Chapter 5

ReLU neural networks

In this section, we discuss feedforward neural networks using the ReLU activation function σ_{ReLU} introduced in Section 2.3. We refer to these functions as ReLU neural networks. Due to its simplicity and the fact that it reduces the vanishing and exploding gradients phenomena, it is one of the most widely used in practice.

A key component of the proofs in the previous chapters was the approximation of derivatives of the activation function to emulate polynomials. Since the ReLU is piecewise linear, this trick is not applicable. This makes the analysis fundamentally different from the case of smoother activation functions. Nonetheless, we will see that even this extremely simple activation function yields a very rich class of functions possessing remarkable approximation capabilities.

To formalize these results, we begin this chapter by adopting a framework from [172]. This framework enables the tracking of the number of network parameters for basic manipulations such as adding up or composing two neural networks. This will allow to bound the network complexity, when constructing more elaborate networks from simpler ones. With these preliminaries at hand, the rest of the chapter is dedicated to the exploration of links between ReLU neural networks and the class of "continuous piecewise linear functions." In Section 5.2, we will see that every such function can be exactly represented by a ReLU neural network. Afterwards, in Section 5.3 we will give a more detailed analysis of the required network complexity. Finally, we will use these results to prove a first approximation theorem for ReLU neural networks in Section 5.4. The argument is similar in spirit to Chapter 4, in that we *transfer* established approximation theory for piecewise linear functions to the class of ReLU neural networks of a certain architecture.

5.1 Basic ReLU calculus

The goal of this section is to formalize how to combine and manipulate ReLU neural networks. We have seen an instance of such a result already in Proposition 2.3. Now we want to make this result more precise under the assumption that the activation function is the ReLU. We sharpen Proposition 2.3 by adding bounds on the number of weights that the resulting neural networks have. The following four operations form the basis of all constructions in the sequel.

- *Reproducing an identity:* We have seen in Proposition 3.16 that for most activation functions, an approximation to the identity can be built by neural networks. For ReLUs, we can have an even stronger result and reproduce the identity exactly. This identity will play a crucial

role in order to extend certain neural networks to deeper neural networks, and to facilitate an efficient composition operation.

- *Composition:* We saw in Proposition 2.3 that we can produce a composition of two neural networks and the resulting function is a neural network as well. There we did not study the size of the resulting neural networks. For ReLU activation functions, this composition can be done in a very efficient way leading to a neural network that has up to a constant not more than the number of weights of the two initial neural networks.
- *Parallelization:* Also the parallelization of two neural networks was discussed in Proposition 2.3. We will refine this notion and make precise the size of the resulting neural networks.
- *Linear combinations:* Similarly, for the sum of two neural networks, we will give precise bounds on the size of the resulting neural network.

5.1.1 Identity

We start with expressing the identity on $\mathbb{R}^d$ as a neural network of depth $L \in \mathbb{N}$.

Lemma 5.1 (Identity)**.** *Let $L \in \mathbb{N}$. Then, there exists a ReLU neural network Φ_L^{id} such that $\Phi_L^{\mathrm{id}}(\boldsymbol{x}) = \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^d$. Moreover,* $\mathrm{depth}(\Phi_L^{\mathrm{id}}) = L$, $\mathrm{width}(\Phi_L^{\mathrm{id}}) = 2d$, *and* $\mathrm{size}(\Phi_L^{\mathrm{id}}) = 2d \cdot (L+1)$.

Proof. Writing $\boldsymbol{I}_d \in \mathbb{R}^{d \times d}$ for the identity matrix, we choose the weights

$$\begin{aligned}&(\boldsymbol{W}^{(0)}, \boldsymbol{b}^{(0)}), \dots, (\boldsymbol{W}^{(L)}, \boldsymbol{b}^{(L)}) \\ &\quad := \left(\begin{pmatrix} \boldsymbol{I}_d \\ -\boldsymbol{I}_d \end{pmatrix}, 0 \right), \underbrace{(\boldsymbol{I}_{2d}, 0), \dots, (\boldsymbol{I}_{2d}, 0)}_{L-1 \text{ times}}, ((\boldsymbol{I}_d, -\boldsymbol{I}_d), 0).\end{aligned}$$

Using that $x = \sigma_{\mathrm{ReLU}}(x) - \sigma_{\mathrm{ReLU}}(-x)$ for all $x \in \mathbb{R}$ and $\sigma_{\mathrm{ReLU}}(x) = x$ for all $x \geq 0$ it is obvious that the neural network Φ_L^{id} associated to the weights above satisfies the assertion of the lemma. □

We will see in Exercise 5.23 that the property to exactly represent the identity is not shared by sigmoidal activation functions. It does hold for polynomial activation functions, though.

5.1.2 Composition

Assume we have two neural networks Φ_1, Φ_2 with architectures $(\sigma_{\mathrm{ReLU}}; d_0^1, \dots, d_{L_1+1}^1)$ and $(\sigma_{\mathrm{ReLU}}; d_0^2, \dots, d_{L_1+1}^2)$ respectively. Moreover, we assume that they have weights and biases given by

$$(\boldsymbol{W}_1^{(0)}, \boldsymbol{b}_1^{(0)}), \dots, (\boldsymbol{W}_1^{(L_1)}, \boldsymbol{b}_1^{(L_1)}), \text{ and } (\boldsymbol{W}_2^{(0)}, \boldsymbol{b}_2^{(0)}), \dots, (\boldsymbol{W}_2^{(L_2)}, \boldsymbol{b}_2^{(L_2)}),$$

respectively. If the output dimension $d_{L_1+1}^1$ of Φ_1 equals the input dimension d_0^2 of Φ_2, we can define two types of concatenations: First $\Phi_2 \circ \Phi_1$ is the neural network with weights and biases given by

$$\begin{aligned}&\left(\boldsymbol{W}_1^{(0)}, \boldsymbol{b}_1^{(0)}\right), \dots, \left(\boldsymbol{W}_1^{(L_1-1)}, \boldsymbol{b}_1^{(L_1-1)}\right), \left(\boldsymbol{W}_2^{(0)} \boldsymbol{W}_1^{(L_1)}, \boldsymbol{W}_2^{(0)} \boldsymbol{b}_1^{(L_1)} + \boldsymbol{b}_2^{(0)}\right), \\ &\quad \left(\boldsymbol{W}_2^{(1)}, \boldsymbol{b}_2^{(1)}\right), \dots, \left(\boldsymbol{W}_2^{(L_2)}, \boldsymbol{b}_2^{(L_2)}\right).\end{aligned}$$

Second, $\Phi_2 \bullet \Phi_1$ is the neural network defined as $\Phi_2 \circ \Phi_1^{\mathrm{id}} \circ \Phi_1$. In terms of weighs and biases, $\Phi_2 \bullet \Phi_1$ is given as

$$\begin{aligned}&\left(\boldsymbol{W}_1^{(0)}, \boldsymbol{b}_1^{(0)}\right), \ldots, \left(\boldsymbol{W}_1^{(L_1-1)}, \boldsymbol{b}_1^{(L_2-1)}\right), \left(\begin{pmatrix} \boldsymbol{W}_1^{(L_1)} \\ -\boldsymbol{W}_1^{(L_1)} \end{pmatrix}, \begin{pmatrix} \boldsymbol{b}_1^{(L_1)} \\ -\boldsymbol{b}_1^{(L_1)} \end{pmatrix}\right), \\ &\left(\left(\boldsymbol{W}_2^{(0)}, -\boldsymbol{W}_2^{(0)}\right), \boldsymbol{b}_2^{(0)}\right), \left(\boldsymbol{W}_2^{(1)}, \boldsymbol{b}_2^{(1)}\right), \ldots, \left(\boldsymbol{W}_2^{(L_2)}, \boldsymbol{b}_2^{(L_2)}\right).\end{aligned}$$

The following lemma collects the properties of the construction above.

Lemma 5.2 (Composition)**.** *Let Φ_1, Φ_2 be neural networks with architectures $(\sigma_{\mathrm{ReLU}}; d_0^1, \ldots, d_{L_1+1}^1)$ and $(\sigma_{\mathrm{ReLU}}; d_0^2, \ldots, d_{L_2+1}^2)$. Assume $d_{L_1+1}^1 = d_0^2$. Then $\Phi_2 \circ \Phi_1(\boldsymbol{x}) = \Phi_2 \bullet \Phi_1(\boldsymbol{x}) = \Phi_2(\Phi_1(\boldsymbol{x}))$ for all $\boldsymbol{x} \in \mathbb{R}^{d_1^0}$. Moreover,*

$$\begin{aligned}\mathrm{width}(\Phi_2 \circ \Phi_1) &\leq \max\{\mathrm{width}(\Phi_1), \mathrm{width}(\Phi_2)\}, \\ \mathrm{depth}(\Phi_2 \circ \Phi_1) &= \mathrm{depth}(\Phi_1) + \mathrm{depth}(\Phi_2), \\ \mathrm{size}(\Phi_2 \circ \Phi_1) &\leq \mathrm{size}(\Phi_1) + \mathrm{size}(\Phi_2) + (d_{L_1}^1 + 1) d_2^1,\end{aligned}$$

and

$$\begin{aligned}\mathrm{width}(\Phi_2 \bullet \Phi_1) &\leq 2\max\{\mathrm{width}(\Phi_1), \mathrm{width}(\Phi_2)\}, \\ \mathrm{depth}(\Phi_2 \bullet \Phi_1) &= \mathrm{depth}(\Phi_1) + \mathrm{depth}(\Phi_2) + 1, \\ \mathrm{size}(\Phi_2 \bullet \Phi_1) &\leq 2(\mathrm{size}(\Phi_1) + \mathrm{size}(\Phi_2)).\end{aligned}$$

Proof. The fact that $\Phi_2 \circ \Phi_1(\boldsymbol{x}) = \Phi_2 \bullet \Phi_1(\boldsymbol{x}) = \Phi_2(\Phi_1(\boldsymbol{x}))$ for all $\boldsymbol{x} \in \mathbb{R}^{d_0^1}$ follows immediately from the construction. The same can be said for the width and depth bounds. To confirm the size bound, we note that $\boldsymbol{W}_2^{(0)} \boldsymbol{W}_1^{(L_1)} \in \mathbb{R}^{d_1^2 \times d_{L_1}^1}$ and hence $\boldsymbol{W}_2^{(0)} \boldsymbol{W}_1^{(L_1)}$ has not more than $d_1^2 \times d_{L_1}^1$ (nonzero) entries. Moreover, $\boldsymbol{W}_2^{(0)} \boldsymbol{b}_1^{(L_1)} + \boldsymbol{b}_2^{(0)} \in \mathbb{R}^{d_1^2}$. Thus, the L_1-th layer of $\Phi_2 \circ \Phi_1(\boldsymbol{x})$ has at most $d_1^2 \times (1 + d_{L_1}^1)$ entries. The rest is obvious from the construction. □

Interpreting linear transformations as neural networks of depth 0, the previous lemma is also valid in case Φ_1 or Φ_2 is a linear mapping.

5.1.3 Parallelization

Next, we wish to put neural networks in parallel. Let $(\Phi_i)_{i=1}^m$ be neural networks with architectures $(\sigma_{\mathrm{ReLU}}; d_0^i, \ldots, d_{L_i+1}^i)$, respectively. We proceed to build a neural network $(\Phi_1, \ldots, \Phi_m)$ such that

$$\begin{aligned}(\Phi_1, \ldots, \Phi_m) \colon \mathbb{R}^{\sum_{j=1}^m d_0^j} &\to \mathbb{R}^{\sum_{j=1}^m d_{L_j+1}^j} \\ (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) &\mapsto (\Phi_1(\boldsymbol{x}_1), \ldots, \Phi_m(\boldsymbol{x}_m)).\end{aligned} \tag{5.1.1}$$

To do so we first assume $L_1 = \cdots = L_m = L$, and define $(\Phi_1, \dots, \Phi_m)$ via the following sequence of weight-bias tuples:

$$\left(\begin{pmatrix} \boldsymbol{W}_1^{(0)} & & \\ & \ddots & \\ & & \boldsymbol{W}_m^{(0)} \end{pmatrix}, \begin{pmatrix} \boldsymbol{b}_1^{(0)} \\ \vdots \\ \boldsymbol{b}_m^{(0)} \end{pmatrix}\right), \dots, \left(\begin{pmatrix} \boldsymbol{W}_1^{(L)} & & \\ & \ddots & \\ & & \boldsymbol{W}_m^{(L)} \end{pmatrix}, \begin{pmatrix} \boldsymbol{b}_1^{(L)} \\ \vdots \\ \boldsymbol{b}_m^{(L)} \end{pmatrix}\right) \tag{5.1.2}$$

where these matrices are understood as block-diagonal filled up with zeros. For the general case where the Φ_j might have different depths, let $L_{\max} := \max_{1 \leq i \leq m} L_i$ and $I := \{1 \leq i \leq m \,|\, L_i < L_{\max}\}$. For $j \in I^c$ set $\widetilde{\Phi}_j := \Phi_j$, and for each $j \in I$

$$\widetilde{\Phi}_j := \Phi^{\mathrm{id}}_{L_{\max} - L_j} \circ \Phi_j. \tag{5.1.3}$$

Finally,

$$(\Phi_1, \dots, \Phi_m) := (\widetilde{\Phi}_1, \dots, \widetilde{\Phi}_m). \tag{5.1.4}$$

We collect the properties of the parallelization in the lemma below.

Lemma 5.3 (Parallelization)**.** *Let $m \in \mathbb{N}$ and $(\Phi_i)_{i=1}^m$ be neural networks with architectures $(\sigma_{\mathrm{ReLU}}; d_0^i, \dots, d_{L_i+1}^i)$, respectively. Then the neural network $(\Phi_1, \dots, \Phi_m)$ satisfies*

$$(\Phi_1, \dots, \Phi_m)(\boldsymbol{x}) = (\Phi_1(\boldsymbol{x}_1), \dots, \Phi_m(\boldsymbol{x}_m)) \text{ for all } \boldsymbol{x} \in \mathbb{R}^{\sum_{j=1}^m d_0^j}.$$

Moreover, with $L_{\max} := \max_{j \leq m} L_j$ it holds that

$$\mathrm{width}((\Phi_1, \dots, \Phi_m)) \leq 2 \sum_{j=1}^m \mathrm{width}(\Phi_j), \tag{5.1.5a}$$

$$\mathrm{depth}((\Phi_1, \dots, \Phi_m)) = \max_{j \leq m} \mathrm{depth}(\Phi_j), \tag{5.1.5b}$$

$$\mathrm{size}((\Phi_1, \dots, \Phi_m)) \leq 2 \sum_{j=1}^m \mathrm{size}(\Phi_j) + 2 \sum_{j=1}^m (L_{\max} - L_j) d_{L_j+1}^j. \tag{5.1.5c}$$

Proof. All statements except for the bound on the size follow immediately from the construction. To obtain the bound on the size, we note that by construction the sizes of the $(\widetilde{\Phi}_i)_{i=1}^m$ in (5.1.3) will simply be added. The size of each $\widetilde{\Phi}_i$ can be bounded with Lemma 5.2. □

If all input dimensions $d_0^1 = \cdots = d_0^m =: d_0$ are the same, we will also use **parallelization with shared inputs** to realize the function $\boldsymbol{x} \mapsto (\Phi_1(\boldsymbol{x}), \dots, \Phi_m(\boldsymbol{x}))$ from $\mathbb{R}^{d_0} \to \mathbb{R}^{d^1_{L_1+1} + \cdots + d^m_{L_m+1}}$. In terms of the construction (5.1.2), the only required change is that the block-diagonal matrix $\mathrm{diag}(\boldsymbol{W}_1^{(0)}, \dots, \boldsymbol{W}_m^{(0)})$ becomes the matrix in $\mathbb{R}^{\sum_{j=1}^m d_1^j \times d_0^1}$ which stacks $\boldsymbol{W}_1^{(0)}, \dots, \boldsymbol{W}_m^{(0)}$ on top of each other. Similarly, we will allow Φ_j to only take some of the entries of $\boldsymbol{x}$ as input. For parallelization with shared inputs we will use the same notation $(\Phi_j)_{j=1}^m$ as before, where the precise meaning will always be clear from context. Note that Lemma 5.3 remains valid in this case.

5.1.4 Linear combinations

Let $m \in \mathbb{N}$ and let $(\Phi_i)_{i=1}^m$ be ReLU neural networks that have architectures $(\sigma_{\mathrm{ReLU}}; d_0^i, \dots, d_{L_i+1}^i)$, respectively. Assume that $d_{L_1+1}^1 = \cdots = d_{L_m+1}^m$, i.e., all $\Phi_1, \dots, \Phi_m$ have the same output dimension. For scalars $\alpha_j \in \mathbb{R}$, we wish to construct a ReLU neural network $\sum_{j=1}^m \alpha_j \Phi_j$ realizing the function

$$\begin{cases} \mathbb{R}^{\sum_{j=1}^m d_0^j} \to \mathbb{R}^{d_{L_1+1}^1} \\ (\boldsymbol{x}_1, \dots, \boldsymbol{x}_m) \mapsto \sum_{j=1}^m \alpha_j \Phi_j(\boldsymbol{x}_j). \end{cases}$$

This corresponds to the parallelization $(\Phi_1, \dots, \Phi_m)$ composed with the linear transformation $(\boldsymbol{z}_1, \dots, \boldsymbol{z}_m) \mapsto \sum_{j=1}^m \alpha_j \boldsymbol{z}_j$. The following result holds.

Lemma 5.4 (Linear combinations)**.** *Let $m \in \mathbb{N}$ and $(\Phi_i)_{i=1}^m$ be neural networks with architectures $(\sigma_{\mathrm{ReLU}}; d_0^i, \dots, d_{L_i+1}^i)$, respectively. Assume that $d_{L_1+1}^1 = \cdots = d_{L_m+1}^m$, let $\alpha \in \mathbb{R}^m$ and set $L_{\max} := \max_{j \le m} L_j$. Then, there exists a neural network $\sum_{j=1}^m \alpha_j \Phi_j$ such that $(\sum_{j=1}^m \alpha_j \Phi_j)(\boldsymbol{x}) = \sum_{j=1}^m \alpha_j \Phi_j(\boldsymbol{x}_j)$ for all $\boldsymbol{x} = (\boldsymbol{x}_j)_{j=1}^m \in \mathbb{R}^{\sum_{j=1}^m d_0^j}$. Moreover,*

$$\operatorname{width}\left(\sum_{j=1}^m \alpha_j \Phi_j\right) \le 2 \sum_{j=1}^m \operatorname{width}(\Phi_j), \tag{5.1.6a}$$

$$\operatorname{depth}\left(\sum_{j=1}^m \alpha_j \Phi_j\right) = \max_{j \le m} \operatorname{depth}(\Phi_j), \tag{5.1.6b}$$

$$\operatorname{size}\left(\sum_{j=1}^m \alpha_j \Phi_j\right) \le 2 \sum_{j=1}^m \operatorname{size}(\Phi_j) + 2 \sum_{j=1}^m (L_{\max} - L_j) d_{L_j+1}^j. \tag{5.1.6c}$$

Proof. The construction of $\sum_{j=1}^m \alpha_j \Phi_j$ is analogous to that of $(\Phi_1, \dots, \Phi_m)$, i.e., we first define the linear combination of neural networks with the same depth. Then the weights are chosen as in (5.1.2), but with the last linear transformation replaced by

$$\left((\alpha_1 \boldsymbol{W}_1^{(L)} \cdots \alpha_m \boldsymbol{W}_m^{(L)}), \sum_{j=1}^m \alpha_j \boldsymbol{b}_j^{(L)} \right).$$

For general depths, we define the sum of the neural networks to be the sum of the extended neural networks $\widetilde{\Phi}_i$ as of (5.1.3). All statements of the lemma follow immediately from this construction. □

In case $d_0^1 = \cdots = d_0^m =: d_0$ (all neural networks have the same input dimension), we will also consider **linear combinations with shared inputs**, i.e., a neural network realizing

$$\boldsymbol{x} \mapsto \sum_{j=1}^m \alpha_j \Phi_j(\boldsymbol{x}) \qquad \text{for } \boldsymbol{x} \in \mathbb{R}^{d_0}.$$

This requires the same minor adjustment as discussed at the end of Section 5.1.3. Lemma 5.4 remains valid in this case and again we do not distinguish in notation for linear combinations with or without shared inputs.

5.2 Continuous piecewise linear functions

In this section, we will relate ReLU neural networks to a large class of functions. We first formally introduce the set of continuous piecewise linear functions from a set $\Omega \subseteq \mathbb{R}^d$ to $\mathbb{R}$. Note that we admit in particular $\Omega = \mathbb{R}^d$ in the following definition.

Definition 5.5. Let $\Omega \subseteq \mathbb{R}^d$, $d \in \mathbb{N}$. We call a function $f : \Omega \to \mathbb{R}$ **continuous, piecewise linear (cpwl)** if $f \in C^0(\Omega)$ and there exist $n \in \mathbb{N}$ affine functions $g_j \colon \mathbb{R}^d \to \mathbb{R}$, $g_j(\boldsymbol{x}) = \boldsymbol{w}_j^\top \boldsymbol{x} + b_j$ such that for each $\boldsymbol{x} \in \Omega$ it holds that $f(\boldsymbol{x}) = g_j(\boldsymbol{x})$ for at least one $j \in \{1, \dots, n\}$. For $m > 1$ we call $f : \Omega \to \mathbb{R}^m$ cpwl if and only if each component of f is cpwl.

Remark 5.6. A "continuous piecewise linear function" as in Definition 5.5 is actually piecewise *affine*. To maintain consistency with the literature, we use the terminology cpwl.

In the following, we will refer to the connected domains on which f is equal to one of the functions g_j, also as **regions** or **pieces**. If f is cpwl with $q \in \mathbb{N}$ regions, then with $n \in \mathbb{N}$ denoting the number of affine functions it holds $n \leq q$.

Note that, the mapping $\boldsymbol{x} \mapsto \sigma_{\mathrm{ReLU}}(\boldsymbol{w}^\top \boldsymbol{x} + b)$, which is a ReLU neural network with a single neuron, is cpwl (with two regions). Consequently, every ReLU neural network is a repeated composition of linear combinations of cpwl functions. It is not hard to see that the set of cpwl functions is closed under compositions and linear combinations. Hence, *every ReLU neural network is a cpwl function.* Interestingly, the reverse direction of this statement is also true, meaning that *every cpwl function can be represented by a ReLU neural network* as we shall demonstrate below. Therefore, we can identify the class of functions realized by arbitrary ReLU neural networks as the class of cpwl functions.

Theorem 5.7. *Let $d \in \mathbb{N}$, let $\Omega \subseteq \mathbb{R}^d$ be convex, and let $f : \Omega \to \mathbb{R}$ be cpwl with $n \in \mathbb{N}$ as in Definition 5.5. Then, there exists a ReLU neural network Φ^f such that $\Phi^f(\boldsymbol{x}) = f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$ and*

$$\mathrm{size}(\Phi^f) = O(dn2^n), \quad \mathrm{width}(\Phi^f) = O(dn2^n), \quad \mathrm{depth}(\Phi^f) = O(n).$$

A statement similar to Theorem 5.7 can be found in [4, 84]. There, the authors give a construction with a depth that behaves logarithmic in d and is independent of n, but with significantly larger bounds on the size. As we shall see, the proof of Theorem 5.7 is a simple consequence of the following well-known result from [224]; also see [167, 236]. It states that every cpwl function can be expressed as a finite maximum of a finite minimum of certain affine functions.

Proposition 5.8. *Let $d \in \mathbb{N}$, $\Omega \subseteq \mathbb{R}^d$ be convex, and let $f : \Omega \to \mathbb{R}$ be cpwl with $n \in \mathbb{N}$ affine functions as in Definition 5.5. Then there exists $m \in \mathbb{N}$ and sets $s_j \subseteq \{1, \dots, n\}$ for $j \in \{1, \dots, m\}$, such that*

$$f(\boldsymbol{x}) = \max_{1 \le j \le m} \min_{i \in s_j} (g_i(\boldsymbol{x})) \qquad \text{for all } \boldsymbol{x} \in \Omega. \tag{5.2.1}$$

Proof. **Step 1.** We start with $d = 1$, i.e., $\Omega \subseteq \mathbb{R}$ is a (possibly unbounded) interval and for each $x \in \Omega$ there exists $j \in \{1, \dots, n\}$ such that with $g_j(x) \coloneqq w_j x + b_j$ it holds that $f(x) = g_j(x)$. Without loss of generality, we can assume that $g_i \neq g_j$ for all $i \neq j$. Since the graphs of the g_j are lines, they intersect at (at most) finitely many points in Ω.

Since f is continuous, we conclude that there exist finitely many intervals covering Ω, such that f coincides with one of the g_j on each interval. For each $x \in \Omega$ let

$$s_x \coloneqq \{1 \le j \le n \,|\, g_j(x) \ge f(x)\}$$

and

$$f_x(y) \coloneqq \min_{j \in s_x} g_j(y) \qquad \text{for all } y \in \Omega.$$

Clearly, $f_x(x) = f(x)$. We claim that, additionally,

$$f_x(y) \le f(y) \qquad \text{for all } y \in \Omega. \tag{5.2.2}$$

This then shows that

$$f(y) = \max_{x \in \Omega} f_x(y) = \max_{x \in \Omega} \min_{j \in s_x} g_j(y) \qquad \text{for all } y \in \mathbb{R}.$$

Since there exist only finitely many possibilities to choose a subset of $\{1, \dots, n\}$, we conclude that (5.2.1) holds for $d = 1$.

It remains to verify the claim (5.2.2). Fix $y \neq x \in \Omega$. Without loss of generality, let $x < y$ and let $x = x_0 < \cdots < x_k = y$ be such that $f|_{[x_{i-1}, x_i]}$ equals some g_j for each $i \in \{1, \dots, k\}$. In order to show (5.2.2), it suffices to prove that there exists at least one j such that $g_j(x_0) \ge f(x_0)$ and $g_j(x_k) \le f(x_k)$. The claim is trivial for $k = 1$. We proceed by induction. Suppose the claim holds for $k - 1$, and consider the partition $x_0 < \cdots < x_k$. Let $r \in \{1, \dots, n\}$ be such that $f|_{[x_0, x_1]} = g_r|_{[x_0, x_1]}$. Applying the induction hypothesis to the interval $[x_1, x_k]$, we can find $j \in \{1, \dots, n\}$ such that $g_j(x_1) \ge f(x_1)$ and $g_j(x_k) \le f(x_k)$. If $g_j(x_0) \ge f(x_0)$, then g_j is the desired function. Otherwise, $g_j(x_0) < f(x_0)$. Then $g_r(x_0) = f(x_0) > g_j(x_0)$ and $g_r(x_1) = f(x_1) \le g_j(x_1)$. Therefore $g_r(x) \le g_j(x)$ for all $x \ge x_1$, and in particular $g_r(x_k) \le g_j(x_k)$. Thus g_r is the desired function.

Step 2. For general $d \in \mathbb{N}$, let $g_j(\boldsymbol{x}) \coloneqq \boldsymbol{w}_j^\top \boldsymbol{x} + b_j$ for $j = 1, \dots, n$. For each $\boldsymbol{x} \in \Omega$, let

$$s_{\boldsymbol{x}} \coloneqq \{1 \le j \le n \,|\, g_j(\boldsymbol{x}) \ge f(\boldsymbol{x})\}$$

and for all $\boldsymbol{y} \in \Omega$, let

$$f_{\boldsymbol{x}}(\boldsymbol{y}) := \min_{j \in s_{\boldsymbol{x}}} g_j(\boldsymbol{y}).$$

For an arbitrary 1-dimensional affine subspace $S \subseteq \mathbb{R}^d$ passing through $\boldsymbol{x}$ consider the line (segment) $I := S \cap \Omega$, which is connected since Ω is convex. By Step 1, it holds

$$f(\boldsymbol{y}) = \max_{\boldsymbol{x} \in \Omega} f_{\boldsymbol{x}}(\boldsymbol{y}) = \max_{\boldsymbol{x} \in \Omega} \min_{j \in s_{\boldsymbol{x}}} g_j(\boldsymbol{y})$$

on all of I. Since I was arbitrary the formula is valid for all $\boldsymbol{y} \in \Omega$. This again implies (5.2.1) as in Step 1. □

Remark 5.9. Using $\min(a, b) = -\max(-a, -b)$, there exists $\tilde{m} \in \mathbb{N}$ and sets $\tilde{s}_j \subseteq \{1, \dots, n\}$ for $j = 1, \dots, \tilde{m}$, such that for all $\boldsymbol{x} \in \mathbb{R}$

$$\begin{aligned} f(\boldsymbol{x}) = -(-f(\boldsymbol{x})) &= -\min_{1 \le j \le \tilde{m}} \max_{i \in \tilde{s}_j} (-\boldsymbol{w}_i^\top \boldsymbol{x} - b_i) \\ &= \max_{1 \le j \le \tilde{m}} (-\max_{i \in \tilde{s}_j} (-\boldsymbol{w}_i^\top \boldsymbol{x} - b_i)) \\ &= \max_{1 \le j \le \tilde{m}} (\min_{i \in \tilde{s}_j} (\boldsymbol{w}_i^\top \boldsymbol{x} + b_i)). \end{aligned}$$

To prove Theorem 5.7, it therefore suffices to show that the minimum and the maximum are expressible by ReLU neural networks.

Lemma 5.10. *For every x, $y \in \mathbb{R}$ it holds that*

$$\min\{x, y\} = \sigma_{\mathrm{ReLU}}(y) - \sigma_{\mathrm{ReLU}}(-y) - \sigma_{\mathrm{ReLU}}(y - x) \in \mathcal{N}_2^1(\sigma_{\mathrm{ReLU}}; 1, 3)$$

and

$$\max\{x, y\} = \sigma_{\mathrm{ReLU}}(y) - \sigma_{\mathrm{ReLU}}(-y) + \sigma_{\mathrm{ReLU}}(x - y) \in \mathcal{N}_2^1(\sigma_{\mathrm{ReLU}}; 1, 3).$$

Proof. We have

$$\begin{aligned} \max\{x, y\} &= y + \begin{cases} 0 & \text{if } y > x \\ x - y & \text{if } x \ge y \end{cases} \\ &= y + \sigma_{\mathrm{ReLU}}(x - y). \end{aligned}$$

Using $y = \sigma_{\mathrm{ReLU}}(y) - \sigma_{\mathrm{ReLU}}(-y)$, the claim for the maximum follows. For the minimum observe that $\min\{x, y\} = -\max\{-x, -y\}$. □

The minimum of $n \ge 2$ inputs can be computed by repeatedly applying the construction of Lemma 5.10. The resulting neural network is described in the next lemma.

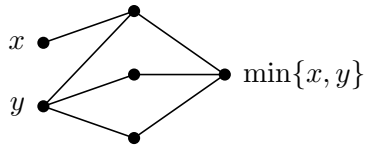


Figure 5.1: Sketch of the neural network in Lemma 5.10. Only edges with non-zero weights are drawn.

Lemma 5.11. *For every $n \geq 2$ there exists a neural network $\Phi_n^{\min} : \mathbb{R}^n \to \mathbb{R}$ with*

$$\mathrm{size}(\Phi_n^{\min}) \leq 16n, \qquad \mathrm{width}(\Phi_n^{\min}) \leq 3n, \qquad \mathrm{depth}(\Phi_n^{\min}) \leq \lceil \log_2(n) \rceil$$

such that $\Phi_n^{\min}(x_1, \ldots, x_n) = \min_{1 \leq j \leq n} x_j$. Similarly, there exists a neural network $\Phi_n^{\max} : \mathbb{R}^n \to \mathbb{R}$ realizing the maximum and satisfying the same complexity bounds.

Proof. Throughout denote by $\Phi_2^{\min} : \mathbb{R}^2 \to \mathbb{R}$ the neural network from Lemma 5.10. It is of depth 1 and size 7 (since all biases are zero, it suffices to count the number of connections in Figure 5.1).

Step 1. Consider first the case where $n = 2^k$ for some $k \in \mathbb{N}$. We proceed by induction of k. For $k = 1$ the claim is proven. For $k \geq 2$ set

$$\Phi_{2^k}^{\min} := \Phi_2^{\min} \circ (\Phi_{2^{k-1}}^{\min}, \Phi_{2^{k-1}}^{\min}). \tag{5.2.3}$$

By Lemma 5.2 and Lemma 5.3 we have

$$\mathrm{depth}(\Phi_{2^k}^{\min}) \leq \mathrm{depth}(\Phi_2^{\min}) + \mathrm{depth}(\Phi_{2^{k-1}}^{\min}) \leq \cdots \leq k.$$

Next, we bound the size of the neural network. Note that all biases in this neural network are set to 0, since the $\Phi_2^{\min}$ neural network in Lemma 5.10 has no biases. Thus, the size of the neural network $\Phi_{2^k}^{\min}$ corresponds to the number of connections in the graph (the number of nonzero weights). Careful inspection of the neural network architecture, see Figure 5.2, reveals that

$$\begin{aligned} \mathrm{size}(\Phi_{2^k}^{\min}) &= 4 \cdot 2^{k-1} + \sum_{j=0}^{k-2} 12 \cdot 2^j + 3 \\ &= 2n + 12 \cdot (2^{k-1} - 1) + 3 = 2n + 6n - 9 \leq 8n, \end{aligned}$$

and that $\mathrm{width}(\Phi_{2^k}^{\min}) \leq (3/2)2^k$. This concludes the proof for the case $n = 2^k$.

Step 2. For the general case, we first let

$$\Phi_1^{\min}(x) := x \qquad \text{for all } x \in \mathbb{R}$$

be the identity on $\mathbb{R}$, i.e. a linear transformation and thus formally a depth 0 neural network. Then, for all $n \geq 2$

$$\Phi_n^{\min} := \Phi_2^{\min} \circ \begin{cases} (\Phi_1^{\mathrm{id}} \circ \Phi_{\lfloor \frac{n}{2} \rfloor}^{\min}, \Phi_{\lceil \frac{n}{2} \rceil}^{\min}) & \text{if } n \in \{2^k + 1 \,|\, k \in \mathbb{N}\} \\ (\Phi_{\lfloor \frac{n}{2} \rfloor}^{\min}, \Phi_{\lceil \frac{n}{2} \rceil}^{\min}) & \text{otherwise.} \end{cases} \tag{5.2.4}$$

This definition extends (5.2.3) to arbitrary $n \geq 2$, since the first case in (5.2.4) never occurs if $n \geq 2$ is a power of two.

To analyze (5.2.4), we start with the depth and claim that

$$\operatorname{depth}(\Phi_n^{\min}) = k \qquad \text{for all } 2^{k-1} < n \leq 2^k$$

and all $k \in \mathbb{N}$. We proceed by induction over k. The case $k = 1$ is clear. For the induction step, assume the statement holds for some fixed $k \in \mathbb{N}$ and fix an integer n with $2^k < n \leq 2^{k+1}$. Then

$$\left\lceil \frac{n}{2} \right\rceil \in (2^{k-1}, 2^k] \cap \mathbb{N}$$

and

$$\left\lfloor \frac{n}{2} \right\rfloor \in \begin{cases} \{2^{k-1}\} & \text{if } n = 2^k + 1 \\ (2^{k-1}, 2^k] \cap \mathbb{N} & \text{otherwise.} \end{cases}$$

Using the induction assumption, (5.2.4) and Lemmas 5.1 and 5.2, this shows

$$\operatorname{depth}(\Phi_n^{\min}) = \operatorname{depth}(\Phi_2^{\min}) + k = 1 + k,$$

and proves the claim.

For the size and width bounds, we only sketch the argument: Fix $n \in \mathbb{N}$ such that $2^{k-1} < n \leq 2^k$. Then $\Phi_n^{\min}$ is constructed from at most as many subnetworks as $\Phi_{2^k}^{\min}$, but with some $\Phi_2^{\min} : \mathbb{R}^2 \to \mathbb{R}$ blocks replaced by $\Phi_1^{\mathrm{id}} : \mathbb{R} \to \mathbb{R}$, see Figure 5.3. Since Φ_1^{id} has the same depth as $\Phi_2^{\min}$, but is smaller in width and number of connections, the width and size of $\Phi_n^{\min}$ is bounded by the width and size of $\Phi_{2^k}^{\min}$. Due to $2^k \leq 2n$, the bounds from Step 1 give the bounds stated in the lemma.

Step 3. For the maximum, define

$$\Phi_n^{\max}(x_1, \dots, x_n) := -\Phi_n^{\min}(-x_1, \dots, -x_n).$$

□

of Theorem 5.7. By Proposition 5.8 the neural network

$$\Phi := \Phi_m^{\max} \bullet (\Phi_{|s_j|}^{\min})_{j=1}^m \bullet ((\boldsymbol{w}_i^\top \boldsymbol{x} + b_i)_{i \in s_j})_{j=1}^m$$

realizes the function f.

Since the number of possibilities to choose subsets of $\{1, \dots, n\}$ equals 2^n we have $m \leq 2^n$. Since each s_j is a subset of $\{1, \dots, n\}$, the cardinality $|s_j|$ of s_j is bounded by n. By Lemma 5.2, Lemma 5.3, and Lemma 5.11

$$\begin{aligned} \operatorname{depth}(\Phi) &\leq 2 + \operatorname{depth}(\Phi_m^{\max}) + \max_{1 \leq j \leq n} \operatorname{depth}(\Phi_{|s_j|}^{\min}) \\ &\leq 1 + \lceil \log_2(2^n) \rceil + \lceil \log_2(n) \rceil = O(n) \end{aligned}$$

and

$$\begin{aligned} \operatorname{width}(\Phi) &\leq 2 \max \left\{ \operatorname{width}(\Phi_m^{\max}), \sum_{j=1}^m \operatorname{width}(\Phi_{|s_j|}^{\min}), \sum_{j=1}^m \operatorname{width}((\boldsymbol{w}_i^\top \boldsymbol{x} + b_i)_{i \in s_j})) \right\} \\ &\leq 2 \max\{3m, 3mn, mdn\} = O(dn2^n) \end{aligned}$$

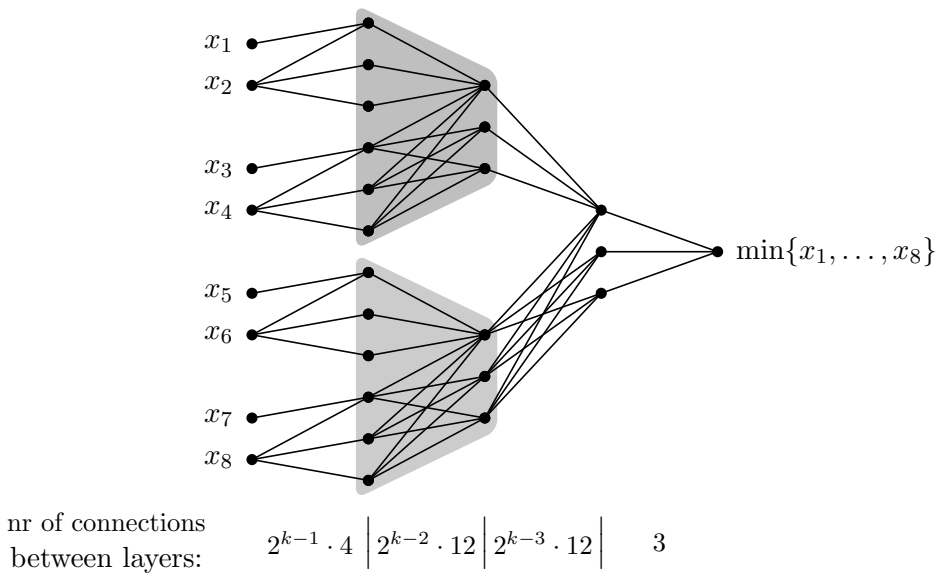


Figure 5.2: Architecture of the $\Phi^{\min}_{2^k}$ neural network in Step 1 of the proof of Lemma 5.11 and the number of connections in each layer for $k = 3$. Each grey box corresponds to 12 connections in the graph.

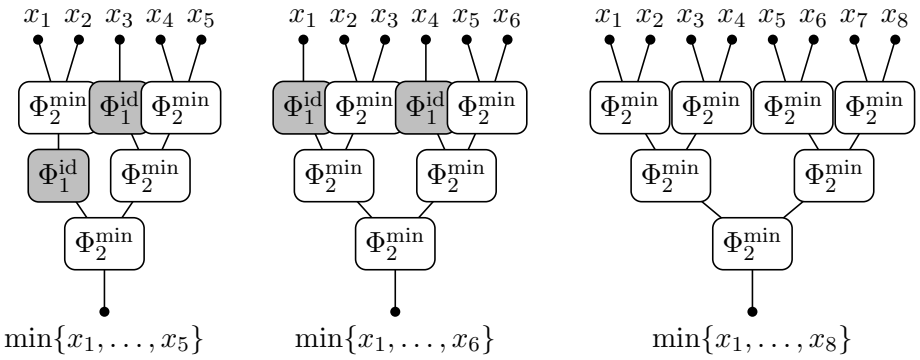


Figure 5.3: Construction of $\Phi^{\min}_n$ for general n in Step 2 of the proof of Lemma 5.11.

and

$$\begin{aligned}\operatorname{size}(\Phi) &\leq 4\Big(\operatorname{size}(\Phi_m^{\max}) + \operatorname{size}((\Phi_{|s_j|}^{\min})_{j=1}^m) + \operatorname{size}((\boldsymbol{w}_i^\top \boldsymbol{x} + b_i)_{i\in s_j})_{j=1}^m)\Big) \\ &\leq 4\left(16m + 2\sum_{j=1}^m (16|s_j| + 2\lceil \log_2(n)\rceil) + nm(d+1)\right) = O(dn2^n).\end{aligned}$$

This concludes the proof. □

5.3 Simplicial pieces

This section studies the case, were we do not have arbitrary cpwl functions, but where the regions on which f is affine are simplices. Under this condition, we can construct neural networks that scale merely *linearly* in the number of such regions, which is a serious improvement from the *exponential* dependence of the size on the number of regions that was found in Theorem 5.7.

5.3.1 Triangulations of Ω

For the ensuing discussion, we will consider $\Omega \subseteq \mathbb{R}^d$ to be partitioned into simplices. This partitioning will be termed a **triangulation** of Ω. Other notions prevalent in the literature include a **tessellation** of Ω, or a **simplicial mesh** on Ω. To give a precise definition, let us first recall some terminology. For a set $S \subseteq \mathbb{R}^d$ we denote the **convex hull** of S by

$$\operatorname{co}(S) := \left\{ \sum_{j=1}^n \alpha_j \boldsymbol{x}_j \;\middle|\; n \in \mathbb{N},\ \boldsymbol{x}_j \in S, \alpha_j \geq 0,\ \sum_{j=1}^n \alpha_j = 1 \right\}.$$

An n**-simplex** is the convex hull of $n \in \mathbb{N}$ points that are independent in a specific sense. This is made precise in the following definition.

Definition 5.12. Let $n \in \mathbb{N}_0$, $d \in \mathbb{N}$ and $n \leq d$. We call $\boldsymbol{x}_0, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$ **affinely independent** if and only if either $n = 0$ or $n \geq 1$ and the vectors $\boldsymbol{x}_1 - \boldsymbol{x}_0, \dots, \boldsymbol{x}_n - \boldsymbol{x}_0$ are linearly independent. In this case, we call $\operatorname{co}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_n) := \operatorname{co}(\{\boldsymbol{x}_0, \dots, \boldsymbol{x}_n\})$ an n**-simplex**.

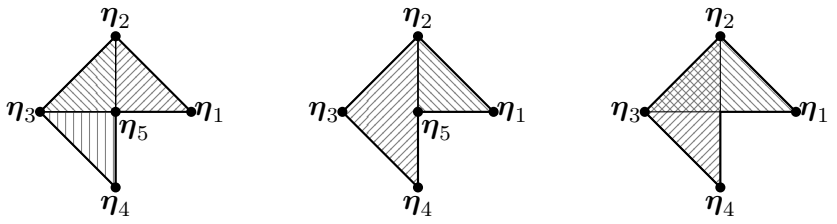
As mentioned before, a triangulation refers to a partition of a space into simplices. We give a formal definition below.

Definition 5.13. Let $d \in \mathbb{N}$, and $\Omega \subseteq \mathbb{R}^d$ be compact. Let $\mathcal{T}$ be a finite set of d-simplices, and for each $\tau \in \mathcal{T}$ let $V(\tau) \subseteq \Omega$ have cardinality $d+1$ such that $\tau = \operatorname{co}(V(\tau))$. We call $\mathcal{T}$ a **regular triangulation** of Ω, if and only if

(i) $\bigcup_{\tau\in\mathcal{T}} \tau = \Omega$,

(ii) for all $\tau, \tau' \in \mathcal{T}$ it holds that $\tau \cap \tau' = \operatorname{co}(V(\tau) \cap V(\tau'))$.

We call $\boldsymbol{\eta} \in \mathcal{V} := \bigcup_{\tau\in\mathcal{T}} V(\tau)$ a **node** (or vertex) and $\tau \in \mathcal{T}$ an **element** of the triangulation.



$\tau_1 = \mathrm{co}(\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \boldsymbol{\eta}_5)$
$\tau_2 = \mathrm{co}(\boldsymbol{\eta}_2, \boldsymbol{\eta}_3, \boldsymbol{\eta}_5)$
$\tau_3 = \mathrm{co}(\boldsymbol{\eta}_3, \boldsymbol{\eta}_4, \boldsymbol{\eta}_5)$

$\tau_1 = \mathrm{co}(\boldsymbol{\eta}_2, \boldsymbol{\eta}_3, \boldsymbol{\eta}_4)$
$\tau_2 = \mathrm{co}(\boldsymbol{\eta}_2, \boldsymbol{\eta}_5, \boldsymbol{\eta}_1)$

$\tau_1 = \mathrm{co}(\boldsymbol{\eta}_2, \boldsymbol{\eta}_3, \boldsymbol{\eta}_4)$
$\tau_2 = \mathrm{co}(\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \boldsymbol{\eta}_3)$

Figure 5.4: The first is a regular triangulation, while the second and the third are not.

For a regular triangulation $\mathcal{T}$ with nodes $\mathcal{V}$ we also introduce the constant

$$k_{\mathcal{T}} := \max_{\boldsymbol{\eta} \in \mathcal{V}} |\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}| \tag{5.3.1}$$

corresponding to the maximal number of elements shared by a single node.

5.3.2 Size bounds for regular triangulations

Throughout this subsection, let $\mathcal{T}$ be a regular triangulation of Ω, and we adhere to the notation of Definition 5.13. We will say that $f : \Omega \to \mathbb{R}$ is cpwl with respect to $\mathcal{T}$ if f is cpwl and $f|_\tau$ is affine for each $\tau \in \mathcal{T}$. The rest of this subsection is dedicated to proving the following result. It was first shown in [134] with a more technical argument, and extends an earlier statement from [84] to general triangulations (also see Section 5.3.3).

Theorem 5.14. *Let $d \in \mathbb{N}$, $\Omega \subseteq \mathbb{R}^d$ be a bounded domain, and let $\mathcal{T}$ be a regular triangulation of Ω. Let $f : \Omega \to \mathbb{R}$ be cpwl with respect to $\mathcal{T}$ and $f|_{\partial\Omega} = 0$. Then there exists a ReLU neural network $\Phi : \Omega \to \mathbb{R}$ realizing f, and it holds*

$$\mathrm{size}(\Phi) = O(|\mathcal{T}|), \qquad \mathrm{width}(\Phi) = O(|\mathcal{T}|), \qquad \mathrm{depth}(\Phi) = O(1), \tag{5.3.2}$$

where the constants in the Landau notation depend on d and $k_{\mathcal{T}}$ in (5.3.1).

We will split the proof into several lemmata. The strategy is to introduce a basis of the space of cpwl functions on $\mathcal{T}$ the elements of which vanish on the boundary of Ω. We will then show that there exist $O(|\mathcal{T}|)$ basis functions, each of which can be represented with a neural network the size of which depends only on $k_{\mathcal{T}}$ and d. To construct this basis, we first point out that an affine function on a simplex is uniquely defined by its values at the nodes.

Lemma 5.15. *Let $d \in \mathbb{N}$. Let $\tau := \mathrm{co}(\boldsymbol{\eta}_0, \dots, \boldsymbol{\eta}_d)$ be a d-simplex. For every $y_0, \dots, y_d \in \mathbb{R}$, there exists a unique $g \in \mathbb{P}_1(\mathbb{R}^d)$ such that $g(\boldsymbol{\eta}_i) = y_i$, $i = 0, \dots, d$.*

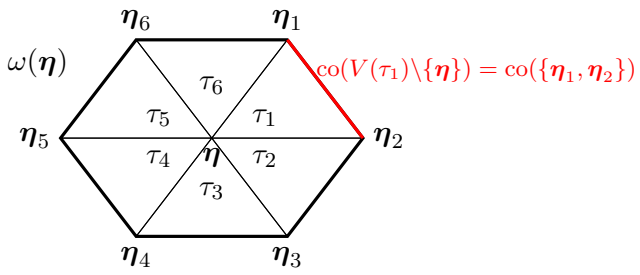


Figure 5.5: Visualization of Lemma 5.16 in two dimensions. The patch $\omega(\boldsymbol{\eta})$ consists of the union of all 2-simplices τ_i containing $\boldsymbol{\eta}$. Its boundary consists of the union of all 1-simplices made up by the nodes of each τ_i without the center node, i.e., the convex hulls of $V(\tau_i)\backslash\{\boldsymbol{\eta}\}$.

Proof. Since $\boldsymbol{\eta}_1 - \boldsymbol{\eta}_0, \dots, \boldsymbol{\eta}_d - \boldsymbol{\eta}_0$ is a basis of $\mathbb{R}^d$, there is a unique $\boldsymbol{w} \in \mathbb{R}^d$ such that $\boldsymbol{w}^\top(\boldsymbol{\eta}_i - \boldsymbol{\eta}_0) = y_i - y_0$ for $i = 1, \dots, d$. Then $g(\boldsymbol{x}) := \boldsymbol{w}^\top \boldsymbol{x} + (y_0 - \boldsymbol{w}^\top \boldsymbol{\eta}_0)$ is as desired. Moreover, for every $g \in \mathbb{P}_1$ it holds that $g(\sum_{i=0}^d \alpha_i \boldsymbol{\eta}_i) = \sum_{i=0}^d \alpha_i g(\boldsymbol{\eta}_i)$ whenever $\sum_{i=0}^d \alpha_i = 1$ (this is in general not true if the coefficients do not sum to 1). Hence, g is uniquely determined by its values at the nodes. □

Since Ω is the union of the simplices $\tau \in \mathcal{T}$, every cpwl function with respect to $\mathcal{T}$ is thus uniquely defined through its values at the nodes. Hence, the desired basis consists of cpwl functions $\varphi_{\boldsymbol{\eta}} : \Omega \to \mathbb{R}$ with respect to $\mathcal{T}$ such that

$$\varphi_{\boldsymbol{\eta}}(\boldsymbol{\mu}) = \delta_{\boldsymbol{\eta}\boldsymbol{\mu}} \qquad \text{for all } \boldsymbol{\mu} \in \mathcal{V}, \tag{5.3.3}$$

where $\delta_{\boldsymbol{\eta}\boldsymbol{\mu}}$ denotes the Kronecker delta. Assuming $\varphi_{\boldsymbol{\eta}}$ to be well-defined for the moment, we can then represent every cpwl function $f : \Omega \to \mathbb{R}$ that vanishes on the boundary $\partial\Omega$ as

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}} f(\boldsymbol{\eta}) \varphi_{\boldsymbol{\eta}}(\boldsymbol{x}) \qquad \text{for all } \boldsymbol{x} \in \Omega.$$

Note that it suffices to sum over the set of **interior nodes** $\mathcal{V} \cap \mathring{\Omega}$, since $f(\boldsymbol{\eta}) = 0$ whenever $\boldsymbol{\eta} \in \partial\Omega$. To formally verify existence and well-definedness of $\varphi_{\boldsymbol{\eta}}$, we first need a lemma characterizing the boundary of so-called "patches" of the triangulation: For each $\boldsymbol{\eta} \in \mathcal{V}$, we introduce the **patch** $\omega(\boldsymbol{\eta})$ of the node $\boldsymbol{\eta}$ as the union of all elements containing $\boldsymbol{\eta}$, i.e.,

$$\omega(\boldsymbol{\eta}) := \bigcup_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}} \tau.$$

Lemma 5.16. *Let $\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}$ be an interior node. Then,*

$$\partial\omega(\boldsymbol{\eta}) = \bigcup_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}} \mathrm{co}(V(\tau)\backslash\{\boldsymbol{\eta}\}).$$

We refer to Figure 5.5 for a visualization of Lemma 5.16. The proof of Lemma 5.16 is quite technical but nonetheless elementary. We therefore only outline the general argument but leave the details to the reader in Excercise 5.27: The boundary of $\omega(\boldsymbol{\eta})$ must be contained in the union of the boundaries of all τ in the patch $\omega(\boldsymbol{\eta})$. Since $\boldsymbol{\eta}$ is an interior point of Ω, it must also be an interior point of $\omega(\boldsymbol{\eta})$. This can be used to show that for every $S := \{\boldsymbol{\eta}_{i_0}, \dots, \boldsymbol{\eta}_{i_k}\} \subseteq V(\tau)$ of cardinality $k+1 \leq d$, the interior of (the k-dimensional manifold) $\mathrm{co}(S)$ belongs to the interior of $\omega(\boldsymbol{\eta})$ whenever $\boldsymbol{\eta} \in S$. Using Exercise 5.27, it then only remains to check that $\mathrm{co}(S) \subseteq \partial\omega(\boldsymbol{\eta})$ whenever $\boldsymbol{\eta} \notin S$, which yields the claimed formula. We are now in position to show well-definenedness of the basis functions in (5.3.3).

Lemma 5.17. *For each interior node $\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}$ there exists a unique cpwl function $\varphi_{\boldsymbol{\eta}} : \Omega \to \mathbb{R}$ satisfying* (5.3.3)*. Moreover, $\varphi_{\boldsymbol{\eta}}$ can be expressed by a ReLU neural network with size, width, and depth bounds that only depend on d and $k_{\mathcal{T}}$.*

Proof. By Lemma 5.15, on each $\tau \in \mathcal{T}$, the affine function $\varphi_{\boldsymbol{\eta}}|_\tau$ is uniquely defined through the values at the nodes of τ. This defines a continuous function $\varphi_{\boldsymbol{\eta}} : \Omega \to \mathbb{R}$. Indeed, whenever $\tau \cap \tau' \neq \emptyset$, then $\tau \cap \tau'$ is a subsimplex of both τ and τ' in the sense of Definition 5.13 (ii). Thus, applying Lemma 5.15 again, the affine functions on τ and τ' coincide on $\tau \cap \tau'$.

Using Lemma 5.15, Lemma 5.16 and the fact that $\varphi_{\boldsymbol{\eta}}(\boldsymbol{\mu}) = 0$ whenever $\boldsymbol{\mu} \neq \boldsymbol{\eta}$, we find that $\varphi_{\boldsymbol{\eta}}$ vanishes on the boundary of the patch $\omega(\boldsymbol{\eta}) \subseteq \Omega$. Thus, $\varphi_{\boldsymbol{\eta}}$ vanishes on the boundary of Ω. Extending by zero, it becomes a cpwl function $\varphi_{\boldsymbol{\eta}} : \mathbb{R}^d \to \mathbb{R}$. This function is nonzero only on elements τ for which $\boldsymbol{\eta} \in \tau$. Hence, it is a cpwl function with at most $n := k_{\mathcal{T}} + 1$ affine functions. By Theorem 5.7, $\varphi_{\boldsymbol{\eta}}$ can be expressed as a ReLU neural network with the claimed size, width and depth bounds. □

Finally, Theorem 5.14 is now an easy consequence of the above lemmata.

of Theorem 5.14. With

$$\Phi(\boldsymbol{x}) := \sum_{\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}} f(\boldsymbol{\eta}) \varphi_{\boldsymbol{\eta}}(\boldsymbol{x}) \qquad \text{for } \boldsymbol{x} \in \Omega, \tag{5.3.4}$$

it holds that $\Phi : \Omega \to \mathbb{R}$ satisfies $\Phi(\boldsymbol{\eta}) = f(\boldsymbol{\eta})$ for all $\boldsymbol{\eta} \in \mathcal{V}$. By Lemma 5.15 this implies that f equals Φ on each τ, and thus f equals Φ on all of Ω. Since each element τ is the convex hull of $d+1$ nodes $\boldsymbol{\eta} \in \mathcal{V}$, the cardinality of $\mathcal{V}$ is bounded by the cardinality of $\mathcal{T}$ times $d+1$. Thus, the summation in (5.3.4) is over $O(|\mathcal{T}|)$ terms. Using Lemma 5.4 and Lemma 5.17 we obtain the claimed bounds on size, width, and depth of the neural network. □

5.3.3 Size bounds for locally convex triangulations

Assuming local convexity of the triangulation, in this section we make the dependence of the constants in Theorem 5.14 explicit in the dimension d and in the maximal number of simplices $k_{\mathcal{T}}$ touching a node, see (5.3.1). As such the improvement over Theorem 5.14 is modest, and the reader may choose to skip this section on a first pass. Nonetheless, the proof, originally from [84], is entirely constructive and gives some further insight on how ReLU networks express functions. Let us start by stating the required convexity constraint.

Definition 5.18. A regular triangulation $\mathcal{T}$ is called **locally convex** if and only if $\omega(\boldsymbol{\eta})$ is convex for all interior nodes $\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}$.

The following theorem is a variant of [84, Theorem 3.1].

Theorem 5.19. *Let $d \in \mathbb{N}$, and let $\Omega \subseteq \mathbb{R}^d$ be a bounded domain. Let $\mathcal{T}$ be a locally convex regular triangulation of Ω. Let $f : \Omega \to \mathbb{R}$ be cpwl with respect to $\mathcal{T}$ and $f|_{\partial\Omega} = 0$. Then, there exists a constant $C > 0$ (independent of d, f and $\mathcal{T}$) and there exists a neural network $\Phi^f : \Omega \to \mathbb{R}$ such that $\Phi^f = f$,*

$$\begin{aligned} \operatorname{size}(\Phi^f) &\leq C \cdot (1 + d^2 k_{\mathcal{T}} |\mathcal{T}|), \\ \operatorname{width}(\Phi^f) &\leq C \cdot (1 + d \log(k_{\mathcal{T}}) |\mathcal{T}|), \\ \operatorname{depth}(\Phi^f) &\leq C \cdot (1 + \log_2(k_{\mathcal{T}})). \end{aligned}$$

Assume in the following that $\mathcal{T}$ is a locally convex triangulation. We will split the proof of the theorem again into a few lemmata. First, we will show that a convex patch can be written as an intersection of finitely many half-spaces. Specifically, with the **affine hull** of a set S defined as

$$\operatorname{aff}(S) := \left\{ \sum_{j=1}^{n} \alpha_j \boldsymbol{x}_j \,\middle|\, n \in \mathbb{N},\ \boldsymbol{x}_j \in S,\ \alpha_j \in \mathbb{R},\ \sum_{j=1}^{n} \alpha_j = 1 \right\}$$

let in the following for $\tau \in \mathcal{T}$ and $\boldsymbol{\eta} \in V(\tau)$

$$H_0(\tau, \boldsymbol{\eta}) := \operatorname{aff}(V(\tau) \backslash \{\boldsymbol{\eta}\})$$

be the affine hyperplane passing through all nodes in $V(\tau)\backslash\{\boldsymbol{\eta}\}$, and let further

$$H_+(\tau, \boldsymbol{\eta}) := \{\boldsymbol{x} \in \mathbb{R}^d \,|\, \boldsymbol{x} \text{ is on the same side of } H_0(\tau, \boldsymbol{\eta}) \text{ as } \boldsymbol{\eta}\} \cup H_0(\tau, \boldsymbol{\eta}).$$

Lemma 5.20. *Let $\boldsymbol{\eta}$ be an interior node. Then a patch $\omega(\boldsymbol{\eta})$ is convex if and only if*

$$\omega(\boldsymbol{\eta}) = \bigcap_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \mathcal{T}\}} H_+(\tau, \boldsymbol{\eta}). \tag{5.3.5}$$

Proof. The right-hand side is a finite intersection of (convex) half-spaces, and thus itself convex. It remains to show that if $\omega(\boldsymbol{\eta})$ is convex, then (5.3.5) holds. We start with "$\supset$". Suppose $\boldsymbol{x} \notin \omega(\boldsymbol{\eta})$. Then the straight line $\operatorname{co}(\{\boldsymbol{x}, \boldsymbol{\eta}\})$ must pass through $\partial\omega(\boldsymbol{\eta})$, and by Lemma 5.16 this implies that there exists $\tau \in \mathcal{T}$ with $\boldsymbol{\eta} \in \tau$ such that $\operatorname{co}(\{\boldsymbol{x}, \boldsymbol{\eta}\})$ passes through $\operatorname{aff}(V(\tau)\backslash\{\boldsymbol{\eta}\}) = H_0(\tau, \boldsymbol{\eta})$.

Hence $\boldsymbol{\eta}$ and $\boldsymbol{x}$ lie on different sides of this affine hyperplane, which shows "$\supseteq$". Now we show "$\subseteq$". Let $\tau \in \mathcal{T}$ be such that $\boldsymbol{\eta} \in \tau$ and fix $\boldsymbol{x}$ in the complement of $H_+(\tau, \boldsymbol{\eta})$. Suppose that $\boldsymbol{x} \in \omega(\boldsymbol{\eta})$. By convexity, we then have $\mathrm{co}(\{\boldsymbol{x}\} \cup \tau) \subseteq \omega(\boldsymbol{\eta})$. This implies that there exists a point in $\mathrm{co}(V(\tau)\backslash\{\boldsymbol{\eta}\})$ belonging to the interior of $\omega(\boldsymbol{\eta})$. This contradicts Lemma 5.16. Thus, $\boldsymbol{x} \notin \omega(\boldsymbol{\eta})$. □

The above lemma allows us to explicitly construct the basis functions $\varphi_{\boldsymbol{\eta}}$ in (5.3.3). To see this, denote in the following for $\tau \in \mathcal{T}$ and $\boldsymbol{\eta} \in V(\tau)$ by $g_{\tau,\boldsymbol{\eta}} \in \mathbb{P}_1(\mathbb{R}^d)$ the affine function such that

$$g_{\tau,\boldsymbol{\eta}}(\boldsymbol{\mu}) = \begin{cases} 1 & \text{if } \boldsymbol{\eta} = \boldsymbol{\mu} \\ 0 & \text{if } \boldsymbol{\eta} \neq \boldsymbol{\mu} \end{cases} \qquad \text{for all } \boldsymbol{\mu} \in V(\tau).$$

This function exists and is unique by Lemma 5.15. Observe that $\varphi_{\boldsymbol{\eta}}(\boldsymbol{x}) = g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \tau$.

Lemma 5.21. *Let $\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}$ be an interior node and let $\omega(\boldsymbol{\eta})$ be a convex patch. Then*

$$\varphi_{\boldsymbol{\eta}}(\boldsymbol{x}) = \max\left\{0, \min_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}} g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x})\right\} \qquad \textit{for all } \boldsymbol{x} \in \mathbb{R}^d. \tag{5.3.6}$$

Proof. First let $\boldsymbol{x} \notin \omega(\boldsymbol{\eta})$. By Lemma 5.20 there exists $\tau \in V(\boldsymbol{\eta})$ such that $\boldsymbol{x}$ is in the complement of $H_+(\tau, \boldsymbol{\eta})$. Observe that

$$g_{\tau,\boldsymbol{\eta}}|_{H_+(\tau,\boldsymbol{\eta})} \geq 0 \qquad \text{and} \qquad g_{\tau,\boldsymbol{\eta}}|_{H_+(\tau,\boldsymbol{\eta})^c} < 0. \tag{5.3.7}$$

Thus

$$\min_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}} g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}) < 0 \qquad \text{for all } \boldsymbol{x} \in \omega(\boldsymbol{\eta})^c,$$

i.e., (5.3.6) holds for all $\boldsymbol{x} \in \mathbb{R}\backslash\omega(\boldsymbol{\eta})$. Next, let τ, $\tau' \in \mathcal{T}$ such that $\boldsymbol{\eta} \in \tau$ and $\boldsymbol{\eta} \in \tau'$. We wish to show that $g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}) \leq g_{\tau',\boldsymbol{\eta}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \tau$. Since $g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}) = \varphi_{\boldsymbol{\eta}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \tau$, this then concludes the proof of (5.3.6). By Lemma 5.20 it holds

$$\boldsymbol{\mu} \in H_+(\tau', \boldsymbol{\eta}) \qquad \text{for all} \qquad \boldsymbol{\mu} \in V(\tau).$$

Hence, by (5.3.7)

$$g_{\tau',\boldsymbol{\eta}}(\boldsymbol{\mu}) \geq 0 = g_{\tau,\boldsymbol{\eta}}(\boldsymbol{\mu}) \qquad \text{for all} \qquad \boldsymbol{\mu} \in V(\tau)\backslash\{\boldsymbol{\eta}\}.$$

Moreover, $g_{\tau,\boldsymbol{\eta}}(\boldsymbol{\eta}) = g_{\tau',\boldsymbol{\eta}}(\boldsymbol{\eta}) = 1$. Thus, $g_{\tau,\boldsymbol{\eta}}(\boldsymbol{\mu}) \geq g_{\tau',\boldsymbol{\eta}}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in V(\tau')$ and therefore

$$g_{\tau',\boldsymbol{\eta}}(\boldsymbol{x}) \geq g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}) \qquad \text{for all } \boldsymbol{x} \in \mathrm{co}(V(\tau')) = \tau'.$$

□

of Theorem 5.19. For every interior node $\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}$, the cpwl basis function $\varphi_{\boldsymbol{\eta}}$ in (5.3.3) can be expressed as in (5.3.6), i.e.,

$$\varphi_{\boldsymbol{\eta}}(\boldsymbol{x}) = \sigma \bullet \Phi^{\min}_{|\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}|} \bullet (g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}))_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}},$$

where $(g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}))_{\{\tau\in\mathcal{T}\,|\,\boldsymbol{\eta}\in\tau\}}$ denotes the parallelization with shared inputs of the functions $g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x})$ for all $\tau\in\mathcal{T}$ such that $\boldsymbol{\eta}\in\tau$.

For this neural network, with $|\{\tau\in\mathcal{T}\,|\,\boldsymbol{\eta}\in\tau\}|\leq k_{\mathcal{T}}$, we have by Lemma 5.2

$$\begin{aligned}\operatorname{size}(\varphi_{\boldsymbol{\eta}}) &\leq 4\big(\operatorname{size}(\sigma)+\operatorname{size}(\Phi^{\min}_{|\{\tau\in\mathcal{T}\,|\,\boldsymbol{\eta}\in\tau\}|})+\operatorname{size}((g_{\tau,\boldsymbol{\eta}})_{\{\tau\in\mathcal{T}\,|\,\boldsymbol{\eta}\in\tau\}})\big)\\ &\leq 4(2+16k_{\mathcal{T}}+k_{\mathcal{T}}d)\end{aligned}\tag{5.3.8}$$

and similarly

$$\operatorname{depth}(\varphi_{\boldsymbol{\eta}})\leq 4+\lceil\log_2(k_{\mathcal{T}})\rceil,\qquad \operatorname{width}(\varphi_{\boldsymbol{\eta}})\leq\max\{1,3k_{\mathcal{T}},d\}.\tag{5.3.9}$$

Since for every interior node, the number of simplices touching the node must be larger or equal to d, we can assume $\max\{k_{\mathcal{T}},d\}=k_{\mathcal{T}}$ in the following (otherwise there exist no interior nodes, and the function f is constant 0). As in the proof of Theorem 5.14, the neural network

$$\Phi(\boldsymbol{x}):=\sum_{\boldsymbol{\eta}\in\mathcal{V}\cap\mathring{\Omega}}f(\boldsymbol{\eta})\varphi_{\boldsymbol{\eta}}(\boldsymbol{x})$$

realizes the function f on all of Ω. Since the number of nodes $|\mathcal{V}|$ is bounded by $(d+1)|\mathcal{T}|$, an application of Lemma 5.4 yields the desired bounds. □

5.4 Convergence rates for Hölder continuous functions

Theorem 5.14 immediately implies convergence rates for certain classes of (low regularity) functions. Recall for example the **space of Hölder continuous functions**: for $s\in(0,1]$ and a bounded domain $\Omega\subseteq\mathbb{R}^d$ we define

$$\|f\|_{C^{0,s}(\Omega)}:=\sup_{\boldsymbol{x}\in\Omega}|f(\boldsymbol{x})|+\sup_{\boldsymbol{x}\neq\boldsymbol{y}\in\Omega}\frac{|f(\boldsymbol{x})-f(\boldsymbol{y})|}{\|\boldsymbol{x}-\boldsymbol{y}\|_2^s}.\tag{5.4.1}$$

Then, $C^{0,s}(\Omega)$ is the set of functions $f\in C^0(\Omega)$ for which $\|f\|_{C^{0,s}(\Omega)}<\infty$.

Hölder continuous functions can be approximated well by certain cpwl functions. Therefore, we obtain the following result.

Theorem 5.22. *Let $d\in\mathbb{N}$. There exists a constant $C=C(d)$ such that for every $f\in C^{0,s}([0,1]^d)$ and every N there exists a ReLU neural network Φ_N^f with*

$$\operatorname{size}(\Phi_N^f)\leq CN,\qquad \operatorname{width}(\Phi_N^f)\leq CN,\qquad \operatorname{depth}(\Phi_N^f)=C$$

and

$$\sup_{\boldsymbol{x}\in[0,1]^d}\Big|f(\boldsymbol{x})-\Phi_N^f(\boldsymbol{x})\Big|\leq C\|f\|_{C^{0,s}([0,1]^d)}N^{-\frac{s}{d}}.$$

Proof. For $M \geq 2$, consider the set of nodes $\{\boldsymbol{\nu}/M \,|\, \boldsymbol{\nu} \in \{-1, \dots, M+1\}^d\}$ where $\boldsymbol{\nu}/M = (\nu_1/M, \dots, \nu_d/M)$. These nodes suggest a partition of $[-1/M, 1+1/M]^d$ into $(2+M)^d$ sub-hypercubes. Each such sub-hypercube can be partitioned into $d!$ simplices, such that we obtain a regular triangulation $\mathcal{T}$ with $d!(2+M)^d$ elements on $[0,1]^d$. According to Theorem 5.14 there exists a neural network Φ that is cpwl with respect to $\mathcal{T}$ and $\Phi(\boldsymbol{\nu}/M) = f(\boldsymbol{\nu}/M)$ whenever $\boldsymbol{\nu} \in \{0, \dots, M\}^d$ and $\Phi(\boldsymbol{\nu}/M) = 0$ for all other (boundary) nodes. It holds

$$\begin{aligned} \operatorname{size}(\Phi) &\leq C|\mathcal{T}| = Cd!(2+M)^d, \\ \operatorname{width}(\Phi) &\leq C|\mathcal{T}| = Cd!(2+M)^d, \\ \operatorname{depth}(\Phi) &\leq C \end{aligned} \tag{5.4.2}$$

for a constant C that only depends on d (since for our regular triangulation $\mathcal{T}$, $k_{\mathcal{T}}$ in (5.3.1) is a fixed d-dependent constant).

Let us bound the error. Fix a point $\boldsymbol{x} \in [0,1]^d$. Then $\boldsymbol{x}$ belongs to one of the interior simplices τ of the triangulation. Two nodes of the simplex have distance at most

$$\left(\sum_{j=1}^{d} \left(\frac{1}{M} \right)^2 \right)^{1/2} = \frac{\sqrt{d}}{M} =: \varepsilon.$$

Since $\Phi|_\tau$ is the linear interpolant of f at the nodes $V(\tau)$ of the simplex τ, $\Phi(\boldsymbol{x})$ is a convex combination of the $(f(\boldsymbol{\eta}))_{\boldsymbol{\eta} \in V(\tau)}$. Fix an arbitrary node $\boldsymbol{\eta}_0 \in V(\tau)$. Then $\|\boldsymbol{x} - \boldsymbol{\eta}_0\|_2 \leq \varepsilon$ and

$$\begin{aligned} |\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{\eta}_0)| \leq \max_{\boldsymbol{\eta}, \boldsymbol{\mu} \in V(\tau)} |f(\boldsymbol{\eta}) - f(\boldsymbol{\mu})| &\leq \sup_{\substack{\boldsymbol{x}, \boldsymbol{y} \in [0,1]^d \\ \|\boldsymbol{x} - \boldsymbol{y}\|_2 \leq \varepsilon}} |f(\boldsymbol{x}) - f(\boldsymbol{y})| \\ &\leq \|f\|_{C^{0,s}([0,1]^d)} \varepsilon^s. \end{aligned}$$

Hence, using $f(\boldsymbol{\eta}_0) = \Phi(\boldsymbol{\eta}_0)$,

$$\begin{aligned} |f(\boldsymbol{x}) - \Phi(\boldsymbol{x})| &\leq |f(\boldsymbol{x}) - f(\boldsymbol{\eta}_0)| + |\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{\eta}_0)| \\ &\leq 2\|f\|_{C^{0,s}([0,1]^d)} \varepsilon^s \\ &= 2\|f\|_{C^{0,s}([0,1]^d)} d^{\frac{s}{2}} M^{-s} \\ &= 2d^{\frac{s}{2}} \|f\|_{C^{0,s}([0,1]^d)} N^{-\frac{s}{d}} \end{aligned} \tag{5.4.3}$$

where $N := M^d$. The statement follows by (5.4.2) and (5.4.3). □

The principle behind Theorem 5.22 can be applied in even more generality. Since we can represent every cpwl function on a regular triangulation with a neural network of size $O(N)$, where N denotes the number of elements, all of classical (e.g. finite element) approximation theory for cpwl functions can be lifted to generate statements about ReLU approximation. For instance, it is well-known, that functions in the Sobolev space $H^2([0,1]^d)$ can be approximated by cpwl functions on a regular triangulation in terms of $L^2([0,1]^d)$ with the rate $2/d$. Similar as in the proof of Theorem 5.22, for every $f \in H^2([0,1]^d)$ and every $N \in \mathbb{N}$ there then exists a ReLU neural network Φ_N such that $\operatorname{size}(\Phi_N) = O(N)$ and

$$\|f - \Phi_N\|_{L^2([0,1]^d)} \leq C\|f\|_{H^2([0,1]^d)} N^{-\frac{2}{d}}.$$

Finally, we can wonder how to approximate even smoother functions, i.e., those that have many continuous derivatives. Since more smoothness is a restrictive assumption on the set of functions to approximate, we would hope that this will allow us to have smaller neural networks. Essentially, we desire a result similar to Theorem 4.9, but with the ReLU activation function.

However, we will see in the following chapter, that the emulation of piecewise affine functions on regular triangulations cannot yield the approximation rates of Theorem 4.9. To harness the smoothness, it will be necessary to build ReLU neural networks that emulate polynomials. Surprisingly, we will see in Chapter 7 that polynomials can be very efficiently approximated by *deep* ReLU neural networks.

Bibliography and further reading

The ReLU calculus introduced in Section 5.1 was similarly given in [172]. The fact that every cpwl function can be expressed as a maximum over a minimum of linear functions goes back to the papers [225, 224]; also see [167, 236].

The main result of Section 5.2, which shows that every cpwl function can be expressed by a ReLU network, is then a straightforward consequence. This was first observed in [4], which also provided bounds on the network size. These bounds were significantly improved in [84] for cwpl functions on triangular meshes that satisfy a local convexity condition. Under this assumption, it was shown that the network size essentially only grows linearly with the number of pieces. The paper [134] showed that the convexity assumption is not necessary for this statement to hold. We give a similar result in Section 5.3.2, using a simpler argument than [134]. The locally convex case from [84] is separately discussed in Section 5.3.3, as it allows for further improvements in some constants.

The implications for the approximation of Hölder continuous functions discussed in Section 5.4, follows by standard approximation theory for cpwl functions. For a general reference on splines and piecewise polynomial approximation see for instance [206]. Finally we mention that similar convergence results can also be shown for other activation functions, see, e.g., [142].

Exercises

Exercise 5.23. Let $p : \mathbb{R} \to \mathbb{R}$ be a polynomial of degree $n \geq 1$ (with leading coefficient nonzero) and let $s : \mathbb{R} \to \mathbb{R}$ be a continuous sigmoidal activation function. Show that the identity map $x \mapsto x : \mathbb{R} \to \mathbb{R}$ belongs to $\mathcal{N}_1^1(p; 1, n+1)$ but not to $\mathcal{N}_1^1(s; L)$ for any $L \in \mathbb{N}$.

Exercise 5.24. Consider cpwl functions $f : \mathbb{R} \to \mathbb{R}$ with $n \in \mathbb{N}_0$ breakpoints (points where the function is not C^1). Determine the minimal size required to exactly express every such f with a depth-1 ReLU neural network.

Exercise 5.25. Show that, the notion of affine independence is invariant under permutations of the points.

Exercise 5.26. Let $\tau = \operatorname{co}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_d)$ be a d-simplex. Show that the coefficients $\alpha_i \geq 0$ such that $\sum_{i=0}^d \alpha_i = 1$ and $\boldsymbol{x} = \sum_{i=0}^d \alpha_i \boldsymbol{x}_i$ are unique for every $\boldsymbol{x} \in \tau$.

Exercise 5.27. Let $\tau = \operatorname{co}(\boldsymbol{\eta}_0, \dots, \boldsymbol{\eta}_d)$ be a d-simplex. Show that the boundary of τ is given by $\bigcup_{i=0}^d \operatorname{co}(\{\boldsymbol{\eta}_0, \dots, \boldsymbol{\eta}_d\} \backslash \{\boldsymbol{\eta}_i\})$.

Chapter 6

Affine pieces for ReLU neural networks

In the previous chapters, we observed some remarkable approximation results of shallow ReLU neural networks. In practice, however, deeper architectures are more common. To understand why, we in this chapter we discuss some potential shortcomings of shallow ReLU networks compared to deep ReLU networks.

Traditionally, an insightful approach to study limitations of ReLU neural networks has been to analyze the number of linear regions these functions can generate.

Definition 6.1. Let $d \in \mathbb{N}$, $\Omega \subseteq \mathbb{R}^d$, and let $f\colon \Omega \to \mathbb{R}$ be cpwl (see Definition 5.5). We say that f has $p \in \mathbb{N}$ **pieces (or linear regions)**, if p is the smallest number of connected open sets $(\Omega_i)_{i=1}^p$ such that $\bigcup_{i=1}^p \overline{\Omega_i} = \Omega$, and $f|_{\Omega_i}$ is an affine function for all $i = 1, \dots, p$. We denote $\mathrm{Pieces}(f, \Omega) \coloneqq p$.

For $d = 1$ we call every point where f is not differentiable a **break point** of f.

To get an accurate cpwl approximation of a function, the approximating function needs to have many pieces. The next theorem, corresponding to [61, Theorem 2], quantifies this statement.

Theorem 6.2. *Let $-\infty < a < b < \infty$ and $f \in C^3([a,b])$ so that f is not affine. Then there exists a constant $c > 0$ depending only on $\int_a^b \sqrt{|f''(x)|}\,\mathrm{d}x$ so that*

$$\|g - f\|_{L^\infty([a,b])} > cp^{-2}$$

for all cpwl g with at most $p \in \mathbb{N}$ pieces.

The proof of the theorem is left to the reader, see Exercise 6.12.

Theorem 6.2 implies that for ReLU neural networks we need architectures allowing for many pieces, if we want to approximate non-linear functions to high accuracy. But how many pieces can

we create for a fixed depth and width? We will establish a simple theoretical upper bound in Section 6.1. Subsequently, we will investigate under which conditions these upper bounds are attainable in Section 6.2. This will reveal that certain functions necessitate very large shallow networks for approximation, whereas relatively small deep networks can also approximate them. These findings are presented in Section 6.3.

Finally, we will question the practical relevance of this analysis by examining how many pieces typical neural networks possess. Surprisingly, in Section 6.4 we will find that randomly initialized deep neural networks on average do not have a number of pieces that is anywhere close to the theoretical upper bound.

6.1 Upper bounds

Neural networks are based on the composition and addition of neurons. These two operations increase the possible number of pieces in a very specific way. Figure 6.1 depicts the two operations and their effect. They can be described as follows:

- *Summation:* Let $\Omega \subseteq \mathbb{R}$. The sum of two cpwl functions $f_1,\ f_2 : \Omega \to \mathbb{R}$ satisfies

$$\mathrm{Pieces}(f_1 + f_2, \Omega) \leq \mathrm{Pieces}(f_1, \Omega) + \mathrm{Pieces}(f_2, \Omega) - 1. \tag{6.1.1}$$

  This holds because the sum is affine in every point where both f_1 and f_2 are affine. Therefore, the sum has at most as many break points as f_1 and f_2 combined. Moreover, the number of pieces of a univariate function equals the number of its break points plus one.

- *Composition:* Let again $\Omega \subseteq \mathbb{R}$. The composition of two functions $f_1 \colon \mathbb{R}^d \to \mathbb{R}$ and $f_2 \colon \Omega \to \mathbb{R}^d$ satisfies

$$\mathrm{Pieces}(f_1 \circ f_2, \Omega) \leq \mathrm{Pieces}(f_1, \mathbb{R}^d) \cdot \mathrm{Pieces}(f_2, \Omega). \tag{6.1.2}$$

  This is because for each of the affine pieces of f_2—let us call one of those pieces $A \subseteq \mathbb{R}$—we have that f_2 is either constant or injective on A. If it is constant, then $f_1 \circ f_2$ is constant. If it is injective, then $\mathrm{Pieces}(f_1 \circ f_2, A) = \mathrm{Pieces}(f_1, f_2(A)) \leq \mathrm{Pieces}(f_1, \mathbb{R}^d)$. Since this holds for all pieces of f_2 we get (6.1.2).

These considerations give the following result, which follows the argument of [226, Lemma 2.1]. We state it for general cpwl activation functions. The ReLU activation function corresponds to $p = 2$.

Theorem 6.3. *Let $L \in \mathbb{N}$. Let σ be cpwl with p pieces. Then, every neural network with architecture $(\sigma; 1, d_1, \ldots, d_L, 1)$ has at most $(p \cdot \mathrm{width}(\Phi))^L$ pieces.*

Proof. The proof is via induction over the depth L. Let $L = 1$, and let $\Phi : \mathbb{R} \to \mathbb{R}$ be a neural network of architecture $(\sigma; 1, d_1, 1)$. Then

$$\Phi(x) = \sum_{k=1}^{d_1} w_k^{(1)} \sigma(w_k^{(0)} x + b_k^{(0)}) + b^{(1)} \qquad \text{for } x \in \mathbb{R},$$

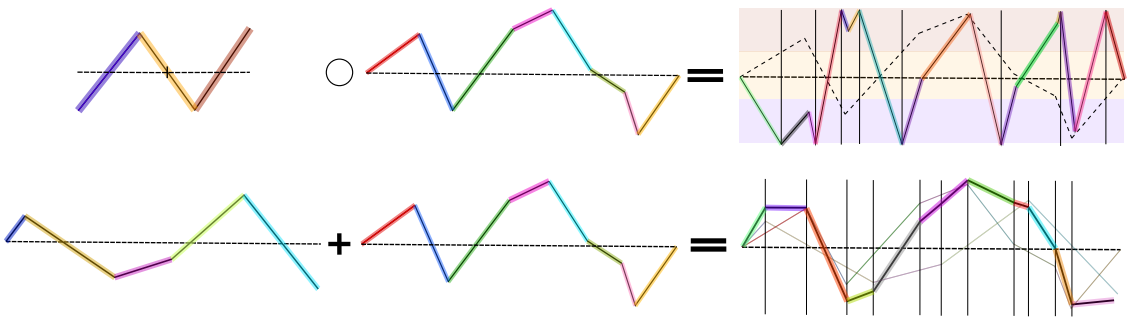


Figure 6.1: **Top:** Composition of two cpwl functions $f_1 \circ f_2$ can create a piece whenever the value of f_2 crosses a level that is associated to a break point of f_1. **Bottom:** Addition of two cpwl functions $f_1 + f_2$ produces a cpwl function that can have break points at positions where either f_1 or f_2 has a break point.

for certain $\boldsymbol{w}^{(0)}$, $\boldsymbol{w}^{(1)}$, $\boldsymbol{b}^{(0)} \in \mathbb{R}^{d_1}$ and $b^{(1)} \in \mathbb{R}$. By (6.1.1), $\mathrm{Pieces}(\Phi) \leq p \cdot \mathrm{width}(\Phi)$.

For the induction step, assume the statement holds for $L \in \mathbb{N}$, and let $\Phi : \mathbb{R} \to \mathbb{R}$ be a neural network of architecture $(\sigma; 1, d_1, \dots, d_{L+1}, 1)$. Then we can write

$$\Phi(x) = \sum_{j=1}^{d_{L+1}} w_j \sigma(h_j(x)) + b \qquad \text{for } x \in \mathbb{R},$$

for some $\boldsymbol{w} \in \mathbb{R}^{d_{L+1}}$, $b \in \mathbb{R}$, and where each h_j is a neural network with architecture $(\sigma; 1, d_1, \dots, d_L, 1)$. Using the induction hypothesis, each $\sigma \circ h_\ell$ has at most $p \cdot (p \cdot \mathrm{width}(\Phi))^L$ affine pieces. Hence Φ has at most $\mathrm{width}(\Phi) \cdot p \cdot (p \cdot \mathrm{width}(\Phi))^L = (p \cdot \mathrm{width}(\Phi))^{L+1}$ affine pieces. This completes the proof. □

Theorem 6.3 shows that there are limits to how many pieces can be created with a certain architecture. It is noteworthy that the effects of the depth and the width of a neural network are vastly different. While increasing the width can polynomially increase the number of pieces, increasing the depth can result in exponential increase. This is a first indication of the prowess of depth of neural networks.

To understand the effect of this on the approximation problem, we apply the bound of Theorem 6.3 to Theorem 6.2.

Theorem 6.4. *Let $d_0 \in \mathbb{N}$ and $f \in C^3([0,1]^{d_0})$. Assume there exists a line segment $\mathfrak{s} \subseteq [0,1]^{d_0}$ of positive length such that $0 < c := \int_{\mathfrak{s}} \sqrt{|f''(x)|}\, \mathrm{d}x$. Then there exists $C > 0$ solely depending on c, such that for all ReLU neural networks $\Phi : \mathbb{R}^{d_0} \to \mathbb{R}$ with L layers*

$$\|f - \Phi\|_{L^\infty([0,1]^{d_0})} \geq c \cdot (2\mathrm{width}(\Phi))^{-2L}.$$

Theorem 6.4 gives a lower bound on achievable approximation rates in dependence of the depth L. As target functions become smoother, we expect that we can achieve faster convergence rates

(cp. Chapter 4). However, without increasing the depth, it seems to be impossible to leverage such additional smoothness.

This observation strongly indicates that deeper architectures can be superior. Before we can make such statements, we first explore whether the upper bounds of Theorem 6.3 are even achievable.

6.2 Tightness of upper bounds

To construct a ReLU neural network, that realizes the upper bound of Theorem 6.3, we first let $h_1 : [0,1] \to \mathbb{R}$ be the hat function

$$h_1(x) := \begin{cases} 2x & \text{if } x \in [0, \frac{1}{2}] \\ 2 - 2x & \text{if } x \in [\frac{1}{2}, 1]. \end{cases}$$

This function can be expressed by a ReLU neural network of depth one and with two nodes

$$h_1(x) = \sigma_{\mathrm{ReLU}}(2x) - \sigma_{\mathrm{ReLU}}(4x - 2) \qquad \text{for all } x \in [0,1]. \tag{6.2.1}$$

We recursively set $h_n := h_{n-1} \circ h_1$ for all $n \geq 2$, i.e., $h_n = h_1 \circ \cdots \circ h_1$ is the n-fold composition of h_1. Since $h_1 : [0,1] \to [0,1]$, we have $h_n : [0,1] \to [0,1]$ and

$$h_n \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n, 2).$$

It turns out that this function has a rather interesting behavior. It is a "sawtooth" function with 2^{n-1} spikes, see Figure 6.2.

Lemma 6.5. *Let $n \in \mathbb{N}$. It holds for all $x \in [0,1]$*

$$h_n(x) = \begin{cases} 2^n(x - i2^{-n}) & \text{if } i \geq 0 \text{ is even and } x \in [i2^{-n}, (i+1)2^{-n}] \\ 2^n((i+1)2^{-n} - x) & \text{if } i \geq 1 \text{ is odd and } x \in [i2^{-n}, (i+1)2^{-n}]. \end{cases}$$

Proof. The case $n = 1$ holds by definition. We proceed by induction, and assume the statement holds for n. Let $x \in [0, 1/2]$ and $i \geq 0$ even such that $x \in [i2^{-(n+1)}, (i+1)2^{-(n+1)}]$. Then $2x \in [i2^{-n}, (i+1)2^{-n}]$. Thus

$$h_n(h_1(x)) = h_n(2x) = 2^n(2x - i2^{-n}) = 2^{n+1}(x - i2^{-n+1}).$$

Similarly, if $x \in [0, 1/2]$ and $i \geq 1$ odd such that $x \in [i2^{-(n+1)}, (i+1)2^{-(n+1)}]$, then $h_1(x) = 2x \in [i2^{-n}, (i+1)2^{-n}]$ and

$$h_n(h_1(x)) = h_n(2x) = 2^n(2x - (i+1)2^{-n}) = 2^{n+1}(x - (i+1)2^{-n+1}).$$

The case $x \in [1/2, 1]$ follows by observing that h_{n+1} is symmetric around $1/2$. □

The neural network h_n has size $O(n)$ but is piecewise linear on $O(2^n)$ pieces. This shows that the upper bound of Theorem 6.3 is tight.

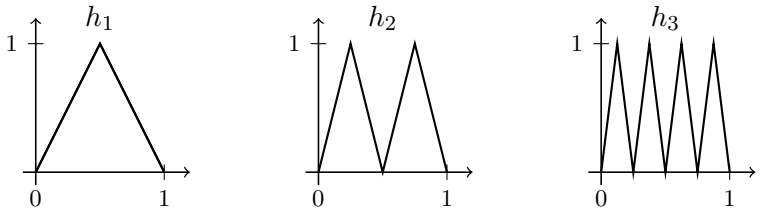


Figure 6.2: The functions h_n in Lemma 6.5.

6.3 Depth separation

Now that we have established how increasing the depth can lead to exponentially more pieces than increasing the width, we can deduce a so-called "depth-separation" result shown by Telgarsky in [226, 227]. Such statements verify the existence of functions that can easily be approximated by deep neural networks, but require much larger size when approximated by shallow neural networks. The following theorem, along with its proof, is presented similarly in Telgarsky's lecture notes [228].

Theorem 6.6. *For every $n \in \mathbb{N}$ there exists a neural network $f \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n^2+3, 2)$ such that for any $g \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n, 2^{n-1})$ holds*

$$\int_0^1 |f(x) - g(x)| \, \mathrm{d}x \geq \frac{1}{32}.$$

The neural network f may have quadratically more layers than g, but $\mathrm{width}(g) = 2^{n-1}$ and $\mathrm{width}(f) = 2$. Hence the *size of g may be exponentially larger than the size of f*, but nonetheless no such g can approximate f. Thus even exponential increase in width cannot necessarily compensate for increase in depth. The proof is based on the following observations stated in [227]:

(i) Functions with few oscillations poorly approximate functions with many oscillations,

(ii) neural networks with few layers have few oscillations,

(iii) neural networks with many layers can have many oscillations.

Proof of Theorem 6.6. Fix $n \in \mathbb{N}$. Let $f := h_{n^2+3} \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n^2+3, 2)$. For arbitrary $g \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n, 2^{n-1})$, by Theorem 6.3, g is piecewise linear with at most $(2 \cdot 2^{n-1})^n = 2^{n^2}$ break points. The function f is the sawtooth function with 2^{n^2+2} spikes. The number of triangles formed by the graph of f and the constant line at $1/2$ equals $2^{n^2+3} - 1$, each with area $2^{-(n^2+5)}$, see Figure 6.3. For the m triangles in between two break points of g, the graph of g does *not* cross at

Figure 6.3: **Left:** The functions h_n form $2^n - 1$ triangles with the line at $1/2$, each with area $2^{-(n+2)}$. **Right:** For an affine function with m (in this sketch $m = 5$) triangles in between two break points, the function can cross at most $\lceil m/2 \rceil + 1 \leq m/2 + 2$ of them. Figure adapted from [228, Section 5].

least $m - (m/2 + 2) = m/2 - 2$ of them. Thus we can bound

$$\begin{aligned}\int_0^1 |f(x) - g(x)| \, \mathrm{d}x &\geq \underbrace{\left(\frac{1}{2}(\underbrace{2^{n^2+3} - 1 - 2^{n^2}}_{\substack{\geq \text{triangles on an interval} \\ \text{without break point of } g}}) - \underbrace{2 \cdot 2^{n^2}}_{\geq 2 \cdot (\text{pieces of } g)} \right)}_{\geq \text{missed triangles}} \underbrace{2^{-(n^2+5)}}_{\text{area of a triangle}} \\ &\geq (2^{n^2+2} - 3 \cdot 2^{n^2}) \cdot 2^{-(n^2+5)} \\ &\geq 2^{n^2} \cdot 2^{-(n^2+5)} = \frac{1}{32},\end{aligned}$$

which concludes the proof. □

6.4 Number of pieces in practice

We have seen in Theorem 6.3 that deep neural networks *can* have many more pieces than their shallow counterparts. This begs the question if deep neural networks tend to generate more pieces in practice. More formally: If we randomly initialize the weights of a neural network, what is the expected number of linear regions? Will this number scale exponentially with the depth? This question was analyzed in [81], and surprisingly, it was found that the number of pieces of randomly initialized neural networks typically does *not* depend exponentially on the depth. In Figure 6.4, we depict two neural networks, one shallow and one deep, that were randomly initialized according to He initialization [85]. Both neural networks have essentially the same number of pieces (114 and 110) and there is no clear indication that one has a deeper architecture than the other.

In the following, we will give a simplified version of the main result of [81] to show why random deep neural networks often behave like shallow neural networks.

We recall from Figure 6.1 that pieces are generated through composition of two functions f_1 and f_2, if the values of f_2 cross a level that is associated to a break point of f_1. In the case of a simple neuron of the form

$$\boldsymbol{x} \mapsto \sigma_{\mathrm{ReLU}}(\langle \boldsymbol{a}, h(\boldsymbol{x}) \rangle + b)$$

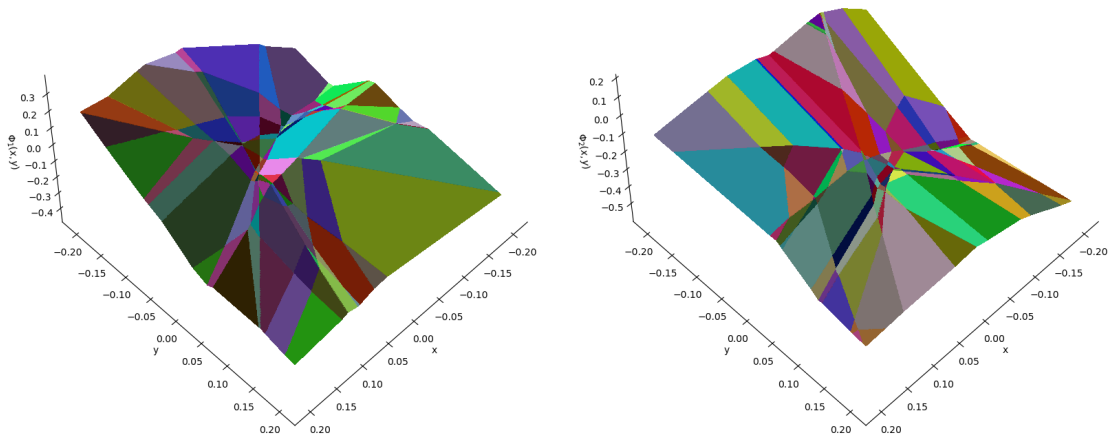


Figure 6.4: Two randomly initialized neural networks Φ_1 and Φ_2 with architectures $(\sigma_{\mathrm{ReLU}}; 1, 10, 10, 1)$ and $(\sigma_{\mathrm{ReLU}}; 1, 5, 5, 5, 5, 5, 1)$. The initialization scheme was He initialization [85]. The number of linear regions equals 114 and 110, respectively.

where h is a cpwl function, $\boldsymbol{a}$ is a vector, and b is a scalar, many pieces can be generated if $\langle \boldsymbol{a}, h(\boldsymbol{x}) \rangle$ crosses the $-b$ level often.

If $\boldsymbol{a}$, b are random variables, and we know that h does not oscillate too much, then we can quantify the probability of $\langle \boldsymbol{a}, h(\boldsymbol{x}) \rangle$ crossing the $-b$ level often. The following lemma from [113, Lemma 3.1] provides the details.

Lemma 6.7. *Let $c > 0$ and let $h\colon [0, c] \to \mathbb{R}$ be a cpwl function on $[0, c]$. Let $t \in \mathbb{N}$, let $A \subseteq \mathbb{R}$ be a Lebesgue measurable set, and assume that for every $y \in A$ it holds that*

$$|\{x \in [0, c] \,|\, h(x) = y\}| \geq t.$$

Then, $c\|h'\|_{L^\infty} \geq \|h'\|_{L^1} \geq |A| \cdot t$, where $|A|$ is the Lebesgue measure of A.

In particular, if h has at most $P \in \mathbb{N}$ pieces and $\|h'\|_{L^1}$ is finite, then it holds for all $\delta > 0$ that for all $t \leq P$

$$\mathbb{P}\left[|\{x \in [0, c] \,|\, h(x) = U\}| \geq t\right] \leq \frac{\|h'\|_{L^1}}{\delta t},$$
$$\mathbb{P}\left[|\{x \in [0, c] \,|\, h(x) = U\}| > P\right] = 0,$$

where U is a uniformly distributed variable on $[-\delta/2, \delta/2]$.

Proof. We will assume $c = 1$. The general case then follows by considering $\tilde{h}(x) = h(x/c)$.

Let for $(c_i)_{i=1}^{P+1} \subseteq [0, 1]$ with $c_1 = 0$, $c_{P+1} = 1$ and $c_i \leq c_{i+1}$ for all $i = 1, \ldots, P+1$ the pieces of h be given by $((c_i, c_{i+1}))_{i=1}^{P}$. We denote

$$V_1 \coloneqq [0, c_2], \quad V_i \coloneqq (c_i, c_{i+1}] \text{ for } i = 1, \ldots, P$$

and for $j = i, \dots, P$

$$\widetilde{V}_i \coloneqq \bigcup_{j=1}^{i-1} V_j.$$

We define, for $n \in \mathbb{N} \cup \{\infty\}$

$$T_{i,n} \coloneqq h(V_i) \cap \left\{ y \in A \,\middle|\, |\{x \in \widetilde{V}_i \,|\, h(x) = y\}| = n - 1 \right\}.$$

In words, $T_{i,n}$ contains the values of A that are hit on V_i for the nth time. Since h is cpwl, we observe that for all $i = 1, \dots, P$

(i) $T_{i,n_1} \cap T_{i,n_2} = \emptyset$ for all $n_1, n_2 \in \mathbb{N} \cup \{\infty\}$, $n_1 \neq n_2$,

(ii) $T_{i,\infty} \cup \bigcup_{n=1}^{\infty} T_{i,n} = h(V_i) \cap A$,

(iii) $T_{i,n} = \emptyset$ for all $P < n < \infty$,

(iv) $|T_{i,\infty}| = 0$.

Note that, since h is affine on V_i it holds that $h' = |h(V_i)|/|V_i|$ on V_i. Hence, for $t \leq P$

$$\begin{aligned}
\|h'\|_{L^1} &\geq \sum_{i=1}^{P} |h(V_i)| \geq \sum_{i=1}^{P} |h(V_i) \cap A| \\
&= \sum_{i=1}^{P} \left(\sum_{n=1}^{\infty} |T_{i,n}| \right) + |T_{i,\infty}| \\
&= \sum_{i=1}^{P} \sum_{n=1}^{\infty} |T_{i,n}| \\
&\geq \sum_{n=1}^{t} \sum_{i=1}^{P} |T_{i,n}|,
\end{aligned}$$

where the first equality follows by (i), (ii), the second by (iv), and the last inequality by (iii). Note that, by assumption for all $n \leq t$ every $y \in A$ is an element of $T_{i,n}$ or $T_{i,\infty}$ for some $i \leq P$. Therefore, by (iv)

$$\sum_{i=1}^{P} |T_{i,n}| \geq |A|,$$

which completes the proof. □

Lemma 6.7 applied to neural networks essentially states that, in a single neuron, if the bias term is chosen uniformly randomly on an interval of length δ, then the probability of generating at least t pieces by composition scales reciprocal to t.

Next, we will analyze how Lemma 6.7 implies an upper bound on the number of pieces generated in a randomly initialized neural network. For simplicity, we only consider random biases in the following, but mention that similar results hold if both the biases and weights are random [81].

Definition 6.8. Let $L \in \mathbb{N}$, $(d_0, d_1, \dots, d_L, 1) \in \mathbb{N}^{L+2}$ and $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ for $\ell = 0, \dots, L$. Furthermore, let $\delta > 0$ and let the bias vectors $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$, for $\ell = 0, \dots, L$, be random variables such that each entry of each $\boldsymbol{b}^{(\ell)}$ is independently and uniformly distributed on the interval $[-\delta/2, \delta/2]$. We call the associated ReLU neural network a **random-bias neural network**.

To apply Lemma 6.7 to a single neuron with random biases, we also need some bound on the derivative of the input to the neuron.

Definition 6.9. Let $L \in \mathbb{N}$, $(d_0, d_1, \dots, d_L, 1) \in \mathbb{N}^{L+2}$, and $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ and $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$ for $\ell = 0, \dots, L$. Moreover let $\delta > 0$.

For $\ell = 1, \dots, L+1$, $i = 1, \dots, d_\ell$ introduce the functions

$$\eta_{\ell,i}(\boldsymbol{x}; (\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)})_{j=0}^{\ell-1}) = (\boldsymbol{W}^{(\ell-1)} \boldsymbol{x}^{(\ell-1)})_i \qquad \text{for } \boldsymbol{x} \in \mathbb{R}^{d_0},$$

where $\boldsymbol{x}^{(\ell-1)}$ is as in (2.1.1). We call

$$\nu\left((\boldsymbol{W}^{(\ell)})_{\ell=1}^{L}, \delta\right) \coloneqq \max\left\{ \left\| \eta'_{\ell,i}(\,\cdot\,; (\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)})_{j=0}^{\ell-1}) \right\|_2 \,\middle|\, (\boldsymbol{b}^{(j)})_{j=0}^{L} \in \prod_{j=0}^{L} [-\delta/2, \delta/2]^{d_{j+1}}, \ell = 1, \dots, L, i = 1, \dots, d_\ell \right\}$$

the **maximal internal derivative** of Φ.

We can now formulate the main result of this section.

Theorem 6.10. *Let $L \in \mathbb{N}$ and let $(d_0, d_1, \dots, d_L, 1) \in \mathbb{N}^{L+2}$. Let $\delta \in (0, 1]$. Let $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$, for $\ell = 0, \dots, L$, be such that $\nu\left((\boldsymbol{W}^{(\ell)})_{\ell=0}^{L}, \delta\right) \leq C_\nu$ for a $C_\nu > 0$.*

For an associated random-bias neural network Φ, we have that for a line segment $\mathfrak{s} \subseteq \mathbb{R}^{d_0}$ of length 1

$$\mathbb{E}[\mathrm{Pieces}(\Phi, \mathfrak{s})] \leq 1 + d_1 + \frac{C_\nu}{\delta}(1 + (L-1)\ln(2\mathrm{width}(\Phi))) \sum_{j=2}^{L} d_j. \tag{6.4.1}$$

Proof. Let $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ for $\ell = 0, \dots, L$. Moreover, let $\boldsymbol{b}^{(\ell)} \in [-\delta/2, \delta/2]^{d_{\ell+1}}$ for $\ell = 0, \dots, L$ be uniformly distributed random variables. We denote

$$\begin{aligned} \theta_\ell \colon \mathfrak{s} &\to \mathbb{R}^{d_\ell} \\ \boldsymbol{x} &\mapsto (\eta_{\ell,i}(\boldsymbol{x}; (\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)})_{j=0}^{\ell-1}))_{i=1}^{d_\ell}. \end{aligned}$$

Let $\kappa\colon \mathfrak{s} \to [0,1]$ be an isomorphism. Since each coordinate of θ_ℓ is cpwl, there are points $\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{q_\ell} \in \mathfrak{s}$ with $\kappa(\boldsymbol{x}_j) < \kappa(\boldsymbol{x}_{j+1})$ for $j = 0, \dots, q_\ell - 1$, such that θ_ℓ is affine (as a function into $\mathbb{R}^{d_\ell}$) on $[\kappa(\boldsymbol{x}_j), \kappa(\boldsymbol{x}_{j+1})]$ for all $j = 0, \dots, q_\ell - 1$ as well as on $[0, \kappa(\boldsymbol{x}_0)]$ and $[\kappa(\boldsymbol{x}_{q_\ell}), 1]$.

We will now inductively find an upper bound on the q_ℓ.

Let $\ell = 2$, then

$$\theta_2(\boldsymbol{x}) = \boldsymbol{W}^{(1)} \sigma_{\mathrm{ReLU}}(\boldsymbol{W}^{(0)} \boldsymbol{x} + \boldsymbol{b}^{(0)}).$$

Since $\boldsymbol{W}^{(1)} \cdot + b^{(1)}$ is an affine function, it follows that θ_2 can only be non-affine in points where $\sigma_{\mathrm{ReLU}}(\boldsymbol{W}^{(0)} \cdot + \boldsymbol{b}^{(0)})$ is not affine. Therefore, θ_2 is only non-affine if one coordinate of $\boldsymbol{W}^{(0)} \cdot + \boldsymbol{b}^{(0)}$ intersects 0 nontrivially. This can happen at most d_1 times. We conclude that we can choose $q_2 = d_1$.

Next, let us find an upper bound on $q_{\ell+1}$ from q_ℓ. Note that

$$\theta_{\ell+1}(\boldsymbol{x}) = \boldsymbol{W}^{(\ell)} \sigma_{\mathrm{ReLU}}(\theta_\ell(\boldsymbol{x}) + \boldsymbol{b}^{(\ell-1)}).$$

Now $\theta_{\ell+1}$ is affine in every point $\boldsymbol{x} \in \mathfrak{s}$ where θ_ℓ is affine and $(\theta_\ell(\boldsymbol{x}) + \boldsymbol{b}^{(\ell-1)})_i \neq 0$ for all coordinates $i = 1, \dots, d_\ell$. As a result, we have that we can choose $q_{\ell+1}$ such that

$$q_{\ell+1} \leq q_\ell + \left| \{ \boldsymbol{x} \in \mathfrak{s} \,|\, (\theta_\ell(\boldsymbol{x}) + \boldsymbol{b}^{(\ell-1)})_i = 0 \text{ for at least one } i = 1, \dots, d_\ell \} \right|.$$

Therefore, for $\ell \geq 2$

$$\begin{aligned} q_{\ell+1} &\leq d_1 + \sum_{j=3}^{\ell} \left| \{ \boldsymbol{x} \in \mathfrak{s} \,|\, (\theta_j(\boldsymbol{x}) + \boldsymbol{b}^{(j)})_i = 0 \text{ for at least one } i = 1, \dots, d_j \} \right| \\ &\leq d_1 + \sum_{j=2}^{\ell} \sum_{i=1}^{d_j} \left| \{ \boldsymbol{x} \in \mathfrak{s} \,|\, \eta_{j,i}(\boldsymbol{x}) = -\boldsymbol{b}_i^{(j)} \} \right|. \end{aligned}$$

By Theorem 6.3, we have that

$$\mathrm{Pieces}\left(\eta_{\ell,i}(\,\cdot\,; (\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)})_{j=0}^{\ell-1}), \mathfrak{s} \right) \leq (2\mathrm{width}(\Phi))^{\ell-1}.$$

We define for $k \in \mathbb{N} \cup \{\infty\}$

$$p_{k,\ell,i} \coloneqq \mathbb{P}\left[\left| \{ \boldsymbol{x} \in \mathfrak{s} \,|\, \eta_{\ell,i}(\boldsymbol{x}) = -\boldsymbol{b}_i^{(\ell)} \} \right| \geq k \right]$$

Then by Lemma 6.7

$$p_{k,\ell,i} \leq \frac{C_\nu}{\delta k}$$

and for $k > (2\mathrm{width}(\Phi))^{\ell-1}$

$$p_{k,\ell,i} = 0.$$

It holds

$$\begin{aligned}
&\mathbb{E}\left[\sum_{j=2}^{L}\sum_{i=1}^{d_j}\left|\left\{\boldsymbol{x}\in\mathfrak{s}\,\middle|\,\eta_{j,i}(\boldsymbol{x})=-\boldsymbol{b}_i^{(j)}\right\}\right|\right]\\
\leq&\sum_{j=2}^{L}\sum_{i=1}^{d_j}\sum_{k=1}^{\infty}k\cdot\mathbb{P}\left[\left|\left\{\boldsymbol{x}\in\mathfrak{s}\,\middle|\,\eta_{j,i}(\boldsymbol{x})=-\boldsymbol{b}_i^{(j)}\right\}\right|=k\right]\\
\leq&\sum_{j=2}^{L}\sum_{i=1}^{d_j}\sum_{k=1}^{\infty}k\cdot(p_{k,j,i}-p_{k+1,j,i}).
\end{aligned}$$

The inner sum can be bounded by

$$\begin{aligned}
\sum_{k=1}^{\infty}k\cdot(p_{k,j,i}-p_{k+1,j,i})&=\sum_{k=1}^{\infty}k\cdot p_{k,j,i}-\sum_{k=1}^{\infty}k\cdot p_{k+1,j,i}\\
&=\sum_{k=1}^{\infty}k\cdot p_{k,j,i}-\sum_{k=2}^{\infty}(k-1)\cdot p_{k,j,i}\\
&=p_{1,j,i}+\sum_{k=2}^{\infty}p_{k,j,i}\\
&=\sum_{k=1}^{\infty}p_{k,j,i}\\
&\leq C_\nu\delta^{-1}\sum_{k=1}^{(2\mathrm{width}(\Phi))^{L-1}}\frac{1}{k}\\
&\leq C_\nu\delta^{-1}\left(1+\int_1^{(2\mathrm{width}(\Phi))^{L-1}}\frac{1}{x}\,\mathrm{d}x\right)\\
&\leq C_\nu\delta^{-1}(1+(L-1)\ln((2\mathrm{width}(\Phi)))).
\end{aligned}$$

We conclude that, in expectation, we can bound q_{L+1} by

$$d_1+C_\nu\delta^{-1}(1+(L-1)\ln(2\mathrm{width}(\Phi)))\sum_{j=2}^{L}d_j.$$

Finally, since $\theta_L=\Phi_{L+1}|_{\mathfrak{s}}$, it follows that

$$\mathrm{Pieces}(\Phi,\mathfrak{s})\leq q_{L+1}+1$$

which yields the result. □

Remark 6.11. We make the following observations about Theorem 6.10:

- *Non-exponential dependence on depth:* If we consider (6.4.1), we see that the number of pieces scales in expectation essentially like $\mathcal{O}(LN)$, where N is the total number of neurons of the architecture. This shows that in expectation, the number of pieces is linear in the number of layers, as opposed to the exponential upper bound of Theorem 6.3.

- *Maximal internal derivative:* Theorem 6.10 requires the weights to be chosen such that the maximal internal derivative is bounded by a certain number. However, if they are randomly initialized in such a way that with high probability the maximal internal derivative is bounded by a small number, then similar results can be shown. In practice, weights in the ℓth layer are often initialized according to a centered normal distribution with standard deviation $\sqrt{2/d_\ell}$, [85]. Due to the anti-proportionality of the variance to the width of the layers it is achieved that the internal derivatives remain bounded with high probability, independent of the width of the neural networks. This explaines the observation from Figure 6.4.

Bibliography and further reading

Establishing bounds on the number of linear regions of a ReLU network has been a popular tool to investigate the complexity of ReLU neural networks, see [150, 183, 4, 209, 81]. The bound presented in Section 6.1, is based on [226]. In addition to this bound, the paper also presents the depth separation result discussed in Section 6.3. The proof techniques employed there have inspired numerous subsequent works in the field.

Together with the lower bound on the number of required linear regions given in [61], this analysis shows how depth can be a limiting factor in terms of achievable convergence rates, as stated in Theorem 6.4.

For the construction of the sawtooth function in Section 6.2, and the depth separation result in Section 6.3 follow the arguments in [226, 227, 228]. Beyond Telgarsky's work, other notable depth separation results include [59, 198, 4]. Moreover, closely related to such statements is the 1987 thesis by Håstad [100], which considers the limitations of logic circuits in terms of depth.

Finally, the analysis of the number of pieces deep neural networks attained with random intialization (Section 6.4) is based on [81] and [113].

Exercises

Exercise 6.12. Let $-\infty < a < b < \infty$ and let $f \in C^3([a,b]) \backslash \mathbb{P}_1$. Denote by $p(\varepsilon) \in \mathbb{N}$ the minimal number of intervals partitioning $[a,b]$, such that a (not necessarily continuous) piecewise linear function on $p(\varepsilon)$ intervals can approximate f on $[a,b]$ uniformly up to error $\varepsilon > 0$. In this exercise, we wish to show

$$\liminf_{\varepsilon \searrow 0} p(\varepsilon)\sqrt{\varepsilon} > 0. \tag{6.4.2}$$

Therefore, we can find a constant $C > 0$ such that $\varepsilon \geq C p(\varepsilon)^{-2}$ for all $\varepsilon > 0$. This shows a variant of Theorem 6.2. Proceed as follows to prove (6.4.2):

(i) Fix $\varepsilon > 0$ and let $a = x_0 < x_1 \cdots < x_{p(\varepsilon)} = b$ be a partitioning into $p(\varepsilon)$ pieces. For $i = 0, \ldots, p(\varepsilon) - 1$ and $x \in [x_i, x_{i+1}]$ let

$$e_i(x) := f(x) - \left(f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) \right).$$

Show that $|e_i(x)| \leq 2\varepsilon$ for all $x \in [x_i, x_{i+1}]$.

(ii) With $h_i := x_{i+1} - x_i$ and $m_i := (x_i + x_{i+1})/2$ show that

$$\max_{x \in [x_i, x_{i+1}]} |e_i(x)| = \frac{h_i^2}{8} |f''(m_i)| + O(h_i^3).$$

(iii) Assuming that $c := \inf_{x \in [a,b]} |f''(x)| > 0$ show that

$$\liminf_{\varepsilon \searrow 0} p(\varepsilon)\sqrt{\varepsilon} \geq \frac{1}{4} \int_a^b \sqrt{|f''(x)|}\, \mathrm{d}x.$$

(iv) Conclude that (6.4.2) holds for general non-linear $f \in C^3([a,b])$.

Exercise 6.13. Show that, for $L = 1$, Theorem 6.3 holds for piecewise smooth functions, when replacing the number of affine pieces by the number of smooth pieces. These are defined by replacing "affine" by "smooth" (meaning C^∞) in Definition 6.1.

Exercise 6.14. Show that, for $L > 1$, Theorem 6.3 does *not*x hold for piecewise smooth functions, when replacing the number of affine pieces by the number of smooth pieces.

Exercise 6.15. For $p \in \mathbb{N}$, $p > 2$ and $n \in \mathbb{N}$, construct a function $h_n^{(p)}$ similar to h_n of (6.5), such that $h_n^{(p)} \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n, p)$ and such that $h_n^{(p)}$ has p^n pieces and size $O(p^2 n)$.

Chapter 7

Deep ReLU neural networks

In the previous chapter, we observed that many layers are a necessary prerequisite for ReLU neural networks to approximate smooth functions with high rates. To complement this observation, we now analyze which depth is sufficient to achieve good approximation rates for smooth functions.

To approximate smooth functions efficiently, one of the main tools in Chapter 4 was to rebuild polynomial-based functions, such as higher-order B-splines. For smooth activation functions, we were able to reproduce polynomials by using the nonlinearity of the activation functions. This argument certainly cannot be repeated for the *piecewise linear* ReLU. On the other hand, up until now, we have seen that deep ReLU neural networks are extremely efficient at producing the strongly oscillating sawtooth functions discussed in Lemma 6.5.

The main observation this chapter is that the efficient representation of sawtooth functions is intimately linked to the approximation of the square function and hence allows very efficient approximations of polynomial functions. This observation was first made by Dmitry Yarotsky [244] in 2016, and the present chapter is primarily based on this paper.

First, in Section 7.1, we will give an efficient neural network approximation of the squaring function. Second, in Section 7.2, we will demonstrate how the squaring neural network can be modified to yield a neural network that approximates the function that multiplies its inputs. Using these two tools, we conclude in Section 7.3 that deep ReLU neural networks can efficiently approximate k-times continuously differentiable functions with Hölder continuous derivatives.

7.1 The square function

In this section, we will show that the square function $x \mapsto x^2$ can be approximated very efficiently by a deep neural network.

Proposition 7.1. *Let $n \in \mathbb{N}$. Then*

$$s_n(x) := x - \sum_{j=1}^{n} \frac{h_j(x)}{2^{2j}}$$

is a piecewise linear function on $[0,1]$ with break points $x_{n,j} = j2^{-n}$, $j = 0, \dots, 2^n$. Moreover, $s_n(x_{n,k}) = x_{n,k}^2$ for all $k = 0, \dots, 2^n$, i.e. s_n is the piecewise linear interpolant of x^2 on $[0,1]$.

Proof. The statement holds for $n = 1$. We proceed by induction. Assume the statement holds for s_n and let $k \in \{0, \dots, 2^{n+1}\}$. By Lemma 6.5, $h_{n+1}(x_{n+1,k}) = 0$ whenever k is even. Hence for even $k \in \{0, \dots, 2^{n+1}\}$

$$\begin{aligned} s_{n+1}(x_{n+1,k}) &= x - \sum_{j=1}^{n+1} \frac{h_j(x_{n+1,k})}{2^{2j}} \\ &= s_n(x_{n+1,k}) - \frac{h_{n+1}(x_{n+1,k})}{2^{2(n+1)}} = s_n(x_{n+1,k}) = x_{n+1,k}^2, \end{aligned}$$

where we used the induction assumption $s_n(x_{n+1,k}) = x_{n+1,k}^2$ for $x_{n+1,k} = k2^{-(n+1)} = \frac{k}{2}2^{-n} = x_{n,k/2}$.

Now let $k \in \{1, \dots, 2^{n+1} - 1\}$ be odd. Then by Lemma 6.5, $h_{n+1}(x_{n+1,k}) = 1$. Moreover, since s_n is linear on $[x_{n,(k-1)/2}, x_{n,(k+1)/2}] = [x_{n+1,k-1}, x_{n+1,k+1}]$ and $x_{n+1,k}$ is the midpoint of this interval,

$$\begin{aligned} s_{n+1}(x_{n+1,k}) &= s_n(x_{n+1,k}) - \frac{h_{n+1}(x_{n+1,k})}{2^{2(n+1)}} \\ &= \frac{1}{2}(x_{n+1,k-1}^2 + x_{n+1,k+1}^2) - \frac{1}{2^{2(n+1)}} \\ &= \frac{(k-1)^2}{2^{2(n+1)+1}} + \frac{(k+1)^2}{2^{2(n+1)+1}} - \frac{2}{2^{2(n+1)+1}} \\ &= \frac{1}{2}\frac{2k^2}{2^{2(n+1)}} = \frac{k^2}{2^{2(n+1)}} = x_{n+1,k}^2. \end{aligned}$$

This completes the proof. □

Lemma 7.2. *For $n \in \mathbb{N}$, it holds*

$$\sup_{x \in [0,1]} |x^2 - s_n(x)| \leq 2^{-2n-1}.$$

Moreover $s_n \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n, 3)$, and $\mathrm{size}(s_n) \leq 7n$ and $\mathrm{depth}(s_n) = n$.

Proof. Set $e_n(x) := x^2 - s_n(x)$. Let x be in the interval $[x_{n,k}, x_{n,k+1}] = [k2^{-n}, (k+1)2^{-n}]$ of length 2^{-n}. Since s_n is the linear interpolant of x^2 on this interval, we have

$$|e_n'(x)| = \left| 2x - \frac{x_{n,k+1}^2 - x_{n,k}^2}{2^{-n}} \right| = \left| 2x - \frac{2k+1}{2^n} \right| \leq \frac{1}{2^n}.$$

Thus $e_n : [0,1] \to \mathbb{R}$ has Lipschitz constant 2^{-n}. Since $e_n(x_{n,k}) = 0$ for all $k = 0, \dots, 2^n$, and the length of the interval $[x_{n,k}, x_{n,k+1}]$ equals 2^{-n} we get

$$\sup_{x \in [0,1]} |e_n(x)| \leq \frac{1}{2} 2^{-n} 2^{-n} = 2^{-2n-1}.$$

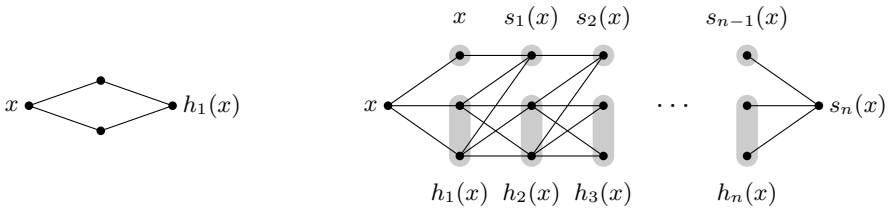


Figure 7.1: The neural networks $h_1(x) = \sigma_{\mathrm{ReLU}}(2x) - \sigma_{\mathrm{ReLU}}(4x-2)$ and $s_n(x) = \sigma_{\mathrm{ReLU}}(s_{n-1}(x)) - h_n(x)/2^{2n}$ where $h_n = h_1 \circ h_{n-1}$.

Finally, to see that s_n can be represented by a neural network of the claimed architecture, note that for $n \geq 2$

$$s_n(x) = x - \sum_{j=1}^{n} \frac{h_j(x)}{2^{2j}} = s_{n-1}(x) - \frac{h_n(x)}{2^{2n}} = \sigma_{\mathrm{ReLU}} \circ s_{n-1}(x) - \frac{h_1 \circ h_{n-1}(x)}{2^{2n}}.$$

Here we used that s_{n-1} is the piecewise linear interpolant of x^2, so that $s_{n-1}(x) \geq 0$ and thus $s_{n-1}(x) = \sigma_{\mathrm{ReLU}}(s_{n-1}(x))$ for all $x \in [0,1]$. Hence s_n is of depth n and width 3, see Figure 7.1. □

In conclusion, we have shown that $s_n : [0,1] \to [0,1]$ approximates the square function uniformly on $[0,1]$ with exponentially decreasing error in the neural network size. Note that due to Theorem 6.4, this would not be possible with a shallow neural network, which can at best interpolate x^2 on a partition of $[0,1]$ with polynomially many (w.r.t. the neural network size) pieces.

7.2 Multiplication

According to Lemma 7.2, depth can help in the approximation of $x \mapsto x^2$, which, on first sight, seems like a rather specific example. However, as we shall discuss in the following, this opens up a path towards fast approximation of functions with high regularity, e.g., $C^k([0,1]^d)$ for some $k > 1$. The crucial observation is that, via the polarization identity we can write the product of two numbers as a sum of squares

$$x \cdot y = \frac{(x+y)^2 - (x-y)^2}{4} \tag{7.2.1}$$

for all x, $y \in \mathbb{R}$. Efficient approximation of the operation of multiplication allows efficient approximation of polynomials. Those in turn are well-known to be good approximators for functions exhibiting $k \in \mathbb{N}$ derivatives. Before exploring this idea further in the next section, we first make precise the observation that neural networks can efficiently approximate the multiplication of real numbers.

We start with the multiplication of two numbers, in which case neural networks of logarithmic size in the desired accuracy are sufficient.

Lemma 7.3. *For every $\varepsilon > 0$ there exists a ReLU neural network $\Phi_\varepsilon^\times : [-1,1]^2 \to [-1,1]$ such that*

$$\sup_{x,y\in[-1,1]} |x \cdot y - \Phi_\varepsilon^\times(x,y)| \leq \varepsilon,$$

and it holds $\operatorname{size}(\Phi_\varepsilon^\times) \leq C \cdot (1 + |\log(\varepsilon)|)$ *and* $\operatorname{depth}(\Phi_\varepsilon^\times) \leq C \cdot (1 + |\log(\varepsilon)|)$ *for a constant $C > 0$ independent of ε. Moreover $\Phi_\varepsilon^\times(x,y) = 0$ if $x = 0$ or $y = 0$.*

Proof. With $n = \lceil |\log_4(\varepsilon)| \rceil$, define the neural network

$$\begin{aligned}\Phi_\varepsilon^\times(x,y) :=& s_n\left(\frac{\sigma_{\mathrm{ReLU}}(x+y) + \sigma_{\mathrm{ReLU}}(-x-y)}{2}\right) \\ &- s_n\left(\frac{\sigma_{\mathrm{ReLU}}(x-y) + \sigma_{\mathrm{ReLU}}(y-x)}{2}\right). \end{aligned} \tag{7.2.2}$$

Since $|a| = \sigma_{\mathrm{ReLU}}(a) + \sigma_{\mathrm{ReLU}}(-a)$, by (7.2.1) we have for all x, $y \in [-1,1]$

$$\begin{aligned}\left|x \cdot y - \Phi_\varepsilon^\times(x,y)\right| &= \left|\frac{(x+y)^2 - (x-y)^2}{4} - \left(s_n\left(\frac{|x+y|}{2}\right) - s_n\left(\frac{|x-y|}{2}\right)\right)\right| \\ &= \left|\frac{4(\frac{x+y}{2})^2 - 4(\frac{x-y}{2})^2}{4} - \frac{4s_n(\frac{|x+y|}{2}) - 4s_n(\frac{|x-y|}{2})}{4}\right| \\ &\leq \frac{4(2^{-2n-1} + 2^{-2n-1})}{4} = 4^{-n} \leq \varepsilon,\end{aligned}$$

where we used $|x+y|$, $|x-y| \in [0,1]$. We have $\operatorname{depth}(\Phi_\varepsilon^\times) = 1 + \operatorname{depth}(s_n) = 1 + n \leq 1 + \lceil \log_4(\varepsilon) \rceil$ and $\operatorname{size}(\Phi_\varepsilon^\times) \leq C + 2\operatorname{size}(s_n) \leq Cn \leq C \cdot (1 - \log(\varepsilon))$ for some constant $C > 0$.

The fact that $\Phi_\varepsilon^\times$ maps from $[-1,1]^2 \to [-1,1]$ follows by (7.2.2) and because $s_n : [0,1] \to [0,1]$. Finally, if $x = 0$, then $\Phi_\varepsilon^\times(x,y) = s_n(|x+y|) - s_n(|x-y|) = s_n(|y|) - s_n(|y|) = 0$. If $y = 0$ the same argument can be made. □

In a similar way as in Proposition 4.8 and Lemma 5.11, we can apply operations with two inputs in the form of a binary tree to extend them to an operation on arbitrary many inputs.

Proposition 7.4. *For every $n \geq 2$ and $\varepsilon > 0$ there exists a ReLU neural network $\Phi_{n,\varepsilon}^\times : [-1,1]^n \to [-1,1]$ such that*

$$\sup_{x_j\in[-1,1]} \left|\prod_{j=1}^n x_j - \Phi_{n,\varepsilon}^\times(x_1,\dots,x_n)\right| \leq \varepsilon,$$

and it holds $\operatorname{size}(\Phi_{n,\varepsilon}^\times) \leq Cn \cdot (1 + |\log(\varepsilon/n)|)$ *and* $\operatorname{depth}(\Phi_{n,\varepsilon}^\times) \leq C\log(n)(1 + |\log(\varepsilon/n)|)$ *for a constant $C > 0$ independent of ε and n.*

Proof. We begin with the case $n = 2^k$. For $k = 1$ let $\tilde{\Phi}^{\times}_{2,\delta} := \Phi^{\times}_{\delta}$. If $k \geq 2$ let

$$\tilde{\Phi}^{\times}_{2^k,\delta} := \Phi^{\times}_{\delta} \circ \left(\tilde{\Phi}^{\times}_{2^{k-1},\delta}, \tilde{\Phi}^{\times}_{2^{k-1},\delta}\right).$$

Using Lemma 7.3, we find that this neural network has depth bounded by

$$\operatorname{depth}\left(\tilde{\Phi}^{\times}_{2^k,\delta}\right) \leq k\operatorname{depth}(\Phi^{\times}_{\delta}) \leq Ck \cdot (1 + |\log(\delta)|) \leq C\log(n)(1 + |\log(\delta)|).$$

Observing that the number of occurences of $\Phi^{\times}_{\delta}$ equals $\sum_{j=0}^{k-1} 2^j \leq n$, the size of $\tilde{\Phi}^{\times}_{2^k,\delta}$ can bounded by $Cn\operatorname{size}(\Phi^{\times}_{\delta}) \leq Cn \cdot (1 + |\log(\delta)|)$.

To estimate the approximation error, denote with $\boldsymbol{x} = (x_j)_{j=1}^{2^k}$

$$e_k := \sup_{x_j \in [-1,1]} \left| \prod_{j \leq 2^k} x_j - \tilde{\Phi}^{\times}_{2^k,\delta}(\boldsymbol{x}) \right|.$$

Then, using short notation of the type $\boldsymbol{x}_{\leq 2^{k-1}} := (x_1, \dots, x_{2^{k-1}})$,

$$\begin{aligned}
e_k &= \sup_{x_j \in [-1,1]} \left| \prod_{j=1}^{2^k} x_j - \Phi^{\times}_{\delta}\left(\tilde{\Phi}^{\times}_{2^{k-1},\delta}(\boldsymbol{x}_{\leq 2^{k-1}}), \tilde{\Phi}^{\times}_{2^{k-1},\delta}(\boldsymbol{x}_{>2^{k-1}})\right) \right| \\
&\leq \delta + \sup_{x_j \in [-1,1]} \left(\left| \prod_{j \leq 2^{k-1}} x_j \right| e_{k-1} + \left| \tilde{\Phi}^{\times}_{2^{k-1},\delta}(\boldsymbol{x}_{>2^{k-1}}) \right| e_{k-1} \right) \\
&\leq \delta + 2e_{k-1} \leq \delta + 2(\delta + 2e_{k-2}) \leq \cdots \leq \delta \sum_{j=0}^{k-2} 2^j + 2^{k-1} e_1 \\
&\leq 2^k \delta = n\delta = \varepsilon.
\end{aligned}$$

Here we used $e_1 \leq \delta$, and that $\tilde{\Phi}^{\times}_{2^k,\delta}$ maps $[-1,1]^{2^{k-1}}$ to $[-1,1]$, which is a consequence of Lemma 7.3.

The case for general $n \geq 2$ (not necessarily $n = 2^k$) is treated similar as in Lemma 5.11, by replacing some $\Phi^{\times}_{\delta}$ neural networks with identity neural networks.

Finally, setting $\delta := \varepsilon/n$ and $\Phi^{\times}_{n,\varepsilon} := \tilde{\Phi}^{\times}_{n,\delta}$ concludes the proof. □

7.3 $C^{k,s}$ functions

We will now discuss the implications of our observations in the previous sections for the approximation of functions in the class $C^{k,s}$.

Definition 7.5. Let $k \in \mathbb{N}_0$, $s \in [0,1]$ and $\Omega \subseteq \mathbb{R}^d$. Then

$$\begin{aligned}
\|f\|_{C^{k,s}(\Omega)} := &\sup_{\boldsymbol{x} \in \Omega} \max_{\{\boldsymbol{\alpha} \in \mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}| \leq k\}} |D^{\boldsymbol{\alpha}} f(\boldsymbol{x})| \\
&+ \sup_{\boldsymbol{x} \neq \boldsymbol{y} \in \Omega} \max_{\{\boldsymbol{\alpha} \in \mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}| = k\}} \frac{|D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) - D^{\boldsymbol{\alpha}} f(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|_2^s},
\end{aligned} \tag{7.3.1}$$

and we denote by $C^{k,s}(\Omega)$ the set of functions $f \in C^k(\Omega)$ for which $\|f\|_{C^{k,s}(\Omega)} < \infty$.

Note that these spaces are ordered according to

$$C^k(\Omega) \supseteq C^{k,s}(\Omega) \supseteq C^{k,t}(\Omega) \supseteq C^{k+1}(\Omega)$$

for all $0 < s \leq t \leq 1$.

In order to state our main result, we first recall a version of Taylor's remainder formula for $C^{k,s}(\Omega)$ functions.

Lemma 7.6. *Let $d \in \mathbb{N}$, $k \in \mathbb{N}$, $s \in [0,1]$, $\Omega = [0,1]^d$ and $f \in C^{k,s}(\Omega)$. Then for all $\boldsymbol{a}$, $\boldsymbol{x} \in \Omega$*

$$f(\boldsymbol{x}) = \sum_{\{\boldsymbol{\alpha} \in \mathbb{N}_0^d \,|\, 0 \leq |\boldsymbol{\alpha}| \leq k\}} \frac{D^{\boldsymbol{\alpha}} f(\boldsymbol{a})}{\boldsymbol{\alpha}!} (\boldsymbol{x} - \boldsymbol{a})^{\boldsymbol{\alpha}} + R_k(\boldsymbol{x}) \tag{7.3.2}$$

where with $h := \max_{i \leq d} |a_i - x_i|$ we have $|R_k(\boldsymbol{x})| \leq h^{k+s} \frac{d^{k+1/2}}{k!} \|f\|_{C^{k,s}(\Omega)}$.

Proof. First, for a function $g \in C^k(\mathbb{R})$ and a, $t \in \mathbb{R}$

$$\begin{aligned} g(t) &= \sum_{j=0}^{k-1} \frac{g^{(j)}(a)}{j!} (t-a)^j + \frac{g^{(k)}(\xi)}{k!} (t-a)^k \\ &= \sum_{j=0}^{k} \frac{g^{(j)}(a)}{j!} (t-a)^j + \frac{g^{(k)}(\xi) - g^{(k)}(a)}{k!} (t-a)^k, \end{aligned}$$

for some ξ between a and t. Now let $f \in C^{k,s}(\mathbb{R}^d)$ and $\boldsymbol{a}$, $\boldsymbol{x} \in \mathbb{R}^d$. Thus with $g(t) := f(\boldsymbol{a} + t \cdot (\boldsymbol{x} - \boldsymbol{a}))$ holds for $f(\boldsymbol{x}) = g(1)$

$$f(\boldsymbol{x}) = \sum_{j=0}^{k-1} \frac{g^{(j)}(0)}{j!} + \frac{g^{(k)}(\xi)}{k!}.$$

By the chain rule

$$g^{(j)}(t) = \sum_{\{\boldsymbol{\alpha} \in \mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}| = j\}} \binom{j}{\boldsymbol{\alpha}} D^{\boldsymbol{\alpha}} f(\boldsymbol{a} + t \cdot (\boldsymbol{x} - \boldsymbol{a}))(\boldsymbol{x} - \boldsymbol{a})^{\boldsymbol{\alpha}},$$

where we use the multivariate notations $\binom{j}{\boldsymbol{\alpha}} = \frac{j!}{\boldsymbol{\alpha}!} = \frac{j!}{\prod_{j=1}^d \alpha_j!}$ and $(\boldsymbol{x} - \boldsymbol{a})^{\boldsymbol{\alpha}} = \prod_{j=1}^d (x_j - a_j)^{\alpha_j}$.

Hence

$$f(\boldsymbol{x}) = \underbrace{\sum_{\{\boldsymbol{\alpha}\in\mathbb{N}_0^d \,|\, 0\leq|\boldsymbol{\alpha}|\leq k\}} \frac{D^{\boldsymbol{\alpha}} f(\boldsymbol{a})}{\boldsymbol{\alpha}!}(\boldsymbol{x}-\boldsymbol{a})^{\boldsymbol{\alpha}}}_{\in\mathbb{P}_k} + \underbrace{\sum_{|\boldsymbol{\alpha}|=k} \frac{D^{\boldsymbol{\alpha}} f(\boldsymbol{a}+\xi\cdot(\boldsymbol{x}-\boldsymbol{a})) - D^{\boldsymbol{\alpha}} f(\boldsymbol{a})}{\boldsymbol{\alpha}!}(\boldsymbol{x}-\boldsymbol{a})^{\boldsymbol{\alpha}}}_{=:R_k},$$

for some $\xi \in [0,1]$. Using the definition of h, the remainder term can be bounded by

$$\begin{aligned}|R_k| &\leq h^k \max_{|\boldsymbol{\alpha}|=k} \sup_{\substack{\boldsymbol{x}\in\Omega \\ t\in[0,1]}} |D^{\boldsymbol{\alpha}} f(\boldsymbol{a}+t\cdot(\boldsymbol{x}-\boldsymbol{a})) - D^{\boldsymbol{\alpha}} f(\boldsymbol{a})| \frac{1}{k!} \sum_{\{\boldsymbol{\alpha}\in\mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}|=k\}} \binom{k}{\boldsymbol{\alpha}} \\ &\leq h^{k+s} \frac{d^{k+\frac{1}{2}}}{k!} \|f\|_{C^{k,s}([0,1]^d)},\end{aligned}$$

where we used (7.3.1), $\|\boldsymbol{x}-\boldsymbol{a}\|_2 \leq \sqrt{d}h$ and $\sum_{\{\boldsymbol{\alpha}\in\mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}|=k\}} \binom{k}{\boldsymbol{\alpha}} = (1+\cdots+1)^k = d^k$ by the multinomial formula. □

We now come to the main statement of this section. Up to logarithmic terms, it shows the convergence rate $(k+s)/d$ for approximating functions in $C^{k,s}([0,1]^d)$.

Theorem 7.7. *Let $d \in \mathbb{N}$, $k \in \mathbb{N}_0$, $s \in [0,1]$, and $\Omega = [0,1]^d$. Then, there exists a constant $C > 0$ such that for every $f \in C^{k,s}(\Omega)$ and every $N \geq 2$ there exists a ReLU neural network Φ_N^f such that*

$$\sup_{\boldsymbol{x}\in\Omega} |f(\boldsymbol{x}) - \Phi_N^f(\boldsymbol{x})| \leq C N^{-\frac{k+s}{d}} \|f\|_{C^{k,s}(\Omega)}, \tag{7.3.3}$$

$\mathrm{size}(\Phi_N^f) \leq CN\log(N)$ and $\mathrm{depth}(\Phi_N^f) \leq C\log(N)$.

Proof. The idea of the proof is to use the so-called "partition of unity method": First we will construct a partition of unity $(\varphi_{\boldsymbol{\nu}})_{\boldsymbol{\nu}}$, such that each $\varphi_{\boldsymbol{\nu}}$ has support on a $O(1/M)$ neighborhood of a point $\boldsymbol{\eta} \in \Omega$. On each of these neighborhoods we will use the local Taylor polynomial $p_{\boldsymbol{\nu}}$ of f around $\boldsymbol{\eta}$ to approximate the function. Then $\sum_{\boldsymbol{\nu}} \varphi_{\boldsymbol{\nu}} p_{\boldsymbol{\nu}}$ gives an approximation to f on Ω. This approximation can be emulated by a neural network of the type $\sum_{\boldsymbol{\nu}} \Phi_\varepsilon^\times(\varphi_{\boldsymbol{\nu}}, \hat{p}_{\boldsymbol{\nu}})$, where $\hat{p}_{\boldsymbol{\nu}}$ is an neural network approximation to the polynomial $p_{\boldsymbol{\nu}}$.

It suffices to show the theorem in the case where

$$\max\left\{\frac{d^{k+1/2}}{k!}, \exp(d)\right\} \|f\|_{C^{k,s}(\Omega)} \leq 1.$$

The general case can then be immediately deduced by a scaling argument.

Step 1. We construct the neural network. Define

$$M := \lceil N^{1/d} \rceil \qquad \text{and} \qquad \varepsilon := N^{-\frac{k+s}{d}}. \tag{7.3.4}$$

Consider a uniform simplicial mesh with nodes $\{\boldsymbol{\nu}/M \,|\, \boldsymbol{\nu} \leq M\}$ where $\boldsymbol{\nu}/M := (\nu_1/M, \dots, \nu_d/M)$, and where "$\boldsymbol{\nu} \leq M$" is short for $\{\boldsymbol{\nu} \in \mathbb{N}_0^d \,|\, \nu_i \leq M \text{ for all } i \leq d\}$. We denote by $\varphi_{\boldsymbol{\nu}}$ the cpwl basis function on this mesh such that $\varphi_{\boldsymbol{\nu}}(\boldsymbol{\nu}/M) = 1$ and $\varphi_{\boldsymbol{\nu}}(\boldsymbol{\mu}/M) = 0$ whenever $\boldsymbol{\mu} \neq \boldsymbol{\nu}$. As shown in Chapter 5, $\varphi_{\boldsymbol{\nu}}$ is a neural network of size $O(1)$. Then

$$\sum_{\boldsymbol{\nu} \leq M} \varphi_{\boldsymbol{\nu}} \equiv 1 \qquad \text{on } \Omega, \tag{7.3.5}$$

is a partition of unity. Moreover, observe that

$$\operatorname{supp}(\varphi_{\boldsymbol{\nu}}) \subseteq \left\{ \boldsymbol{x} \in \Omega \,\middle|\, \left\| \boldsymbol{x} - \frac{\boldsymbol{\nu}}{M} \right\|_\infty \leq \frac{1}{M} \right\}, \tag{7.3.6}$$

where $\|\boldsymbol{x}\|_\infty = \max_{i \leq d} |x_i|$.

For each $\boldsymbol{\nu} \leq M$ define the multivariate polynomial

$$p_{\boldsymbol{\nu}}(\boldsymbol{x}) := \sum_{|\boldsymbol{\alpha}| \leq k} \frac{D^{\boldsymbol{\alpha}} f\left(\frac{\boldsymbol{\nu}}{M}\right)}{\boldsymbol{\alpha}!} \left(\boldsymbol{x} - \frac{\boldsymbol{\nu}}{M}\right)^{\boldsymbol{\alpha}} \in \mathbb{P}_k,$$

and the approximation

$$\hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x}) := \sum_{|\boldsymbol{\alpha}| \leq k} \frac{D^{\boldsymbol{\alpha}} f\left(\frac{\boldsymbol{\nu}}{M}\right)}{\boldsymbol{\alpha}!} \Phi^\times_{|\boldsymbol{\alpha}|,\varepsilon}\left(x_{i_{\boldsymbol{\alpha},1}} - \frac{\nu_{i_{\boldsymbol{\alpha},1}}}{M}, \dots, x_{i_{\boldsymbol{\alpha},k}} - \frac{\nu_{i_{\boldsymbol{\alpha},k}}}{M}\right),$$

where $(i_{\boldsymbol{\alpha},1}, \dots, i_{\boldsymbol{\alpha},k}) \in \{0, \dots, d\}^k$ is arbitrary but fixed such that $|\{j \,|\, i_{\boldsymbol{\alpha},j} = r\}| = \alpha_r$ for all $r = 1, \dots, d$.

Finally, define

$$\Phi_N^f := \sum_{\boldsymbol{\nu} \leq M} \Phi_\varepsilon^\times(\varphi_{\boldsymbol{\nu}}, \hat{p}_{\boldsymbol{\nu}}), \tag{7.3.7}$$

where the precise values of $\varepsilon > 0$ and $M \in \mathbb{N}$ will be chosen at the end of Step 2.

Step 2. We bound the approximation error. First, for each $\boldsymbol{x} \in \Omega$, using (7.3.5) and (7.3.6)

$$\begin{aligned}\left| f(\boldsymbol{x}) - \sum_{\boldsymbol{\nu} \leq M} \varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) p_{\boldsymbol{\nu}}(\boldsymbol{x}) \right| &\leq \sum_{\boldsymbol{\nu} \leq M} |\varphi_{\boldsymbol{\nu}}(\boldsymbol{x})| |p_{\boldsymbol{\nu}}(\boldsymbol{x}) - f(\boldsymbol{x})| \\ &\leq \max_{\boldsymbol{\nu} \leq M} \sup_{\{\boldsymbol{y} \in \Omega \,|\, \|\frac{\boldsymbol{\nu}}{M} - \boldsymbol{y}\|_\infty \leq \frac{1}{M}\}} |f(\boldsymbol{y}) - p_{\boldsymbol{\nu}}(\boldsymbol{y})|.\end{aligned}$$

By Lemma 7.6 we obtain

$$\sup_{\boldsymbol{x} \in \Omega} \left| f(\boldsymbol{x}) - \sum_{\boldsymbol{\nu} \leq M} \varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) p_{\boldsymbol{\nu}}(\boldsymbol{x}) \right| \leq M^{-(k+s)} \frac{d^{k+\frac{1}{2}}}{k!} \|f\|_{C^{k,s}(\Omega)} \leq M^{-(k+s)}. \tag{7.3.8}$$

Next, fix $\boldsymbol{\nu} \leq M$ and $\boldsymbol{y} \in \Omega$ such that $\|\boldsymbol{\nu}/M - \boldsymbol{y}\|_\infty \leq 1/M \leq 1$. Then by Proposition 7.4

$$\begin{aligned}|p_{\boldsymbol{\nu}}(\boldsymbol{y}) - \hat{p}_{\boldsymbol{\nu}}(\boldsymbol{y})| &\leq \sum_{|\boldsymbol{\alpha}|\leq k} \frac{D^{\boldsymbol{\alpha}} f\left(\frac{\boldsymbol{\nu}}{M}\right)}{\boldsymbol{\alpha}!} \left| \prod_{j=1}^{k} \left(y_{i_{\boldsymbol{\alpha},j}} - \frac{\nu_{i_{\boldsymbol{\alpha},j}}}{M}\right) \right. \\ &\qquad \left. - \Phi^{\times}_{|\boldsymbol{\alpha}|,\varepsilon}\left(y_{i_{\boldsymbol{\alpha},1}} - \frac{\nu_{i_{\boldsymbol{\alpha},1}}}{M}, \dots, y_{i_{\boldsymbol{\alpha},k}} - \frac{i_{\boldsymbol{\alpha},k}}{M}\right)\right| \\ &\leq \varepsilon \sum_{|\boldsymbol{\alpha}|\leq k} \frac{D^{\boldsymbol{\alpha}} f(\frac{\boldsymbol{\nu}}{M})}{\boldsymbol{\alpha}!} \leq \varepsilon \exp(d) \|f\|_{C^{k,s}(\Omega)} \leq \varepsilon, \end{aligned} \tag{7.3.9}$$

where we used $|D^{\boldsymbol{\alpha}} f(\boldsymbol{\nu}/M)| \leq \|f\|_{C^{k,s}(\Omega)}$ and

$$\sum_{\{\boldsymbol{\alpha}\in\mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}|\leq k\}} \frac{1}{\boldsymbol{\alpha}!} = \sum_{j=0}^{k} \frac{1}{j!} \sum_{\{\boldsymbol{\alpha}\in\mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}|=j\}} \frac{j!}{\boldsymbol{\alpha}!} = \sum_{j=0}^{k} \frac{d^j}{j!} \leq \sum_{j=0}^{\infty} \frac{d^j}{j!} = \exp(d).$$

Similarly, one shows that

$$|\hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x})| \leq \exp(d)\|f\|_{C^{k,s}(\Omega)} \leq 1 \qquad \text{for all } \boldsymbol{x} \in \Omega.$$

Fix $\boldsymbol{x} \in \Omega$. Then $\boldsymbol{x}$ belongs to a simplex of the mesh, and thus $\boldsymbol{x}$ can be in the support of at most $d+1$ (the number of nodes of a simplex) functions $\varphi_{\boldsymbol{\nu}}$. Moreover, Lemma 7.3 implies that $\operatorname{supp} \Phi^{\times}_{\varepsilon}(\varphi_{\boldsymbol{\nu}}(\cdot), \hat{p}_{\boldsymbol{\nu}}(\cdot)) \subseteq \operatorname{supp} \varphi_{\boldsymbol{\nu}}$. Hence, using Lemma 7.3 and (7.3.9)

$$\begin{aligned}&\left| \sum_{\boldsymbol{\nu}\leq M} \varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) p_{\boldsymbol{\nu}}(\boldsymbol{x}) - \sum_{\boldsymbol{\nu}\leq M} \Phi^{\times}_{\varepsilon}(\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}), \hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x})) \right| \\ &\quad \leq \sum_{\{\boldsymbol{\nu}\leq M \,|\, \boldsymbol{x}\in\operatorname{supp}\varphi_{\boldsymbol{\nu}}\}} \left(|\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) p_{\boldsymbol{\nu}}(\boldsymbol{x}) - \varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) \hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x})| \right. \\ &\qquad \left. + |\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) \hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x}) - \Phi^{\times}_{\varepsilon}(\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}), \hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x}))| \right) \\ &\quad \leq \varepsilon + (d+1)\varepsilon = (d+2)\varepsilon. \end{aligned}$$

In total, together with (7.3.8)

$$\sup_{\boldsymbol{x}\in\Omega} |f(\boldsymbol{x}) - \Phi^f_N(\boldsymbol{x})| \leq M^{-(k+s)} + \varepsilon \cdot (d+2).$$

With our choices in (7.3.4) this yields the error bound (7.3.3).

Step 3. It remains to bound the size and depth of the neural network in (7.3.7).

By Lemma 5.17, for each $0 \leq \boldsymbol{\nu} \leq M$ we have

$$\operatorname{size}(\varphi_{\boldsymbol{\nu}}) \leq C \cdot (1 + k_{\mathcal{T}}), \qquad \operatorname{depth}(\varphi_{\boldsymbol{\nu}}) \leq C \cdot (1 + \log(k_{\mathcal{T}})), \tag{7.3.10}$$

where $k_{\mathcal{T}}$ is the maximal number of simplices attached to a node in the mesh. Note that $k_{\mathcal{T}}$ is independent of M, so that the size and depth of $\varphi_{\boldsymbol{\nu}}$ are bounded by a constant C_φ independent of M.

Lemma 7.3 and Proposition 7.4 thus imply with our choice of $\varepsilon = N^{-(k+s)/d}$

$$\begin{aligned}\operatorname{depth}(\Phi_N^f) &= \operatorname{depth}(\Phi_\varepsilon^\times) + \max_{\boldsymbol{\nu}\leq M} \operatorname{depth}(\varphi_{\boldsymbol{\eta}}) + \max_{\boldsymbol{\nu}\leq M} \operatorname{depth}(\hat{p}_{\boldsymbol{\nu}}) \\ &\leq C\cdot(1+|\log(\varepsilon)|+C_\varphi) + \operatorname{depth}(\Phi_{k,\varepsilon}^\times) \\ &\leq C\cdot(1+|\log(\varepsilon)|+C_\varphi) \\ &\leq C\cdot(1+\log(N))\end{aligned}$$

for some constant $C>0$ depending on k and d (we use "C" to denote a generic constant that can change its value in each line).

To bound the size, we first observe with Lemma 5.4 that

$$\operatorname{size}(\hat{p}_{\boldsymbol{\nu}}) \leq C\cdot\left(1+\sum_{|\boldsymbol{\alpha}|\leq k}\operatorname{size}\left(\Phi_{|\boldsymbol{\alpha}|,\varepsilon}^\times\right)\right) \leq C\cdot(1+|\log(\varepsilon)|)$$

for some C depending on k. Thus, for the size of Φ_N^f we obtain with $M=\lceil N^{1/d}\rceil$

$$\begin{aligned}\operatorname{size}(\Phi_N^f) &\leq C\cdot\left(1+\sum_{\boldsymbol{\nu}\leq M}\left(\operatorname{size}(\Phi_\varepsilon^\times)+\operatorname{size}(\varphi_{\boldsymbol{\nu}})+\operatorname{size}(\hat{p}_{\boldsymbol{\nu}})\right)\right) \\ &\leq C\cdot(1+M)^d(1+|\log(\varepsilon)|+C_\varphi) \\ &\leq C\cdot(1+N^{1/d})^d(1+C_\varphi+\log(N)) \\ &\leq CN\log(N),\end{aligned}$$

which concludes the proof. □

Theorem 7.7 shows the convergence rate $(k+s)/d$ for approximating a $C^{k,s}$-function $f:[0,1]^d\to\mathbb{R}$. As long as k is large, in principle we can achieve arbitrarily large (and d-independent if $k\geq d$) convergence rates. Crucially, and in contrast to Theorem 5.22, achieving error $N^{-\frac{k+s}{d}}$ requires the neural networks to be of size $O(N\log(N))$ and depth $O(\log(N))$, i.e. to get more and more accurate approximations, the neural network depth is required to increase.

Remark 7.8. Under the stronger assumption that f is an analytic function (in particular such an f is in C^∞), one can show exponential convergence rates for ReLU networks of the type $\exp(-\beta N^{1/(d+1)})$ for some fixed $\beta>0$ and where N corresponds again to the neural network size (up to logarithmic terms), see [57, 164].

Remark 7.9. Let $L:\boldsymbol{x}\mapsto\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}:\mathbb{R}^d\to\mathbb{R}^d$ be a bijective affine transformation and set $\Omega := L([0,1]^d)\subseteq\mathbb{R}^d$. Then for a function $f\in C^{k,s}(\Omega)$, by Theorem 7.7 there exists a neural network Φ_N^f such that

$$\begin{aligned}\sup_{\boldsymbol{x}\in\Omega}|f(\boldsymbol{x})-\Phi_N^f(L^{-1}(\boldsymbol{x}))| &= \sup_{\boldsymbol{x}\in[0,1]^d}|f(L(\boldsymbol{x}))-\Phi_N^f(L^{-1}(\boldsymbol{x}))| \\ &\leq C\|f\circ L\|_{C^{k,s}([0,1]^d)}N^{-\frac{k+s}{d}}.\end{aligned}$$

Since for $\boldsymbol{x}\in[0,1]^d$ holds $|f(L(\boldsymbol{x}))|\leq\sup_{\boldsymbol{y}\in\Omega}|f(\boldsymbol{y})|$ and if $\mathbf{0}\neq\boldsymbol{\alpha}\in\mathbb{N}_0^d$ is a multiindex $|D^{\boldsymbol{\alpha}}(f(L(\boldsymbol{x}))|\leq\|A\|_2^{|\boldsymbol{\alpha}|}\sup_{\boldsymbol{y}\in\Omega}|D^{\boldsymbol{\alpha}}f(\boldsymbol{y})|$, we have $\|f\circ L\|_{C^{k,s}([0,1]^d)}\leq(1+\|A\|_2^{k+s})\|f\|_{C^{k,s}(\Omega)}$. Thus the convergence rate $N^{-\frac{k+s}{d}}$ is achieved on every set of the type $L([0,1]^d)$ for an affine map L, and in particular on every hypercube $\times_{j=1}^d[a_j,b_j]$.

Bibliography and further reading

This chapter is based on the seminal 2017 paper by Yarotsky [244], where the construction of approximating the square function, the multiplication, and polynomials (discussed in Sections 7.1 and 7.2) was first introduced and analyzed. The construction relies on the sawtooth function discussed in Section 6.2 and originally introduced by Telgarsky in [226]. Yarotsky's work has since sparked a large body of research, as it allows to lift polynomial approximation theory to neural network classes. Convergence results based on this type of argument include for example [172, 58, 148, 57, 164].

The approximation result derived in Section 7.3 for Hölder continuous functions follows by standard approximation theory for piecewise polynomial functions. We point out that similar results for the approximation of functions in C^k or functions that are analytic can also be shown for other activation function than ReLU; see in particular the works of Mhaskar [142, 143] and Section 6 in Pinkus' Acta Numerica article [174] for sigmoidal and smooth activations. Additionally, the more recent paper [47] specifically addresses the hyperbolic tangent activation. Finally, [80] studies general activation functions that allow for the construction of approximate partitions of unity.