{\mathbf{m},\alpha}.$$

There are at most $d^\beta (2N + 1)^d$ possible values for (\mathbf{m}, α) .

The approximation error of \tilde{f} is:

$$|\tilde{f}(\mathbf{x}) - f_1(\mathbf{x})| \leq 2^d d^\beta (d + \beta - 1)\delta.$$

Choose

$$\delta = \frac{\epsilon}{2^{d+1} d^\beta (d + \beta - 1)},$$

then $\|\tilde{f} - f_1\|_\infty \leq \frac{\epsilon}{2}$. And $\|\tilde{f} - f\|_\infty \leq \epsilon$.

□

Since a full approximation neural network is a linear combination of fewer than $d^\beta \prod_{i=1}^d (2N + 1)$ sub-networks, the resulting network width is $H = O((2N + 1)^d) = O(\epsilon^{-\frac{d}{\beta}})$.

B.2 Proof of Theorem 3

Proof. Based on the proof above, first notice that we can directly represent $\psi(\cdot)$ by a ReLU neural network,

$$\psi(x) = \sigma_R(2 - \sigma_R(x) - \sigma_R(-x)) - \sigma_R(1 - \sigma_R(x) - \sigma_R(-x)).$$

Since $\phi_{\mathbf{m}}(\cdot)$ is a product of $\psi(\cdot)$, we can easily approximate it using Lemma 7. The

fundamental constraint is that $a_{\mathbf{m}, \alpha}$ admits at most $d^\beta(2N + 1)^d$ distinct values. Under this condition, a narrower network architecture can maintain identical approximation error bounds. This is achieved solely through an appropriate reconstruction of $\tilde{f}_{\mathbf{m}, \alpha}(\mathbf{x})$.

Constrained by the bound $W \leq C \cdot H^2 L$ for fully connected neural networks, our goal is to construct a neural network with width $O(2N + 1)^{\frac{d}{2}}$. If d is an even number, the modification is straightforward. For odd d , however, a more delicate construction is required.

The proof is separated into three cases:

Case 1: if d is even, we can write:

$$\begin{aligned} & \phi_{\mathbf{m}}(\mathbf{x})(\mathbf{x} - \frac{\mathbf{m}}{N})^\alpha \\ &= \prod_{k=1}^d \psi(3N(x_k - \frac{m_k}{N}))(x_k - \frac{m_k}{N})^{\alpha_k} \\ &= \left(\prod_{k=1}^{\frac{d}{2}} \psi(3N(x_k - \frac{m_k}{N}))(x_k - \frac{m_k}{N})^{\alpha_k} \right) \left(\prod_{k=\frac{d}{2}+1}^d \psi(3N(x_k - \frac{m_k}{N}))(x_k - \frac{m_k}{N})^{\alpha_k} \right). \end{aligned}$$

Denote $\mathbf{m}^1 = (m_1, \dots, m_{\frac{d}{2}})$, $\mathbf{m}^2 = (m_{\frac{d}{2}+1}, \dots, m_d)$, $\alpha^1 = (\alpha_1, \dots, \alpha_{\frac{d}{2}})$, $\alpha^2 = (\alpha_{\frac{d}{2}+1}, \dots, \alpha_d)$, $\mathbf{x}^1 = (x_1, \dots, x_{\frac{d}{2}})$, $\mathbf{x}^2 = (x_{\frac{d}{2}}, \dots, x_d)$.

We can rewrite $f_1(\mathbf{x})$, so that

$$\begin{aligned} f_1(\mathbf{x}) &= \sum_{\mathbf{m} \in \{-N, \dots, -1, 0, 1, \dots, N\}^d} \sum_{\alpha: |\alpha| < \beta} a_{\mathbf{m}, \alpha} \phi_{\mathbf{m}}(\mathbf{x})(\mathbf{x} - \frac{\mathbf{m}}{N})^\alpha \\ &= \sum_{\mathbf{m}^2 \in \{-N, \dots, -1, 0, 1, \dots, N\}^{\frac{d}{2}}} \sum_{\alpha^2: |\alpha^2| < \beta} \phi_{\mathbf{m}^2}(\mathbf{x}^2)(\mathbf{x}^2 - \frac{\mathbf{m}^2}{N})^{\alpha^2} \\ &\quad \times \left(\sum_{\mathbf{m}^1 \in \{-N, \dots, -1, 0, 1, \dots, N\}^{\frac{d}{2}}} \sum_{\alpha^1: |\alpha^1| < \beta - |\alpha^2|} a_{\mathbf{m}, \alpha} \phi_{\mathbf{m}^1}(\mathbf{x}^1)(\mathbf{x}^1 - \frac{\mathbf{m}^1}{N})^{\alpha^1} \right). \end{aligned}$$

Using the definition in equation (10), we can approximate $\phi_{\mathbf{m}^1}(\mathbf{x}^1)(\mathbf{x}^1 - \frac{\mathbf{m}^1}{N})^{\alpha^1}$ by $\tilde{f}_{\mathbf{m}^1, \alpha^1}$ and $\phi_{\mathbf{m}^2}(\mathbf{x}^2)(\mathbf{x}^2 - \frac{\mathbf{m}^2}{N})^{\alpha^2}$ by $\tilde{f}_{\mathbf{m}^2, \alpha^2}$. From Lemma 7, $\forall i = 1, 2, \tilde{f}_{\mathbf{m}^i, \alpha^i}$ can be approximated by a fully connected neural network such that its depth is $O(\ln(1/\delta) + 1)$ and width is $O(\frac{d}{2})$.

Since (\mathbf{m}^1, α^1) and (\mathbf{m}^2, α^2) both have at most $d^\beta(2N + 1)^{\frac{d}{2}}$ possible values, with $2 * d^\beta(2N + 1)^{\frac{d}{2}}$ output units, depth $O(\ln(1/\delta) + 1)$ and width $O((2N + 1)^{\frac{d}{2}})$, by Lemma 7, we can use a

(fully connected) neural network to represent all possible values for $\tilde{f}_{\mathbf{m}^i, \alpha^i}, i = 1, 2$.

For any pair of (\mathbf{m}^2, α^2) , if $\alpha^1 + \alpha^2 > \beta$, we set $a_{\mathbf{m}, \alpha} = 0$. By equation (9), $|a_{\mathbf{m}, \alpha}| \leq 1$, which satisfies the definition of the function $\tilde{\times}(\cdot)$.

Therefore, the full approximation function is redefined as

$$\tilde{f} = 2^{\frac{d}{2}} d^\beta \sum_{\mathbf{m}^2, \alpha^2} \tilde{\times}\left(\frac{1}{2^{\frac{d}{2}} d^\beta} \sum_{\mathbf{m}^1, \alpha^1} a_{\mathbf{m}, \alpha} \tilde{f}_{\mathbf{m}^1, \alpha^1}(\mathbf{x}^1), \tilde{f}_{\mathbf{m}^2, \alpha^2}(\mathbf{x}^2)\right),$$

where $\frac{1}{2^{\frac{d}{2}} d^\beta}$ is to ensure the inputs of $\tilde{\times}(\cdot)$ satisfies the definition.

By equation (11), for $i = 1, 2$,

$$\left| \tilde{f}_{\mathbf{m}^i, \alpha^i}(\mathbf{x}^i) - \phi_{\mathbf{m}^i}(\mathbf{x}^i) \left(\mathbf{x}^i - \frac{\mathbf{m}^i}{N} \right)^{\alpha^i} \right| \leq \left(\frac{d}{2} + |\alpha^i| - 1 \right) \delta.$$

Thus,

$$|\tilde{f}(\mathbf{x}) - f_1(\mathbf{x})| \leq 2^d d^n (d + \beta) \delta,$$

we choose $\delta = \frac{\epsilon}{2^{d+1} d^\beta (d + \beta - 1)}$, and $\|\tilde{f} - f\| \leq \epsilon$.

By using Lemma 8, the full approximation neural network is fully connected and has depth $\mathcal{O}(\ln(1/\delta) + 1)$, and width at most $\mathcal{O}((2N+1)^{\frac{d}{2}})$. The remainder of the proof follows unchanged. Thus, the weight count scales as $\mathcal{O}((2N+1)^d \ln(1/\delta))$.

Case 2: if d is odd and $d > 1$, since $d - 1$ is even, only the extra dimension requires special treatment.⁹

Let $\bar{d} = \frac{d+1}{2}$ denote the central dimension index. Denote $\mathbf{m}^1 = (m_1, \dots, m_{\bar{d}-1}), \mathbf{m}^2 = (m_{\bar{d}+1}, \dots, m_d), \alpha^1 = (\alpha_1, \dots, \alpha_{\bar{d}-1}), \alpha^2 = (\alpha_{\bar{d}+1}, \dots, \alpha_d), \mathbf{x}^1 = (x_1, \dots, x_{\bar{d}-1}), \mathbf{x}^2 = (x_{\bar{d}+1}, \dots, x_d)$. Similar to the even case, denote $\tilde{f}_{\mathbf{m}^1, \alpha^1}(\mathbf{x}^1)$ as the neural network to approximate $f_{\mathbf{m}^1, \alpha^1}(\mathbf{x}^1) := \prod_{k=1}^{\bar{d}-1} \psi(3N(x_k - \frac{m_k}{N})) (x_k - \frac{m_k}{N})^{\alpha_k}$ and let $\tilde{f}_{\mathbf{m}^2, \alpha^2}(\mathbf{x}^2)$ be the neural network to approximate $f_{\mathbf{m}^2, \alpha^2}(\mathbf{x}^2) := \prod_{k=\bar{d}+1}^d \psi(3N(x_k - \frac{m_k}{N})) (x_k - \frac{m_k}{N})^{\alpha_k}$.

⁹We implicitly assume $N \geq 4$, which is justified since N becomes large as ϵ approaches zero. Therefore, this assumption is made without loss of generality.

Decompose the function:

$$\begin{aligned}
& \phi_{\mathbf{m}}(\mathbf{x}) \left(\mathbf{x} - \frac{\mathbf{m}}{N} \right)^\alpha \\
&= \prod_{k=1}^d \psi(3N(x_k - \frac{m_k}{N})) (x_k - \frac{m_k}{N})^{\alpha_k} \\
&= \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})) (x_{\bar{d}} - \frac{m_{\bar{d}}}{N})^{\alpha_{\bar{d}}} f_{\mathbf{m}^1, \alpha^1}(\mathbf{x}^1) f_{\mathbf{m}^2, \alpha^2}(\mathbf{x}^2),
\end{aligned}$$

since $m_{\bar{d}} \in \{-N, \dots, -1, 0, 1, \dots, N\}$, the key is to separate all $2N+1$ possible values of $\psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N}))$ into two parts, each with $\sqrt{2N+1}$ units and the combination of two parts can represent $\psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N}))$.

Denote $A = \lceil \sqrt{2N+1} \rceil$. And define sets:

$$\mathcal{G}_g = \{-N + i * A + g, i = 0, \dots, A-1\}, g = 0, \dots, A-1.$$

Each set has A elements. $\forall g = 0, \dots, A-1, \forall x_{\bar{d}} \in [-1, 1]$, there are at most two $m_{\bar{d}}$, each in separate sets, such that $\psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})) > 0$.

Also, define

$$\begin{aligned}
\mathcal{G}^1 &:= \mathcal{G}_0 \cup \mathcal{G}_1 \cup \mathcal{G}_{A-1}, \\
\mathcal{G}^2 &:= \left(\bigcup_{g=2}^{A-2} \mathcal{G}_g \right).
\end{aligned}$$

By the construction,

$$\mathcal{G}^1 \cup \mathcal{G}^2 \supseteq \{-N, \dots, -1, 0, 1, \dots, N\},$$

$$\mathcal{G}_{g'} \cap \mathcal{G}_g = \emptyset, \quad \forall g, g' = 0, \dots, A-1, g \neq g'.$$

Define a new function:

$$\kappa_b(x_{\bar{d}}) = \sum_{j=-N+1+Ab}^{-N+Ab+A-1} \psi(3N(x_{\bar{d}} - \frac{j}{N})), \quad b = 0, 1, \dots, A-1.$$

Notice that j doesn't include values from set \mathcal{G}_0 .

By the construction, $\forall m_{\bar{d}} \in \{-N, \dots, -1, 0, 1, \dots, N\}$, either $m_{\bar{d}} \in \mathcal{G}^1$, or $m_{\bar{d}} \in \mathcal{G}^2$. If $m_{\bar{d}} \in \mathcal{G}^2$, there is a unique $m_{(b,g)}$, such that $m_{\bar{d}} \in [-N + 1 + Ab, -N + Ab + A - 1]$ and $m_{\bar{d}} \in \mathcal{G}_g, g = 2, \dots, A - 2$ and $m_{\bar{d}} = m_{(b,g)}$. Notice that there are $(A - 3)A$ possible values for (g, b) .

Fix a $x_{\bar{d}} \in [-1, 1]$, $\exists m \in \{-N, \dots, -1, 0, 1, \dots, N\}$, such that $x_{\bar{d}} \in [\frac{m}{N}, \frac{m+1}{N}]$. Then, $\forall m' \in \{-N, \dots, -1, 0, 1, \dots, N\}$, such that $m' \neq m, m' \neq m + 1$,

$$\psi(3N(x_{\bar{d}} - \frac{m'}{N})) = 0.$$

If $m \in \mathcal{G}_g$,

$$\psi(3N(x_{\bar{d}} - \frac{m}{N})) = \sum_{i \in \mathcal{G}_g} \psi(3N(x_{\bar{d}} - \frac{i}{N})).$$

If $m + 1 \in \mathcal{G}_g$,

$$\psi(3N(x_{\bar{d}} - \frac{m+1}{N})) = \sum_{i \in \mathcal{G}_g} \psi(3N(x_{\bar{d}} - \frac{i}{N})).$$

If $m, m + 1 \notin \mathcal{G}_g$,

$$\sum_{i \in \mathcal{G}_g} \psi(3N(x_{\bar{d}} - \frac{i}{N})) = 0.$$

There are four possible situations.

First, if $m, m + 1 \in \mathcal{G}^2$, then $\exists b \in \{0, 1, \dots, A - 1\}$, such that $m, m + 1 \in \{-N + 1 + Ab, \dots, N + Ab + A - 1\}$ and

$$\kappa_b(x_{\bar{d}}) = 1,$$

$$\kappa_{b'}(x_{\bar{d}}) = 0, \quad b' \in \{0, 1, \dots, A - 1\}, b' \neq b.$$

Then, $\forall b \in \{0, 1, \dots, A - 1\}$,

$$\begin{aligned} \psi(3N(x_{\bar{d}} - \frac{m_{(b,g)}}{N})) &= \kappa_b(x_{\bar{d}}) \sum_{i \in \mathcal{G}_g} \psi(3N(x_{\bar{d}} - \frac{i}{N})), \quad g = 2, \dots, A - 2, \\ \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})) &= 0, \quad \forall m_{\bar{d}} \in \mathcal{G}^1, \\ \psi(3N(x_{\bar{d}} - \frac{m_{(b,g)}}{N})) &= 0, \quad \forall m_{(b,g)} \in \mathcal{G}^2, m_{(b,g)} \neq m, m + 1. \end{aligned}$$

For the second situation, if $m, m+1 \in \mathcal{G}^1$, then, $\sum_{i \in \mathcal{G}_g} \psi(3N(x_{\bar{d}} - \frac{i}{N})) = 0, g = 2, \dots, A-2$.

$\forall b \in \{0, 1, \dots, A-1\}$,

$$\begin{aligned}\psi(3N(x_{\bar{d}} - \frac{m_{(b,g)}}{N})) &= \kappa_b(x_{\bar{d}}) \sum_{i \in \mathcal{G}_g} \psi(3N(x_{\bar{d}} - \frac{i}{N})) = 0, \quad g = 2, \dots, A-2, \\ \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})) &= \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})), \quad \forall m_{\bar{d}} \in \mathcal{G}^1.\end{aligned}$$

For the third situation, if $m \in \mathcal{G}^2, m+1 \in \mathcal{G}^1$, then $\exists b \in \{0, 1, \dots, A-1\}$, such that $m, m+1 \in \{-N+1+Ab, \dots, -N+Ab+A-1\}$ and

$$\begin{aligned}\kappa_b(x_{\bar{d}}) &= 1, \\ \kappa_{b'}(x_{\bar{d}}) &= 0, \quad b' \in \{0, 1, \dots, A-1\}, b' \neq b.\end{aligned}$$

Then, $\forall b \in \{0, 1, \dots, A-1\}$,

$$\begin{aligned}\psi(3N(x_{\bar{d}} - \frac{m_{(b,g)}}{N})) &= \kappa_b(x_{\bar{d}}) \sum_{i \in \mathcal{G}_g} \psi(3N(x_{\bar{d}} - \frac{i}{N})), \quad g = 2, \dots, A-2, \\ \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})) &= \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})), \quad \forall m_{\bar{d}} \in \mathcal{G}^1, \\ \psi(3N(x_{\bar{d}} - \frac{m_{(b,g)}}{N})) &= 0, \quad \forall m_{(b,g)} \in \mathcal{G}^2, m_{(b,g)} \neq m, \\ \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})) &= 0, \quad \forall m_{\bar{d}} \in \mathcal{G}^1, m_{\bar{d}} \neq m+1.\end{aligned}$$

For the last situation, if $m \in \mathcal{G}^1, m+1 \in \mathcal{G}^2, \forall b \in \{0, 1, \dots, A-1\}$,

$$\begin{aligned}\psi(3N(x_{\bar{d}} - \frac{m_{(b,g)}}{N})) &= \kappa_b(x_{\bar{d}}) \sum_{i \in \mathcal{G}_g} \psi(3N(x_{\bar{d}} - \frac{i}{N})), \quad g = 2, \dots, A-2, \\ \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})) &= \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})), \quad \forall m_{\bar{d}} \in \mathcal{G}^1, \\ \psi(3N(x_{\bar{d}} - \frac{m_{(b,g)}}{N})) &= 0, \quad \forall m_{(b,g)} \in \mathcal{G}^2, m_{(b,g)} \neq m+1, \\ \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})) &= 0, \quad \forall m_{\bar{d}} \in \mathcal{G}^1, m_{\bar{d}} \neq m.\end{aligned}$$

In conclusion, if $m_{\bar{d}} \in \mathcal{G}^1$,

$$\begin{aligned} & \phi_{\mathbf{m}}(\mathbf{x}) \left(\mathbf{x} - \frac{\mathbf{m}}{N} \right)^\alpha \\ &= \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})) \left(x_{\bar{d}} - \frac{m_{\bar{d}}}{N} \right)^{\alpha_{\bar{d}}} \tilde{f}_{\mathbf{m}^1, \alpha^1}(\mathbf{x}^1) \tilde{f}_{\mathbf{m}^2, \alpha^2}(\mathbf{x}^2), \end{aligned}$$

if $m_{\bar{d}} \in \mathcal{G}^2, m_{\bar{d}} = m_{(b,g)}$,

$$\begin{aligned} & \phi_{\mathbf{m}}(\mathbf{x}) \left(\mathbf{x} - \frac{\mathbf{m}}{N} \right)^\alpha \\ &= \psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})) \left(x_{\bar{d}} - \frac{m_{\bar{d}}}{N} \right)^{\alpha_{\bar{d}}} \tilde{f}_{\mathbf{m}^1, \alpha^1}(\mathbf{x}^1) \tilde{f}_{\mathbf{m}^2, \alpha^2}(\mathbf{x}^2) \\ &= \left(\sum_{i \in \mathcal{G}_g} \psi(3N(x_{\bar{d}} - \frac{i}{N})) \left(x_{\bar{d}} - \frac{i}{N} \right)^{\alpha_{\bar{d}}} \tilde{f}_{\mathbf{m}^1, \alpha^1}(\mathbf{x}^1) \right) \left(\tilde{f}_{\mathbf{m}^2, \alpha^2}(\mathbf{x}^2) \kappa_b(x_{\bar{d}}) \right), \end{aligned}$$

Adjusting the definition,

$$\begin{aligned} \tilde{f}_{\mathbf{m}^1, \alpha^1, g, \alpha_{\bar{d}}}(\mathbf{x}^1, x_{\bar{d}}) &:= \tilde{\times} \left(\sum_{i \in \mathcal{G}_g} \tilde{\times}(\psi(3N(x_{\bar{d}} - \frac{i}{N})), \tilde{\times}(x_{\bar{d}} - \frac{i}{N}, \dots)), \tilde{f}_{\mathbf{m}^1, \alpha^1}(\mathbf{x}^1) \right), \\ \tilde{f}_{\mathbf{m}^2, \alpha^2, b}(\mathbf{x}^2, x_{\bar{d}}) &:= \tilde{\times} \left(\tilde{f}_{\mathbf{m}^2, \alpha^2}(\mathbf{x}^2), \kappa_b(x_{\bar{d}}) \right). \end{aligned}$$

For each neural network, with at most $(d-1)^\beta (2N+1)^{\frac{d-1}{2}} \lceil \sqrt{2N+1} \rceil$ output units, depth $\mathcal{O}(\ln(1/\delta) + 1)$ and width $\mathcal{O}((2N+1)^{\frac{d}{2}})$, by Lemma 7, we can use a neural network to represent all possible values.

For each (\mathbf{m}^2, α^2) and (\mathbf{m}^1, α^1) , if $m_{\bar{d}} \in \mathcal{G}^2$, let $a_{\mathbf{m}, \alpha, (b,g), \alpha_{\bar{d}}}$ denote $a_{\mathbf{m}, \alpha}$ in the case $m_{\bar{d}} = m_{(b,g)}$; if $m_{\bar{d}} \in \mathcal{G}^1$, let $a_{\mathbf{m}, \alpha}$ denote $a_{\mathbf{m}, \alpha}$. We set $a_{\mathbf{m}, \alpha, (b,g), \alpha_{\bar{d}}} = 0$ if it doesn't satisfy the definition. We can rewrite $f_1(\mathbf{x})$ as

$$\begin{aligned} f_1(\mathbf{x}) &= \sum_{\mathbf{m}^2, \alpha^2, b} \left(\sum_{\mathbf{m}^1, \alpha^1, g \in \mathcal{G}^2, \alpha_{\bar{d}}} a_{\mathbf{m}^1, \alpha^1, (b,g), \alpha_{\bar{d}}} \tilde{f}_{\mathbf{m}^1, \alpha^1, g, \alpha_{\bar{d}}}(\mathbf{x}^1, x_{\bar{d}}) \right. \\ &\quad \times \tilde{f}_{\mathbf{m}^2, \alpha^2, b}(\mathbf{x}^2, x_{\bar{d}}) \\ &+ \sum_{\mathbf{m}^2, \alpha^2} \left(\sum_{\mathbf{m}^1, \alpha^1, m_{\bar{d}} \in \mathcal{G}^1, \alpha_{\bar{d}}} a_{\mathbf{m}, \alpha} \left(x_{\bar{d}} - \frac{m_{\bar{d}}}{N} \right)^{\alpha_{\bar{d}}} \psi \left(3N \left(x_{\bar{d}} - \frac{m_{\bar{d}}}{N} \right) \right) f_{\mathbf{m}^1, \alpha^1}(\mathbf{x}^1) \right. \\ &\quad \times \tilde{f}_{\mathbf{m}^2, \alpha^2}(\mathbf{x}^2). \end{aligned}$$

Define the full approximation by the following:

$$\begin{aligned}
\tilde{f} = & 2^{\frac{d}{2}} d^\beta \sum_{\mathbf{m}^2, \alpha^2, b} \tilde{\times} \left(\frac{1}{2^{\frac{d}{2}} d^\beta} \sum_{\mathbf{m}^1, \alpha^1, g \in \mathcal{G}^2, \alpha_{\bar{d}}} a_{\mathbf{m}^1, \alpha^1, (b, g), \alpha_{\bar{d}}} \tilde{f}_{\mathbf{m}^1, \alpha^1, g, \alpha_{\bar{d}}}(\mathbf{x}^1, x_{\bar{d}}), \right. \\
& \quad \left. \tilde{f}_{\mathbf{m}^2, \alpha^2, b}(\mathbf{x}^2, x_{\bar{d}}) \right) \\
& + 2^{\frac{d}{2}} d^\beta \sum_{\mathbf{m}^2, \alpha^2} \tilde{\times} \left(\tilde{f}_{\mathbf{m}^2, \alpha^2}(\mathbf{x}^2), \right. \\
& \quad \left. \frac{1}{2^{\frac{d}{2}} d^\beta} \sum_{\mathbf{m}^1, \alpha^1, m_{\bar{d}} \in \mathcal{G}^1, \alpha_{\bar{d}}} a_{\mathbf{m}, \alpha} \tilde{\times} \left(\tilde{f}_{\mathbf{m}^1, \alpha^1}(\mathbf{x}^1), \tilde{\times} \left(\psi(3N(x_{\bar{d}} - \frac{m_{\bar{d}}}{N})), \tilde{\times}(x_{\bar{d}} - \frac{m_{\bar{d}}}{N}, \dots) \right) \right) \right).
\end{aligned}$$

For the neural network in the first and second lines, the depth is $\mathcal{O}(\ln(1/\delta) + 1)$ and width is $\mathcal{O}((2N+1)^{\frac{d}{2}})$; since \mathcal{G}^1 has at most $3\lceil\sqrt{2N+1}\rceil$ elements, we need neural networks in the third and fourth lines to have depth $\mathcal{O}(\ln(1/\delta) + 1)$ and width $\mathcal{O}((2N+1)^{\frac{d}{2}})$. To construct the first line's neural network, we need one more layer of composition $\tilde{\times}(\cdot)$. Thus, we must adjust the term $d + \beta - 1$ in Appendix B.1 into $d + \beta$. That is, we choose $\delta = \frac{\epsilon}{2^{d+1} d^\beta (d+\beta) \delta}$. The rest of the proof remains the same as in Appendix B.1.

Case 3: if $d = 1$, we need a special treatment.¹⁰ The method used above requires dividing $[-1, 1]$ into N grids and N different functions of $\psi(x)$, the width is $\mathcal{O}(N)$. Since $d \geq 2$, the largest width is $\mathcal{O}(N^{\frac{d}{2}})$, this approach is applicable to the previous cases. However, when $d = 1$, we need the largest width to be $\mathcal{O}(\sqrt{N})$. We will follow the notation used above.

First, notice that the function $\kappa_b(x)$ requires a width $\mathcal{O}(N)$; we first modify this function.

Define a new function, $\forall b = 0, 1, \dots, A-1$,

$$\Psi_b(x) = \begin{cases} 1, & x \in [-N + Ab + 1, -N + Ab + A - 1], \\ \psi(3N(x - \frac{-N+Ab+1}{N})), & x \in [-N + Ab, -N + Ab + 1], \\ \psi(3N(x - \frac{-N+Ab+A-1}{N})), & x \in [-N + Ab + A - 1, -N + Ab + A], \\ 0, & \text{otherwise.} \end{cases}$$

This function can be directly represented by a ReLU neural network with finite width and depth.

¹⁰Although there is no theoretical advantage in using deep neural networks when $d = 1$, we include a brief proof for completeness.

Notice that

$$\Psi_b(x) = \kappa_b(x).$$

To represent all possible values of $\Psi_b(x_{\bar{d}})$, we only need a neural network with finite depth and width $O(\sqrt{N})$.

$\forall m' \in \mathcal{G}^2, \exists m \in [-N+2, -N+A-2], \exists b = 0, 1, \dots, A-1$, such that $m' = m + b$. We denote $a_{m',\alpha}$ as $a_{m,b,\alpha}$. We notice that $f_1(x)$ can be written as:

$$\begin{aligned} f_1(x) &= \sum_b \sum_\alpha \sum_{m=-N+2}^{-N+A-2} a_{m,b,\alpha} \Psi_b(x) \cdot \psi(x - \frac{m+bA}{N})(x - \frac{m+bA}{N})^\alpha \\ &\quad + \sum_\alpha \sum_{m \in \mathcal{G}^1} a_{m,\alpha} \psi(x - \frac{m}{N})(x - \frac{m}{N})^\alpha \\ &= \sum_b \Psi_b(x) \cdot \sum_{m=-N+2}^{-N+A-2} \sum_\alpha a_{m,b,\alpha} \psi\left(-\frac{m}{N} + \sum_{b'} \Psi_{b'}(x) \cdot (x - \frac{b'A}{N})\right) \\ &\quad \cdot \left(-\frac{m}{N} + \sum_{b'} \Psi_{b'}(x) \cdot (x - \frac{b'A}{N})\right)^\alpha \\ &\quad + \sum_\alpha \sum_{m \in \mathcal{G}^1} a_{m,\alpha} \psi(x - \frac{m}{N})(x - \frac{m}{N})^\alpha \end{aligned}$$

Define the full approximation by the following:

$$\begin{aligned} \tilde{f} &= 2d^\beta \sum_b \tilde{\times}\left(\Psi_b(x), \right. \\ &\quad \left. \frac{1}{2d^\beta} \sum_{m=-N+2}^{-N+A-2} \sum_\alpha a_{m,b,\alpha} \tilde{\times}\left(\psi\left(-\frac{m}{N} + \sum_{b'} \tilde{\times}\left(\Psi_{b'}(x), x - \frac{b'A}{N}\right)\right), \right. \right. \\ &\quad \left. \left. \tilde{\times}\left(-\frac{m}{N} + \sum_{b'} \tilde{\times}\left(\Psi_{b'}(x), x - \frac{b'A}{N}\right), \dots\right)\right)\right) \\ &\quad + \sum_\alpha \sum_{m \in \mathcal{G}^1} a_{m,\alpha} \tilde{\times}\left(\Psi_b(x), \tilde{\times}\left(\psi(x - \frac{m}{N}), \tilde{\times}(x - \frac{m}{N}, \dots)\right)\right) \end{aligned}$$

The width needed for this approximation is at most $O(\sqrt{N})$ and depth $O(\ln(1/\delta) + 1)$. The rest of the proof remains the same as above.

In conclusion, there is a ReLU network that has depth $c(\ln(1/\epsilon) + 1)$, width at most $c\epsilon^{-\frac{d}{2\beta}}$, and weights at most $c\epsilon^{-\frac{d}{\beta}}(\ln(1/\epsilon) + 1)$, with some constant $c = c(d, \beta)$. Lemma 8 shows that

this network can be fully connected with the same level of depth, width, and weights. The only difference is the constant term c .

□

References

- Bauer, B. and Kohler, M. (2019). On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *The Annals of Statistics*.
- Brown, C. (2024). Statistical properties of deep neural networks with dependent data. *arXiv preprint arXiv:2410.11113*.
- Chronopoulos, I., Chrysikou, K., Kapetanios, G., Mitchell, J., and Raftapostolos, A. (2023). Deep neural network estimation in panel data models. *arXiv preprint arXiv:2305.19921*.
- Colangelo, K. and Lee, Y.-Y. (2025). Double debiased machine learning nonparametric inference with continuous treatments. *Journal of Business & Economic Statistics*, (just-accepted):1–26.
- Farrell, M. H., Liang, T., and Misra, S. (2021). Deep neural networks for estimation and inference. *Econometrica*, 89(1):181–213.
- Feng, X., Jiao, Y., Kang, L., Zhang, B., and Zhou, F. (2023). Over-parameterized deep nonparametric regression for dependent data with its applications to reinforcement learning. *Journal of Machine Learning Research*, 24(383):1–40.
- Jiao, Y., Kang, L., Liu, J., Lu, X., and Yang, J. Z. (2025). Deep approximate policy iteration. *The Annals of Statistics*, 53(2):802–821.
- Kohler, M., Krzyżak, A., and Langer, S. (2022). Estimation of a function of low local dimensionality by deep neural networks. *IEEE transactions on information theory*, 68(6):4032–4042.
- Liu, R., Boukai, B., and Shang, Z. (2022). Optimal nonparametric inference via deep neural network. *Journal of Mathematical Analysis and Applications*, 505(2):125561.
- Milnor, J. W. and Weaver, D. W. (1997). *Topology from the differentiable viewpoint*, volume 21. Princeton university press.
- Petersen, P. and Zech, J. (2024). Mathematical theory of deep learning. *arXiv preprint arXiv:2407.18384*.

- Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. (2017). Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519.
- Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with relu activation function. *The Annals of Statistics*.
- Shaham, U., Cloninger, A., and Coifman, R. R. (2018). Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, 44(3):537–557.
- Shen, G., Jiao, Y., Lin, Y., Horowitz, J. L., and Huang, J. (2021). Deep quantile regression: Mitigating the curse of dimensionality through composition. *arXiv preprint arXiv:2107.04907*.
- Stone, C. J. (1982). Optimal global rates of convergence for nonparametric regression. *The annals of statistics*, pages 1040–1053.
- Stone, C. J. (1994). The use of polynomial splines and their tensor products in multivariate function estimation. *The annals of statistics*, pages 118–171.
- Yarotsky, D. (2017). Error bounds for approximations with deep relu networks. *Neural networks*, 94:103–114.
- Zhang, Y. and Bradic, J. (2024). Causal inference through multi-stage learning and doubly robust deep neural networks. *arXiv preprint arXiv:2407.08560*.
- Zhang, Z., Shi, L., and Zhou, D.-X. (2024). Classification with deep neural networks and logistic loss. *Journal of Machine Learning Research*, 25(125):1–117.