# CONSTRUCTIVE UNIVERSAL APPROXIMATION AND FINITE SAMPLE MEMORIZATION BY NARROW DEEP RELU NETWORKS

MARTÍN HERNÁNDEZ[†] AND ENRIQUE ZUAZUA [*†‡]

Abstract. We present a fully constructive analysis of deep ReLU neural networks for classification and function approximation tasks. First, we prove that any dataset with $N$ distinct points in $\mathbb{R}^d$ and $M$ output classes can be exactly classified using a multilayer perceptron (MLP) of width 2 and depth at most $2N + 4M - 1$, with all network parameters constructed explicitly. This result is sharp with respect to width and is interpreted through the lens of simultaneous or ensemble controllability in discrete nonlinear dynamics.

Second, we show that these explicit constructions yield uniform bounds on the parameter norms and, in particular, provide upper estimates for minimizers of standard regularized training loss functionals in supervised learning. As the regularization parameter vanishes, the trained networks converge to exact classifiers with bounded norm, explaining the effectiveness of overparameterized training in the small-regularization regime.

We also prove a universal approximation theorem in $L^p(\Omega; \mathbb{R}_+)$ for any bounded domain $\Omega \subset \mathbb{R}^d$ and $p \in [1, \infty)$, using MLPs of fixed width $d + 1$. The proof is constructive, geometrically motivated, and provides explicit estimates on the network depth when the target function belongs to the Sobolev space $W^{1,p}$. We also extend the approximation and depth estimation results to $L^p(\Omega; \mathbb{R}^m)$ for any $m \geq 1$.

Our results offer a unified and interpretable framework connecting controllability, expressivity, and training dynamics in deep neural networks.

## Contents



## 1. Introduction and main results

1.1. **Motivation and summary of the main results.** Given a training dataset $\{x_i, y_i\}_{i=1}^N \subset \mathcal{X} \times \mathcal{Y}$, where each $x_i$ represents an input data point and $y_i$ its corresponding label or output, and a model $\phi(x, \theta)$ parameterized by $\theta$, the property of *finite sample memorization* [45, 46] holds if the model $\phi$ can correctly assign the label $y_i$ to each training instance $x_i$, i.e.,

$$\phi(x_i, \theta) = y_i, \qquad \text{for every } i \in \{1, \dots, N\}.$$

2020 *Mathematics Subject Classification.* 68T07, 93C10, 34H05.

*Key words and phrases.* Deep neuronal network; Finite sample memorization; Simultaneous controllability; Nonlinear discrete dynamics; Universal approximation theorem.

[†] Chair for Dynamics, Control, Machine Learning, and Numerics, Alexander von Humboldt-Professorship, Department of Mathematics, Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany.

[*] Departamento de Matemáticas, Universidad Autónoma de Madrid, 28049 Madrid, Spain.

[‡] Chair of Computational Mathematics, University of Deusto. Av. de las Universidades, 24, 48007 Bilbao, Basque Country, Spain.

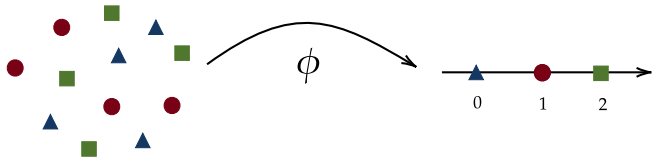



FIGURE 1. Classification of a two-dimensional dataset via the neural network map $\phi$, with input dimension $d = 2$ and and scalar output $m = 1$.

We analyze the finite sample memorization property when $\phi(\cdot, \theta)$ corresponds to the output of a neural network, with $\mathcal{X} = \mathbb{R}^d$ for $d \geq 1$, and $\mathcal{Y} = \mathbb{R}^m$ for $m \geq 1$.

When $\phi(x, \theta)$ is determined as the output of a continuous or discrete dynamical system, the problem can also be interpreted as an *ensemble* or *simultaneous controllability* problem, ensuring that the initial data $\{x_i\}$ are mapped simultaneously to the corresponding targets $\{y_i\}$ [35, 30, 42, 36].

This memorization property is particularly valuable for classification and interpolation tasks involving an unknown function $f : \mathcal{X} \to \mathcal{Y}$, as it guarantees exact fitting at the data points. Once pointwise interpolation is achieved, one can extend this construction to approximate $f$ in $L^p$-norms for $p \in [1, \infty)$, aligning with universal approximation theorems. These results establish that various neural network architectures are dense in functional spaces, such as $L^p$ or Sobolev spaces, thus enabling global approximation from finite data [10, 12, 17, 18, 32, 33, 43, 46].

In this article, we present the following main results.

- The first shows one that a ReLU multilayer perceptron with a width 2 and at most $2N + 4M - 1$ layers satisfies the finite sample memorization (or universal interpolation) property for $N$ input points and $M$ classes. Equivalently, the discrete dynamics generated by this MLP fulfils the property of simultaneous and/or ensemble controllable. This result is sharp, as memorization cannot be achieved by MLPs with width 1.

  Our proof constructs the network parameters in a systematic manner, based on a geometric and dynamical interpretation of the neural network's architecture. Specifically, the parameters at each layer define hyperplanes that partition the state space into regions where the nonlinear activation function induces distinct dynamical behaviors. By strategically selecting these parameters and iteratively applying them across layers, we ensure that the network satisfies the memorization property (see Section 2). To the best of our knowledge, such a constructive and purely geometric interpretation of how narrow MLPs achieve finite sample memorization has not been previously explored in the literature (see Section 1.4).
- We then examine the implications of our constructive approach in the context of supervised training with $\ell^2$-regularization. Although the networks we construct are not obtained through optimization (but rather through geometric and dynamical considerations), we demonstrate that the explicit interpolating parameters from our first result can be used to establish upper bounds on the optimal value of the regularized empirical loss. In particular, we prove that minimizing a standard training objective with $\ell^2$-regularization produces parameter norms that are uniformly bounded by those of our explicit construction. Moreover, in the vanishing regularization limit, the resulting parameters converge to a minimal-norm interpolating network. This result offers a theoretical explanation for the behavior of trained networks in the small-$\lambda$ regime and reinforces the idea that exact data fitting can be achieved without uncontrolled growth in parameter magnitude.
- Our final contribution establishes a universal approximation theorem in $L^p(\Omega; \mathbb{R}_+)$, where $\Omega \subset \mathbb{R}^d$ is bounded and $p \in [1, \infty)$. This result is obtained using an MLP of fixed width $d + 1$. As in our first result, the network parameters are constructed explicitly, without relying on any optimization procedure. The proof is geometrically motivated and nonlinear, marking a departure from prior approaches to universal approximation with fixed-width networks [7, 25, 26, 31, 33]. Crucially, our explicit construction allows for quantitative estimates on the required depth when the target function belongs to $W^{1,p}(\Omega; \mathbb{R}_+)$. As a corollary, we extend this approximation result to the vector-valued settings $L^p(\Omega; \mathbb{R}^m_+)$ and $L^p(\Omega; \mathbb{R}^m)$ for any $m \geq 1$, with corresponding estimates on the necessary network width.

1.2. **Problem formulation.** Let $x \in \mathbb{R}$ and define the ReLU activation function as $\sigma(x) = \max\{0, x\}$. We consider a sequence of positive integers $\{d_j\}_{j=1}^{L}$. For each $j \in \{1, \dots, L\}$ and $x = (x^{(1)}, \dots, x^{(d_j)})^\top$ in $\mathbb{R}^{d_j}$, we introduce the vector-valued version of $\sigma$, defined by

$$\boldsymbol{\sigma}_j : \mathbb{R}^{d_j} \to \mathbb{R}^{d_j}, \qquad \boldsymbol{\sigma}_j(x) = \left(\sigma(x^{(1)}) \, , \dots, \sigma(x^{(d_j)})\right). \tag{1.1}$$

Given positive integers $L$, $d$, $N$, $M$, and the dataset $\{x_i, y_i\}_{i=1}^{N} \subset \mathbb{R}^d \times \{0, \dots, M-1\}$. We consider the following multilayer perceptron:

$$\begin{cases} x_i^j = \boldsymbol{\sigma}_j(W_j x_i^{j-1} + b_j), & \text{for } j \in \{1, \dots, L\}, \\ x_i^0 = x_i, \end{cases} \tag{1.2}$$

where $i \in \{1, \dots, N\}$. In this context, $W_j \in \mathbb{R}^{d_j \times d_{j-1}}$ and $b_j \in \mathbb{R}^{d_j}$, for $j \in \{1, \dots, L\}$, represent the weight matrices and biases, respectively. Each $d_j$ determines the width of the $j$-layer, i.e., the dimension of the Euclidean space where the data reside at each iteration. The depth or the number of hidden layers of the neural network is represented by $L$, which is the number of iterations in the discrete dynamical system (1.2), referred to as a $L-$hidden layer neural network (see Figure 2).

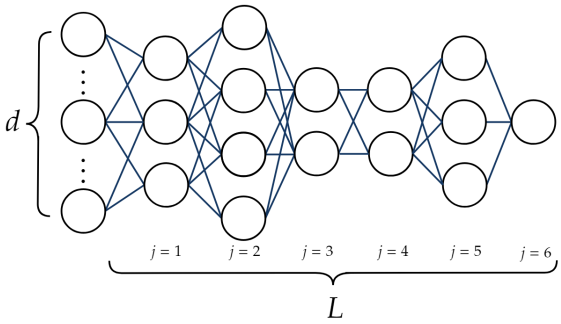



FIGURE 2. Example of a deep neural network defined by the architecture (1.2). Here, $d$ indicates the dimension of the input data, while $j$ is the index of the layer, $L = 6$ being the total number of layers, i.e., the depth of the neural network. In this particular example, we have $d_1 = 3$, $d_2 = 4$, $d_3 = 2$, etc. Moreover, the maximum width of the neural network is 4, determined by the second layer.

In the following, we denote the sequences of weights and biases defining the neural network (1.2) as $\mathcal{W}^L = \{W_j\}_{j=1}^{L}$ and $\mathcal{B}^L = \{b_j\}_{j=1}^{L}$, respectively. The *width* of the neural network is defined as $w_{max} = \max_{j \in \{1,\dots,L\}}\{d_j\}$, i.e., the number of neurons in the widest layer. System (1.2) is said to be a *$w_{max}$-wide deep neural network*. The width and depth of a neural network are determined by its architecture and serve as an intrinsic measure of its complexity and approximation capacity.

1.3. **Statement of the main results and strategies of the proofs.**

1.3.1. *Finite Sample Memorization.* Let us define the input-output map $\phi : \mathbb{R}^d \to \mathbb{R}$ of the neural network (1.2) as

$$\phi^L(x_i) := \phi(\mathcal{W}^L, \mathcal{B}^L, x_i) = x_i^L, \qquad \text{for every } i \in \{1, \dots, N\},$$

where $x^L$ represents the output of (1.2).

Note that we are considering the particular case in which the output lies in $\mathbb{R}$, which means that $W_L \in \mathbb{R}^{1 \times d_{L-1}}$.

With this definition, we present our first main theorem.

**Theorem 1.1** (Finite Sample Memorization)**.** *Let the integers $d$, $N$, $M \geq 1$ and consider the dataset $\{x_i, y_i\}_{i=1}^{N} \subset \mathbb{R}^d \times \{0, \dots, M-1\}$. Assume that $x_i \neq x_j$ if $i \neq j$. Then, there exist parameters $\mathcal{W}^L$ and $\mathcal{B}^L$ with width $w_{max} = 2$ and depth $L = 2N + 4M - 1$ such that the input-output map of (1.2) satisfies*

$$\phi(\mathcal{W}^L, \mathcal{B}^L, x_i) = y_i, \qquad \textit{for every } i \in \{1, \dots, N\}. \tag{1.3}$$

*Moreover, this result is sharp since the memorization property cannot be achieved with width 1.*

**Remark 1.1.** *Some comments are in order.*

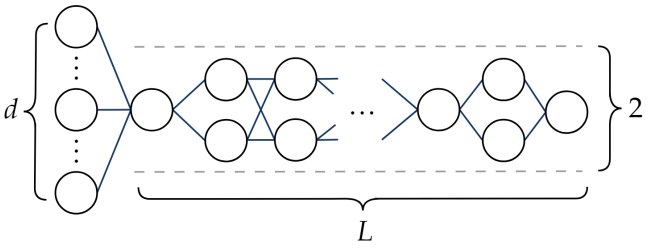



FIGURE 3. Deep neural network of width $w_{max} = 2$ as in Theorem 1.1.

- *Theorem 1.1 guarantees that there exists a* $2-$*wide deep neural network satisfying the finite sample memorization property, or equivalently, that system* (1.2) *is simultaneously or ensemble controllable.*
- *Obviously, Theorem 1.1 also ensures finite sample memorization with* $w_{max} \geq 2$.
- *Theorem 1.1 provides an estimate for the number of layers sufficient for the neural network to exhibit the finite sample memorization property. The depth of the network is directly related to both the number of data points* $N$ *and distinct labels* $M$*. However, it is independent of the dimension* $d$*, to which the data set belongs.*
- *The neural network depth estimation is obtained from the constructive proof of Theorem 1.1, which is based on the worst-case scenario. This construction does not guarantee optimality in the estimated depth* $L$*, and for specific datasets, memorization could be achieved with fewer layers.*
- *Although the width of the neural network is* 2*, some layers have only one neuron, as in Figure 3. Namely, the total number of neurons and parameters in our neural network is* $4N + 6M + d - 2$ *and* $8N + 12M + 2d - 4$*, respectively.*
- *In Theorem 1.1, we considered* $\{y_i\}_{i=1}^N \subset \{0, \dots, M-1\}$ *to simplify the exposition. However, the labels* $\{0, \dots, M-1\}$ *could be replaced by any other choice of* $M$ *distinct values in* $\mathbb{R}_+$*. This does not impact the width and depth of the neural network needed for memorization.*

*Strategy of proof of Theorem 1.1.* The proof of this result is grounded in three fundamental tools, all of which are derived from the geometrical properties of the system (1.2):

- **Dimension Reduction:** Given a family of distinct points $\{x_k\}_{k=1}^N \subset \mathbb{R}^d$, $d \geq 1$, we can construct a projection $\phi^1 : \mathbb{R}^d \to \mathbb{R}$ so that its images are all different.
- **Distance Scaling:** Let $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$. For $x^0 \in \mathbb{R}^d$, if $w \cdot x^0 + b > 0$, the value of $\sigma(w \cdot x^0 + b) \in \mathbb{R}$ corresponds to $\|w\| d(x_0, H)$, where $\|w\|$ is the Euclidean norm of $w$ and $d(x_0, H)$ is the distance between $x_0$ and the hyperplane

$$H := \{x \in \mathbb{R}^d \, : \, w \cdot x + b = 0\}. \tag{1.4}$$

- **Collapse:** The hyperplane (1.4) divides the space into two half-spaces and the function $\sigma(w \cdot x + b)$ collapses the half-space $w \cdot x + b \leq 0$ into the null point.

A more detailed discussion of these tools can be found in Section 2. The map $\phi$ in Theorem 1.1 is built in four steps:

(1) **Preconditioning of the data:** Data is driven from the $d-$dimensional space to the one-dimensional one to reduce complexity.
(2) **Compression process:** A recursive process is built to drive the $N$ data points to $M$ representative elements, according to their labels.
(3) **Data sorting:** Data are mapped to ordered one-dimensional points according to their labels.
(4) **Mapping to the respective labels:** Finally, each data point is mapped to its corresponding label.

In each of these steps, we employ an input-output map of the neuronal network (1.2), utilizing at most two neurons per layer. As this is a purely constructive process, we can determine the number of layers required at each stage, and therefore, we can estimate the depth of the neural network. Further details on these key steps can be found in Section 3.

Let us consider the norms

$$\left|\!\left|\!\left|(\mathcal{W}^L,\mathcal{B}^L)\right|\!\right|\!\right|_2 := \left(\sum_{j=1}^{L} \|W_j\|_F^2 + \|b_j\|_2^2\right)^{1/2},$$

and

$$\left|\!\left|\!\left|(\mathcal{W}^L,\mathcal{B}^L)\right|\!\right|\!\right|_\infty := \sup_{j\in\{1,\dots,L\}} \{\|W_j\|_\infty, \|b_j\|_\infty\},$$

where $\|\cdot\|_F$ denotes the Frobenius norm, $\|\cdot\|_2$ the $\ell^2$-norm, and $\|\cdot\|_\infty$ the $\ell^\infty$-norm. Then, due to the explicit construction of the parameters in Theorem 1.1, we obtain the following estimate for their norms.

**Corollary 1.1.** *Let $d$, $N$, $M > 1$ be integers, and consider the dataset $\{(x_i, y_i)\}_{i=1}^N \subset B_{R_x}^d(0) \times B_{R_y}^1(0)$, where $B_{R_x}^d(0) \subset \mathbb{R}^d$ and $B_{R_y}^1(0) \subset \mathbb{R}$ denote balls of radius $R_x > 0$ and $R_y > 0$ centered at the origin. Then, the parameters $\mathcal{W}^L$ and $\mathcal{B}^L$ provided by Theorem 1.1 satisfy*

$$\left|\!\left|\!\left|(\mathcal{W}^L,\mathcal{B}^L)\right|\!\right|\!\right|_2 \le C(1 + R_x\sqrt{N} + R_x N\sqrt{M} + R_y M), \tag{1.5}$$

$$\left|\!\left|\!\left|(\mathcal{W}^L,\mathcal{B}^L)\right|\!\right|\!\right|_\infty \le C(R_x N + M + R_y), \tag{1.6}$$

*where $C > 0$ is a constant independent of $M$, $N$, $R_x$, $R_y$, and $d$, but depends on $\min_{i\neq j\in\{1,\dots,N\}} \|x_i - x_j\|$.*

*Strategy of the proof of Corollary 1.1.* The result follows directly from the explicit construction given in Theorem 1.1. Since the parameters $\mathcal{W}^L$ and $\mathcal{B}^L$ are specified layer by layer, the norm bounds are obtained by computing their contributions at each step and applying standard estimates.

**Remark 1.2.** *As mentioned in Remark 1.1, Theorem 1.1 ensures exact classification for any $w_{\max} \ge 2$ by zero-padding the parameters. Consequently, Corollary 1.1 also holds for all $w_{\max} \ge 2$.*

**Remark 1.3** (Other activation functions)**.** *Our techniques can also be applied to other activation functions $\sigma$ that satisfy the following conditions:*

- *$\sigma$ is monotonically non-decreasing on $\mathbb{R}_+$, being strictly monotonic in a subinterval $T$ of $\mathbb{R}_+$. This permits scaling distances between different points.*
- *There exists and open subset $S$ of $\mathbb{R}_-$ in which $\sigma$ vanishes. This allows the collapse of different points to merge them according to their labels.*

*The essential features of the activation function employed are described below in Section 3 (see Figure 4).*

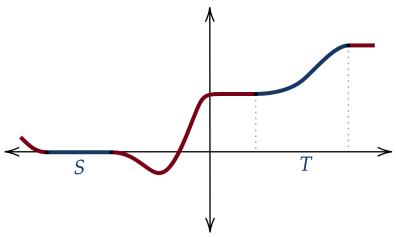



FIGURE 4. Activation functions for which the results of this paper can be generalized.

*Note, however, that despite this generalization being possible, its practical interest is limited. The weights and biases will be more difficult to construct and will generally have larger norms compared to those provided by Corollary 1.1, which uses the ReLU activation function to ensure minimal complexity.*

As a consequence of Theorem 1.1, we can address the case of $m$-dimensional labels for $m \ge 1$, that is, when $\{y_i\}_{i=1}^N$ is a subset of $M$ distinct points in $\mathbb{R}^m$. This leads to the following corollary.

**Corollary 1.2** (Finite Sample Memorization for $m-$dimensional labels)**.** *Let the integers $d$, $N$, $M$, $m \ge 1$ and a dataset $\{x_i, y_i\}_{i=1}^N$ so that $\{x_i\}_{i=1}^N \subset \mathbb{R}^d$, $x_i \neq x_j$ if $i \neq j$, and $\{y_i\}_{i=1}^N \subset \{\ell_k\}_{k=0}^{M-1}$ with $\ell_k \in \mathbb{R}_+^m$.*

*Then, there exist parameters $\mathcal{W}^L$ and $\mathcal{B}^L$ with $L = 2N + 4M - 1$ and $w_{max} = 2m$, such that the input-output map of (1.2) satisfies*

$$\phi(\mathcal{W}^L, \mathcal{B}^L, x_i) = y_i, \qquad \text{for every } i \in \{1, \dots, N\}.$$

*Strategy of proof of Corollary 1.2.* Corollary 1.2 follows directly from Theorem 1.1 by considering $m$ independent neural networks, each constructed as in Theorem 1.1 to handle one component of the $m$-dimensional labels, and combining them in parallel to obtain a single network with vector-valued outputs (see Figure 5).

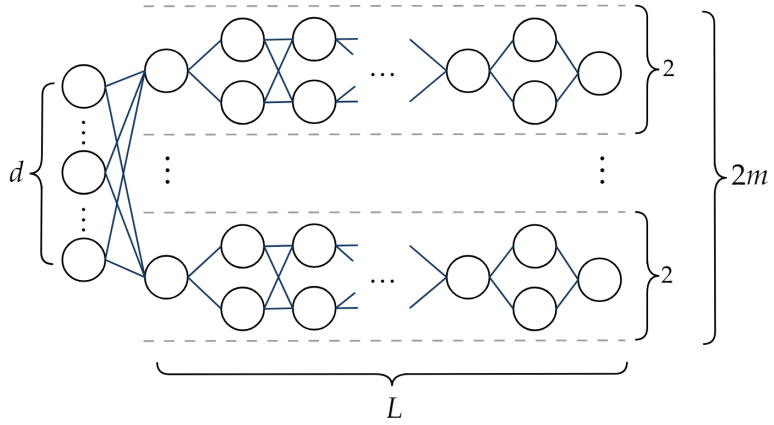



FIGURE 5. Deep neural network of width $w_{max} = 2m$ as in Corollary 1.2.

In the previous results, all the labels were assumed to be non-negative. However, it is reasonable to consider the case where the signs on the labels may vary. Let $\mathrm{Im}(\phi^L)$ denote the image of the input-output map defined by (1.2). Since $\mathrm{Im}(\phi^L) \subset \mathbb{R}_+$, data cannot be mapped to negative labels by the architecture defined in (1.2). However, we can consider the following MLP

$$\begin{cases} x_i^j = A_j \boldsymbol{\sigma}_j(W_j x_i^{j-1} + b_j), & \text{for } j \in \{1, \dots, L\}, \\ x_i^0 = x_i, \end{cases} \tag{1.7}$$

where the extra parameters $A_j \in \mathbb{R}^{d_j \times d_j}$ for $j \in \{1, \dots, L\}$ allow the neural network, in particular, to map data to negative values. We denote by $\mathcal{A}^L = \{A_j\}_{j=1}^L$ this new sequence of parameters.

The following corollary states that the neural network architecture (1.7) satisfies the finite sample memorization for labels in $\mathbb{R}$.

**Corollary 1.3** (Finite Sample Memorization for real labels)**.** *Consider the integers $d$, $N$, $M \geq 1$ and a dataset $\{x_i, y_i\}_{i=1}^N \subset \mathbb{R}^d \times \{\alpha_0, \dots, \alpha_{M-1}\}$ with $\{\alpha_k\}_{k=0}^{M-1} \subset \mathbb{R}$. Assume that $x_i \neq x_j$ if $i \neq j$. Then, for $L = 2N + 4M$ and $w_{max} = 2$, there exist parameters $\mathcal{A}^L$, $\mathcal{W}^L$ and $\mathcal{B}^L$ such that the input-output map of* (1.7) *satisfies*

$$\phi(\mathcal{A}^L, \mathcal{W}^L, \mathcal{B}^L, x_i) = y_i, \qquad \text{for every } i \in \{1, \dots, N\}. \tag{1.8}$$

*Strategy of Proof for Corollary 1.3.* The proof hinges on two key observations: first, that the architecture (1.7) can drive data to negative labels; and second, that this architecture coincides with (1.2) when $A_j = \mathrm{Id}_{d_j}$ (the identity matrix in $\mathbb{R}^{d_j \times d_j}$). Initially, we define a set of auxiliary positive labels. Then, taking $A_j = \mathrm{Id}_{d_j}$, we apply Theorem 1.1 to map the data points to these auxiliary labels by using $2N + 4M - 1$ layers. Finally, by using a particular matrix $A$, we construct a 2-wide, one-layer neural network that maps the auxiliary labels to the original ones. The proof concludes by composing these two neural networks, obtaining a 2-wide neural network with $2N + 4M$ layers.

**Remark 1.4.** *Following the same approach used in the proof of Corollary 1.2, we can extend Corollary 1.3 to establish the finite sample memorization property for labels in $\mathbb{R}^m$. In this case, the construction yields a neural network of width $2m$ and depth $2N + 4M$.*

1.3.2. *Implications for Neural Network Training.* The training of neural networks is typically formulated as the minimization of a regularized empirical risk functional. Given a dataset $\{x_i, y_i\}_{i=1}^N \subset \mathbb{R}^d \times \mathbb{R}^m$, the standard training objective reads

$$\min_{(\mathcal{W}^L, \mathcal{B}^L)} \left\{ \mathcal{J}_\lambda(\mathcal{W}^L, \mathcal{B}^L) = \lambda \left\| (\mathcal{W}^L, \mathcal{B}^L) \right\|_2^2 + \frac{1}{N} \sum_{i=1}^N \mathrm{loss}\left(\phi(\mathcal{W}^L, \mathcal{B}^L, x_i), y_i\right) \right\}, \tag{1.9}$$

where the first term penalizes large weights with a parameter $\lambda > 0$, and the second term measures the prediction error on the training dataset through the input-output map $\phi(\mathcal{W}^L, \mathcal{B}^L, \cdot)$ defined in (1.2). Here, $\mathrm{loss}(\cdot,\cdot)$ is a given continuous and nonnegative function such that $\mathrm{loss}(x,x) = 0$. For instance, $\mathrm{loss}(x,y) := \|x-y\|_p^2$ with $p \in \{1,2\}$ is commonly used for regression tasks, while $\mathrm{loss}(x,y) = \log(1+e^x) - yx$, with $y \in \{0,1\}$ and $x \in \mathbb{R}$, is used for binary classification (binary logistic loss). The regularization parameter $\lambda$ controls the trade-off between data fidelity and model complexity.

Note that for every $\lambda > 0$, the functional $\mathcal{J}_\lambda$ is coercive. Moreover, since both loss and the activation functions $\boldsymbol{\sigma}_j$ are continuous, the Bolzano–Weierstrass theorem ensures the existence of optimal parameters for every $\lambda > 0$.

Recall that Theorem 1.1 establishes the existence of $(\mathcal{W}_*^L, \mathcal{B}_*^L)$ such that exact interpolation of arbitrary finite datasets is possible. In particular, this guarantees that

$$\mathcal{J}_0(\mathcal{W}_*^L, \mathcal{B}_*^L) = 0, \tag{1.10}$$

and since $\mathcal{J}_0 \geq 0$, we deduce that $(\mathcal{W}_*^L, \mathcal{B}_*^L)$ is a minimizer of $\mathcal{J}_0$. Moreover, we have the following result:

**Theorem 1.2.** *Let $d$, $N$, $m \in \mathbb{N}$ with $d$, $N$, $m > 1$, and let $\{(x_i, y_i)\}_{i=1}^N \subset \mathbb{R}^d \times \mathbb{R}^m$. Let $(\mathcal{W}_\lambda^L, \mathcal{B}_\lambda^L)$ be a minimizer of $\mathcal{J}_\lambda$, and let $(\mathcal{W}_*^L, \mathcal{B}_*^L)$ be a minimizer of $\mathcal{J}_0$ for the ReLU activation function. Then, we have*

$$\mathcal{J}_\lambda(\mathcal{W}_\lambda^L, \mathcal{B}_\lambda^L) \leq \lambda \left|\!\left|\!\left| (\mathcal{W}_*^L, \mathcal{B}_*^L) \right|\!\right|\!\right|_2^2, \quad \text{for all } \lambda > 0. \tag{1.11}$$

*In particular, we have*

$$\frac{1}{N}\sum_{i=1}^N \mathrm{loss}\left(\phi(\mathcal{W}_\lambda^L, \mathcal{B}_\lambda^L, x_i), y_i\right) \to 0 \quad \text{as } \lambda \to 0, \tag{1.12}$$

*and*

$$\left|\!\left|\!\left| (\mathcal{W}_\lambda^L, \mathcal{B}_\lambda^L) \right|\!\right|\!\right|_2 \leq \left|\!\left|\!\left| (\mathcal{W}_*^L, \mathcal{B}_*^L) \right|\!\right|\!\right|_2, \quad \text{for all } \lambda > 0. \tag{1.13}$$

*Moreover, the family of parameters $\{(\mathcal{W}_\lambda^L, \mathcal{B}_\lambda^L)\}_{\lambda>0}$ admits a subsequence that converges to a minimal-norm minimizer of $\mathcal{J}_0$.*

**Remark 1.5.** *Several remarks are in order:*

- *The proof of Theorem 1.2 can be found in Section 6. It relies on the fact that a minimizer of $\mathcal{J}_0$ is known explicitly, which is possible due to the interpolation result from Theorem 1.1. However, Theorem 1.2 is not tight to the 2-width MLP used in Theorem 1.1; it can be applied to any architecture (or discrete dynamical system) whose input-output map is continuous with respect to the parameters.*
- *This result shows that, as the regularization parameter $\lambda$ tends to zero, the trained network tends to interpolate the dataset while remaining uniformly bounded in parameter norm. In particular, it guarantees that any accumulation point of the sequence of minimizers corresponds to a minimum-norm interpolant. Indeed, this holds given that the arguments of Theorem 1.2 apply to any minimizer of $\mathcal{J}_0$, and in particular to those of minimal norm.This has practical implications in training: it justifies the use of small values of $\lambda$ for training since this pushes the model to fit the training data exactly, with finite values of the parameters, close to the optimal ones.*
- *In general, we cannot guarantee that $(\mathcal{W}_0^L, \mathcal{B}_0^L) = (\mathcal{W}_*^L, \mathcal{B}_*^L)$. This is a consequence of the lack of convexity of the functionals $\mathcal{J}_\lambda$ for $\lambda \geq 0$.*
- *Observe that we can replace the norm $|\!|\!|\cdot|\!|\!|$ in (1.9) by $|\!|\!|\cdot|\!|\!|_\infty$ and derive Theorem 1.2 again, but now involving the $\ell^\infty$-norm.*
- *Due to Corollary 1.1, when we consider the dataset $\{(x_i, y_i)\}_{i=1}^N \subset B_{R_x}^d(0) \times B_{R_y}^m(0)$, we can provide an explicit estimate for the optimal value of $\mathcal{J}_\lambda$ in terms of $N$, $M$, $R_x$, and $R_y$. Namely, in the case $m = 1$, combining Corollary 1.1 with Theorem 1.2, we obtain*

$$\mathcal{J}_\lambda(\mathcal{W}_\lambda^L, \mathcal{B}_\lambda^L) \leq \lambda C(1 + R_x^2 N + R_x^2 N^2 M + R_y^2 M^2). \tag{1.14}$$

*However, since the parameters provided in Theorem 1.1 are constructed to handle worst-case scenarios, the bound in (1.14) may be conservative. In practical situations, significantly tighter estimates may be achieved.*

**Remark 1.6** (Practical application of Theorem 1.2)**.** *These results, and in particular* (1.14)*, facilitate defining stopping criteria for the training process. Indeed, the optimal value of $\mathcal{J}_\lambda$ is always bounded above by $\lambda \left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^2$. Therefore, for any fixed $\lambda > 0$ – even for small values of $\lambda$ – the optimizer can be designed to terminate once the value of $\mathcal{J}_\lambda$ falls close to this explicit threshold.*

In practice, neural networks are frequently trained using activation functions that do not satisfy the properties in Remark 1.3. Then, constructing or estimating the parameters assuring classification becomes challenging.

This motivates the following question: Can the knowledge of parameter norms that ensure exact classification for a given activation function, for instance the ReLU, still provide useful information to the regularized functional for a perturbed activation function?

To analyze this point, we consider the following neural network

$$\begin{cases} \hat{x}_i^j = \hat{\boldsymbol{\sigma}}_j(W_j \hat{x}_i^{j-1} + b_j), & \text{for } j \in \{1, \dots, L\}, \\ \hat{x}_i^0 = x_i, \end{cases} \tag{1.15}$$

where each $\hat{\boldsymbol{\sigma}}_j : \mathbb{R}^{d_j} \to \mathbb{R}^{d_j}$ is a continuous activation function, not necessarily the ReLU. Denote by $\hat{\phi}(\mathcal{W}^L, \mathcal{B}^L, \cdot)$ the input-output map associated with (1.15). We then train this new model:

$$\min_{(\mathcal{W}^L, \mathcal{B}^L)} \left\{ \hat{\mathcal{J}}_\lambda(\mathcal{W}^L, \mathcal{B}^L) = \lambda \left|\!\left|\!\left|(\mathcal{W}^L, \mathcal{B}^L)\right|\!\right|\!\right|_2^2 + \frac{1}{N} \sum_{i=1}^N \text{loss}\left(\hat{\phi}(\mathcal{W}^L, \mathcal{B}^L, x_i), y_i\right) \right\}, \tag{1.16}$$

where $\text{loss}(\cdot, \cdot)$ is a continuous loss function, as before. Thanks to the continuity of loss and $\hat{\boldsymbol{\sigma}}_j$, and the coercivity of $\hat{\mathcal{J}}_\lambda$, we can guarantee the existence of a minimizer $(\hat{\mathcal{W}}_\lambda^L, \hat{\mathcal{B}}_\lambda^L)$ for every $\lambda > 0$. Moreover, the following result holds.

**Theorem 1.3.** *Let $d$, $N$, $m \in \mathbb{N}$ with $d$, $N$, $m > 1$, and let $\{(x_i, y_i)\}_{i=1}^N \subset \mathbb{R}^d \times \mathbb{R}^m$. Let $(\hat{\mathcal{W}}_\lambda^L, \hat{\mathcal{B}}_\lambda^L)$ be a minimizer of $\hat{\mathcal{J}}_\lambda$, and let $(\mathcal{W}_*^L, \mathcal{B}_*^L)$ be a minimizer of $\mathcal{J}_0$ for the ReLU. Define $R_0 := \max_{i \in \{1,\dots,N\}} \|x_i\|$, and assume that*

$$\nu_j := \sup_{z \in B_{R_j}^{d_j}(0)} \|\hat{\boldsymbol{\sigma}}_j(z) - \boldsymbol{\sigma}_j(z)\| < \infty, \quad \text{with} \quad R_j := \|W_j\|_2 R_{j-1} + \|b_j\|_2, \quad \text{for } j \in \{1, \dots, L\}, \tag{1.17}$$

*$R_j$ being defined through the optimal parameters of the ReLU model whose values we estimated. Set $\nu := (\nu_1, \dots, \nu_L)$.*

*Then, for every $\lambda > 0$, we have*

$$\hat{\mathcal{J}}_\lambda(\hat{\mathcal{W}}_\lambda^L, \hat{\mathcal{B}}_\lambda^L) \leq \lambda \left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 + \mathscr{A}_{\text{loss}}(\nu, \mathcal{W}_*^L, \mathcal{B}_*^L),$$

*where $\mathscr{A}_{\text{loss}}$ is a nonnegative function depending on the loss function* loss*, and satisfies $\mathscr{A}_{\text{loss}}(\nu, \cdot, \cdot) \to 0$ as $\|\nu\|_2 \to 0$.*

The proof of Theorem 1.3 can be found in Section 6.

**Remark 1.7.** *Several observations are worth:*

- *The proof consists on analyzing the deviation between the input-output maps for both activation functions, using arguments similar to those used in Theorem 1.2. The radii $R_j$ introduced in* (1.17) *are key to estimating the deviation between activations at each layer.*
- *Observe that the only assumption imposed on $\hat{\boldsymbol{\sigma}}_j$ is continuity. No further structural properties are required.*
- *We have assumed that both architectures are identical, i.e., data evolve through layers of the same dimensions. This assumption is not strictly necessary. When the dynamics* (1.15) *evolve along a different architecture, with $\hat{d}_j \neq d_j$, one can always extend the networks by zero-padding to the maximal dimension $\max\{d_j, \hat{d}_j\}$ and proceed to a stability analysis. A similar estimate then follows, now involving a modified term $\hat{\mathscr{A}}_{\text{loss}}$, depending on the padded architecture.*
- *In the particular case $\text{loss}(x, y) = \|x - y\|_2^2$, we obtain the explicit estimate*

$$\mathscr{A}_{\text{loss}}(\nu, \mathcal{W}_*^L, \mathcal{B}_*^L) = \begin{cases} 2\|\nu\|_2^2 \left( \dfrac{\left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^{2L} - 1}{\left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 - 1} \right) & \text{if } \left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 \neq 1, \\ 2\|\nu\|_2^2 L & \text{if } \left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 = 1. \end{cases}$$

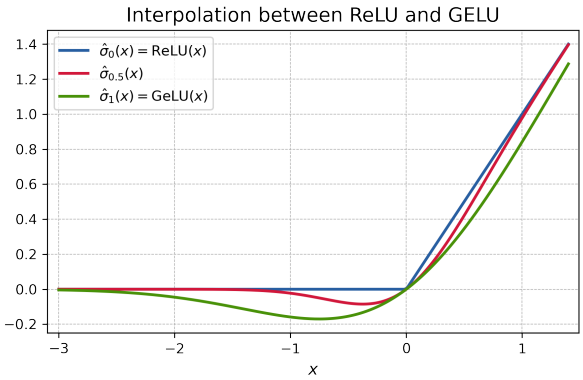



FIGURE 6. Illustration of the interpolation between ReLU and GELU via $\hat{\sigma}_\varepsilon(x)$ for different values of $\varepsilon$. The case $\varepsilon = 0$ corresponds to ReLU, and $\varepsilon = 1$ to GELU.

The previous result is particularly relevant when, instead of considering a fixed activation function $\hat{\boldsymbol{\sigma}}$, for instance the ReLU, we study a family of continuous activation functions $\hat{\boldsymbol{\sigma}}_\varepsilon$ parametrized by $\varepsilon > 0$, for instance a regularization of the ReLU. In this case, we can again formulate a training problem, now depending on the parameter $\varepsilon > 0$. Namely, we consider

$$\min_{(\mathcal{W}^L, \mathcal{B}^L)} \left\{ \mathcal{J}_{\lambda,\varepsilon}(\mathcal{W}^L, \mathcal{B}^L) = \lambda \left|\!\left|\!\left| (\mathcal{W}^L, \mathcal{B}^L) \right|\!\right|\!\right|_2^2 + \frac{1}{N} \sum_{i=1}^{N} \mathrm{loss}\left(\phi_\varepsilon(\mathcal{W}^L, \mathcal{B}^L, x_i), y_i\right) \right\}, \tag{1.18}$$

where $\phi_\varepsilon$ denotes the input-output map of the dynamics (1.15) when $\hat{\boldsymbol{\sigma}}$ is replaced by $\hat{\boldsymbol{\sigma}}_\varepsilon$.

By similar arguments, one can ensure the existence of a minimizer of (1.18) for every $\varepsilon$, $\lambda > 0$. We denote by $(\hat{\mathcal{W}}^L_{\lambda,\varepsilon}, \hat{\mathcal{B}}^L_{\lambda,\varepsilon})$ the corresponding minimizer.

A particularly interesting case arises when we define the family $\hat{\sigma}_\varepsilon : \mathbb{R} \to \mathbb{R}$ by

$$\hat{\sigma}_\varepsilon(x) = \frac{x}{2}\left(1 + \mathrm{erf}\left(\frac{x}{\varepsilon\sqrt{2}}\right)\right), \tag{1.19}$$

so that $\hat{\sigma}_1(x) = \mathrm{GELU}(x)$, the so-called Gaussian Error Linear Unit (see Figure 6). Let $\hat{\boldsymbol{\sigma}}_\varepsilon : \mathbb{R}^{d_j} \to \mathbb{R}^{d_j}$ denote the vector-valued extension of $\hat{\sigma}_\varepsilon$ (as in (1.1)). Then, we obtain the following corollary.

**Corollary 1.4.** *Let $d$, $N$, $m \in \mathbb{N}$ with $d$, $N$, $m > 1$, and let $\{(x_i, y_i)\}_{i=1}^N \subset \mathbb{R}^d \times \mathbb{R}^m$. Let $(\mathcal{W}^L_{\lambda,\varepsilon}, \mathcal{B}^L_{\lambda,\varepsilon})$ be a minimizer of $\mathcal{J}_{\lambda,\varepsilon}$, and let $(\mathcal{W}^L_*, \mathcal{B}^L_*)$ be a minimizer of $\mathcal{J}_0$. Then, under the assumptions of Theorem 1.3, we have*

$$\limsup_{\varepsilon \to 0} \mathcal{J}_{\lambda,\varepsilon}(\mathcal{W}^L_{\lambda,\varepsilon}, \mathcal{B}^L_{\lambda,\varepsilon}) \leq \lambda \left|\!\left|\!\left| (\mathcal{W}^L_*, \mathcal{B}^L_*) \right|\!\right|\!\right|_2^2,$$

*for every $\lambda > 0$.*

**Remark 1.8.** *Corollary 1.4 follows directly from Theorem 1.3, observing that $\sigma_\varepsilon \to \sigma$ in $C(\mathbb{R})$ as $\varepsilon \to 0$, where $\sigma_\varepsilon$ is defined in (1.19), and $\sigma$ denotes the ReLU activation function. The proof is provided in Section 6.*

1.3.3. *Universal Approximation Theorem.* We now analyze the property of universal approximation.

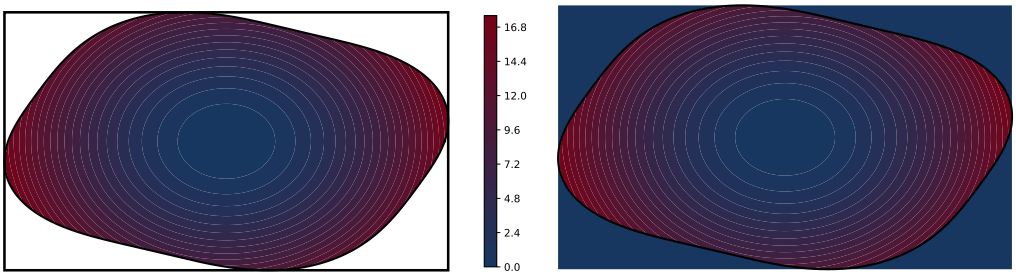
**Theorem 1.4** (Universal Approximation Theorem for $L^p(\Omega; \mathbb{R}_+)$)**.** *Let be $1 \leq p < \infty$, $d \geq 1$ an integer, and $\Omega \subset \mathbb{R}^d$ a bounded domain. For any $f \in L^p(\Omega; \mathbb{R}_+)$ and $\varepsilon > 0$, there exist a depth $\mathcal{L} = \mathcal{L}(\varepsilon) \geq 1$ and parameters $\mathcal{W}^\mathcal{L}$ and $\mathcal{B}^\mathcal{L}$ such that the input-output map of (1.2) with $w_{max} = d + 1$ satisfies*

$$\|\phi(\mathcal{W}^\mathcal{L}, \mathcal{B}^\mathcal{L}, \cdot) - f(\cdot)\|_{L^p(\Omega;\mathbb{R}_+)} < \varepsilon. \tag{1.20}$$

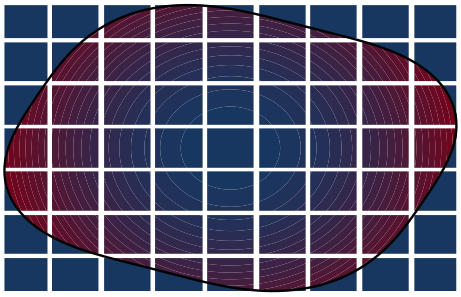
*Additionally, for all $f \in W^{1,p}(\Omega; \mathbb{R}_+)$, we have*

$$\mathcal{L} \leq C \|f\|^d_{W^{1,p}(\Omega;\mathbb{R}_+)} \varepsilon^{-d}, \tag{1.21}$$

*where $C$ is a positive constant depending on $m_d(\Omega)$, $d$ and $p$, $m_d(\cdot)$ being the Lebesgue measure in $\mathbb{R}^d$.*




(A) (B)



(C)

FIGURE 7. (A) Level sets of the paraboloid $f(x,y) = x^2 + y^2$ on $\Omega \subset \mathbb{R}^2$. The rectangle represents the set $\mathcal{C}$. (B) The function $f$ extended by zero to $\mathcal{C}$. (C) Level sets of $f$ on $\mathcal{H} = \mathcal{C} \setminus G_\delta^h$, $G_\delta^h$ being the white mesh.

**Remark 1.9.** *Theorem 1.4 ensures the existence of a neural network with a fixed width $d+1$ neurons and sufficient depth approximating any function in $L^p(\Omega;\mathbb{R}_+)$. As illustrated in the strategy of proof below, the neural network is constructed using geometrical arguments. Therefore, Theorem 1.4 not only merely asserts the existence of parameters, but also provides explicit ones.*

*Strategy of Proof of Theorem 1.4.* The proof is based on a two-step approximation procedure. First, a function $f \in L^p(\Omega;\mathbb{R}_+)$ is approximated by a simple function supported in a family of hyperrectangles. Then, the simple function is approximated by a deep enough neural network. The proof is outlined as follows:

**Step 1.** Let $f \in L^p(\Omega;\mathbb{R}_+)$ be a given function and $\varepsilon > 0$. Denote by $\mathcal{C}$ the smallest hyperrectangle containing $\Omega$, oriented according to the axes of the canonical basis of $\mathbb{R}^d$. We extend $f$ by zero into $\mathcal{C}$. Then, we construct a particular simple function $f_h$ that approximates $f$. For its construction, we consider a family $\mathcal{H}_h$ of hyperrectangles of size $h > 0$ such that $\mathcal{H}_h \cup G_h^\delta = \mathcal{C}$, where $G_h^\delta$ is a grid with thickness $\delta \leq h^{1+p}$ that satisfies $m_d(G_h^\delta) \to 0$ as $\delta \to 0$. This is illustrated in Figure 7. Then, $f_h$ is defined as the average value of the function $f$ on each hyperrectangle of the family $\mathcal{H}_h$. With this simple function, we can guarantee that there exists $h_1 > 0$ such that for every $h < h_1$, we have $\|f - f_h\|_{L^p(\mathcal{C};\mathbb{R}_+)} < \varepsilon/2$. We denote by $N_h$ the number of hyperrectangles in $\mathcal{H}_h$.

**Step 2.** Then, we construct a neural network $\phi^{\mathcal{L}}$ with a width $d+1$ and depth $\mathcal{L}$, ensuring the existence of $h_2 > 0$ such that for all $h < h_2$, we have $\|f_h - \phi^{\mathcal{L}}\|_{L^p(\mathcal{C};\mathbb{R}_+)} < \varepsilon/2$. This is done in two steps:

**Step 2.1.** Let $\mathcal{H}_h^E$ be the subset of hyperrectangles in $\mathcal{H}_h$ that are closest to the edges of $\mathcal{C}$, with $N_h^E$ denoting the number of hyperrectangles in $\mathcal{H}_h^E$. For each hyperrectangle $H \in \mathcal{H}_h^E$, we construct a two-layer neural network with an input-output map $\phi_1^2 : \mathbb{R}^d \to \mathbb{R}^d$. This map, which has a width $d+1$, drives

$H$ to a single point in $\mathbb{R}^d$ while mapping the remaining hyperrectangles in $\mathcal{H}_h$ to distinct, non-overlapping hyperrectangles in different locations. Here, the chosen parameters ensure the injectivity of the neural network with respect to the hyperrectangles in $\mathcal{H}_h$.

By iteratively applying these maps, we define a sequence of maps $\phi_i^2$ for $i \in \{1, \dots, N_h^E\}$, such that the composition $\phi^{2N_h^E} := (\phi_{N_h^E}^2 \circ \cdots \circ \phi_1^2)$ eventually drives all hyperrectangles in $\mathcal{H}_h^E$ to distinct points. This process results in the dimensional reduction of the remaining hyperrectangles in $\mathcal{H}_h$, where each $\phi_i^2$ transforms $n$-dimensional hyperrectangles into $(n-1)$-dimensional hyperrectangles. Eventually, all hyperrectangles in $\mathcal{H}_h$ are mapped to distinct points $\{x_i\}_{i=1}^{N_h} \subset \mathbb{R}^d$. This approach leverages the specific choice of $\mathcal{H}_h^E$, the dimensionality reduction, and the injectivity of the map $\phi^{2N_h^E}$.

**Step 2.2** Then, we apply Theorem 1.1 to construct an input-output map $\phi^L$ driving the points $\{x_i\}_{i=1}^{N_h}$ to the $M_h$ values of $f_h$, ensuring that the composition $\phi^{\mathcal{L}} := (\phi^L \circ \phi^{2N_h^E})$ equals $f_h$ on $\mathcal{H}_h$, and therefore $\|f_h - \phi^{\mathcal{L}}\|_{L^p(\mathcal{H}_h;\mathbb{R}_+)} = 0$. Additionally, we estimate the error introduced by the neural network on $G_h^\delta$ and show that, for a fixed $h < 1$, the norm $\|\phi^{\mathcal{L}}\|_{L^\infty(G_h^\delta;\mathbb{R}_+)}$ is bounded. This estimation allows us to ensure that there exists a $h_2 > 0$ such that for all $h < h_2$, we have $\|f_h - \phi^{\mathcal{L}}\|_{L^p(G_h^\delta;\mathbb{R}_+)} < \epsilon/2$.

**Step 3.** As a consequence of the triangle inequality and by choosing $h < \min\{h_1, h_2\}$, we obtain $\|f - \phi^{\mathcal{L}}\|_{L^p(\Omega;\mathbb{R}_+)} < \varepsilon$.

**Step 4.** Due to the explicit construction of the neural network $\phi^{\mathcal{L}}$, the depth $\mathcal{L}$ can be estimated in terms of $h$. Additionally, assuming $f$ to be in $W^{1,p}$, explicit estimates for $h_1$ and $h_2$ can be obtained in terms of $\|f\|_{W^{1,p}(\Omega;\mathbb{R}_+)}$, $\varepsilon$, $p$ and $d$ (see [11] and its application to isotropic partitions), and therefore conclude (1.21).

**Remark 1.10.** *Several remarks concerning the strategy of the proof of Theorem 1.4 are in order:*

- **Step 2**, *involving the neural network approximation, is the most challenging one in the proof.*
- *As shown in Section 2, each neuron in the neural network represents a hyperplane. In* **step 2.1**, $d+1$ *neurons are necessary because this is the number of hyperplanes required to separate a single edge hyperrectangle in* $\mathcal{H}_h^E$. *For instance, in the 2-dimensional case, a hyperrectangle on the left edge of* $\mathcal{C}$ *has only one adjacent hyperrectangle above, one below, and one to its right (there are no hyperrectangles outside* $\mathcal{C}$*). Thus, only* $d+1 = 3$ *hyperplanes are necessary to separate this hyperrectangle from the others, as opposed to the case where a hyperrectangle is in the interior of* $\mathcal{C}$, *where four hyperrectangles would surround it, and it would be necessary to use* $2d = 4$ *hyperplanes. These* $d+1$ *hyperplanes, in particular, ensure the compression of the separated hyperrectangle into a point.*
- *The set* $\mathcal{H}_h^E$ *allows us to define a neural network with* $2N_h^E$ *layers, mapping the* $N_h$ *hyperrectangles of* $\mathcal{H}_h$ *to* $N_h$ *distinct points. Focusing on* $\mathcal{H}_h^E$ *is essential, as it reduces the number of required layers. Mapping each hyperrectangle of* $\mathcal{H}_h$ *individually would result in a neural network with significantly more layers, i.e.,* $N_h \gg N_h^E$.
- *The neural network in Theorem 1.4, defined as* $\phi^{\mathcal{L}} = \phi^L \circ \phi^{2N_h^E}$ *requires a width* $d+1$. *However, only the first* $2N_h^E$ *layers require this width. The width of the* $L$ *remaining layers can be reduced to* 2 *(as a consequence of Theorem 1.1).*
- *The neural network of width* 2 *from Theorem 1.1 maps the compressed hyperrectangles to their corresponding labels and requires* $L = 2N_h + 4M_h - 1$ *layers, where* $M_h$ *is the number of distinct values taken by the approximating simple functions* $f_h$ *(which act as labels). In particular,* $M_h \leq N_h$, *and* $N_h$ *can be estimated in terms of* $h$ *and* $\delta$. *Moreover, since* $N_h^E \leq N_h$ *and* $M_h \leq N_h$, *the depth of the neural network* $\phi^{\mathcal{L}}$ *is bounded by* $\mathcal{L} = 2N_h^E + 2N_h + 4M_h - 1 \leq 8N_h - 1$. *Finally, following Step 4, we can estimate the total depth.*

**Remark 1.11** (An equivalent statement of Theorem 1.4)**.** *Observe that Theorem 1.4 asserts that for every* $\mathcal{L} > 0$, *there exist parameters* $\mathcal{W}^{\mathcal{L}}$ *and* $\mathcal{B}^{\mathcal{L}}$ *such that the input-output map of* (1.2) *with* $w_{\max} = d+1$ *satisfies*

$$\|\phi^{\mathcal{L}}(\mathcal{W}^{\mathcal{L}}, \mathcal{B}^{\mathcal{L}}, \cdot) - f(\cdot)\|_{L^p(\Omega;\mathbb{R}_+)} \leq C' \|f\|_{W^{1,p}(\Omega;\mathbb{R}_+)} \mathcal{L}^{-1/d},$$

*where* $C'$ *is a positive constant depending on* $m_d(\Omega)$, $d$, *and* $p$.

*It is important to mention that in the absence of a finite $W^{1,p}$-norm, a function $f \in L^p(\Omega;\mathbb{R}_+)$ may exhibit arbitrarily rapid oscillations, making any nontrivial approximation rate unattainable. Therefore, the finiteness of the $W^{1,p}$-norm is essential.*

As a consequence of the preceding theorem and Corollary 1.2, we have the following universal approximation theorem for $L^p(\Omega;\mathbb{R}^m_+)$ functions.

**Corollary 1.5** (Universal Approximation Theorem in $L^p(\Omega;\mathbb{R}^m_+)$)**.** *Let us consider $1 \leq p < \infty$, two integers $m$, $d \geq 1$, and a bounded domain $\Omega \subset \mathbb{R}^d$. Then, for any $f \in L^p(\Omega;\mathbb{R}^m_+)$ and $\varepsilon > 0$, there exist $\mathcal{L} = \mathcal{L}(\varepsilon) \geq 1$ and parameters $\mathcal{W}^{\mathcal{L}}$, and $\mathcal{B}^{\mathcal{L}}$ such that the input-output map of* (1.2) *with width $w_{max} = \max\{d+1, 2m\}$, satisfies*

$$\|\phi(\mathcal{W}^{\mathcal{L}}, \mathcal{B}^{\mathcal{L}}, \cdot) - f(\cdot)\|_{L^p(\Omega;\mathbb{R}^m_+)} < \varepsilon. \tag{1.22}$$

*Moreover, estimate* (1.21) *for the depth $\mathcal{L}$ is still valid for $f \in W^{1,p}(\Omega;\mathbb{R}^m_+)$.*

*Strategy of proof for Corollary 1.5.* The proof follows a methodology analogous to that of Theorem 1.4, approximating, first, the function $f$ by simple functions with support in hyperrectangles. Then, we construct a neural network of width $d+1$ approximating the second simple function. This is done by mapping the hyperrectangles to a set of different points in $\mathbb{R}^d$. Subsequently, to map these points to their $m$-dimensional targets, we utilize $2m$ hyperplanes by applying Corollary 1.2 instead of Theorem 1.1. The width of the network is then determined by the maximum between $d+1$ and $2m$. Furthermore, since Corollary 1.2 also employs a depth of $2N + 4M - 1$, the estimated depth of the neural network of Corollary 1.5 is that in Theorem 1.4.

Clearly, the input-output map of (1.2) cannot approximate functions with negative values. However, in view of Corollary 1.3, we know that this can be done by means of the more general architecture (1.7). Using (1.7) the universal approximation result can be extended to functions in $L^p(\Omega;\mathbb{R}^m)$.

**Corollary 1.6** (Universal Approximation Theorem in $L^p(\Omega;\mathbb{R}^m)$)**.** *Let us consider $1 \leq p < \infty$, integers $m$, $d \geq 1$, and a bounded domain $\Omega \subset \mathbb{R}^d$. Then for any $f \in L^p(\Omega;\mathbb{R}^m)$ and $\varepsilon > 0$, there exist parameters $\mathcal{A}^{\mathcal{L}}$, $\mathcal{W}^{\mathcal{L}}$, $\mathcal{B}^{\mathcal{L}}$, and $\mathcal{L} = \mathcal{L}(\varepsilon) \geq 1$ such that the input-output map of* (1.7) *with $w_{max} = \max\{d+1, 2m\}$, satisfies*

$$\|\phi(\mathcal{A}^{\mathcal{L}}, \mathcal{W}^{\mathcal{L}}, \mathcal{B}^{\mathcal{L}}, \cdot) - f(\cdot)\|_{L^p(\Omega;\mathbb{R}^m)} < \varepsilon. \tag{1.23}$$

*Moreover, estimate* (1.21) *for the depth $\mathcal{L}$ is still valid for $f \in W^{1,p}(\Omega;\mathbb{R}^m)$.*

*Strategy of Proof of Corollary 1.6.* The proof is analogous to the proof of Corollary 1.5, and it concludes by replacing Corollary 1.2 with Corollary 1.3.

1.4. **Related Work.** Deep learning has gained popularity due to its state-of-the-art performance in various machine learning applications [24, 38]. In practice, neural networks are typically trained using optimization methods minimizing a least-squares error functional, with stochastic gradient descent algorithms serving as an essential tool to search for minimizers. While this numerical approach, combined with backpropagation techniques to compute the gradients, often leads to solutions that outperform human experts, we still lack a solid mathematical understanding of why deep learning works so well. The results in this paper aim to contribute to explain such performance by explicit constructions, which yield concrete estimates of the complexity required for neural networks to achieve the desired goals – in particular memorization, and universal approximation – and explicit bounds for the trained parameters.

In this section, we describe some recent related advancements in this broad field.

**Finite sample memorization:** The literature on the memorization capacity of linear threshold networks, employing a step function $\sigma$, dates back to the 1960s [9, 5, 27]. In the 1990s, the analysis of single-hidden layer neural networks (FNNs) with more general nonlinear bounded activation functions, such as sigmoids, was conducted ([20, 21]). These studies show that a single-hidden layer neural network of width $N$ can memorize $N$ points with $N$ classes. A similar result was proven in [47], showing that a single hidden layer ReLU network with $N$ neurons can memorize $N$ arbitrary real points, see also [19]. In [46], it was proved that a 2-hidden layer ReLU network with widths $d_1$ and $d_2$ can memorize a dataset with $N$ points with $N$ classes if $d_1 d_2 \geq 4Nm$, where $m$ is the dimension of the labels. Therefore, for $m = 1$, the width of the neural network in [46] is $d_1 = d_2 = 2\sqrt{N}$. The above shows that a 2-hidden layer ReLU network can memorize $N$ points with $O(\sqrt{N})$ neurons.

In the context of deep neural networks, [45] constitutes one of the first attempts with sigmoid activation functions. For the ReLU activation function, [46] demonstrated that for a fixed depth $L$, a neural network with a width depending on $N$ and satisfying a technical assumption can memorize $N$ data. In particular, this result holds if there exists $l > 1$ such that $d_j d_j + l = O(Nm)$ for some $j > 1$. In [32], it is shown that ReLU networks with a width greater than 3 and $O(N^{2/3}log(M))$ neurons suffice to approximately memorize $N$ points with $M$ one-dimensional classes, in the sense that the data can be driven to be $\varepsilon$-close to the labels, where $\varepsilon > 0$ (this constitutes an approximate simultaneous or ensemble controllability result). The same article also establishes that a 3-wide neural network with $O(N^{2/3}log(M))$ layers (or neurons) suffices for such approximate memorization. Finally, [43] showed that, by fixing a width 12, networks can memorize any $N$ points with $M$ one-dimensional classes using $O(N^{1/2} + log(M))$ layers (or neurons). See [43] for the exact expression of the required depth. We highlight that none of the previously mentioned works provides an explicit estimate of the parameter norms ($\ell^2$ or $\ell^\infty$); they focus mainly on the number of parameters.

From the control theory perspective, in [35, 3], simultaneous controllability for ResNets and neural ordinary differential equations is proven (see also [1]). This implies the memorization property. The novelty in [35] lies in the genuinely constructive approach to building parameters. An extension of [35] can be found in [8], where it is shown that for neural networks with sufficiently large depth but fixed width, interpolation can be guaranteed through the use of non-linear activation functions. Finally, we mention that constructive methods to prove controllability have been widely used in classical control theory; see [6, 44, 14, 22, 23] and the references therein for a detailed discussion.

**Universal Approximation Theorem:** Classical results in this field primarily focus on shallow neural networks [4, 10, 18, 28, 34], with Cybenko's celebrated work [10] as a notable example, who proved that a single hidden layer neural network could approximate any continuous function within a compact set of $\mathbb{R}^n$ using a sigmoidal activation function. However, such density results trace back to 1932, with Wiener's Tauberian theorem, which provides necessary and sufficient conditions under which any function in $L^1(\mathbb{R})$ or $L^2(\mathbb{R})$ can be approximated by linear combinations of translations of a given $L^1(\mathbb{R})$ profile, the gaussian, for instance.

On the other hand, recent years have demonstrated that deep networks typically offer better approximation capabilities compared to shallow networks. In this context, [41] shows that if a ReLU deep neural network is capable of approximating a function with a given error $\varepsilon$ using $L$ layers and relatively narrow width, then a shallower network with a fixed depth of $O(L^{1/3})$ layers would require a width that increases exponentially with $L$ to achieve the same approximation error $\varepsilon$. This finding highlights one of the principal advantages of deeper architectures in neural networks.

With regard to universal approximation in $L^p$ spaces, [31] demonstrates that a deep neural network with the ReLU activation function and width $d + 4$ can approximate any function in $L^1(\mathbb{R}^d; \mathbb{R})$. They allocate $d$ neurons for transferring input information to subsequent layers, two neurons to carry the information of the approximation made by the previous layers, and two neurons for approximation on each layer. The same article also proves that if the width of a deep neural network is less than $d$, it is impossible to approximate $L^1(\mathbb{R}^d; \mathbb{R})$ or $L^1(\Omega; \mathbb{R})$ for a compact $\Omega$. In [25], for $p \in [1, \infty)$, a compact set $\Omega \subset \mathbb{R}^d$, and $m \geq 1$, it is established that it is possible to approximate the spaces $L^p(\mathbb{R}^d; \mathbb{R}^m)$ and $L^p(\Omega; \mathbb{R}^m)$ with a ReLU network of width $d + m + 1$. Their main argument for approximating $L^p(\mathbb{R}^d; \mathbb{R}^m)$ involves using a neural network to approximate cutoff functions. For $L^p(\Omega; \mathbb{R}^m)$ functions, they prove universal approximation for $C(\Omega; \mathbb{R}^m)$ functions using $m+d+1$ neurons, concluding by density. More precise estimates of the minimal width in $L^p(\mathbb{R}^d; \mathbb{R}^m)$ and $L^p(\Omega; \mathbb{R}^m)$ are presented in [33], which determine the minimal widths to be $\max\{d + 1, m\}$ and $\max\{d + 2, m + 1\}$, respectively. The proof of this theorem utilizes a coding scheme, consisting of encoding (projecting) $x \in \Omega$ into a codeword (scalar values) containing information about $x$, and then a decoder transforming each codeword into a target function $f(x)$. This scheme is applied to approximate continuous functions and completed with density arguments in $L^p$. In the particular case of the *Leaky-ReLU* activation function, a variant of ReLU, in [7] it was proven that for a compact $\Omega \subset \mathbb{R}^d$, the minimum width required to approximate functions in $L^p(\Omega; \mathbb{R}^m)$ is $\max\{d, m, 2\}$. Recently, [26] has shown that the minimum width of a neural network with a ReLU activation function necessary to approximate $L^p(\Omega; \mathbb{R}^m)$ is $\max\{d, m, 2\}$, $\Omega$ being a compact set. The proof of this result, based on [33], employs the coding scheme to approximate continuous functions in a compact $\Omega$, concluding the result for $L^p(\Omega; \mathbb{R}^m)$ functions by density.

Universal approximation theorems for the space of continuous functions in the case of arbitrary depth are discussed in [15, 16, 25, 33, 40, 29]. In particular, the minimal width for approximating $L^p$ functions

using the recurrent neural networks (RNN) architecture has been studied in [39]. We refer to [2] for universal approximation theorems in $L^p$, using transformers. For an extended introduction to the universal approximation theorem in more general spaces, see the survey article [12].

1.5. **Our contribution.** In Theorem 1.1, we present a constructive proof of the fact that the neural network defined by (1.2) satisfies the finite sample memorization property with width 2 and depth $L = 2N + 4M - 1$, which implies a total number of neurons of order $O(N)$. To the best of our knowledge, this is the first purely constructive and geometrically interpretable proof of memorization for narrow MLPs. Our approach allows not only for an explicit step-by-step construction of the network parameters but also provides explicit upper bounds on their norms. This has important implications for regularized training: minimizing a standard empirical loss with $\ell^2$-regularization leads to parameters whose norms are uniformly bounded by the ones of our constructed parameters. In the limit of vanishing regularization, this guarantees exact interpolation while maintaining control over the norm of the trained parameters.

Regarding universal approximation, for MLP of fixed width $d+1$, we establish universal approximation results in $L^p(\Omega; \mathbb{R}+)$ and $L^p(\Omega; \mathbb{R})$, for any $p \in [1, \infty)$ and bounded domain $\Omega \subset \mathbb{R}^d$. Again, our primary contribution lies in our purely constructive proof based on a detailed geometric and recursive argument. In contrast to other constructive results such as [33, 12, 26], our approach emphasizes geometric intuition and visualization at each layer, following ideas similar to [35]. While the width requirement in [26] is optimal (namely, width equals $d$), our construction requires width $d+1$ but yields a fully interpretable and elementary strategy to understand how approximation is achieved. Moreover, all parameters involved are given explicitly, which allows us to estimate the depth required to approximate functions with a prescribed accuracy.

1.6. **Outline.** The rest of the paper is organized as follows: In Section 2, we conduct a geometric analysis of the discrete system (1.2), introducing fundamental tools essential to our proofs. In Section 3, we offer an informal demonstration on constructing parameters to guarantee the finite sample memorization property, illustrated with a specific example. Section 4 contains the formal proof of Theorem 1.1, followed by the proof of the universal approximation theorem (Theorem 1.4). Section 6 provides the proof of the theorems related to the regularized training and vanishing regularization. Finally, in Section 7, we discuss extensions and open problems.

1.7. **Notation.** Throughout this article, we will use the following notation:

- We denote by $[\![1, L]\!]$ the set of numbers $\{1, \dots, L\}$.
- The symbol $\cdot$ denotes the Euclidean scalar product between two vectors.
- Given a set $Q$, its cardinal is denoted by $|Q|$.
- $\mathbb{S}^d$ denotes the unit $d-$sphere in $\mathbb{R}^{d+1}$.
- $\mathcal{W}^L$ and $\mathcal{B}^L$ denote the families of parameters $\{W_j\}_{j=1}^L$ and $\{b_j\}_{j=1}^L$, respectively.
- $w_{max}$ denotes the width of the neural network defined as $\max_{j\in[\![1,L]\!]}\{d_j\}$.
- $m_d(\Omega)$ denotes the Lebesgue measure of $\Omega$ in $\mathbb{R}^d$.
- $\|w\|$ stands for the Euclidean norm of the vector $w$ in $\mathbb{R}^d$.
- $\sigma$ denotes the ReLU function and $\boldsymbol{\sigma}$ denotes its vector-valued version.

## 2. Preliminaries

2.1. **Geometrical interpretation.** This section illustrates the dynamics of the system (1.2) from a geometric perspective. In what follows, we will refer to the property of finite sample memorization as simply data classification. To simplify the notation, we also avoid the dependence of $\boldsymbol{\sigma}_j =: \boldsymbol{\sigma}$ with respect to the dimension $d_j$ for every $j \in [\![1, L]\!]$.

2.1.1. *A single hyperplane:* Let us begin by analyzing the simple case $(N, L, d, d_1) = (1, 1, 2, 1)$. Consider $x^0 \in \mathbb{R}^2$, $w \in \mathbb{R}^{1\times 2}$ and $b \in \mathbb{R}$. Under these conditions, the system (1.2) corresponds to

$$x^1 = \sigma(w \cdot x^0 + b) \in \mathbb{R}.$$

Let the hyperplane

$$H := \{x \in \mathbb{R}^2 \,:\, w \cdot x + b = 0\}, \tag{2.1}$$

which divides the space into two half-spaces determined by $w \cdot x + b > 0$ and $w \cdot x + b \leq 0$ respectively. Thus, the value of $\sigma(w \cdot x^0 + b)$ is either zero or equals to $w \cdot x^0 + b = \|w\| d(x^0, H)$, depending on the sign

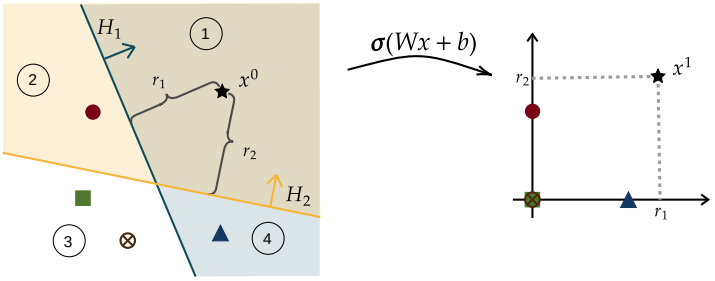



FIGURE 9. Left: Two hyperplanes split the space into four regions. Different points are chosen in each region. Right: Output of the nonlinear map $\boldsymbol{\sigma}(Wx+b)$. The green square and brown cross in region 3 are both mapped to the same point, $(0,0)$. The black star is mapped to a new one in the first quadrant, according to its distances to the two hyperplanes. The other two points are mapped to the coordinate axes according to the distance to the hyperplane of the active component.

of $w \cdot x^0 + b$. Here $d(x^0, H)$ denotes the distance between $x^0$ and the hyperplane $H$. This is illustrated in Figure 8.

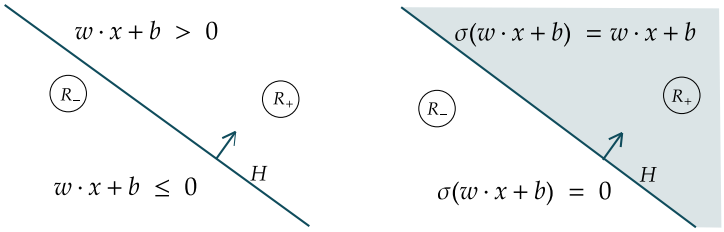



FIGURE 8. Left: $H$ divides the space into two half-spaces $R_+$ and $R_-$. Right: $R_+$ represents the half-space where $\sigma$ is active, while $R_-$ represents its null half-space.

For future reference, we will say that $x^0$ *is in the activation sector (or region) of* $H$ if $w \cdot x^0 + b > 0$. In Figure 8, the sector where the hyperplane $H$ is activated is denoted by $R_+$.

Note that by appropriately choosing the norm of $w$, the distance of the points $x$ within the activation sector can be scaled, either by moving the points closer to or further away from the hyperplane $H$.

2.1.2. *Two and more hyperplanes.* Consider two vectors $w^1, w^2 \in \mathbb{R}^{1\times 2}$, and scalars $b^1, b^2 \in \mathbb{R}$. Let us define the matrix $W = (w^1, w^2)^\top$ and the vector $b = (b^1, b^2)^\top$. Then, for $x^0 \in \mathbb{R}^2$, we have

$$x^1 = \boldsymbol{\sigma}(Wx^0 + b) = \begin{pmatrix} \sigma(w^1 \cdot x^0 + b^1) \\ \sigma(w^2 \cdot x^0 + b^2) \end{pmatrix}. \tag{2.2}$$

Denote by $H_1$ and $H_2$ the two hyperplanes defined by $w^1 \cdot x + b^1 = 0$ and $w^2 \cdot x + b^2 = 0$, respectively. Let $r_1 = \|w^1\| d(x^0, H_1)$ and $r_2 = \|w^2\| d(x^0, H_2)$, then we have that

$$\sigma(w^1 \cdot x^0 + b^1) = \begin{cases} r_1, & \text{if } x^0 \text{ is in the activation sector of } H_1, \\ 0, & \text{otherwise,} \end{cases}$$

while for the second coordinate,

$$\sigma(w^2 \cdot x^0 + b^2) = \begin{cases} r_2, & \text{if } x^0 \text{ is in the activation sector of } H_2, \\ 0, & \text{otherwise.} \end{cases}$$

The hyperplanes $H_1$ and $H_2$ partition the plane into four disjoint regions. Depending on the region where a given point $x \in \mathbb{R}^2$ lies, the function $\boldsymbol{\sigma}(Wx+b)$ takes a particular value, as depicted in Figure 9, mapping $x^0$ into a new point $x^1$ with coordinates $(r_1, r_2)$, which depend, in particular, on the sector where $x^0$ lies. All points in region 1 are mapped to the first quadrant of the plane. Points in regions 2 and 4 are mapped to the coordinate axes. Meanwhile, all points in region 3, the kernel of the map $\boldsymbol{\sigma}(Wx+b)$, are mapped to the origin.

**Remark 2.1.** *Some comments are in order.*
• *As discussed,* $K := \{x \in \mathbb{R}^2 \,:\, w^1 \cdot x + b^1 \leq 0, \text{ and } w^2 \cdot x + b^2 \leq 0\}$ *is the kernel of* $\boldsymbol{\sigma}(Wx+b)$*. It is determined by the parameters* $W$ *and* $b$*, that are to be designed to map points to the null point via* $\boldsymbol{\sigma}$*. This allows clustering data. However, it is crucial to choose parameters* $W$*, and* $b$ *carefully to ensure that the kernel* $K$ *does not contain data related to different labels; otherwise, the map* $\boldsymbol{\sigma}(Wx+b)$ *would collapse different labeled points into the same one. If this were to happen, this would render the data classification task impossible.*

• *The same construction can be extended to any number of hyperplanes by considering* $W \in \mathbb{R}^{r\times d}$ *and* $b \in \mathbb{R}^r$*. In this case, the function* $\boldsymbol{\sigma}(Wx+b)$ *defines a partition of* $\mathbb{R}^d$ *determined by the family of hyperplanes* $H_j = \{x \in \mathbb{R}^d \,:\, w_j \cdot x + b_j = 0\}$ *for* $j \in [\![1, r]\!]$*. These hyperplanes determine the convex kernel* $K$*, which is not necessarily unbounded since* $d+1$ *hyperplanes (or more) in a* $d-$*dimensional space, can determine a bounded kernel, which is then a convex polyhedron.*

• *The hyperplanes and the norm of* $W$ *can be determined first by geometric considerations, and then their parameters are extracted a posteriori to set the neural network's weight and bias.*

2.2. **Projection Lemma.** We present a technical result that will be systematically applied in our proof. This lemma ensures that, given a finite number of points in a $d$-dimensional space, we can always find a direction determining an injective one-dimensional projection of the data.

**Lemma 2.1.** *Let us consider a finite set of distinct data* $\{x_i\}_{i=1}^N \subset \mathbb{R}^d$ *such that* $x_j \neq x_i$ *if* $i \neq j$*. Then there exists a vector* $v \in \mathbb{S}^{d-1}$ *such that*

$$v \cdot x_i \neq v \cdot x_j, \qquad \text{for every } i \neq j \in [\![1, N]\!]. \tag{2.3}$$

*Proof.* For each pair $1 \leq i < j \leq N$, define the set of directions

$$H_{ij} = \big\{v \in \mathbb{S}^{d-1} \,:\, v \cdot (x_i - x_j) = 0\big\}.$$

Since $x_i \neq x_j$, the vector $q_{ij} := x_i - x_j \in \mathbb{R}^d \setminus \{0\}$, so $H_{ij}$ is the intersection of the unit sphere with the hyperplane orthogonal to $q_{ij}$. Hence, $H_{ij}$ is a closed subsphere of dimension $d-2$ and thus a proper, nowhere-dense subset of $\mathbb{S}^{d-1}$. Set

$$G = \bigcup_{1\leq i<j\leq N} H_{ij}.$$

Then $G$ is a finite union of closed, nowhere-dense subsets of the compact manifold $\mathbb{S}^{d-1}$. By the Baire Category Theorem (or by observing that each $H_{ij}$ has surface-measure zero and hence their finite union cannot exhaust the full-measure sphere), we conclude $\mathbb{S}^{d-1} \setminus G \neq \emptyset$. Any $v \in \mathbb{S}^{d-1} \setminus G$ satisfies

$$v \cdot (x_i - x_j) \neq 0, \quad \forall i \neq j,$$

which is equivalent to $v \cdot x_i \neq v \cdot x_j$ whenever $i \neq j$, concluding the proof. □

**Remark 2.2.** *A few remarks are necessary.*
• *The above argument extends without modification to any countable collection of pairwise distinct points* $x_{i\,i\in\mathbb{N}} \subset \mathbb{R}^d$*. Indeed,* $G$ *is a countable union of closed subspheres of codimension 1, so by the Baire Category Theorem* $\mathbb{S}^{d-1} \setminus G \neq \varnothing$*. However, for an uncountable (e.g., continuous) data set, one may have* $G = \mathbb{S}^{d-1}$*, and thus, no separating direction exists.*
• *If the data set is finite, each* $H_{ij}$ *is a closed subsphere of dimension* $d-2$*, and their finite union* $G$ *is a closed set with empty interior. Hence* $G^c = \mathbb{S}^{d-1} \setminus G$ *is open and dense. In particular, for any* $g \in G$ *and any* $\varepsilon > 0$*, there exists* $g_\varepsilon$ *in an* $\varepsilon-$*neighborhood of* $g$ *such that* $g_\varepsilon \cdot (x_i - x_j) \neq 0$ *for all* $i \neq j$*, showing that the projection property is stable under small perturbations.*

## 3. Sketch of the proof of Theorem 1.1: An example

In this section, we illustrate the proofs of Theorem 1.1 through a specific example. The formal proof and the estimation of the depth $L$ are provided in the subsequent section.

Let us consider the dataset $\{x_i, y_i\}_{i=1}^8 \subset \mathbb{R}^3 \times [\![0, 3]\!]$. For ease of visualization, we assume their labels correspond to four shapes of different colors: red circle, blue triangle, green square, and brown cross. We aim to drive blue triangles to 0, red circles to 1, brown crosses to 2, and green squares to 3. In the following, we will refer to a *class of elements* as a set that contains points associated with the same label. Therefore, in our example, we have four such classes of elements.

In the ensuing discussion, we illustrate the main steps needed to complete the classification process.

**1) Preconditioning of the data:** We choose the parameters $w_1$ and $b_1$ to project the 3−dimensional data injectively into a one-dimensional space, ensuring that all the projected data remain distinct. Note that the neural network's width in this step is one since we use one hyperplane.

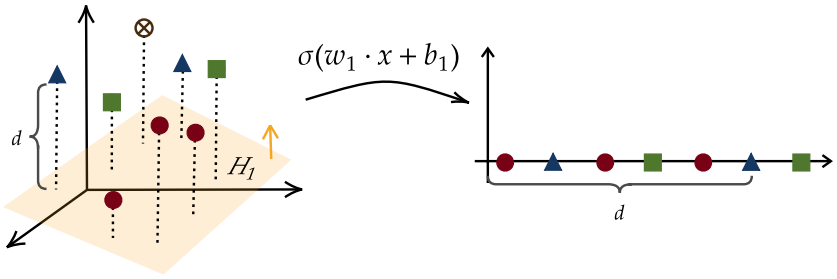
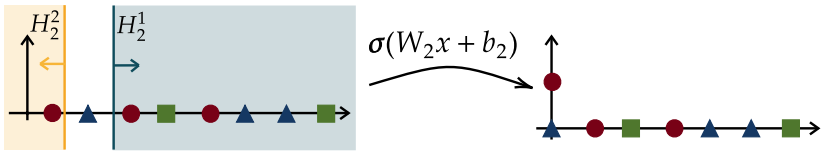


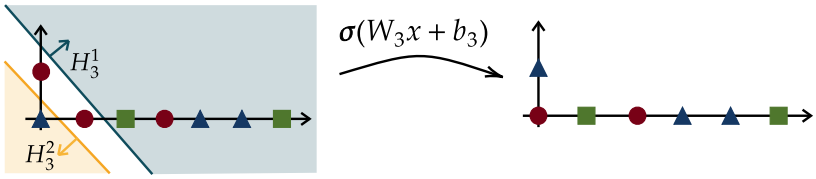

FIGURE 10. Projection of the data in a one-dimensional space using the hyperplane $H_1$.

**2) Compression process:** Inspired by Remark 2.1, in this step, we aim to collapse each class of elements into a single point. For this purpose, it is enough to show how to collapse a single class while keeping the other three classes well separated throughout the process, and then proceed inductively. We illustrate this step by compressing the red circles, indicating which hyperplanes are needed to carry out this process.

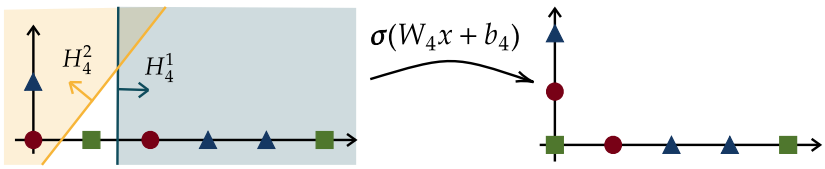
• Step 2.1: Starting from the output of the previous step, we define two hyperplanes, $H_2^1$ and $H_2^2$. These are chosen such that the red circle at the left is mapped to the $y-$axis, the blue triangle in between is driven to $(0,0)$, and the remaining data points are mapped to the $x-$axis.




• Step 2.2: We now consider two diagonal hyperplanes $H_3^1$ and $H_3^2$ enclosing the red circles between them. Their activation sector is chosen to make the two red circles collapse to the origin (see Remark 2.1).




• Step 2.3: With the same goal as in Step 2.1, we consider two hyperplanes $H_4^1$ and $H_4^2$ that place one red circle on the $y-$axis and another red circle on the $x-$axis.




The slope of $H_4^2$ plays an important role since, if the hyperplane was vertical, the blue triangle and red circle to its left (and all the points that lie on $y-$axis) would be at the same distance from the hyperplane $H_4^2$ and thus would be driven to the same value.

We can iteratively apply Steps 2.2 and 2.3, as many times as necessary, to collapse all the red circles.

By applying this process to each class, we can define the input-output map $\phi^{L_2} : \mathbb{R} \to \mathbb{R}^2$ described by Figure 11.

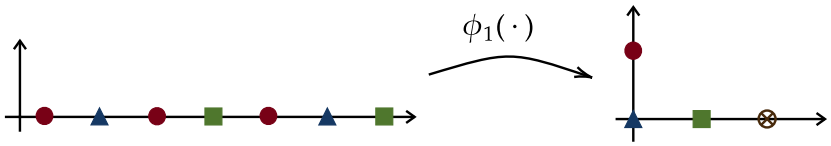


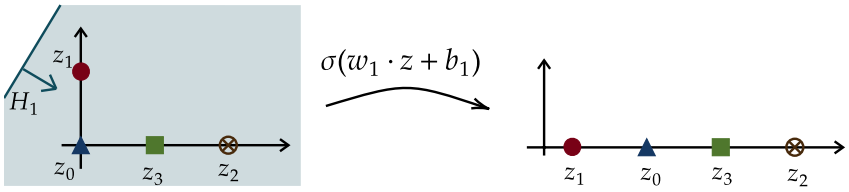

FIGURE 11. Compression of each class into a single point using the mapping $\phi_1$.

**3) Data sorting:** After completing the previous step, all points within each class have been driven into the same position. Hence, the points of each group become indistinguishable and inseparable, allowing us to treat them as a single reference point. Let us denote the reference point associated with the label $i$ as $z_i$.

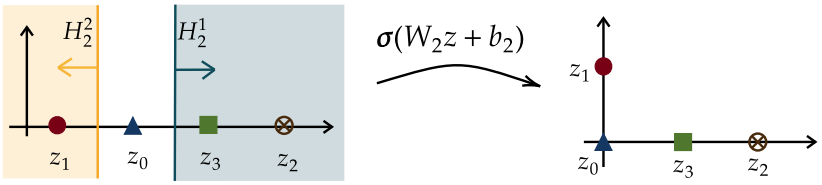
Note, however, that the outputs of the last step do not provide any specific ordering of these reference points. In this third step, our aim is to show how to reorder these reference points along the real line according to their labels.

We will outline the first steps and illustrate how to carry out the inductive process.

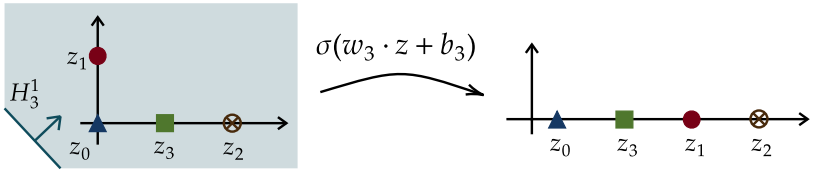
• Step 3.1: We begin by projecting the two-dimensional data into the real line using any hyperplane, ensuring that all projected data remain distinct.



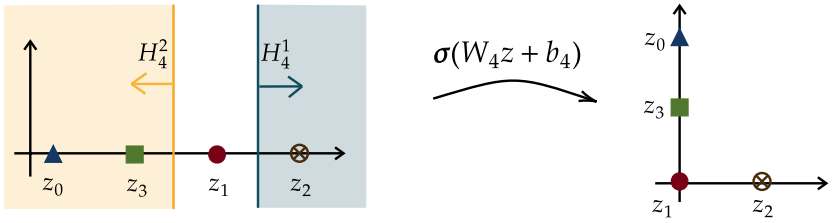

• Step 3.2: We consider two vertical hyperplanes to drive only the first point $z_0$ to $(0, 0)$.



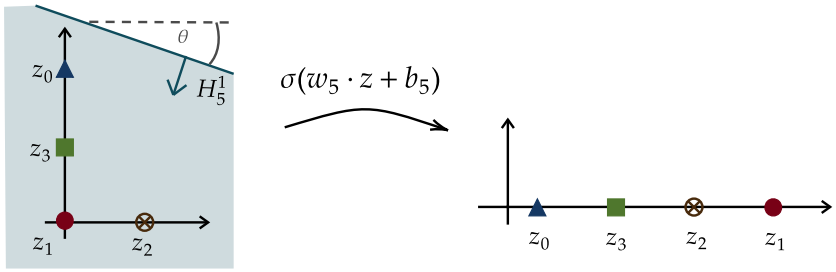

• Step 3.3: We consider a hyperplane such that the activated semi-space contains all the points, and the closest point to it is the one in $(0, 0)$. This allows us to sort the first point.



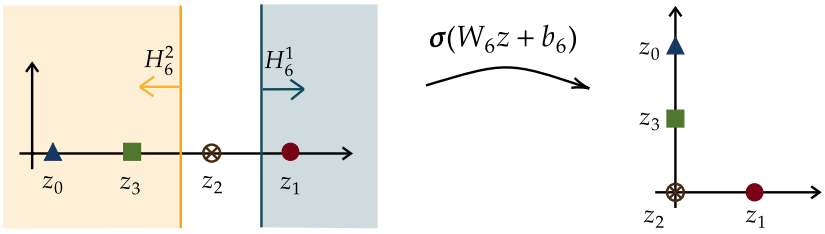

• Step 3.4: Again, we consider vertical hyperplanes $H_4^1$ and $H_4^2$, this time to drive only $z_1$ to $(0, 0)$.



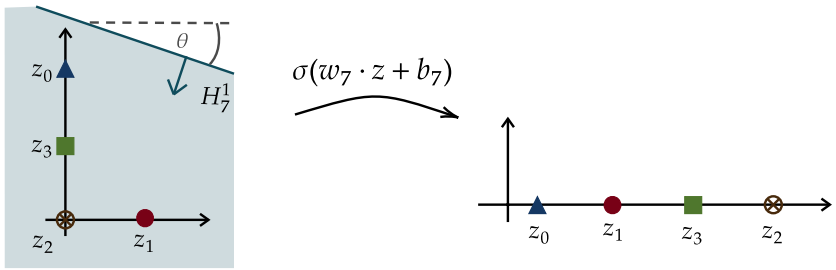

• Step 3.5: We consider a hyperplane $H_5^1$ such that the nearest point is $z_0$ and the farthest point is $z_1 = (0, 0)$. This allows us to ensure that $z_1$ will be the farthest point from zero.




• Step 3.6: We consider vertical hyperplanes to drive only $z_2$ to $(0, 0)$.



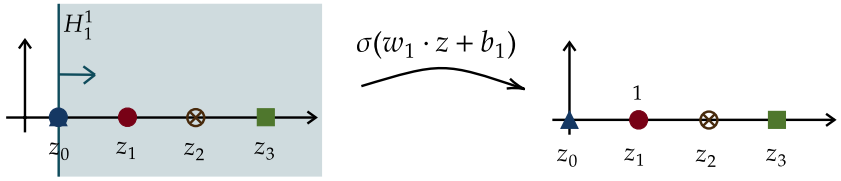

• Step 3.7: We consider a hyperplane such that the closest point is $z_0$, then $z_1$, and the farthest point is $z_2 = (0, 0)$.



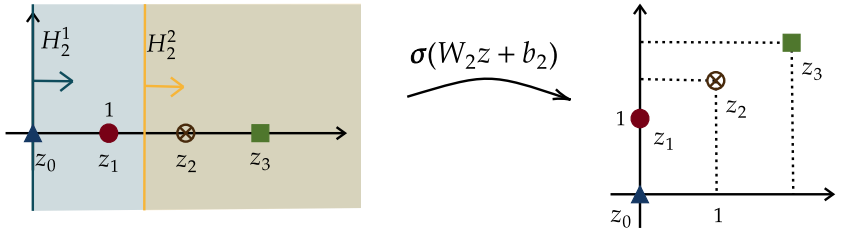

The first two points are well collocated, and the data we want to sort next, $z_2$, is at the end. Applying steps 3.6 and 3.7 iteratively, we can sort $z_2$ and all the remaining positions, always taking a suitable slope $\theta$ for the hyperplane in Step 3.5.

**4) Mapping to the respective labels:** We will show how to drive each point to its corresponding label. This is done by applying projections and choosing the weights properly to have a correct distance scaling.

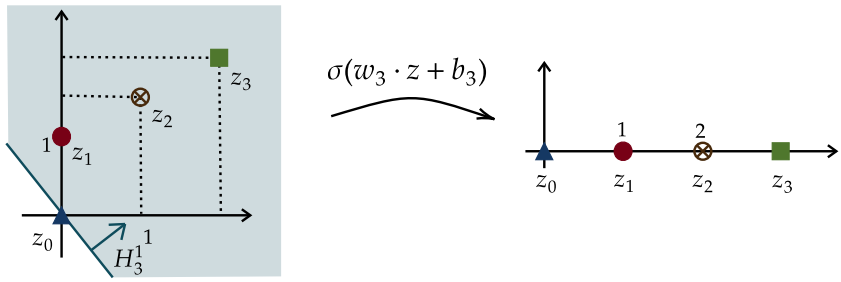
• Step 4.1: We begin by considering a hyperplane containing $z_0$, so that it can contract or dilate the position of $z_1$, to drive this point to 1, while sending $z_0$ to $(0, 0)$. This maps $z_0$ to 0 and $z_1$ to 1.




• Step 4.2: We now consider two hyperplanes. The first hyperplane passes through $z_0$ and ensures that the existing order of the data along the $x$-axis is preserved (after applying $\sigma$, this order is maintained along the $y$-axis). The second hyperplane is placed between $z_1$ and $z_2$ and is used to push or pull $z_2$ along the $x$-axis, moving it closer to the value 1 on the $x$-axis after applying $\sigma$.




• Step 4.3: We define a hyperplane of equation $w_3 \cdot x = 0$, so it contains $z_0$. The parameter $w_3$ is such that $\sigma(w_3 \cdot z_1) = 1$ and $\sigma(w_3 \cdot z_2) = 2$.




We note that we can again go back to step 4.2, considering the same hyperplane containing $z_0$ and the second one located between $z_2$ and $z_3$. Then, when reproducing step 4.3, we would choose $w_3$ such that $w_3 \cdot z_1 = 1$, $w_3 \cdot z_2 = 2$, and $w_3 \cdot z_3 = 3$. This is feasible because the first two conditions coincide (given that $z_2 = 2z_1$) (see Step 4 in Section 4 for the explicit construction of $w_3$).

By iterating this procedure and combining steps 4.2 and 4.3, we can bring all data to their respective labels.

**Remark 3.1** (Minimal width deep neural network)**.** *At this point, it becomes evident that at least two hyperplanes are necessary to develop an algorithm for classifying data. Indeed, if we were restricted to using just a single hyperplane, we would be unable to develop a compression process for every data set. Therefore, it is not feasible to classify any $d-$dimensional dataset using a 1-wide neural network. Thus, 2 is the minimal width for a neural network (as defined by (1.2)) capable of addressing any classification problem.*

The computation of the depth of the process described above is detailed in the following section. It shows that for the first stage, 1 layer is necessary; $2N$ layers for the second stage; $2M+1$ for the third one; and, finally, $2M-3$ layers for the fourth stage. In total, the depth of the neural network is $2N+4M-1$ layers. Moreover, it is possible to see that in the first stage (since we project from $\mathbb{R}^d$ to $\mathbb{R}$), the number of neurons is $d$. In the second stage, the number of neurons is $2(2N)$. In the third stage, we alternate between one and two dimensions, and the number of neurons becomes $(M+1)+2M$. For the fourth stage, the number of neurons also alternates and is equal to $(M-3)+2M$. The total number of neurons is then $4N+6M+d-2$.

**Remark 3.2.** *The strategy described above has been implemented in Python for a binary classification; the code is available in github.com/Martinshs. This implementation demonstrates how the explicit parameters provided in the proof of Theorem 1.1 can be used to drive each input to its target level set without any training process.*

## 4. Proof of Theorem 1.1

This section is devoted to presenting the details of the proof of Theorem 1.1. Each stage described in the previous section is developed in a separate subsection. The proofs of the steps 2, 3, and 4 will be done by induction.

4.1. **Preconditioning of the data:** To prove the first step, let us consider the dataset $\{x_i, y_i\}_{i=1}^N \subset \mathbb{R}^d \times [\![0, M-1]\!]$. Lemma 2.1 assures the existence of a vector $w_1 \in \mathbb{R}^d$ satisfying

$$w_1 \cdot x_i \neq w_1 \cdot x_j,$$

for every $i \neq j \in [\![1, N]\!]$. Now, let $b_1 \in \mathbb{R}$ be large enough such that

$$w_1 \cdot x_i + b_1 > 0, \quad \forall i \in [\![1, N]\!],$$

which implies

$$\sigma(w_1 \cdot x_i + b_1) \neq \sigma(w_1 \cdot x_j + b_1), \quad \forall i \neq j \in [\![1, N]\!],$$

and also $\sigma(w_1 \cdot x_i + b_1) > 0$ for all $i \in [\![1, N]\!]$. We denote by $\{x_i^1\}_{i=1}^N \subset \mathbb{R}$ the projected one-dimensional new data given by

$$x_i^1 = \sigma(w_1 \cdot x_i + b_1), \quad i \in [\![1, N]\!]. \tag{4.1}$$

Note that the points $\{x_i\}_{i=1}^N$ are collocated according to their distance to the hyperplane $w_1 \cdot x + b_1 = 0$. In other words, for $L_0 = 1$ and with the choice of the parameters $\mathcal{W} = \{w_1\}$, and $\mathcal{B} = \{b_1\}$ we have that

$$\phi^{L_0}(x_i) = \phi(\mathcal{W}^1, \mathcal{B}^1, x_i) = x_i^1. \tag{4.2}$$

This is illustrated in Figure 10.

4.2. **Compression process.** We divide this section into two parts. In the first part, we show that an induction procedure suffices. The second part focuses on showing that a single class can be compressed.

4.2.1. *The induction strategy.* Consider the sets corresponding to the different classes of points:

$$\mathcal{C}_k = \{x_i \text{ with } i \in [\![1, N]\!] : y_i = k\}, \quad \text{and} \quad \mathcal{C} = \bigcup_{k=0}^{M-1} \mathcal{C}_k. \tag{4.3}$$

In the sequel we write $\phi(\mathcal{W}^L, \mathcal{B}^L, \mathcal{C}_k) = z_k$ when $\phi(\mathcal{W}^L, \mathcal{B}^L, x) = z$ for every $x \in \mathcal{C}_k$, Therefore, compressing all the classes is equivalent to proving the existence of parameters $\mathcal{W}^{\tilde{L}}$ and $\mathcal{B}^{\tilde{L}}$, $\tilde{L} > 0$, and a sequence of different vectors $\{z_k\}_{k=0}^{M-1} \subset \mathbb{R}^2$ such that

$$\phi(\mathcal{W}^{\hat{L}}, \mathcal{B}^{\hat{L}}, \mathcal{C}_k) = z_k, \qquad \text{for every } k \in [\![0, M-1]\!]. \tag{4.4}$$

The following proposition guarantees that this problem can be handled in an inductive manner.

**Proposition 4.1.** *For any $k \in [\![0, M-1]\!]$ fixed but arbitrary, we assume that there exist $\tilde{z}_0 \in \mathbb{R}^2$, $\tilde{L} \geq 1$, $\mathcal{W}^{\tilde{L}}$ and $\mathcal{B}^{\tilde{L}}$ such that*

$$\phi(\mathcal{W}^{\tilde{L}}, \mathcal{B}^{\tilde{L}}, \mathcal{C}_k) = \tilde{z}_0, \qquad \phi(\mathcal{W}^{\tilde{L}}, \mathcal{B}^{\tilde{L}}, \mathcal{C} \setminus \mathcal{C}_k) \neq \tilde{z}_0, \tag{4.5}$$

*and*

$$\phi(\mathcal{W}^{\tilde{L}}, \mathcal{B}^{\tilde{L}}, \tilde{z}_1) \neq \phi(\mathcal{W}^{\tilde{L}}, \mathcal{B}^{\tilde{L}}, \tilde{z}_2), \quad \textit{for all } \tilde{z}_1, \tilde{z}_2 \in \mathcal{C} \setminus \mathcal{C}_k, \ \tilde{z}_1 \neq \tilde{z}_2. \tag{4.6}$$

*Then, there exist $L_1 \geq 1$, $\mathcal{W}^{L_1}$, $\mathcal{B}^{L_1}$ and different vectors $\{z_k\}_{k=0}^{M-1} \subset \mathbb{R}^2$ such that*

$$\phi(\mathcal{W}^{L_1}, \mathcal{B}^{L_1}, \mathcal{C}_k) = z_k,$$

*for $k \in [\![0, M-1]\!]$.*

In other words, Proposition 4.1 shows that to compress all the classes of points, it is sufficient to compress a single (but arbitrary) class of points without collapsing the points not belonging to that class.

The proof of Proposition 4.1 can be found in Appendix A.

4.2.2. *Compression of a single class.* Let us take $k \in [\![0, M-1]\!]$ arbitrarily but fixed. Our goal is to drive the class $\mathcal{C}_k$ to some vector $z_k \in \mathbb{R}^2$ in $\tilde{L} \geq 1$ steps. We will do it by induction.

We focus on the worst-case scenario in which points in the class $\mathcal{C}_k$ are isolated, not having neighboring points of the same class, which could be treated simultaneously as a single point, reducing the number of layers needed.

In this procedure, we will combine two operations, in an alternating manner:

• Data structuring: Construction of hyperplanes driving the data set to some particular structure.

• Compression process: Using the structure established in the prior step, introduce hyperplanes to collapse points belonging to the same class.

**(1) Initial Step:** We show that the two first points of the class $\mathcal{C}_k$ closer to zero can be compressed.

*Data structuring:* Given that data have been projected into the one-dimensional real line, without loss of generality, we can assume that the new data $\{x_i^1\}_{i=1}^N$, defined in (4.1), are indexed according to their order, i.e., $x_i^1 \leq x_j^1$ for every $i \leq j$. Let $\mathcal{C}_k^1$ be given by

$$\mathcal{C}_k^1 = \{\sigma(w_1 \cdot x + b_1) \in \mathbb{R} \,:\, x \in \mathcal{C}_k\},$$

where $w_1$ and $b_1$ are the parameters defined in the *Preconditioning of the data* step. Let us denote by $x_{r_1}^1$ the smallest element of the class $\mathcal{C}_k^1$. Then, we introduce the parameters $W_2 = (w_2^1, w_2^2)^\top$ and $b_2 = (b_2^1, b_2^2)^\top$ with

$$w_2^1 = 1, \quad w_2^2 = -1, \quad b_2^1 = -\left(\frac{x_{r_1+2}^1 + x_{r_1+1}^1}{2}\right), \quad \text{and} \quad b_2^2 = \frac{x_{r_1+1}^1 + x_{r_1}^1}{2}.$$

Data are then mapped into the 2-dimensional vectors (see Figure 12)

$$x_i^2 = \boldsymbol{\sigma}(W_2 x_i^1 + b_2) = \begin{pmatrix} \sigma(w_2^1 x_i^1 + b_2^1) \\ \sigma(w_2^2 x_i^1 + b_2^2) \end{pmatrix},$$

such that

$$\begin{cases} x_i^2 = (0, a_i^2) \quad \text{for all } i \in [\![1, r_1]\!], \\ x_{r_1+1}^2 = (0, 0), \\ x_i^2 = (a_i^2, 0) \quad \text{for all } i \in [\![r_1+2, N]\!], \end{cases} \tag{4.7}$$

for some $\{a_i^2\}_{i=1}^N \subset \mathbb{R}_+$. Denote by $H_2^1$ and $H_2^2$ the vertical hyperplanes defined by the parameters $(w_2^1, b_2^1)$ and $(w_2^2, b_2^2)$, respectively.

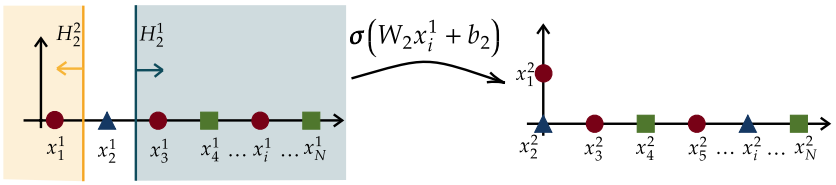



FIGURE 12. In these figures, $\mathcal{C}_k$ corresponds to the class of red circles. The vertical hyperplane $H_2^2$ separates $x_1^1$ and $x_2^1$, and the hyperplane $H_2^1$ is placed between $x_2^1$ and $x_3^1$. The ReLU vector-valued function maps the points to the left of $H_2^2$ to the $y-$axis and those to the right of $H_2^1$ to the $x-$axis. The point between the two planes is mapped to the origin.

*Compression:* Let $x_{r_1}^2 = \boldsymbol{\sigma}(W_2 x_{r_1}^1 + b_2)$ with $x_{r_1}^1$ defined in the previous step, and

$$\mathcal{C}_k^2 = \left\{\boldsymbol{\sigma}(W_2 x + b_2) \in \mathbb{R} \,:\, x \in \mathcal{C}_k^1\right\}.$$

Denote by $x_{r_2}^2 \in \mathcal{C}_k^2$ the closest element to the null vector on the $x-$axis in the class $\mathcal{C}_k^2$. Then, define $W_3 = (w_3^1, w_3^2)^\top$ and $b_3 = (b_3^1, b_3^2)^\top$ with

$$w_3^1 = \left(\frac{a_{r_1-1}^2 + a_{r_1}^2}{a_{r_2+1}^2 + a_{r_2}^2}, 1\right), \quad w_3^2 = \left(-\frac{a_{r_1}^2 + a_{r_1+1}^2}{a_{r_2}^2 + a_{r_2+1}^2}, -1\right), \tag{4.8}$$

$$b_3^1 = -\left(\frac{a_{r_1-1}^2 + a_{r_1}^2}{2}\right), \quad b_3^2 = \frac{a_{r_1}^2 + a_{r_1+1}^2}{2} \tag{4.9}$$

and set $x_i^3 = \boldsymbol{\sigma}(W_3 x_i^2 + b_3)$, which are of the form

$$\begin{cases} x_i^3 = (0, a_i^3) & \text{for all } i \in [\![r_1 + 1, r_2 - 1]\!], \\ x_{r_{1,2}}^3 := x_{r_1}^3 = x_{r_2}^3 = (0,0), \\ x_i^3 = (a_i^3, 0) & \text{for all } i \in [\![1, r_1 - 1]\!] \cup [\![r_2 + 1, N]\!] \end{cases} \tag{4.10}$$

for some $\{a_i^3\}_{i=1}^N \subset \mathbb{R}_+$. The hyperplanes $H_3^1$ and $H_3^2$ are defined by the parameters $(w_3^1, b_3^1)$ and $(w_3^2, b_3^2)$, respectively. The argument above is illustrated in Figure 13.

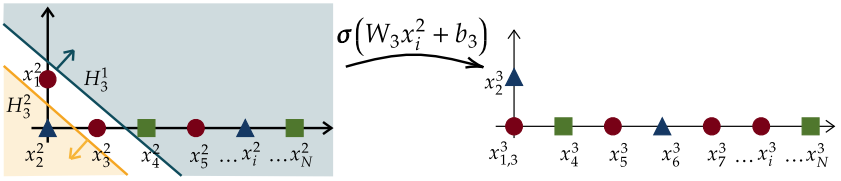



FIGURE 13. The points above the hyperplane $H_3^1$ are mapped to the $x-$axis, and the points below the hyperplane $H_3^2$ are mapped to the $y-$axis, while the points between the two hyperplanes are compressed to the null vector. We denote by $x_{1,3}^3 = x_1^3 = x_3^3$ the null vector to which the two red circles collapse.

**(2) Inductive Step:** The initial step has been achieved. We now aim to show that induction can also be applied successfully. In fact, to do that, it suffices to apply the arguments of the initial step again and again. Note that the model under consideration collapses point to the exact same location, and once this happens, they will never split again in the forthcoming iterations. In this way, if $\alpha_k$ denotes the number of elements in $\mathcal{C}_k$ i.e. $|\mathcal{C}_k| = \alpha_k$, all points in $\mathcal{C}_k$ will collapse applying $2\alpha_k$ times the nonlinear mapping $\sigma$, so that $L_k = 2\alpha_k$.

Let us assume that we have compressed the first $j \in [\![1, \alpha_k]\!]$ elements of the class $\mathcal{C}_k$. Note that to compress $j$ elements, it is necessary to apply two steps per element (*data structuring* and *compression* steps). Therefore, it is necessary to apply $2j$ steps. Denote by $\mathcal{C}_k^{2j+1}$ the class $\mathcal{C}_k^1$ after having applied $2j$ steps to it. Let us show that we can compress the $j+1$-th element of $\mathcal{C}_k^{2j+1}$.

*Data structuring:* Denote by $x_{r_t}^{2j+1}$ the elements of $\mathcal{C}_k^{2j+1}$ for $t \in [\![1, \alpha_k]\!]$. Observe that, after compressing the first $j$ elements of $\mathcal{C}_k^{2j+1}$, we will always have that $x_{r_t}^{2j+1} = (0,0)$ for $t \in [\![1, j]\!]$ and $x_{r_{j+1}}^{2j+1} = (a_{j+1}^{2j+1}, 0)$ for some $a_{j+1}^{2j+1} \in \mathbb{R}$. Since we assumed from the beginning that they are not neighboring points of the same class, let $x_s^{2j+1} = (a_s^{2j+1}, 0) \notin \mathcal{C}_k^{2j+1}$ be the nonzero closest point to $x_{r_1}^{2j} = (0,0)$ in the $x-$axis. We also consider $x_{s+1}^{2j+1} = (a_{s+1}^{2j+1}, 0)$ with $a_{s+1}^{2j+1} \neq 0$, the point to the right of $x_s^{2j+1}$ on the $x$-axis (if it does not exist, we take $x_{s+1}^{2j+1} := x_s^{2j+1} + (0.5, 0)$).

Let us consider the parameters

$$w_{2j+2}^1 = (0,1), \quad w_{2j+2}^2 = \left(\frac{1}{2}, -\frac{1}{2}\right), \quad b_{2j+2}^1 = -\left(\frac{a_{s+1}^{2j+1} + a_s^{2j+1}}{2}\right), \quad b_{2j+2}^2 = \frac{a_s^{2j+1}}{4}. \tag{4.11}$$

Define $W_{2j+2} = (w_{2j+2}^1, w_{2j+2}^2)^\top$ and $b_{2j+2} = (b_{2j+2}^1, b_{2j+2}^2)^\top$ and set

$$x_i^{2j+2} = \boldsymbol{\sigma}(W_{2j+2} x_i^{2j+1} + b_{2j+2}),$$

so that

$$\begin{cases} x_{r_t}^{2j+2} = (0, a_j^{2j+2}) & \text{for all } t \in [\![1, j]\!], \\ x_s^{2j+2} = (0,0), \\ x_{r_{j+1}}^{2j+2} = (a_{j+1}^{2j+2}, 0). \end{cases}$$

*Compression:* We are going to compress $x_{r_j}^{2j+2}$ with $x_{r_{j+1}}^{2j+2}$. Let $x_u^{2j+2} = (0, a_u^{2j+2})$ and $x_d^{2j+2} = (0, a_d^{2j+2})$ be the points lying above and below $x_{r_j}^{2j+2}$ on the $y-$axis, respectively. Also, denote by $x_{rr}^{2j+2} = (a_{rr}^{2j+2}, 0)$

and $x_{ll}^{2j+2} = (a_{ll}^{2j+2}, 0)$ the points that are to the right and left of $x_{r_{j+1}}^{2j+2}$ in the $x-$axis. Thus, we consider the parameters

$$w_{2j+3}^1 = \left(\frac{a_u^{2j+2} + a_j^{2j+2}}{a_{rr}^{2j+2} + a_{j+1}^{2j+2}}, 1\right), \quad w_{2j+3}^2 = \left(-\frac{a_d^{2j+2} + a_j^{2j+2}}{a_{ll}^{2j+2} + a_{j+1}^{2j+2}}, -1\right),$$
$$b_{2j+3}^1 = -\left(\frac{a_u^{2j+2} + a_j^{2j+2}}{2}\right), \quad b_{2j+3}^2 = \frac{a_d^{2j+2} + a_j^{2j+2}}{2}. \tag{4.12}$$

Define $W_{2j+3} = (w_{2j+3}^1, w_{2j+3}^2)^\top$ and $b_{2j+2} = (b_{2j+3}^1, b_{2j+3}^2)^\top$. Therefore, we have

$$x_i^{2j+3} = \boldsymbol{\sigma}(W_{2j+3} x_i^{2j+2} + b_{2j+3}).$$

This selection of parameters allows us to ensure that $x_{r_t}^{2j+3} = (0,0)$ for all $t \in [\![1, j+1]\!]$. This concludes the induction argument.

As we already observed, we need $\hat{L}_k = 2\alpha_k$ layers to compress all the elements of $\mathcal{C}_k$. Consequently, we have shown that for any arbitrary $k \in [\![0, M-1]\!]$ there exist $z_k \in \mathbb{R}^2$, a depth $\hat{L}_k \geq 1$, and parameters $\mathcal{W}^{\hat{L}_k}$ and $\mathcal{B}^{\hat{L}_k}$ such that (4.5) and (4.6) hold. Furthermore, we can explicitly construct the parameters by following (4.11) and (4.12).

As a consequence of the Proposition 4.1, there exist $L_1 \geq 1$, parameters $\mathcal{W}^{L_1}$, and $\mathcal{B}^{L_1}$, and a sequence of different points $\{z_k\}_{k=0}^{M-1} \subset \mathbb{R}^2$ such that

$$\phi(\mathcal{W}^{L_1}, \mathcal{B}^{L_1}, \mathcal{C}_k) = z_k, \qquad \text{for all } k \in [\![0, M-1]\!]. \tag{4.13}$$

Therefore, to compress all classes, the vector-valued $\boldsymbol{\sigma}$ function must be applied

$$\sum_{k=0}^{M-1} 2\alpha_k = 2 \sum_{k=0}^{M-1} |\mathcal{C}_k| = 2N,$$

times. In other words, the depth of the neural network has to be $L_1 = 2N$ to compress all classes.

4.3. **Data sorting.** In the previous step, we have shown that we can reduce our dataset $\{x_i\}_{i=1}^N \subset \mathbb{R}^d$ to a set $\{z_k\}_{k=0}^{M-1} \subset \mathbb{R}^2$, in which each element represents a class or label. Without loss of generality, we assume that each $z_k$ is associated with a label $k$.

In this section, we aim to find $L_2 > 0$ and parameters $\mathcal{W}^{L_2}$, $\mathcal{B}^{L_2}$ such that for a strictly increasing sequence $\{\xi_k\}_{k=0}^{M-1} \subset \mathbb{R}$, we have

$$\phi(\mathcal{W}^{L_2}, \mathcal{B}^{L_2}, z_k) = \xi_k, \quad \text{for all } k \in [\![0, M-1]\!]. \tag{4.14}$$

Let $\{\beta_\eta\}_{\eta=0}^{M-1}$ be a sequence of positive numbers and $\{\hat{\xi}_k\}_{k=0}^{M-1} \subset \mathbb{R}$ a strictly increasing sequence. Note that to prove (4.14), it is sufficient to show that

$$\begin{aligned} \phi(\mathcal{W}^{\beta_\eta}, \mathcal{B}^{\beta_\eta}, z_k) &= \hat{\xi}_k, \qquad \text{for all } k \in [\![0, \eta]\!], \\ \phi(\mathcal{W}^{\beta_\eta}, \mathcal{B}^{\beta_\eta}, z_{\eta+1}) &= \hat{\xi}_{M-1}, \end{aligned} \tag{4.15}$$

for every $\eta \in [\![0, M-2]\!]$. This is equivalent to asserting that we can order the first $\eta$ points, and place $z_{\eta+1}$ as the farthest point from zero. Clearly, when $\eta = M-2$ we recover (4.14).

We will prove (4.15) by induction on $\eta$, applying a data structuring process, similar to the *"compression of a single class"* step, and a projection process in which Lemma 2.1 will be consistently utilized.

**(1) Initial Step:** Our goal is to prove that (4.15) is fulfilled for $\eta = 0$. We proceed in several steps.

*Projection:* We start by projecting the data to the one-dimensional line. By Lemma 2.1, there exist $w_1$ and $b_1$ such that

$$z_k^1 = \sigma(w_1 \cdot z_k + b_1) \in \mathbb{R},$$

satisfies $z_i^1 \neq z_j^1$ for all $i \neq j \in [\![0, M-1]\!]$. In the following, when $z_k^l \in \mathbb{R}$ for some $l \geq 1$, we add an extra sub-index $j_k$ in $z_{k,j_k}^l$ to denote the actual position with respect to the other elements counting from left to right (see Figure 14). Clearly, depending on which point $k$ we consider, its position $(j_k)$ can be different. However, we will only make the dependence of $j_k$ on $k$ explicit when necessary.

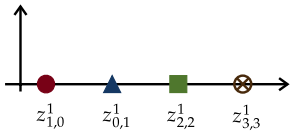



FIGURE 14. Sequence $\{z^1_{k,j}\}_{k=0}^{M-1} \subset \mathbb{R}$. The index $k$ indicates the class to which $z^1_{k,j}$ belongs, while the index $j$ indicates its position from left to right.

*Data structuring:* Let us define

$$w_2^1 = 1, \quad w_2^2 = -1, \quad b_2^1 = -\left(\frac{z^1_{0,j+1} + z^1_{0,j}}{2}\right), \quad \text{and} \quad b_2^2 = \frac{z^1_{0,j} + z^1_{0,j-1}}{2},$$

and denote $W_2 = (w_2^1, w_2^2)^\top$ and $b_2 = (b_2^1, b_2^2)^\top$. Then, we define $z_k^2 = \boldsymbol{\sigma}(W_2 z_k^1 + b_2)$. With the above parameters, we ensure that $z_0^1 = (0,0)$. See Figure 15.

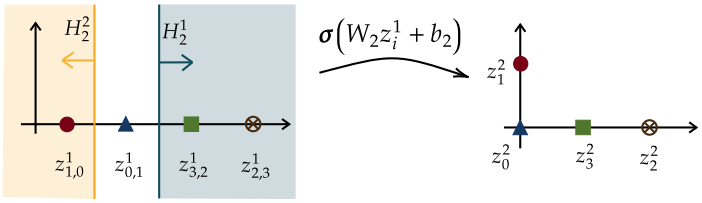



FIGURE 15. The hyperplanes $H_2^1$ and $H_2^2$ are defined by the equations $w_2^1 \cdot x + b_2^1 = 0$ and $w_2^2 \cdot x + b_2^2 = 0$, respectively. This step is similar to the first one in the compression process (see Figure 12)

*Projection:* Due Lemma 2.1 there exist $w_3 \in \mathbb{R}^2$ and $b_3 \in \mathbb{R}$ such that

$$0 \leq w_3 \cdot z_0^2 + b_3 < w_3 \cdot z_k^2 + b_3, \quad \text{for all } k \in [\![1, M-1]\!].$$

Define $z^3_{k,j} = \sigma(w_3 \cdot z_k^2 + b_3)$. By construction, $z_0$ is the closest point to the hyperplane $w_3 \cdot z + b_3 = 0$, so it will be the closest point to zero after the projection step. Consequently, $j_0 = 0$, and the first point has been sorted. It remains to prove that the second point can be moved to the last position.

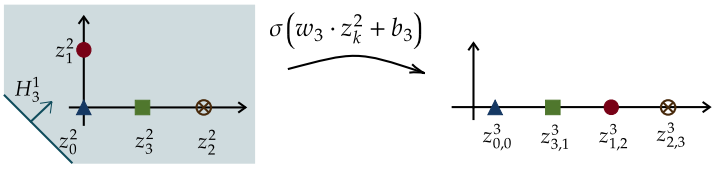



FIGURE 16. The hyperplane $H_3^1$ is defined by the equation $w_3^1 \cdot x + b_3 = 0$ so that $z_0^3$ is the first point from left to right.

*Data structuring:* Let us define

$$w_4^1 = 1, \quad w_4^2 = -1, \quad b_4^1 = -\left(\frac{z^3_{1,j+1} + z^3_{1,j}}{2}\right), \quad \text{and} \quad b_4^2 = \frac{z^3_{1,j} + z^3_{1,j-1}}{2},$$

and denote $W_4 = (w_4^1, w_4^2)^\top$ and $b_4 = (b_4^1, b_4^2)^\top$. Then, define $z_k^4 = \boldsymbol{\sigma}(W_4 z_k^3 + b_4)$. With the above parameters, $z_1^4 = (0,0)$, while $z_0^4$ is the farthest point from the origin on the $y-$axis (see Figure 17).

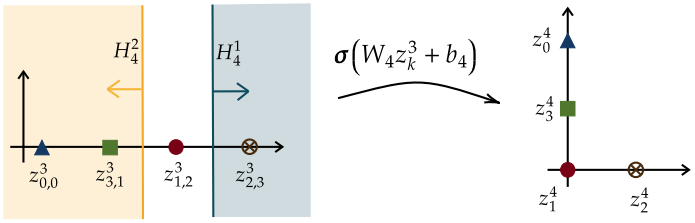



FIGURE 17. The hyperplanes $H_4^1$ and $H_4^2$ are defined by the equations $w_4^1 \cdot x + b_4^1 = 0$ and $w_4^2 \cdot x + b_4^2 = 0$, respectively.

*Projection:* Let us consider a vector $w_5 \in \mathbb{R}^2$ and $b_5 \in \mathbb{R}$ such that

$$0 \leq w_5 \cdot z_0^4 + b_5 < w_5 \cdot z_k^4 + b_5, \qquad \text{for all } k \in [\![1, M-1]\!], \tag{4.16}$$

and also

$$w_5 \cdot z_k^4 + b_5 < w_5 \cdot z_1^4 + b_5, \qquad \text{for all } k \in [\![2, M-1]\!]. \tag{4.17}$$

With these parameter values: $z_{k,j}^5 = \sigma(w_5 \cdot z_k^4 + b_5)$. By construction, we have that (4.15) is satisfied when $\eta = 0$, concluding the initial step.

**Remark 4.1.** *The following aspects should be highlighted.*

- *As observed in Figure 18 the hyperplane $H_5^1$ considered satisfies the conditions* (4.16) *and* (4.17)*. They are satisfied whenever $\theta \in (0, \theta^*)$, where $\theta$ is the angle between the $x-$axis and the hyperplane $H_5^1$, and $\theta^*$ is the angle between the $x-$axis and hyperplane containing $z_0^4$ and the farthest point of $\{z_k^4\}_{k=1}^{M-1}$ on the $x-$axis.*
- *Note that, in the initial step of the proof, we iterate five times to obtain* (4.15) *with $\eta = 0$. But given the configuration $\{z_k^5\}_k$, only two extra steps (specifically, the last data structuring and projection steps) are necessary to satisfy* (4.15) *with $\eta = 1$. Therefore, to sort the first $\eta \in [\![1, M-3]\!]$ classes, one requires $5 + 2\eta$ iterations.*
- *By construction, the point to be sorted is always the one that is placed the furthest from the origin. Therefore, it is enough to sort the first $M-1$ points, and the point $M$ will automatically be sorted.*

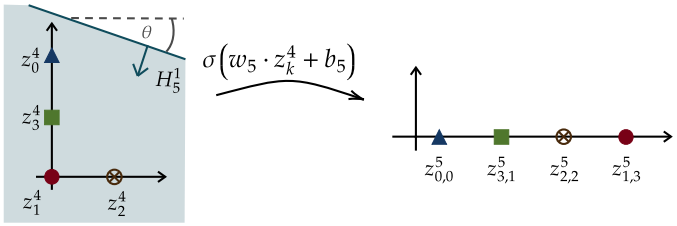



FIGURE 18. The hyperplane $H_5^1$ is defined by $w_5^1 \cdot x + b_5^1 = 0$, so that $j_0 = 0$ and $j_1 = 3$.

**(2) Inductive Step:** Let $\eta \in [\![1, M-2]\!]$ and $\ell := 5 + 2(\eta + 1)$. Assume that we have sorted the first $\eta$ points of $\{z_k^\ell\}_{k=0}^{M-1}$, that is $j_k = k$ for $k \in [\![0, \eta]\!]$. We will show that we can sort one extra element.

By construction, we can assume that the element that we have to sort is the farthest one in the $x-$axis, i.e., $j_{\eta+1} = M - 1$.

*Data structuring:* Let $k_1, k_2 \in [\![\eta, M-1]\!]$ be such that $z_{k_1, j_{\eta+2}-1}^\ell$ and $z_{k_2, j_{\eta+2}+1}^\ell$ are the left and right neighborhood points of $z_{\eta+2, j_{\eta+2}}^\ell$, respectively (if $z_{k_2, j_{\eta+2}+1}^\ell$ does not exist, it is enough to take $z_{k_2, j_{\eta+2}+1}^\ell := z_{k_1, j_{\eta+2}}^\ell + 1$). Thus, consider the parameters

$$w_{\ell+1}^1 = 1, \quad w_{\ell+1}^2 = -1, \quad b_{\ell+1}^1 = -\left(\frac{z_{k_2, j_{\eta+2}+1}^\ell + z_{\eta+2, j_{\eta+2}}^\ell}{2}\right),$$

$$\text{and} \quad b_{\ell+1}^2 = \frac{z_{\eta+2, j_{\eta+2}}^\ell + z_{k_1, j_{\eta+2}-1}^\ell}{2}.$$

Define $W_{\ell+1} = (w^1_{\ell+1}, w^2_{\ell+1})^\top$ and $b_{\ell+1} = (b^1_{\ell+1}, b^2_{\ell+1})^\top$ and set

$$z^{\ell+1}_k = \boldsymbol{\sigma}(W_{\ell+1} z^\ell_k + b_{\ell+1}).$$

We obtain that $z^{\ell+1}_{\eta+2} = (0,0)$ and $z^{\ell+1}_{\eta+1} = (a,0)$, for some $a > 0$, is the farthest point in the $x-$axis. Moreover, for a decreasing sequence of positive numbers $\{a_k\}_{k=0}^{M-1}$, we deduce

$$z^{\ell+1}_k = (0, a_k), \quad \text{for all } k \in [\![0, k_1]\!]. \tag{4.18}$$

*Projection:* Let $w_{\ell+2} \in \mathbb{R}^2$ and $b_{\ell+2} \in \mathbb{R}$ be such that

$$0 \le w_{\ell+2} \cdot z^{\ell+1}_k + b_{\ell+2} < w_{\ell+2} \cdot z^{\ell+1}_{k+1} + b_{\ell+2} \quad \text{for all } k \in [\![0, \eta]\!], \tag{4.19}$$

$$w_{\ell+2} \cdot z^{\ell+1}_k + b_{\ell+2} < w_{\ell+2} \cdot z^{\ell+1}_{\eta+2} + b_{\ell+2} \quad \text{for all } k \in [\![0, M-1]\!]. \tag{4.20}$$

The assumptions about $W_{\ell+2}$ mentioned above are not restrictive. Condition (4.19) is feasible, as shown by (4.18), where the sequence $z^{\ell+1}_k$ is arranged on the $y-$axis for $k \in [\![0, M-2]\!]$ as a decreasing sequence. Furthermore, (4.20) is achievable by selecting the appropriate slope of the hyperplane defined by the parameters (see Remark 4.1). We then set $z^{\ell+2}_{k,j} = \sigma(w_{\ell+2} \cdot z^{\ell+1}_k + b_{\ell+2})$, and, by construction, we have that $j_k = k$ for $k \in [\![1, \eta+1]\!]$ and $j_{\eta+2} = M-1$, concluding the induction.

Therefore, taking $\eta = M-2$ in (4.15), we can sort all the data. Thus for $L_2 = 5 + 2(M-2) = 1 + 2M$, there exist $\mathcal{W}^{L_2}$, $\mathcal{B}^{L_2}$ and a strictly increasing sequence $\{\xi_k\}_{k=0}^{M-1} \subset \mathbb{R}$ such that (4.14) holds.

4.4. **Mapping to the respective labels.** We start from the output of the previous step, where we have shown that for $L_2 = 1 + 2M$ there exist parameters $\mathcal{W}^{L_2}$and $\mathcal{B}^{L_2}$ such that for a strictly increasing sequence $\{\xi_k\}_{k=0}^{M-1} \subset \mathbb{R}$, we have

$$\phi^{L_2}(x_i) = \phi(\mathcal{W}^{L_2}, \mathcal{B}^{L_2}, z_k) = \xi_k, \quad \text{for all } k \in [\![0, M-1]\!]. \tag{4.21}$$

Our goal in this step is to prove, again by induction, that there exist $L_3 > 0$, and parameters $\mathcal{W}^{L_3}$ and $\mathcal{B}^{L_3}$ such that

$$\phi^{L_3}(x_i) = \phi(\mathcal{W}^{L_3}, \mathcal{B}^{L_3}, \xi_k) = k, \quad \text{for all } k \in [\![0, M-1]\!]. \tag{4.22}$$

**(1) Initial Step:** We begin by sorting the first three elements.

*Data projection.* Consider the parameters

$$w_1 = \frac{1}{\xi_1 - \xi_0}, \text{ and } b_1 = \frac{-\xi_0}{\xi_1 - \xi_0},$$

and $\xi^1_k = \sigma(w_1 \cdot \xi_k + b_1)$. We have $\xi^1_0 = 0$ and $\xi^1_1 = 1$.

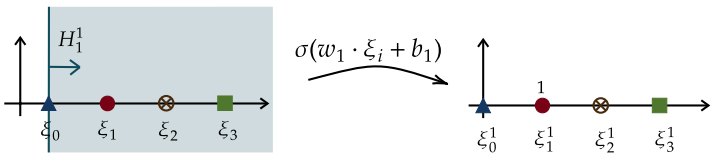



FIGURE 19. The hyperplane $H^1_1$ is defined by the equation $w_1 \cdot x + b_1 = 0$.

*Data structuring.* With the parameters

$$w^1_2 = \frac{2}{\xi^1_2 - \xi^1_1}, \quad w^2_2 = 1, \quad b^1_2 = -\left(\frac{\xi^1_2 + \xi^1_1}{\xi^1_2 - \xi^1_1}\right), \quad \text{and} \quad b^2_2 = 0$$

define $w_2 = (w^1_2, w^2_2)^\top$ and $b_2 = (b^1_2, b^2_2)^\top$, and set $\xi^2_k = \boldsymbol{\sigma}(W_2 \xi^1_k + b_2)$. By construction, we deduce

$$\xi^2_0 = (0,0),\ \xi^2_1 = (0,1),\ \xi^2_2 = (1, \xi^1_2),\ \xi^2_k = (a_k, c_k), \quad \text{for all } k \in [\![3, M]\!],$$

where $\{a_k\}_{k=3}^M$, and $\{c_k\}_{k=3}^M$ are two increasing sequences satisfying that $a_3 > 1$ and $c_3 > \xi^1_2$.

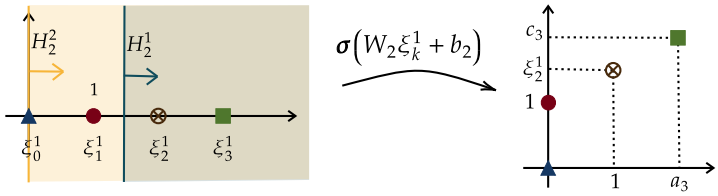



FIGURE 20. The hyperplanes $H_2^1$ and $H_2^2$ are respectively defined by the equations $w_2^1 x + b_2^1 = 0$ and $w_2^2 x + b_2^2 = 0$.

*Data projection.* Let us consider the parameters $w_3 = (2-\xi_2^1, 1)$ and $b_3 = 0$, and define $\xi_k^3 = \sigma(w_3 \cdot \xi_k^2 + b_3)$. Clearly, we have that $\xi_k^3 = k$ for all $k \in [\![0, 2]\!]$.

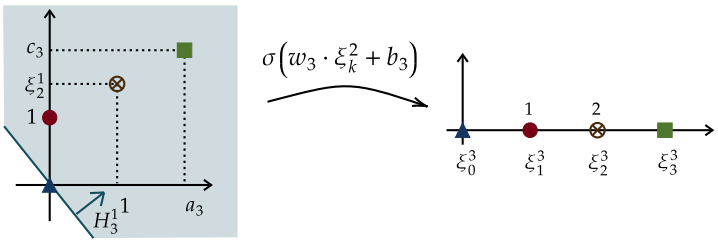



FIGURE 21. The hyperplane $H_3^1$ is defined by the equation $w_3 \cdot x + b_3 = 0$.

**Remark 4.2.** *We have applied $\boldsymbol{\sigma}$ (or $\sigma$) three times to sort the first three points. However, to sort one more point, only two further steps are needed: data structuring and projection. Therefore, to sort $n \geq 2$ points (from the configuration of the previous step), $1 + 2(n-2) = 2n - 3$ applications of $\boldsymbol{\sigma}$ (or $\sigma$) are needed.*

**(2) Inductive Step:** Consider $\eta \in [\![0, M-1]\!]$ and define $l := 1 + 2(\eta - 2)$. Let us assume that $\xi_k^l = k$ for all $k \in [\![0, \eta]\!]$. We will show that there exist parameters such that $\xi_k^{l+2} = k$ for all $k \in [\![0, \eta+1]\!]$. We will proceed in two steps: data structuring and data projection.

*Data structuring.* Let us consider the parameters

$$w_{l+1}^1 = \frac{2}{\xi_{\eta+1}^l - \xi_\eta^l}, \quad w_{l+1}^2 = 1, \quad b_{l+1}^1 = -\left(\frac{\xi_{\eta+1}^l + \xi_\eta^l}{\xi_{\eta+1}^l - \xi_\eta^l}\right), \quad \text{and} \quad b_{l+1}^2 = 0.$$

Define $W_{l+1} = (w_{l+1}^1, w_{l+1}^2)^\top$ and $b_{l+1} = (b_{l+1}^1, b_{l+1}^2)^\top$, and consider

$$\xi_k^{l+1} = \boldsymbol{\sigma}(W_{l+1}\xi_k^l + b_{l+1})$$

so that $\xi_{\eta+1}^{l+1} = (1, \xi_{\eta+1}^l)$ and

$$\begin{aligned} \xi_k^{l+1} &= (0, k), \quad \text{for all } k \in [\![0, \eta]\!], \\ \xi_k^{l+1} &= (a_k, c_k), \quad \text{for all } k \in [\![\eta+2, M-1]\!], \end{aligned}$$

where $\{a_k\}_{k=\eta+2}^{M-1}$ and $\{c_k\}_{k=\eta+2}^{M-1}$ are two sequences of strictly increasing numbers such that $a_k > 1$ and $c_k > \xi_{\eta+1}^l$ for all $k \geq \eta + 2$.

*Data projection.* Finally, define $\xi_k^{l+2} = \sigma(w_{l+2} \cdot \xi_k^{l+1} + b_{l+2})$ with

$$w_{l+2} = ((\eta+1) - \xi_{\eta+1}^l, 1), \quad \text{and} \quad b_{l+2} = 0$$

so that $\xi_k^{l+2} = k$ for all $k \in [\![0, \eta+1]\!]$.

Observe that, in order to drive the $M$ points to their respective labels, we need to apply $L_3 = 2M - 3$ steps.

Summarizing, the input-output map $\phi^L$ of Theorem 1.1, for $N$ points with $M$ classes, is given by the composition of the mappings $\phi_i^{L_i}$ given by (4.2), (4.13), (4.14) and (4.22) respectively, i.e.,

$$\phi^L = (\phi^{L_3} \circ \phi^{L_2} \circ \phi^{L_1} \circ \phi^{L_0}),$$

with $L = L_0 + L_1 + L_2 + L_3 = 1 + 2N + (2M + 1) + (2M - 3) = 2N + 4M - 1$.

**Proof of Corollary 1.1:** Now, we continue with the proof of Corollary 1.1. To this end, let us recall that we have assumed $\{x_i, y_i\} \subset B_{R_x}^d(0) \times B_{R_y}^1(0)$, with $R_x, R_y > 0$. We proceed step by step to estimate the norms.

**Preconditioning of the data:** Denote by $(\mathcal{W}_1, \mathcal{B}_1) = (W_1, b_1)$ the parameters in this step. Recall that $\|w_1\|_2 = 1$ and we take $b_1 = 2R_x$, so that

$$\|W_1\|_F = \|w_1\|_2 = 1, \quad \|b_1\|_2 = 2R_x, \quad \|W_1\|_\infty \leq 1, \quad \|b_1\|_\infty = 2R_x.$$

Hence, we have that

$$|\!|\!|(\mathcal{W}_1, \mathcal{B}_1)|\!|\!|_2^2 = \sum_{j=1}^{1} \big(\|W_j\|_F^2 + \|b_j\|_2^2\big) = 1^2 + (2R_x)^2 = 1 + 4R_x^2, \tag{4.23}$$

and for the $l^\infty-$ norm we get

$$|\!|\!|(\mathcal{W}_1, \mathcal{B}_1)|\!|\!|_\infty = \max_{j\in\{1\}} \{\|W_j\|_\infty, \|b_j\|_\infty\} = \max\{1, 2R_x\} = 2R_x. \tag{4.24}$$

**Compression step:** The total depth of the network is $L_2 = 2N$. For each $j \in \{0, \ldots, N-1\}$, we first analyze the data structuring layer $(W_{2j},\, b_{2j})$:

$$W_{2j} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \text{so that} \quad \|W_{2j}\|_F^2 = 2, \quad \|b_{2j}\|_2^2 \leq 2M_1^2.$$

Now, for the Compression layer $(W_{2j+1},\, b_{2j+1})$:

$$W_{2j+1} = \begin{pmatrix} A_j & 1 \\ -B_j & -1 \end{pmatrix}, \quad \text{with} \quad A_j, B_j \leq 1, \quad \|W_{2j+1}\|_F^2 \leq 4, \quad \|b_{2j+1}\|_2^2 \leq 2M_1^2.$$

Here $M_1 = \max_i\{\|W_1 x_1 + b_1\|_\infty\}$ and therefore $M_1 \leq 3R_x$. To compute the full norm in this step, we sum over $j$ from 0 to $N-1$, obtaining

$$\begin{aligned} |\!|\!|(\mathcal{W}_2, \mathcal{B}_2)|\!|\!|_2^2 &= \sum_{j=0}^{N-1} \big(\|W_{2j}\|_F^2 + \|b_{2j}\|_2^2 + \|W_{2j+1}\|_F^2 + \|b_{2j+1}\|_2^2\big) \\ &= N(2 + 2M_1^2) + N(4 + 2M_1^2) = N(6 + 4M_1^2). \end{aligned} \tag{4.25}$$

For the $\ell^\infty$-norm we have that $\|W_j\|_\infty \leq 1$, and $\|b_j\|_\infty \leq M_1$, thus

$$|\!|\!|(\mathcal{W}_2, \mathcal{B}_2)|\!|\!|_\infty \leq 3R_x. \tag{4.26}$$

**Data sorting:** Let $R_z := \max_k \|z_k\|_\infty$ being $\{z_k\}$ the output from the compression step. The network depth is $L_3 = 2M + 1$. We consider projection layers indexed by $j \in \{0, 2, \ldots, 2M\}$ and data structuring layers indexed by $j \in \{1, 3, \ldots, 2M-1\}$. For each projection layer $W_{2j}$ and $b_{2j}$:

$$\|W_{2j}\|_F^2 = 1, \quad \|b_{2j}\|_2^2 \leq R_z^2.$$

There are $M + 1$ such layers. For each structuring layer $W_{2j+1}$ and $b_{2j+1}$:

$$\|W_{2j+1}\|_F^2 = 2, \quad \|b_{2j+1}\|_2^2 \leq 2R_z^2.$$

There are $M$ such layers. The total norm is

$$\begin{aligned} |\!|\!|(\mathcal{W}_3, \mathcal{B}_3)|\!|\!|_2^2 &= \sum_{j=0}^{M} \big(\|W_{2j}\|_F^2 + \|b_{2j}\|_2^2\big) + \sum_{j=0}^{M-1} \big(\|W_{2j+1}\|_F^2 + \|b_{2j+1}\|_2^2\big) \\ &= (M+1)(1 + R_z^2) + M(2 + 2R_z^2) = (3M+1)(1 + R_z^2). \end{aligned}$$

To estimate $R_z$, let $z_i^{(0)} = x_i^1$ be the output of Step 1, and assume $\|z_i^{(0)}\|_\infty \leq M_1$ for all $i$. Then each iteration in Step 2 satisfies:

$$\|z_i^{(\ell)}\|_\infty \leq \|W_\ell\|_\infty \cdot \|z_i^{(\ell-1)}\|_\infty + \|b_\ell\|_\infty \leq \|z_i^{(\ell-1)}\|_\infty + M_1.$$

By induction over $2N$ layers, we obtain:

$$R_z := \max_i \|z_i^{(2N)}\|_\infty \leq (2N+1)M_1 \leq 2R_x(2N+1).$$

Using the bound for $R_z$ derived above,

$$\|\!|(\mathcal{W}_3, \mathcal{B}_3)|\!\|_2^2 \leq (3M+1)(1+4R_x^2(2N+1)^2). \tag{4.27}$$

For the $\ell^\infty$-norm, since all parameters are bounded by $R_z$, we deduce

$$\|\!|(\mathcal{W}_3, \mathcal{B}_3)|\!\|_\infty \leq 2R_x(2N+1). \tag{4.28}$$

**Mapping to the labels:** We denote by $(\mathcal{W}_4, \mathcal{B}_4)$ the collection of parameters used in the final step of the construction. Recall that the total number of layers in this step is $L_4 = 2M-3$. The first three layers implement the initial projection and positioning of the first three values. All parameters involved in these layers are explicitly defined and uniformly bounded in terms of the geometry of the values $\{\xi_k\}_{k=0}^{M-1}$. Therefore, there exists a constant $C_\xi > 0$, depending only on the relative distance of the data $\{x_i\}_{i=1}^N$, such that

$$\|W_j\|_F^2 + \|b_j\|_2^2 \leq C_\xi(1+R_y^2), \quad \|W_j\|_\infty, \|b_j\|_\infty \leq \max\{1, C_\xi\}, \quad \text{for } j=1,2,3.$$

In the inductive step, we have that for each $\eta \in [\![2, M-2]\!]$, two layers are required to map $\xi_{\eta+1} \mapsto \eta+1$. This is done with the Data structuring layer, for which we have Parameters satisfy again

$$\|W_j\|_F^2 + \|b_j\|_2^2 \leq C_\xi\,(1+R_y^2), \quad \|W_j\|_\infty, \|b_j\|_\infty \leq \max\{1,\, C_\xi\}.$$

Then we apply the Projection layer with weight $w = (\eta + 1 - \xi_{\eta+1}, 1)$, and bias 0, and hence

$$\|W_j\|_F^2 = (\eta+1-\xi_{\eta+1})^2 + 1 \leq (M+R_y)^2, \quad \|b_j\|_2^2 = 0, \quad \|W_j\|_\infty \leq M+R_y, \quad \|b_j\|_\infty = 0.$$

Then, putting all together, we have three initial layers plus $(M-3)$ inductive steps. Therefore, we deduce

$$\begin{aligned}\|\!|(\mathcal{W}_4, \mathcal{B}_4)|\!\|_2^2 &= 3\,C_\xi\,(1+R_y^2) + (M-3)\big[C_\xi\,(1+R_y^2) + (M+R_y)^2\big] \\ &= M\,C_\xi\,(1+R_y^2) + (M-3)\,(M+R_y)^2. \end{aligned}\tag{4.29}$$

Analogously for the $\ell^\infty$ norm, taking the maximum over all layers yields

$$\|\!|(\mathcal{W}_4, \mathcal{B}_4)|\!\|_\infty \leq \max\big\{\max\{1, C_\xi\},\, M+R_y\big\}. \tag{4.30}$$

Finally, combining estimation (4.23)-(4.25)-(4.27)-(4.29), we deduce that

$$\begin{aligned}\|\!|(\mathcal{W}, \mathcal{B})|\!\|_2^2 \leq \big(1+4R_x^2\big) + N\big(6+36R_x^2\big) + (3M+1)\big[\,1+9R_x^2(2N+1)^2\,\big] \\ + \big[M\,C_\xi(1+R_y^2) + (M-3)(M+R_y)^2\big],\end{aligned}$$

and for the $l^\infty-$norm we combine estimations (4.24)-(4.26)-(4.28)-(4.30) to obtain

$$\|\!|(\mathcal{W}, \mathcal{B})|\!\|_\infty = \max\big\{\,2R_x,\; 3R_x,\; 3R_x(2N+1),\; \max\{C_\xi,\, M+R_y\}\big\}.$$

In particular, we can guarantee the existence of a constant $C > 0$ independent of $N$, $M$, $R_x$, $R_y$ such that

$$\|\!|(\mathcal{W}, \mathcal{B})|\!\|_2 \leq C(1 + R_x\sqrt{N} + R_x N\sqrt{M} + R_y M),$$

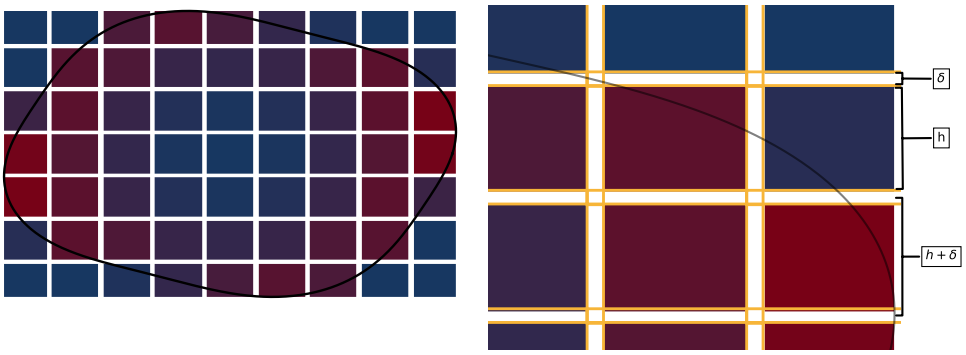
and

$$\|\!|(\mathcal{W}, \mathcal{B})|\!\|_\infty \leq C\big(R_x N + M + R_y\big).$$

## 5. Universal approximation theorem

In this section, we prove the Universal Approximation Theorem in $L^p(\Omega; \mathbb{R}_+)$ (Theorem 1.4).

*Proof.* We proceed according to the *Strategy of the proof* after Theorem 1.4, in Section 1.3.




(A) (B)

FIGURE 22. (A) Illustration of the simple function $f_h$ on $\mathcal{H}$, defined in (5.3). (B) The main features of $G_\delta^h$ are represented: The mesh thickness $\delta$, the size $h > 0$, and the hyperrectangles $\mathcal{H}_i^G$, illustrated with orange boundary.

**Step 1 (Hyperrectangles construction):** Let us consider $\mathcal{C}$, the smallest hyperrectangle containing $\Omega$, oriented according to the axes of the canonical basis of $\mathbb{R}^d$. Consider $0 < h < 1$ and $0 < \delta \ll h$. Define an equispaced grid $G_\delta^h \subset \mathbb{R}^d$, of thickness $\delta$, oriented according to the axes of the canonical basis of $\mathbb{R}^d$. Define the family of hyperrectangles $\mathcal{H} = \{\mathcal{H}_i\}_{i=1}^{N_h}$ as $\mathcal{H} = \mathcal{C} \setminus G_\delta^h$. We also consider $\mathcal{H}^G := \{\mathcal{H}_i^G\}_{i=1}^{N_h^G}$ a family of hyperrectangles such that

$$G_\delta^h = \bigcup_{i=1}^{N_G} \mathcal{H}_i^G,$$

see Figure 7. Note that the family $\mathcal{H}$ depends on $h$, and the family $\mathcal{H}^G$ depends on $h$ and $\delta$; however, this dependency will be omitted to simplify the notation.

The number of hyperrectangles $N_h$ on $\mathcal{H}$ satisfies

$$N_h \leq h^{-d} C_\Omega, \tag{5.1}$$

with $C_\Omega$ a constant depending on $m_d(\Omega)$, where $m_d(\cdot)$ denotes the Lebesgue measure in $\mathbb{R}^d$. Taking into account that the number of edges of a $d$-dimensional hypercube is $2d(d-1)$, the Lebesgue measure $m_d(G_\delta^h)$ of the grid $G_\delta^h$ intersecting $\mathcal{C}$ is bounded by

$$m_d(G_\delta^h) \leq C_{\Omega,d}\delta(h+\delta)^{d-1}h^{-d}. \tag{5.2}$$

Thus, for any $\gamma > 0$, taking $\delta = h^{1+\gamma}$, the volume of $G_\delta^h$ tends to zero as $h \to 0$. In the following, we will take $\gamma = p$.

Let us fix a function $f \in L^p(\Omega; \mathbb{R}_+)$. Extending it by zero, we assume that $f \in L^p(\mathcal{C}; \mathbb{R}_+)$. By the density of simple functions, we know that $f$ can be approximated by a sequence of simple functions. In particular, we can construct a simple function supported on hyperrectangles as follows: Let us consider the constants

$$f_{1,i}^h := \frac{1}{m_d(\mathcal{H}_i)} \int_{\mathcal{H}_i} f(x)\, dx, \quad \text{for } i \in [\![1, N_h]\!],$$

and

$$f_{2,i}^h := \frac{1}{m_d(\mathcal{H}_i^G)} \int_{\mathcal{H}_i^G} f(x)\, dx, \quad \text{for } i \in [\![1, N_G]\!].$$

That is, $f_{1,i}^h$ and $f_{2,i}^h$ are the average value of the function $f$ in the hyperrectangle $\mathcal{H}_i$ and $\mathcal{H}_i^G$, respectively. Then, we introduce the simple function

$$f_h(x) = \sum_{i=1}^{N_h} f_{1,i}^h \chi_{\mathcal{H}_i}(x) + \sum_{i=1}^{N_G} f_{2,i}^h \chi_{\mathcal{H}_i^G}(x), \tag{5.3}$$

where $\chi_{\mathcal{H}_i}$ denotes the characteristic function on the set $\mathcal{H}_i$, similar for $\mathcal{H}_i^G$. In the following, we denote by $M_h > 0$ the number of values that the function $f_h$ takes on the family of hyperrectangles $\mathcal{H}$. Note that $M_h \leq N_h$.

Let us observe that by the Lebesgue differentiation theorem, the sequence $f_h$ approximates $f$ a.e. on $\mathcal{C}$ as $h \to 0$, and therefore, due to the dominated convergence theorem, we have that $\|f - f_h\|_{L^p(\mathcal{C};\mathbb{R}_+)} \to 0$ as $h \to 0$. In particular, for all $\varepsilon > 0$ there exists $h_1 > 0$ small enough such that for every $0 < h < h_1$ we have

$$\|f - f_h\|_{L^p(\mathcal{C};\mathbb{R}_+)} < \varepsilon/2. \tag{5.4}$$

Moreover, as shown in [13, Section 6.2], if $f \in W^{1,p}(\Omega;\mathbb{R}_+)$, there exists a constant $C > 0$ independent of $h > 0$ such that

$$\|f - f_h\|_{L^p(\Omega;\mathbb{R}_+)} \leq C \max\{\operatorname{diam}(\mathcal{H}), \operatorname{diam}(\mathcal{H}^{\mathcal{G}})\}\|f\|_{W^{1,p}(\Omega;\mathbb{R}_+)} \tag{5.5}$$

where $\operatorname{diam}(\mathcal{H}) = \max_{\mathcal{H}_i \in \mathcal{H}}\{\operatorname{diam}(\mathcal{H}_i)\}$. Since $\operatorname{diam}(\mathcal{H}^{\mathcal{G}}) < \operatorname{diam}(\mathcal{H})$, and $\operatorname{diam}(\mathcal{H})$ is at most $\sqrt{d}h > 0$, (5.5) reduces to

$$\|f - f_h\|_{L^p(\Omega;\mathbb{R}_+)} \leq Ch\|f\|_{W^{1,p}(\Omega;\mathbb{R}_+)} \tag{5.6}$$

Thus, estimate (5.4) is ensured by taking

$$h_1 = \frac{\varepsilon}{2C\|f\|_{W^{1,p}(\Omega;\mathbb{R}_+)}}. \tag{5.7}$$

**Step 2 (Approximation of $f_h$ using a neural network):** In this step, we will construct a neural network approximating the simple function $f_h$. This is done by mapping the hyperrectangles of $\mathcal{H}$ into the $M_h$ values of $f_h$ via a neural network.

**Step 2.1 (Compresion of one hyperrectanle)**: In the same spirit as the compression process in Section 4.2.2, we first show that a single $\mathcal{H}_i$ can be compressed without mixing the other hyperrectangles. This allows compressing the whole family $\{\mathcal{H}_i\}_{i=1}^{N_h}$.

This is done in two stages. First, we apply a compression process driving the $d-$dimensional hyperrectangle into a $(d+1)-$dimensional Euclidean space, allowing us to drive a hyperrectangle to a point. In the second stage, we project the data to the $d-$dimensional space, keeping the structure of the hyperrectangle.

**Step 2.1.1 (First layer):** Let us suppose we want to compress a fixed hyperrectangle $\mathcal{H}_*$ that is located on one edge of the hyperrectangle $\mathcal{C}$, and oriented in a canonical direction $e_{\hat{\eta}}$ for some $\hat{\eta} \in [\![1, d]\!]$. We consider the following family of hyperplanes

$$\begin{aligned} H_\eta &:= \{x \in \mathbb{R}^d \,:\, e_\eta x + b_\eta = 0\}, \quad \text{for } \eta \in [\![1, d]\!], \\ H_{d+1} &= \{x \in \mathbb{R}^d \,:\, -e_{\hat{\eta}} x + b^*_{\hat{\eta}} = 0\}, \end{aligned} \tag{5.8}$$

and the respective activation regions (see Section 2.1.1)

$$\begin{aligned} R_\eta &:= \{x \in \mathbb{R}^d \,:\, e_\eta x + b_\eta \geq 0\}, \quad \text{for } \eta \in [\![1, d]\!], \\ R_{d+1} &:= \{x \in \mathbb{R}^d \,:\, -e_{\hat{\eta}} x + b^*_{\hat{\eta}} \geq 0\}. \end{aligned} \tag{5.9}$$

In particular, observe that

$$R_{\hat{\eta}} \cap R_{d+1} = \varnothing. \tag{5.10}$$

Here, we have one hyperplane for each canonical direction and one extra hyperplane in the direction $e_{\hat{\eta}}$, so there are in total $d+1$ hyperplanes. The constants $b_k$ are chosen in such a way that the hyperplanes are placed around the hyperrectangle $\mathcal{H}_*$, that is, $b_k$'s are taken such that

$$\sigma(e_\eta x + b_\eta) = 0, \quad \sigma(-e_{\hat{\eta}} x + b^*_{\hat{\eta}}) = 0, \quad \text{for all } x \in \mathcal{H}_*,\ \eta \in [\![1, d]\!].$$

Let us apply the map defined by the parameters from the hyperplanes (5.8). This defines the new family of hyperrectangles $\{\mathcal{H}_i^1\}_{i=1}^N$. Namely, denote by $W^1 \in \mathbb{R}^{d+1\times d}$ and $b^1 \in \mathbb{R}^{d+1}$ the matrices given by

$$W^1 = \left(e_1|e_2|\dots|e_{\hat{\eta}}|(-1)e_{\hat{\eta}}|\dots|e_d\right)^\top, \qquad b^1 = \left(b_1|b_2|\dots|b_{\hat{\eta}}|b^*_{\hat{\eta}}|\dots|b_d\right)^\top.$$

Then, the new family of hyperrectangles is given as

$$x_1 = \boldsymbol{\sigma}(W^1 x + b^1) \in \mathcal{H}_i^1, \quad \text{for all } x \in \mathcal{H}_i,\ i \in [\![1, N]\!]. \tag{5.11}$$

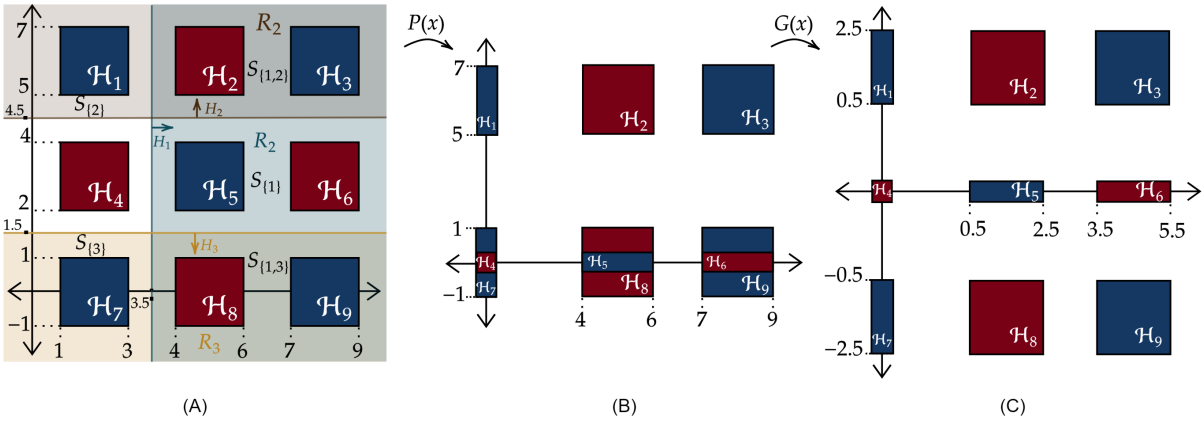



FIGURE 23. (A) We illustrate how the hyperplanes $H_\eta$ (three of them in this 2−dimensional setting) enclose the hyperrectangle $\mathcal{H}_4$. Additionally, this defines the regions $R_\eta$ and the subregions $\mathcal{S}_{I_n}$ introduced in (5.9) and (5.12), respectively. Here, we can observe that the parameters of $H_1$ are $w_1 = (1, 0)$ and $b_1 = -3.5$, for $H_2$ are $w_2 = (0, 1)$ and $b_2 = -4.5$, and for $H_3$, $w_3 = -(0, 1)$ and $b_3 = 1.5$ . (B) We show how the function $P(x)$, defined in (5.14), maps the hyperrectangles. In (B), we keep the same labels for each hyperrectangle as in (A), but it needs to be understood as $P(\mathcal{H}_i)$ for every $i$. The rectangles here, as $\mathcal{H}_1$, illustrate 1−dimensional hyperrectangles (lines). For $\mathcal{H}_1$, we have that $p_1(x) = p_3(x) = 0$ and $p_2(x) = 1$ for all $x \in \mathcal{H}_1$, thus $P(x) = (0, x^{(2)})$. The small square illustrates a 0−dimensional hyperrectangle, i.e., a point. The chosen parameters ensure that $P(\mathcal{H}_4) = \{\mathbf{0}_d\}$. (C) Here, we illustrate the image of $G$, taking as an input $P(\mathcal{H})$. Observe that $G$ translates the hyperrectangles such that there is no overlapping between them, recovering the same structure of (A), but with some hyperrectangles of smaller dimensions. In particular, those that belong to one subregion in (A) in (C) become lines. The function $F$ introduced in (5.14) maps the hyperrectangles in (A) to the hyperrectangles in (C).

An illustration of the family $\{\mathcal{H}_i^1\}_{i=1}^N$ for a 2−dimensional example is given in Figure 25 (Appendix B).

**Step 2.1.2 (Second layer):** The previous selection of parameters can drive all points in $\mathcal{H}_*$ into $\mathcal{H}_*^1 = \{\mathbf{0}_{d+1}\}$, where $\mathbf{0}_{d+1}$ denotes the null vector in $\mathbb{R}^{d+1}$, and since $d + 1$ hyperplanes have been employed, our hyperrectangles are carried into a $d + 1$-dimensional space. To recursively apply this process, we need to project the hyperrectangles into a $d$-dimensional space, ensuring that the previous steps define an injective mapping for the hyperrectangles, i.e., distinct hyperrectangles are carried to distinct locations without mixing them.

For this purpose, let $I_n \subset [\![1, d + 1]\!]$ be a set of indices with $n \geq 1$ elements. We introduce the subregions $\mathcal{S}_{I_n}$ defined as

$$\mathcal{S}_{I_n} = \left\{ x \in \mathbb{R}^d : x \in \bigcap_{\eta \in I_n} R_\eta \text{ and } x \notin \bigcup_{\eta \in I_n^c} R_\eta \right\}, \tag{5.12}$$

where $I_n^c$ is the complement of $I_n$ with respect to $[\![1, d + 1]\!]$. Observe that if $x \in \mathcal{S}_{I_n}$, then $x$ belongs to $n$ regions. Likewise, if $\mathcal{H}_i \subset \mathcal{S}_{I_n}$, $\mathcal{H}_i$, it is in $n$ regions. Due to the construction of the hyperplanes, a hyperrectangle can belong to at most $d$ regions. See Figure 23 (A).

Additionally, if $x$ belongs to $n$ regions, $\boldsymbol{\sigma}(W^1 x + b^1)$ is a vector with $n$ non zero coordinates. This motivates the introduction of the pattern activation function $p_\eta : \mathbb{R}^d \to \mathbb{R}$ defined as

$$p_\eta(x) = \begin{cases} 1 & \text{if } x \in R_\eta, \\ 0 & \text{if } x \notin R_\eta, \end{cases} \qquad \text{for every } \eta \in [\![1, d + 1]\!].$$

Since (5.10), we have that either

$$p_{\hat{\eta}}(x) = 0 \quad \text{or} \quad p_{d+1}(x) = 0, \quad \text{for every } x \in \mathbb{R}^d. \tag{5.13}$$

Furthermore, $p_\eta(x) = 1$ if $x \in \mathcal{S}_{I_n}$ with $\eta \in I_n$, and $p_\eta(x) = 0$ otherwise. Let us introduce $F : \mathbb{R}^d \to \mathbb{R}^d$ defined as

$$F(x) = P(x) + G(x) = \left(\sum_{\eta=1}^{d+1} p_\eta(x) x^{(\eta)} e_\eta\right) + \left(\sum_{\eta=1}^{d+1} p_\eta(x) b_\eta e_\eta\right). \tag{5.14}$$

Figure 23 illustrates how the map $F(x)$ acts on the hyperrectangles $\mathcal{H}$ in a two-dimensional example. The following lemma, proved in Appendix B, gives us important properties of $F$.

**Lemma 5.1.** *We have that $F(\mathcal{H}_*) = \mathbf{0}_d$, where $\mathbf{0}_d$ denotes the null vector in $\mathbb{R}^d$. Moreover, for any hyperrectangles $\mathcal{H}_i, \mathcal{H}_j \subset \mathcal{H}$ with $i \neq j$, we have that $F(\mathcal{H}_i) \cap F(\mathcal{H}_j) = \emptyset$.*

For each $\mathcal{H}_i$, there are two possibilities for $F(\mathcal{H}_i)$: it either retains the shape of a hyperrectangle, in which case $P(\mathcal{H}_i)$ is equal to $\mathcal{H}_i$ and $G(\mathcal{H}_i)$ acts as a translation, or it collapses into a lower-dimensional hyperrectangle, where $P(\mathcal{H}_i)$ projects the hyperrectangle and $G(\mathcal{H}_i)$ translates it. In fact, the dimension of the resulting hyperrectangle depends on the number of regions the original hyperrectangle $\mathcal{H}_i$ is associated with. See Figure 23 for an illustration.

Consider $W^2 := (W^1)^\top \in \mathbb{R}^{d \times d+1}$ as the transpose of $W^1$, and a vector $b^2 \in \mathbb{R}^d$ such that

$$e_k(W^2 x_1 + b^2) > 0, \quad \text{for all } x_1 \in \mathcal{H}_i^1,\ i \in [\![1, N_h]\!],$$

and for all $k \in [\![1, d]\!]$. The family of hyperrectangles $\{\mathcal{H}_i^2\}_{i=1}^N$ is defined by

$$x_2 = \boldsymbol{\sigma}(W^2 x_1 + b^2) \in \mathcal{H}_i^2, \quad \text{for all } x_1 \in \mathcal{H}_i^1,\ i \in [\![1, N_h]\!]. \tag{5.15}$$

Let us note that the hyperrectangles $\{\mathcal{H}_i^2\}_{i=1}^N$ are not mixed, that is, for $\mathcal{H}_i, \mathcal{H}_j \in \mathcal{H}$ with $i \neq j$ we have that $\mathcal{H}_i^2 \cap \mathcal{H}_j^2 = \emptyset$. Furthermore, $\mathcal{H}_*$ is mapped to a single point. To verify the above, let us observe that for $x \in \mathcal{H}$, we have

$$x_2 = \boldsymbol{\sigma}(W^2 \boldsymbol{\sigma}(W^1 x + b^1) + b^2) = W^2 \boldsymbol{\sigma}(W^1 x + b^1) + b^2. \tag{5.16}$$

Let $x^{(i)}$ denote the $i-$th coordinate of $x$. Using the fact that $\boldsymbol{\sigma}(W^1 x^{(1)} + b^1) = p_1(x)(W^1 x^{(1)} + b^1)$, for every $x \in \mathcal{H}_i$ and given (5.13), we obtain

$$x_2 = W^2 \begin{pmatrix} p_1(x)(e_1 x + b_1) \\ p_2(x)(e_2 x + b_2) \\ \vdots \\ p_{\hat{\eta}}(x)(e_{\hat{\eta}} x + b_{\hat{\eta}}) \\ p_{d+1}(x)(-e_{\hat{\eta}} x + b^*_{\hat{\eta}}) \\ \vdots \\ p_d(x)(e_d x + b_d) \end{pmatrix} + b^2 = \begin{pmatrix} p_1(x)(e_1 x + b_1) \\ p_2(x)(e_2 x + b_2) \\ \vdots \\ p_{\hat{\eta}}(x)(e_{\hat{\eta}} x + b_{\hat{\eta}}) + p_{d+1}(x)(e_{\hat{\eta}} x - b^*_{\hat{\eta}}) \\ \vdots \\ p_d(x)(e_d x + b_d) \end{pmatrix} + b^2$$

$$= \left(\sum_{\eta=1}^{d} p_\eta(x) e_\eta x + p_{d+1}(x) e_{\hat{\eta}} x\right) + \left(\sum_{\eta=1}^{d} p_\eta(x) b_\eta e_\eta - p_{d+1}(x) b^*_{\hat{\eta}} e_{\hat{\eta}}\right) + b^2 = F(x) + b^2.$$

The last constant, $b^2$, simply translates all hyperrectangles by the same magnitude. Therefore, due to Lemma 5.1, we conclude that the transformation $F(\mathcal{H}) + b^2$ does not mix the hyperrectangles, preserves their structure, and maps the hyperrectangle $\mathcal{H}_*$ to $\{\mathbf{0}_d\}$.

For the example shown in Figure 23, the hyperrectangles $\{\mathcal{H}_i^2\}_{i=1}^{N_h}$ correspond to those in part (C), but translated to the positive quadrant $\mathbb{R}^2_+$. This translation is carried out by $b^2$, which, in this example, can be chosen as $b^2 = (0.1, 2.6)$.

**Step 2.2 (Compression of all hyperrectangles):** In the previous step, we successfully compressed $\mathcal{H}_*$ into a single point without mixing the other hyperrectangles. However, this process also slightly perturbed the remaining hyperrectangles, transforming them into hyperrectangles of lower dimensions. Specifically, hyperrectangles that were originally $n$-dimensional are now $(n-1)$-dimensional.

Let $E$ represent the set of edges of $\mathcal{C}$. We define $\{E_i\}_{i=1}^d \subset E$ as a set of orthogonal edges. The family of hyperrectangles $\mathcal{H}^E$ is then given by $\mathcal{H}^E = \{\mathcal{H}_i \in \mathcal{H} : \mathcal{H}_i \cap \{E_i\}_{i=1}^d \neq \emptyset\}$, that is, the set of hyperrectangles in $\mathcal{H}$ intersecting at least one edge in $\{E_i\}_{i=1}^d$. We will prove that compressing the hyperrectangles in $\mathcal{H}^E$ into points also compresses all the hyperrectangles in $\mathcal{H}$ into points.

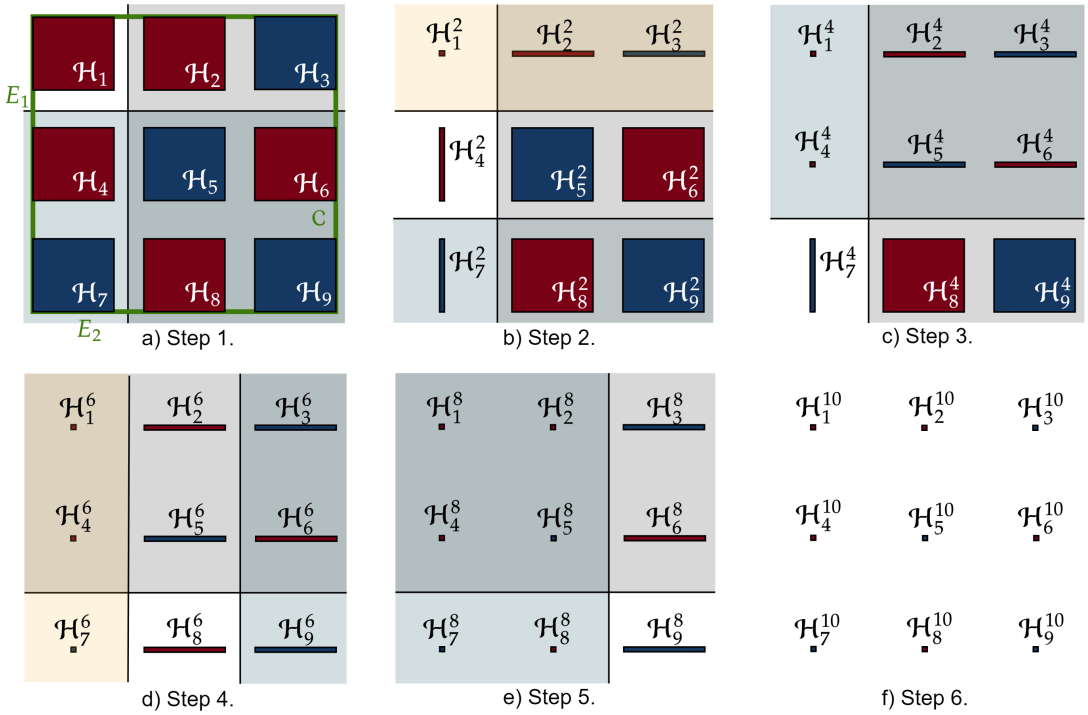



FIGURE 24. In the figure, we show an example of the compression process following the guidelines from Steps 2.1-2.2. First, in Step 1, we select two edges, $E_1$ and $E_2$ of $\mathcal{C}$, oriented according to different canonical vectors. Thus, $\mathcal{H}^E = \{\mathcal{H}_1, \mathcal{H}_4, \mathcal{H}_7, \mathcal{H}_8, \mathcal{H}_9\}$. Then, we compress the first hyperrectangle, $\mathcal{H}_1$, using two hyperplanes (a two-layer neural network with two neurons in the first layer and two neurons in the second layer). Here, $\mathcal{H}_2$ and $\mathcal{H}_3$ belong to $S_{\{1\}}$, and intersects $\mathcal{H}_4$ and $\mathcal{H}_7$ $S_{\{2\}}$. Next, in Step 2, we compress the second hyperrectangle on $E_1$, corresponding to $\mathcal{H}_4^2$. We use three hyperplanes to compress it (a two-layer neural network with three neurons in the first layer and two neurons in the second layer). Here, $\mathcal{H}_5^2$ and $\mathcal{H}_6^2$ belongs to $S_{\{1\}}$. In Step 3, we continue with $\mathcal{H}_7^4$. After compressing all the hyperrectangles along $E_1$, we have reduced every hyperrectangle to $(d-1)$-dimensional objects ($d-1=1$ in this example). By applying the same process to the hyperrectangles along $E_2$ and continuing with Steps 4 and 5, we complete the compression process.

We refer to Figure 24 for a graphical description of what follows.

Let $E_1$ be one of the selected orthogonal edges of $\mathcal{C}$. By Step 2.1, we can compress any hyperrectangle $\mathcal{H}_1 \in \mathcal{H}^E$ intersecting $E_1$ using a two-layer neural network. Moreover, observe that if a $d$-dimensional hyperrectangle $\mathcal{H}_i$ belongs to a subregion $SI_n$ with $n \leq d$, then it collapses into a $(d-1)$-dimensional hyperrectangle. Indeed, the condition $\mathcal{H}_i \subset S_{I_n}$ with $n \leq d$ implies that there exists an index $\bar{\eta} \in [\![1, d+1]\!]$ such that $p_{\bar{\eta}}(x) = 0$ for all $x \in \mathcal{H}_i$. Consequently, the $\bar{\eta}$-th coordinate of every point in $\mathcal{H}_i$ vanishes, collapsing $\mathcal{H}_i$ in to a $(d-1)-$dimensional hyperrectangle.

By applying the previous argument to all hyperrectangles in $\mathcal{H}^E$ intersecting $E_1$, we deduce that each $\mathcal{H}_i \in \mathcal{H}$ eventually enters a region $S_{I_n}$ with $n \leq d-1$. As a result, there exists a coordinate $\bar{\eta} \in [\![1, d+1]\!]$ such that $p_{\bar{\eta}}(x) = 0$ for all $x \in \mathcal{H}_i$, and thus $\mathcal{H}_i$ collapses into a $(d-1)$-dimensional hyperrectangle. Then, we continue with the hyperrectangles that intersect $E_2$. Since $E_2$ is orthogonal to $E_1$, the vanishing coordinate is now $\tilde{\eta} \neq \bar{\eta}$, and each $(d-1)$-dimensional hyperrectangle is further collapsed into a $(d-2)$-dimensional hyperrectangle. Repeating this process for each edge $E_k$ with $k \in \{1, \ldots, d\}$, and noting that each step corresponds to the vanishing of a new and different coordinate, we conclude that every $\mathcal{H}_i \in \mathcal{H}$ is eventually mapped to a 0-dimensional object (a point). This concludes the compression process.

**Remark 5.1.** *If we had not followed the edge-based strategy, we would have needed to compress hyperrectangles located inside $\mathcal{C}$. In the worst case, enclosing a $d$-dimensional hyperrectangle requires $2d$ hyperplanes, implying that the neural network would need to have width $2d$. By restricting the compression to hyperrectangles on the edges of $\mathcal{C}$, we reduce this requirement to $d+1$ hyperplanes per step. Therefore, this strategy allows us to reduce the width of the neural network from $2d$ to $d+1$ neurons per layer.*

**Step 2.3 (Neuronal network construction):** Let $N_E^j$ be the number of hyperrectangles on the edge $E_j$ for every $j \in \{1,\dots,d\}$. After applying Step 2, we have applied a two-layer neural network $N_E := \sum_{j=1}^d N_E^j$ times, and we have constructed a family of parameters $\mathcal{W}^{2N_E}$ and $\mathcal{B}^{2N_E}$ such that

$$\phi^{2N_E}(\mathcal{H}) := \phi(\mathcal{W}^{2N_E}, \mathcal{B}^{2N_E}, \mathcal{H}) = \{x_i\}_{i=1}^{N_h}, \tag{5.17}$$

where $\{x_i\}_{i=1}^{N_h} \subset \mathbb{R}^d$ is a sequence of points. Then we can apply Theorem 1.1 to find the parameters $\mathcal{W}^L$ and $\mathcal{B}^L$ with $L = 2N_h + 4M_h - 1$, and an input-output map $\phi^L$ of (1.2) such that

$$\phi^L(x_i) := \phi(\mathcal{W}^L, \mathcal{B}^L, x_i) = f_i^h, \quad \text{for all } i \in [\![1, N_h]\!]. \tag{5.18}$$

Finally, composing the maps given by (5.17) and (5.18), i.e., $\phi^{\mathcal{L}} = \phi^L \circ \phi^{2N_E}$ with $\mathcal{L} = L + 2N_E$, we have that

$$\phi^{\mathcal{L}}(x) := \phi(\mathcal{W}^{\mathcal{L}}, \mathcal{B}^{\mathcal{L}}, x) = f_h(x), \quad \text{for every } x \in \mathcal{H}, \tag{5.19}$$

where $\mathcal{W}^{\mathcal{L}} = \mathcal{W}^L \cup \mathcal{W}^{2N_h}$ and $\mathcal{B}^{\mathcal{L}} = \mathcal{B}^L \cup \mathcal{B}^{2N_h}$.

**Step 3 (Error estimation):** Let us recall that in **Step 1**, for a given $f \in L^p(\Omega;\mathbb{R}_+)$ and $\varepsilon > 0$, we have chosen $h > 0$ small enough such that (5.4) holds.

To estimate the error between $f_h$ and $\phi^{\mathcal{L}}$, since $\Omega \subset \mathcal{C} = \mathcal{H} \cup G_\delta^h$, we can write

$$\|\phi^{\mathcal{L}}(x) - f_h(x)\|_{L^p(\Omega;\mathbb{R}_+)}^p \le \int_{\mathcal{H}} |\phi^{\mathcal{L}}(x) - f_h(x)|^p \, dx + \int_{G_\delta^h} |\phi^{\mathcal{L}}(x) - f_h(x)|^p \, dx.$$

Owing to (5.19), the first term on the right-hand side vanishes. Thus,

$$\|\phi^{\mathcal{L}}(x) - f_h(x)\|_{L^p(\Omega;\mathbb{R}_+)}^p \le \int_{G_\delta^h} |\phi^{\mathcal{L}}(x) - f_h(x)|^p \, dx \le \int_{G_\delta^h} |f_h(x)|^p \, dx + \|\phi^{\mathcal{L}}\|_{L^\infty(G_\delta^h)}^p m_d(G_\delta^h). \tag{5.20}$$

Let us estimate each term on the right-hand side of (5.20). Since $f_h$ converge to $f$ in $L^p(\Omega;\mathbb{R}_+)$, in particular we have that

$$\|f_h - f\|_{L^p(\mathcal{H}_i^G;\mathbb{R}_+)} = \|f_i^h - f\|_{L^p(\mathcal{H}_i^G;\mathbb{R}_+)} \to 0, \quad \text{as } h \to 0.$$

Due to the triangular inequality, we deduce that

$$|f_i^h| \le C_1(1 + \|f\|_{L^p(\Omega;\mathbb{R}_+)}), \tag{5.21}$$

with $C_1 > 0$ a constant independent of $h$. Then, it follows that

$$\begin{aligned}\int_{G_\delta^h} |f_h(x)|^p \, dx &= \int_{G_\delta^h} \left| \sum_{i=1}^{N_h^G} f_i^h \chi_{\mathcal{H}_i^G}(x) \right|^p dx = \sum_{i=1}^{N_h^G} |f_i^h|^p m_d(\mathcal{H}_i^G) \\ &\le C_1^p(1 + \|f\|_{L^p(\Omega;\mathbb{R}_+)}^p) \sum_{i=1}^{N_h^G} m_d(\mathcal{H}_i^G) = C_1^p(1 + \|f\|_{L^p(\Omega;\mathbb{R}_+)}^p) m_d(G_\delta^h).\end{aligned}$$

Thus, using (5.2) we have

$$\int_{G_\delta^h} |f_h(x)|^p \, dx \le C_2 \delta (h+\delta)^{d-1} h^{-d},$$

with $C_2 = C_1^p C_{\Omega,d}(1 + \|f\|_{L^p(\Omega;\mathbb{R}_+)}^p)$. Then, since we are taking $\delta = h^{1+p}$, we obtain

$$\int_{G_\delta^h} |f_h(x)|^p \, dx \le C_2(1 + \|f\|_{L^p(\Omega;\mathbb{R}_+)}^p) h^p. \tag{5.22}$$

Before continuing, let us consider the following lemma, whose proof can be found in Appendix B.

**Lemma 5.2.** *Let $\phi^{\mathcal{L}}$ be the map defined by (5.19), and denote by $l_{\mathcal{C}}$ the longest edge of $\mathcal{C}$. Then, for $h < l_{\mathcal{C}}\log(2)/(d+1)$ we have*

$$\|\phi^{\mathcal{L}}\|_{L^\infty(\mathcal{C};\mathbb{R}_+)} \leq C_3\left(1+\delta(h+\delta)+h\right),$$

*with $C_3 > 0$ a constant depending on $d$, $m_d(\mathcal{C})$, and $\|f\|_{L^p(\Omega;\mathbb{R}_+)}$.*

Let us assume that $h < l_{\mathcal{C}}\log(2)/(d+1)$. Then, using Lemma 5.2, (5.2), and since $\delta < h^{\gamma+1}$ we deduce

$$\|\phi^{\mathcal{L}}\|_{L^\infty(G_\delta^h;\mathbb{R}_+)} m_d(G_\delta^h) \leq C_3\left(1+\delta(h+\delta)+h\right) C_{\Omega,d}\delta(h+\delta)^{d-1}h^{-d} \leq 4C_3C_{\Omega,d}h^p. \tag{5.23}$$

Finally, putting together (5.22) and (5.23), we deduce that

$$\|\phi^{\mathcal{L}} - f_h\|^p_{L^p(\Omega;\mathbb{R}_+)} \leq C_2h^p + (4C_3C_{\Omega,d}h^p)^p \leq \hat{C}h^p, \tag{5.24}$$

where $\hat{C} = C_2 + (4C_3C_{\Omega,d})^p$ is a positive constant depending on $d$, $p$, $m_d(\mathcal{C})$, and $\|f\|_{L^p(\Omega;\mathbb{R}_+)}$. Thus, taking $h < \varepsilon/(2\hat{C}^{1/p})$, we deduce that

$$\|\phi^{\mathcal{L}} - f_h\|_{L^p(\Omega;\mathbb{R}_+)} < \frac{\varepsilon}{2}. \tag{5.25}$$

Therefore, taking $h < \min\{\varepsilon/(2\hat{C}^{1/p}), h_1\}$, with $h_1$ defined in (5.7), we have that (5.4) and (5.24) hold, and consequently, we conclude that

$$\|f - \phi^{\mathcal{L}}\|_{L^p(\Omega;\mathbb{R}_+)} < \varepsilon.$$

**Step 4 (Depth Estimation):** To estimate the depth of the neural network, we first estimate $\min\{\varepsilon/(2\hat{C}^{1/p}), h_1\}$. Assume that $f \in W^{1,p}(\Omega;\mathbb{R}_+)$. Then, let us observe that in an analogous way to (5.6), we can deduce that

$$|f_i^h| \leq \|f\|_{L^p(\Omega;\mathbb{R}_+)} + h\|f\|_{W^{1,p}(\Omega;\mathbb{R}_+)}. \tag{5.26}$$

Therefore, using (5.26) instead (5.21), we have $C_2 \leq C_1^pC_{\Omega,d}\|f\|^p_{W^{1,p}(\Omega;\mathbb{R}_+)}$. Moreover, due to Lemma 5.2, we have that $C_3 = C\|f\|^p_{W^{1,p}(\Omega;\mathbb{R}_+)}$. Consequently, we deduce that the constant $\hat{C}$ can be taken as

$$\hat{C} \leq (C_1^pC_{\Omega,d} + 4CC_{\Omega,d})\|f\|^p_{W^{1,p}(\Omega;\mathbb{R}_+)} =: \tilde{C}^p\|f\|^p_{W^{1,p}(\Omega;\mathbb{R}_+)}$$

Consequently, using (5.7) we have

$$\begin{aligned}\min\left\{\frac{\varepsilon}{2\hat{C}^{1/p}}, h_1\right\} &= \min\left\{\frac{\varepsilon}{2\hat{C}^{1/p}}, \frac{\varepsilon}{2C\|f\|_{W^{1,p}(\Omega;\mathbb{R}_+)}}\right\} \geq \frac{\varepsilon}{2\|f\|_{W^{1,p}(\Omega;\mathbb{R}_+)}}\min\left\{\frac{1}{\tilde{C}}, \frac{1}{C}\right\}\\ &=: \frac{\varepsilon C_5}{2\|f\|_{W^{1,p}(\Omega;\mathbb{R}_+)}}.\end{aligned}$$

Then, in particular, we take

$$h = \frac{\varepsilon C_5}{2\|f\|_{W^{1,p}(\Omega;\mathbb{R}_+)}}. \tag{5.27}$$

Now, due to Step 2.3, we know that $\mathcal{L} = 2N_h + 4M_h - 1 + 2N_E$. Moreover, observe that $N_E$ and $N$ can be estimated by

$$N_E \leq d\left\lceil\frac{l_{\mathcal{C}}}{h+\delta}\right\rceil, \quad \text{and} \quad N_h \leq d\left\lceil\frac{l_{\mathcal{C}}}{h+\delta}\right\rceil^d. \tag{5.28}$$

with $\delta = h^{1+p}$. Then, using $M_h \leq N_h$ and the estimations (5.28) and (5.27), we deduce the upper bound for the depth

$$\mathcal{L} \leq C_6\left(\|f\|^d_{W^{1,p}(\Omega;\mathbb{R}_+)}\varepsilon^{-d} + \|f\|_{W^{1,p}(\Omega;\mathbb{R}_+)}\varepsilon^{-1} + 1\right),$$

where $C_6$ is a positive constant that depends on $m_d(\mathcal{C})$, $p$, and $d$. This concludes the proof of (1.21). □

## 6. Proof of the training theorems

In this section, we provide the proof of Theorems 1.2 and 1.3 and Corollary 1.4.

*Proof of Theorem 1.2.* By the definition of minimizer, we have

$$\begin{aligned}\mathcal{J}_\lambda(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L) \le \mathcal{J}_\lambda(\mathcal{W}_*^L,\mathcal{B}_*^L) &= \lambda \left|\!\left|\!\left|(\mathcal{W}_*^L,\mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 + \frac{1}{N}\sum_{i=1}^N \text{loss}\left(\phi(\mathcal{W}_*^L,\mathcal{B}_*^L,x_i),y_i\right)\\ &= \lambda \left|\!\left|\!\left|(\mathcal{W}_*^L,\mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 + \mathcal{J}_0(\mathcal{W}_*^L,\mathcal{B}_*^L) = \lambda \left|\!\left|\!\left|(\mathcal{W}_*^L,\mathcal{B}_*^L)\right|\!\right|\!\right|_2^2, \end{aligned} \tag{6.1}$$

where we have used (1.10). Now, by the definition of the functional $\mathcal{J}_\lambda$ and (6.1), we obtain

$$\frac{1}{N}\sum_{i=1}^N \text{loss}\left(\phi(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L,x_i),y_i\right) \le \mathcal{J}_\lambda(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L) \le \lambda \left|\!\left|\!\left|(\mathcal{W}_*^L,\mathcal{B}_*^L)\right|\!\right|\!\right|_2^2. \tag{6.2}$$

Now, due to Corollary 1.1, we know that $\left|\!\left|\!\left|(\mathcal{W}_*^L,\mathcal{B}_*^L)\right|\!\right|\!\right|_2$ can be uniformly bounded with respect to $\lambda$. Therefore, taking $\lambda \to 0$ in (6.2), we conclude (1.12).

Next, by the definition of $\mathcal{J}_\lambda$, we obtain

$$\lambda \left|\!\left|\!\left|(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L)\right|\!\right|\!\right|_2 \le \mathcal{J}_\lambda(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L) \le \lambda \left|\!\left|\!\left|(\mathcal{W}_*^L,\mathcal{B}_*^L)\right|\!\right|\!\right|_2^2. \tag{6.3}$$

Then, dividing (6.3) by $\lambda > 0$, we deduce that the sequence $\{(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L)\}_{\lambda>0}$ is bounded by $\left|\!\left|\!\left|(\mathcal{W}_*^L,\mathcal{B}_*^L)\right|\!\right|\!\right|_2$. Moreover, the Bolzano–Weierstrass Theorem ensures that this sequence has a subsequence that converges to some $(\mathcal{W}_0^L,\mathcal{B}_0^L)$. Denoting such a subsequence again by $\{(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L)\}_{\lambda>0}$, we observe that

$$\mathcal{J}_0(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L) = \mathcal{J}_\lambda(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L) - \lambda \left|\!\left|\!\left|(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L)\right|\!\right|\!\right|_2 \le \mathcal{J}_\lambda(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L). \tag{6.4}$$

Then, combining (6.1) and (6.4), and using the continuity of $\mathcal{J}_0$, we get

$$\mathcal{J}_0(\mathcal{W}_0^L,\mathcal{B}_0^L) = \mathcal{J}_0\left(\lim_{\lambda\to 0}(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L)\right) = \lim_{\lambda\to 0}\mathcal{J}_0(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L) = 0,$$

concluding that $(\mathcal{W}_0^L,\mathcal{B}_0^L)$ is a minimizer of $\mathcal{J}_0$. Now, let us consider

$$(\tilde{\mathcal{W}}_0^L,\tilde{\mathcal{B}}_0^L) \in \text{argmin}\left\{\left|\!\left|\!\left|(\mathcal{W}^L,\mathcal{B}^L)\right|\!\right|\!\right|_2 : \mathcal{J}_0(\mathcal{W}^L,\mathcal{B}^L) = 0\right\},$$

that is, $(\tilde{\mathcal{W}}_0^L,\tilde{\mathcal{B}}_0^L)$ is a parameter of minimal norm minimizing $\mathcal{J}_0$. In particular, since $(\tilde{\mathcal{W}}_0^L,\tilde{\mathcal{B}}_0^L)$ is a minimizer of $\mathcal{J}_0$, (1.13) implies that

$$\left|\!\left|\!\left|(\mathcal{W}_\lambda^L,\mathcal{B}_\lambda^L)\right|\!\right|\!\right|_2 \le \left|\!\left|\!\left|(\tilde{\mathcal{W}}_0^L,\tilde{\mathcal{B}}_0^L)\right|\!\right|\!\right|_2, \quad \text{for all } \lambda > 0. \tag{6.5}$$

Taking $\lambda \to 0$ in (6.5), we obtain

$$\left|\!\left|\!\left|(\mathcal{W}_0^L,\mathcal{B}_0^L)\right|\!\right|\!\right|_2 \le \left|\!\left|\!\left|(\tilde{\mathcal{W}}_0^L,\tilde{\mathcal{B}}_0^L)\right|\!\right|\!\right|_2. \tag{6.6}$$

If (6.6) holds with strict inequality, then we contradict the fact that $(\tilde{\mathcal{W}}_0^L,\tilde{\mathcal{B}}_0^L)$ is a parameter of minimal norm. Therefore, we must have

$$\left|\!\left|\!\left|(\mathcal{W}_0^L,\mathcal{B}_0^L)\right|\!\right|\!\right|_2 = \left|\!\left|\!\left|(\tilde{\mathcal{W}}_0^L,\tilde{\mathcal{B}}_0^L)\right|\!\right|\!\right|_2,$$

that is, $(\mathcal{W}_0^L,\mathcal{B}_0^L)$ is also a parameter of minimal norm that minimize $\mathcal{J}_0$. □

*Proof of Theorem 1.3.* We begin by analyzing the deviation between $x_i^j$ and $\hat{x}_i^j$, the solutions of (1.2) and (1.15) at layer $j$. We have

$$\begin{aligned}\|x_i^j - \hat{x}_i^j\| &\le \|\hat{\boldsymbol{\sigma}}_j((\hat{W}_j\hat{x}_i^{j-1}+\hat{b}_j) - \boldsymbol{\sigma}_j(W_j\hat{x}_i^{j-1}+b_j)\| + \|\boldsymbol{\sigma}_j(W_j\hat{x}_i^{j-1}+b_j) - \boldsymbol{\sigma}_j(W_jx_i^{j-1}+b_j)\|\\ &\le \nu_j + \|W_j\|\|x_i^{j-1}-\hat{x}_i^{j-1}\| \le \nu_j + \left|\!\left|\!\left|(\mathcal{W}_*^L,\mathcal{B}_*^L)\right|\!\right|\!\right|_2 \|x_i^{j-1}-\hat{x}_i^{j-1}\|, \end{aligned} \tag{6.7}$$

where we have used the fact that the radii $R_j$ defined in (1.17) are large enough to control the deviation between the activations at each layer. We also used the Lipschitz continuity of the activation functions on compact sets.

Since $\|x_i^0 - \hat{x}_i^0\| = 0$, an iterative application of (6.7) yields

$$\|\hat{\phi}(\hat{\mathcal{W}}_\lambda^L,\hat{\mathcal{B}}_\lambda^L,x_i) - \phi(\mathcal{W}_*^L,\mathcal{B}_*^L,x_i)\|^2 = \|x_i^L - \hat{x}_i^L\|^2 \le \left(\sum_{j=1}^L \nu_j \left|\!\left|\!\left|(\mathcal{W}_*^L,\mathcal{B}_*^L)\right|\!\right|\!\right|_2^{L-j}\right)^2$$

$$\leq \|\nu\|_2^2 \sum_{j=1}^{L} \left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^{2(L-j)}$$
$$= \|\nu\|_2^2 \left( \frac{\left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^{2L} - 1}{\left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 - 1} \right) =: \mathscr{E}(\nu, \mathcal{W}_*^L, \mathcal{B}_*^L),$$

where we used the Cauchy–Schwarz inequality and the formula for the sum of a geometric series.

Now, consider the following compact set

$$\mathcal{K}_\nu := \left\{(z, y) \in \mathbb{R}^m \times \mathbb{R}^m \,:\, \|z\| \leq \mathscr{E}(\nu, \mathcal{W}_*^L, \mathcal{B}_*^L),\ y \in \{y_i\}_{i=1}^N\right\},$$

and let us define

$$\mathscr{A}_{\text{loss}}(\nu, \mathcal{W}_*^L, \mathcal{B}_*^L) := \max_{(z,y)\in\mathcal{K}_\nu} \text{loss}(z + y, y). \tag{6.8}$$

Due to the continuity and nonnegativity of loss, it follows that $\mathscr{A}_{\text{loss}} < \infty$ and $\mathscr{A}_{\text{loss}} \geq 0$. Moreover, as $\|\nu\|_2 \to 0$, we have $\mathscr{E} \to 0$, and hence the compact set $\mathcal{K}_\nu$ converges to

$$\mathcal{K}_0 = \left\{(z, y) \in \mathbb{R}^m \times \mathbb{R}^m \,:\, \|z\| \leq 0,\ y \in \{y_i\}_{i=1}^N\right\}.$$

Consequently, we deduce that

$$\mathscr{A}_{\text{loss}}(0, \mathcal{W}_*^L, \mathcal{B}_*^L) = \max_{(z,y)\in\mathcal{K}_0} \text{loss}(y, y) = 0,$$

since $\text{loss}(y, y) = 0$. Thus, $\mathscr{A}_{\text{loss}}(\nu, \mathcal{W}_*^L, \mathcal{B}_*^L) \to 0$ as $\|\nu\|_2 \to 0$.

Now, by the definition of minimizer and using (6.8), we obtain

$$\hat{\mathcal{J}}_\lambda(\hat{\mathcal{W}}_\lambda^L, \hat{\mathcal{B}}_\lambda^L) \leq \hat{\mathcal{J}}_\lambda(\mathcal{W}_*^L, \mathcal{B}_*^L) = \lambda \left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 + \frac{1}{N}\sum_{i=1}^N \text{loss}\left(\hat{\phi}(\mathcal{W}_*^L, \mathcal{B}_*^L, x_i), y_i\right)$$
$$= \lambda \left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 + \frac{1}{N}\sum_{i=1}^N \text{loss}\left(\hat{\phi}(\mathcal{W}_*^L, \mathcal{B}_*^L, x_i) - \phi(\mathcal{W}_*^L, \mathcal{B}_*^L, x_i) + \phi(\mathcal{W}_*^L, \mathcal{B}_*^L, x_i), y_i\right)$$
$$= \lambda \left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 + \frac{1}{N}\sum_{i=1}^N \text{loss}\left(\left[\hat{\phi}(\mathcal{W}_*^L, \mathcal{B}_*^L, x_i) - \phi(\mathcal{W}_*^L, \mathcal{B}_*^L, x_i)\right] + y_i, y_i\right)$$
$$\leq \lambda \left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 + \frac{1}{N}\sum_{i=1}^N \mathscr{A}_{\text{loss}}(\nu, \mathcal{W}_*^L, \mathcal{B}_*^L) = \lambda \left|\!\left|\!\left|(\mathcal{W}_*^L, \mathcal{B}_*^L)\right|\!\right|\!\right|_2^2 + \mathscr{A}_{\text{loss}}(\nu, \mathcal{W}_*^L, \mathcal{B}_*^L),$$

which concludes the proof. □

*Proof of Corollary 1.4.* Let us analyze the behavior of $\sigma_\varepsilon$ when $\varepsilon \to 0$. We first observe that

$$\lim_{u\to 0^+} \text{erf}\left(\frac{1}{u}\right) = 1, \quad \text{and} \quad \lim_{u\to 0^-} \text{erf}\left(\frac{1}{u}\right) = -1.$$

Now, taking $u = \varepsilon\sqrt{2}/x$, we observe that since $\varepsilon > 0$, we have

$$\lim_{\varepsilon\to 0^+} \text{erf}\left(\frac{x}{\varepsilon\sqrt{2}}\right) = \begin{cases} 1 & \text{if } x > 0, \\ -1 & \text{if } x < 0. \end{cases}$$

Consequently, we have

$$\lim_{\varepsilon\to 0^+} \sigma_\varepsilon(x) = \lim_{\varepsilon\to 0^+} \frac{x}{2}\left(1 + \text{erf}\left(\frac{x}{\varepsilon\sqrt{2}}\right)\right) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{if } x < 0, \end{cases} = \text{ReLU}(x) = \sigma(x). \tag{6.9}$$

Therefore, since (6.9) holds for every $x$, we deduce that $\sigma_\varepsilon \to \sigma$ uniformly in $\mathbb{R}$. The rest of the proof follows as a direct application of Theorem 1.3, by observing that when $\varepsilon \to 0$ then $\nu \to 0$ as well. □

## 7. Further comments and open problems

In this paper, we have demonstrated that a 2-wide deep neural network can address any classification problem with no more than $O(N)$ layers. Additionally, we have established a universal approximation theorem for $L^p(\Omega;\mathbb{R}_+)$ functions, requiring a neural network width $d+1$. Notably, our proofs are fully constructive, explicitly detailing the parameters to be utilized, giving a fully geometric interpretation of the architecture employed, and providing a formal proof of each statement.

These explicit constructions yield bounds on the parameters and provide a priori estimates for minimizers of standard regularized training loss functionals in supervised learning. As the regularization parameter vanishes, the trained networks converge to exact classifiers.

In the following, we present some interesting open-related questions.

**(1) Understanding $n$-wide deep neural networks.** The construction of the parameters in Theorem 1.1 provides a clear and geometric interpretation of why and how the neural network achieves memorization. In this context, it would also be interesting to describe and explain geometrically other results in the literature, such as the 3-wide deep neural networks constructed in [32], or the two-layer network constructed in [46] that achieves memorization with $O(N^{1/2})$ neurons.

**(2) Topology of the Dataset.** The first step in the proof of Theorem 1.1 involves projecting the data into a one-dimensional space. This reduction simplifies the data structure, facilitating the development of our algorithm. However, this projection results in the possible loss of the original data distribution, since it could, for instance, disperse initially clustered points. Therefore, to take advantage of the initial data distribution, we could project the data in a space of dimension greater than one, using more hyperplanes (neurons) at the first step (layer). One possibility is to find a low-dimensional space in which the points can be embedded, preserving distances. This is precisely what the Johnson-Lindenstrauss lemma states, ensuring the existence of a linear map that projects points in a lower dimensional space, preserving distances. This could reduce the number of hidden layers that Theorem 1.1 uses. Manipulating data in dimensions higher than 2 can considerably reduce the number of layers needed, [32].

**(3) Width versus Depth.** As we have seen in the bibliographic discussion of Section 1.4, networks with one hidden layer can memorize $N$ data points with $O(N)$ neurons, while adding an extra layer, making it possible using $O(\sqrt{N})$ neurons, [46]. In the context of deep neural networks, width 12 allows memorization with $O(N^{1/2}+\log(N))$ neurons, [32], and width 3 with $O(N^{2/3}\log(N))$ neurons. In this paper, we have shown that $O(N)$ neurons suffice for width 2. These results exhibit a trade-off between the depth and width of the network. A systematic analysis of this compromise between depth and width would be desirable.

**(4) Extension of the Universal Approximation Theorem.** As observed in the proof of Theorem 1.4, Theorem 1.1 was utilized to map the resulting points to their respective labels. However, a universal approximation theorem can also be concluded using other networks, not necessarily of width 2 [32, 43, 46, 47]. By maintaining a width of $d+1$, we can incorporate the neural network with width 3 introduced in [32] to ensure universal approximation when $d \geq 2$. Combining this result with our strategy may lead to a neural network with reduced depth compared to the one given by Theorem 1.4.

On the other hand, the first step of the proof of Theorem 1.4 uses an approximation by simple functions of finite volume type on a regular set of hyperrectangles, ensuring an error of order $h$ [11]. However, more sophisticated nonlinear approximation procedures, such as those based on *dyadic partitions*, could achieve better convergence rates (see [11]). Nevertheless, to approximate functions using the neural network, it would be necessary to develop an iterative algorithm operating over a non-uniform grid.

**(5) Minimal width universal approximation theorem for more general spaces.** Universal approximation theorems have been extended to Sobolev and Besov spaces, as discussed in [12, 37]. In [37], it is shown that the class $W^{s,q}$, when compactly embedded in $L^p$, can be approximated by neural networks. For $p \leq q$, piecewise polynomial approximations on uniform grids are applicable, which neural networks can approximate. Additionally, the neural network in [37] uses a width $25d+31$. The uniformity of the grid allows us to extend our methodology to estimate the network depth while maintaining a small width and understanding the parameters involved in the approximation. However, for $p > q$, nonlinear or adaptive methods are required. Although [37] constructs a neural network to approximate nonlinear functions on non-uniform grids, the choice of parameters lacks clear geometric intuition, and no algorithmic procedure is provided.

## Appendix A.

*Proof.* (Proof of Proposition 4.1) Recall that the classes under consideration are defined as follows:

$$\mathcal{C}_k = \{x_i \text{ with } i \in [\![1, N]\!] : y_i = k\}, \quad \text{and} \quad \mathcal{C} = \bigcup_{k=0}^{M-1} \mathcal{C}_k. \tag{A.1}$$

To prove that we can compress the $M$ classes, we proceed by induction. For the first class, i.e. $k = 0$, the proof follows directly by noting that the assumptions of the statement ensure the existence of parameters $\mathcal{W}^{L_0}, \mathcal{B}^{L_0}$ and $\tilde{z}_0 \in \mathbb{R}^2$ such that $\phi(\mathcal{W}^{L_0}, \mathcal{B}^{L_0}, \mathcal{C}_0) = \tilde{z}_0$.

Let us assume that the statement holds for some $0 < k < M - 1$. Thus, there exists a collection of points $\{z_j\}_{j=0}^k \subset \mathbb{R}^2$, $\tilde{L}_1 > 1$, and parameters $\mathcal{W}^{\tilde{L}_1}, \mathcal{B}^{\tilde{L}_1}$ such that the input-output map satisfies

$$\phi_k(\mathcal{W}^{\tilde{L}_1}, \mathcal{B}^{\tilde{L}_1}, \mathcal{C}_j) = z_j, \quad \text{for every } j \in [\![0, k]\!].$$

In particular, we have that

$$\phi_k(\mathcal{W}^{\tilde{L}_1}, \mathcal{B}^{\tilde{L}_1}, \mathcal{C}_{k+1}) = \hat{\mathcal{C}}_{k+1}, \quad \text{where } \hat{\mathcal{C}}_{k+1} = \{z_i \text{ with } i \in [\![1, N]\!] : y_i = k + 1\}.$$

Let us prove the statement for $k + 1$. Denote by $\hat{\mathcal{C}} = \phi^k(\mathcal{W}^{\tilde{L}_1}, \mathcal{B}^{\tilde{L}_1}, \mathcal{C})$. By hypothesis, there exist $\hat{z}_{k+1} \in \mathbb{R}^2$, $\tilde{L}_2 \geq 1$, $\mathcal{W}^{\tilde{L}_2}$ and $\mathcal{B}^{\tilde{L}_2}$ such that

$$\phi_{k+1}(\mathcal{W}^{\tilde{L}_2}, \mathcal{B}^{\tilde{L}_2}, \hat{\mathcal{C}}_{k+1}) = \hat{z}_{k+1}, \qquad \phi_{k+1}(\mathcal{W}^{\tilde{L}_2}, \mathcal{B}^{\tilde{L}_2}, \hat{\mathcal{C}} \setminus \hat{\mathcal{C}}_{k+1}) \neq \hat{z}_{k+1}, \tag{A.2}$$

and

$$\phi_{k+1}(\mathcal{W}^{\tilde{L}_2}, \mathcal{B}^{\tilde{L}_2}, z^1) \neq \phi_{k+1}(\mathcal{W}^{\tilde{L}_2}, \mathcal{B}^{\tilde{L}_2}, z^2), \quad \text{for all } z^1, z^2 \in \hat{\mathcal{C}} \setminus \hat{\mathcal{C}}_{k+1},\ z^1 \neq z^2. \tag{A.3}$$

Since the composition of input-output maps is again an input-output map of (1.2), we can compose $\phi_{k+1}$ and $\phi_k$. Then, by (A.2)-(A.3), the map $\hat{\phi} := \phi_{k+1} \circ \phi_k$ satisfies that

$$\hat{\phi}(\mathcal{W}^{\tilde{L}_3}, \mathcal{B}^{\tilde{L}_3}, \mathcal{C}_j) = \hat{z}_j \quad \text{for every } j \in [\![0, k + 1]\!], \tag{A.4}$$

where $\tilde{L}_3 = \tilde{L}_2 + \tilde{L}_1$, $\mathcal{W}^{\tilde{L}_3} = \mathcal{W}^{\tilde{L}_2} \cup \mathcal{W}^{\tilde{L}_1}$, and $\mathcal{B}^{\tilde{L}_3} = \mathcal{B}^{\tilde{L}_2} \cup \mathcal{B}^{\tilde{L}_1}$. This concludes the induction. The result follows by taking $k = M - 2$ in (A.4). □

*Proof.* (Proof of Corollary 1.3) Let us consider the dataset $\{(x_i, y_i)\}_{i=1}^N \subset \mathbb{R}^d \times \{\alpha_0, \dots, \alpha_{M-1}\}$, where $\{\alpha_k\}_{k=0}^{M-1} \subset \mathbb{R}$. Without loss of generality, assume that $y_i < y_j$ for every $i < j$, where $i, j \in \{0, \dots, M-1\}$. If $y_0 \geq 0$, we conclude by applying Theorem 1.1. If $y_0 < 0$, we consider a new set of labels given by $\hat{y}_i = y_i - y_0$ for every $i \in \{0, \dots, M - 1\}$, noting that $\hat{y}_0 = 0$. Furthermore, $\{\hat{y}_i\}_{i=1}^N \subset \{\hat{\alpha}_k\}_{k=0}^{M-1} \subset \mathbb{R}_+$. Then, according to Theorem 1.1, there exist parameters $\mathcal{W}^L$ and $\mathcal{B}^L$ such that for $L = 2N + 4M - 1$ and $w_{\max} = 2$, the input-output map of (1.7) with $A_j^1 = \mathrm{Id}_{d_j}$ (the identity matrix in $\mathbb{R}^{d_j \times d_j}$) for all $j \in \{1, \dots, L\}$, satisfies

$$\phi^L(x_i) = \phi(\mathcal{A}^L, \mathcal{W}^L, \mathcal{B}^L, x_i) = \hat{y}_i, \quad \text{for all } i \in \{1, \dots, N\}. \tag{A.5}$$

It now suffices to construct a mapping that transforms each $\hat{y}_i$ into $y_i$ for every $i \in \{1, \dots, N\}$. Consider the parameters

$$w_1^1 = -1, \quad w_2^1 = 1, \quad b_1^1 = -y_0, \quad b_2^1 = y_0, \quad \text{and } A^1 = (-1, 1),$$

where $W^1 = (w_1^1, w_2^1)^\top$ and $b^1 = (b_1^1, b_2^1)^\top$. We denote by $\phi^1$ the input-output map of (1.7) defined by $W^1, b^1$, and $A^1$. For every $\hat{y}_i$, we have that $-\hat{y}_i + y_0 \geq 0$ or $\hat{y}_i - y_0 \geq 0$. If $-\hat{y}_i + y_0 \geq 0$, then

$$\begin{aligned} \phi^1(\hat{y}_i) &= A^1 \cdot \boldsymbol{\sigma}(W^1 \hat{y}_i + b^1) = (-1, 1) \cdot \begin{pmatrix} \sigma(-\hat{y}_i + y_0) \\ \sigma(\hat{y}_i - y_0) \end{pmatrix} \\ &= -\sigma(-\hat{y}_i + y_0) + \sigma(\hat{y}_i - y_0) = -(-\hat{y}_i + y_0) = y_i. \end{aligned}$$

Similarly, for $\hat{y}_i$ such that $\hat{y}_i - y_0 \geq 0$, we obtain $\phi^1(\hat{y}_i) = (\hat{y}_i - y_0) = y_i$. Therefore, the input-output map $\phi^{L+1} := \phi^1 \circ \phi^L$ can memorize the dataset $\{(x_i, y_i)\}_{i=1}^N$. Moreover, since the width and depth of the neural network defined by $\phi^1$ are 2 and 1, respectively, the resulting neural network defined by $\phi^{L+1}$ has a width 2 and a depth $2N + 4M$.

□

## Appendix B.

**Lemma B.1.** *For every $I_n$ with $n \geq 1$ and for all $\mathcal{H}_j, \mathcal{H}_i \in \mathcal{S}_{I_n}$ with $i \neq j$, we have that $F(\mathcal{H}_i) \cap F(\mathcal{H}_j) = \emptyset$.*

*Proof.* (Proof of Lemma B.1) By contradiction, assume that there exist $\mathcal{H}_1$ and $\mathcal{H}_2$ in some $\mathcal{S}_{I_n}$ such that $F(\mathcal{H}_1) \cap F(\mathcal{H}_2) \neq \emptyset$. Therefore, there exist $x_1 \in \mathcal{H}_1$ and $x_2 \in \mathcal{H}_2$ such that $F(x_1) = F(x_2)$. Since $\mathcal{H}_1$ and $\mathcal{H}_2$ are different hyperrectangles, there exists $k \in [\![1, d+1]\!]$ such that $x_1^{(k)} \neq x_2^{(k)}$. Due to the fact that $\mathcal{H}_1$ and $\mathcal{H}_2$ belong to the same subregion, we have that

$$p_\eta(x_1) = p_\eta(x_2), \quad \text{for every } \eta \in [\![1, d+1]\!]. \tag{B.1}$$

Since $F(x_1) = F(x_2)$ on each coordinate, we will have

$$p_\eta(x_1)x_1^{(\eta)} + p_\eta(x_1)b_\eta = p_\eta(x_2)x_2^{(\eta)} + p_\eta(x_2)b_\eta, \quad \text{for every } \eta \in [\![1, d+1]\!]. \tag{B.2}$$

Therefore, using (B.1), we conclude that $x_1^{(\eta)} = x_2^{(\eta)}$ for every $\eta \in [\![1, d+1]\!]$, which is a contradiction since $x_1^{(k)} \neq x_2^{(k)}$. □

*Proof.* (Proof of Lemma 5.1) From the definition of $p_\eta(x)$, we immediately have that $F(\mathcal{H}_*) = \mathbf{0}_d$. For the second part of the lemma, we proceed by contradiction. Let us assume that there exist $\mathcal{H}_1, \mathcal{H}_2$ such that $F(\mathcal{H}_1) \cap F(\mathcal{H}_2) \neq \emptyset$. If $\mathcal{H}_1$ and $\mathcal{H}_2$ belong to the same subregion, we are done due to Lemma B.1.

Therefore, we can assume that $\mathcal{H}_1$ and $\mathcal{H}_2$ are in different subregions. This implies that a hyperplane separates them. Thus, there exists $k \in [\![1, d+1]\!]$ such that

$$x_1^{(k)} < b_k < x_2^{(k)} \quad \text{or} \quad x_2^{(k)} < b_k < x_1^{(k)}, \tag{B.3}$$

and that $p_k(x_1) \neq p_k(x_2)$ for every $x_1 \in \mathcal{H}_1$ and $x_2 \in \mathcal{H}_2$. Without loss of generality, we assume that

$$p_k(x_1) = 1 \quad \text{and} \quad p_k(x_2) = 0, \tag{B.4}$$

that is, $\mathcal{H}_1 \subset R_k$, and due to the fact that the hyperrectangles do not intersect the hyperplanes, we have that

$$e_k \cdot x_1 + b_k > 0. \tag{B.5}$$

We continue the proof by dividing it into two cases.

• The case $P(x_1) = P(x_2)$: In such case, we have that $p_\eta(x_1)x_1^{(\eta)} = p_\eta(x_2)x_2^{(\eta)}$ for all $\eta \in [\![1, d+1]\!]$. Using (B.4), we deduce that $x_1^{(k)} = 0$. Thus, due to (B.3), we have that $b_k \neq 0$. Since $F(x_1) = F(x_2)$ and we have assumed that $P(x_1) = P(x_2)$, then $G(x_1) = G(x_2)$. The last equality implies that $p_\eta(x_1)b_\eta = p_\eta(x_2)b_\eta$ for all $\eta \in [\![1, d+1]\!]$, therefore, applying (B.4), we conclude that $b_k = 0$, which is a contradiction.

• The case $P(x_1) \neq P(x_2)$. When $p_k(x_1)x_1^{(k)} \neq p_k(x_2)x_2^{(k)}$, due to (B.4), necessarily $x_1^{(k)} \neq 0$. As before, using (B.4) and the fact that $F(x_1) = F(x_2)$, we deduce that $x_1^{(k)} = -b^k$. Therefore, considering (B.5), we face a contradiction. □

*Proof of the Lemma 5.2.* Let us begin by observing that Corollary 1.1 cannot be directly applied since now we have an infinite number of data points. However, we can provide a similar estimation by carefully analyzing the parameters used in the proof of Theorem 1.4. According to Step 2.3 in the proof of Theorem 1.4, the map $\phi^{\mathcal{C}} = \phi^L \circ \phi^{2N_E}$ drives the hyperrectangles defined in $\mathcal{H}$ into their respectively labels. Therefore, to estimate the norm of $\phi^{\mathcal{C}}$, we divide the proof into two parts.

**Norm of $\phi^{2N_E}$.** To estimate the norm of $\phi^{2N_E}$, we make the following observations:

1) Due to the fact that the hyperplanes defined in (5.8) must belong to $G_\delta^h$, we have

$$\|b_\eta\|_\infty \leq C_\eta(h + \delta/2) + m_d(\Omega), \quad \text{for every } \eta \in [\![1, d+1]\!],$$

where $m_d(\mathcal{C})$ is the Lebesgue measure of $\mathcal{C}$ and the $C_\eta$'s are positive uniformly bounded constants. Thus, $\|b^1\|_\infty \leq Ch + m_d(\Omega)$. Moreover, by definition, $\|W^1\|_\infty = 1$.

2) With the parameters derived in Step 2 of the proof of Theorem 1.4, the hyperrectangles are mapped to a $d+1$-dimensional space. Since $\|W^1\| = 1$, the hyperrectangles are mapped according to their distance to the hyperplane, which is less than $\delta/2$. Furthermore, all hyperrectangles are no farther away than $C(h + \delta/2)$. Thus, the parameters $b_\eta^2$ introduced in Step 2 of the proof of Theorem 1.4 satisfy $\|b^2\|_\infty \leq C(h + \delta/2)$. By definition, again, $\|W^2\|_\infty = 1$.

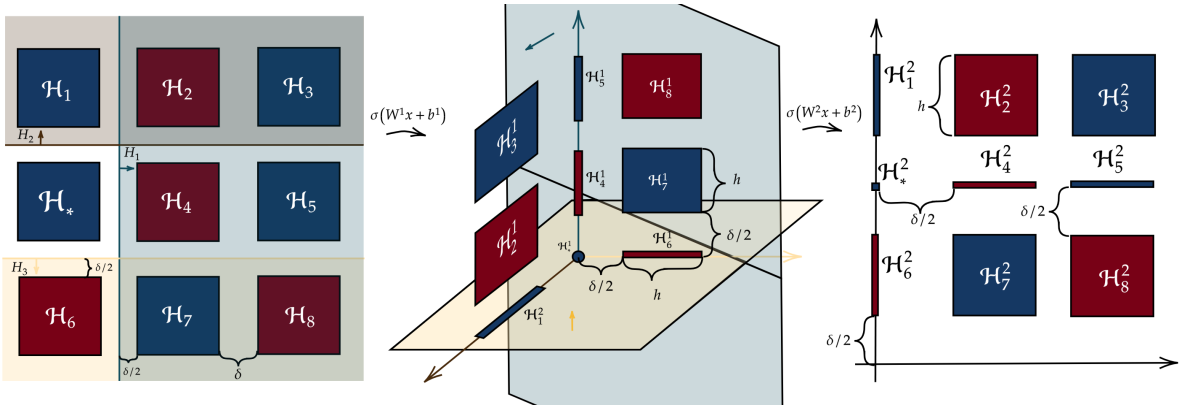



FIGURE 25. Illustration of the initial steps in the compression process. For a specific 2−dimensional example, we show how the parameters $(W^1, b^1)$ and $(W^2, b^2)$ affect the hyperrectangles, reducing distances. The first figure shows hyperrectangles separated by a $\delta$ distance. We choose hyperplanes with normal vectors $(0,1)$, $(0,-1)$, and $(1,0)$ to collapse the hypercube $\mathcal{H}^*$. This action maps the hyperrectangles into a $3-d$ space. Subsequently, using two hyperplanes with normal vectors $(0,0,1)$ and $(1,-1,0)$, we map the hyperrectangles onto a $2-d$ space, where the distance between the hyperrectangles is now $\delta/2$ close to zero, and the farthest hyperrectangle is at most at distance $C(h+\delta/2)$.

3) When projecting the hyperrectangles into the $d$-dimensional space, they remain no farther away than $C(h+\delta/2)$. Additionally, the hyperrectangles are contained within the interior of a ball $B_0(C(h+\delta/2))$, centered at zero with radius $C(h+\delta/2)$ (see Figure 25).

4) Note that if we apply similar parameters, this time to compress $\mathcal{H}_2^2$, from Figure 25, the distance between hyperrectangles becomes $\delta/4$. Generally, the distance between hyperrectangles in step $j$ is $\delta/(2^{\lfloor j/2 \rfloor})$.

In the compression phase, we apply an iterative process where the parameters are selected based on the same criteria. Therefore, we conclude that

$$\|b^j\|_\infty = \begin{cases} C(h+\delta/2) + m_d(\mathcal{C}) & \text{if } j=1, \\ C(h+\delta/(2^{\lfloor j/2 \rfloor})) & \text{otherwise,} \end{cases} \qquad \|W^j\|_\infty = 1, \quad \forall j \in [\![1, 2dN_e]\!].$$

Consequently,

$$\|\phi^{2N_E}(x)\| \leq \|x\| + m_d(\mathcal{C}) + 2dN_E C\left(h + \frac{\delta}{2}\right). \tag{B.6}$$

**Norm of $\phi^L$.** We have shown that in the compression process, the data is driven into a ball $B_0(C(h+\delta/2^{2N_E}))$, where $C$ is a constant depending on $m_d(\mathcal{C})$. Moreover, the distance between the points does not exceed $\delta/(2^{2N_E})$. Therefore, the resulting $N$ points from the compression process reside within that ball. Note that the map $\phi^L = (\phi_3^{L_3} \circ \phi_2^{L_2} \circ \phi_1^{L_1} \circ \phi_0^{L_0})$ corresponds to the map constructed in Section 4. Thus, starting from the output of $\phi^{2N_E}$, we analyze each map $\phi_i^{L_i}$.

1) Precondition of the data: In this phase, $b_1$ is chosen large enough such that $\boldsymbol{\sigma}$ acts as the identity function. Considering as an input data point the output of the map $\phi^{2N_E}$, then it is enough to take $b$ larger than $C(h+\delta/2^{2N_E})$. This implies:

$$\|W_0\|_\infty = 1, \quad \|b_0\|_\infty \leq 2C(h+\delta/2^{2N_E}).$$

2) Compression process: After data preconditioning, all the datasets have been projected to the real line, and the distance between points does not exceed $C\left(\delta/2^{2N_E}\right)$. We place the hyperplanes in the *"Compression process"* depending on the data location. Therefore, we deduce that

$$\|W_j\|_\infty \leq 1, \quad \|b_j\|_\infty \leq C\left(\frac{\delta}{2^{2N_E}}\right), \qquad \text{for all } j \in [\![1, \ldots, 2N+1]\!].$$

3) Data sorting: In this step, we place the hyperplanes depending on the data location. Then, we obtain that

$$\|W_j\|_\infty \leq 1, \quad \|b_j\|_\infty \leq C\left(\frac{\delta}{2^{2N_E}}\right), \qquad \text{for all } j \in [\![2N+2, \dots, 2N+2M+2]\!].$$

4) Mapping to the respective label: In this step, we expand or contract the data to map them to their respective labels. In Theorem 1.4, the labels are defined by the different values $\{f_i^h\}_i$ that the function $f_h$ takes. Then, we deduce that

$$\|\phi^{L_3}\|_{L^\infty(\mathcal{C};\mathbb{R}_+)} \leq \max\{\max_i\{f_i^h\}, m_d(\mathcal{C})\}.$$

Then, analogous to (5.21), we obtain that

$$|f_i^h| \leq C(1 + \|f\|_{L^p(\mathcal{H}_i;\mathbb{R}_+)}) \leq C(1 + \|f\|_{L^p(\Omega;\mathbb{R}_+)}).$$

Consequently, there exists a constant $C > 0$ that only depends on $\|f\|_{L^p(\Omega;\mathbb{R}_+)}$ and $m_d(\mathcal{C})$, such that $\|\phi^{L_3}\|_{L^\infty(\mathcal{C};\mathbb{R}_+)} \leq C$.

On the other hand, given $\hat{L} > 0$ and a family of parameters $\mathcal{W}^{\hat{L}} = \{W^i\}_{i=1}^{\hat{L}}$ and $\mathcal{B}^{\hat{L}} = \{b^i\}_{i=1}^{\hat{L}}$, the norm of $\phi^{\hat{L}} := \phi^{\hat{L}}(\mathcal{W}^{\hat{L}}, \mathcal{B}^{\hat{L}}, \cdot)$, the input-output map of (1.2), can be bounded by

$$\|\phi^{\hat{L}}\|_{L^\infty(\mathcal{C};\mathbb{R}_+)} \leq \operatorname*{ess\,sup}_{x\in\mathcal{C}} \left\| \prod_{j=1}^{\hat{L}} W^j x + \sum_{i=1}^{\hat{L}-1} \left( \prod_{j=i}^{\hat{L}-1} W^{j+1} \right) b^i + b^{\hat{L}} \right\|_\infty \tag{B.7}$$

Consequently, using the fact that in the compression process and data sorting, we are using $2N_h$ and $2M_h + 1$ layers, respectively, and the estimation of the parameters norms, we can apply (B.7) to deduce that

$$\begin{aligned}\|\phi^{\mathcal{L}}\|_{L^\infty(\mathcal{C};\mathbb{R}_+)} = \|\phi^L \circ \phi^{2N_E}\| \leq C \operatorname*{ess\,sup}_{x\in\mathcal{C}} \|\phi^{2N_E}(x)\| \\ + 2N_h C\left(\frac{\delta}{2^{2N_E}}\right) + (2M_h+1)C\left(\frac{\delta}{2^{2N_E}}\right) + 2C\left(h + \frac{\delta}{2^{N_E}}\right).\end{aligned}$$

Then, using (B.6) we have

$$\|\phi^{\mathcal{L}}\|_{L^\infty(\mathcal{C};\mathbb{R}_+)} \leq 2Cm_d(\mathcal{C}) + 2dN_E C\left(h + \frac{\delta}{2}\right) + (2N_h + 2M_h + 1)C\left(\frac{\delta}{2^{2N_E}}\right) + 2C\left(h + \frac{\delta}{2^{N_E}}\right). \tag{B.8}$$

Denote by $l_{\mathcal{C}}$ the largest edge of $\mathcal{C}$. Then, applying (5.28) in (B.8) there exists a positive constant $C_1$, independent of $h$ and $\delta$, such that

$$\|\phi^{\mathcal{L}}\|_{L^\infty(\mathcal{C};\mathbb{R}_+)} \leq C_1\left(1 + \delta\frac{2^{\frac{-C_2}{h+\delta}}}{(h+\delta)^d} + \delta 2^{\frac{-C_2}{h+\delta}} + h\right), \tag{B.9}$$

where $C_2 = 2l_{\mathcal{C}}$. Now, since $e^y \geq \sum_{k=0}^{d+1} y^k/k!$ using the change of variable $y = log(2)C_2/(h+\delta)$ we deduce that

$$\begin{aligned}\frac{2^{\frac{-C_2}{h+\delta}}}{(h+\delta)^d} &\leq \frac{1}{(h+\delta)^d}\left(\sum_{k=0}^{d+1}\frac{(C_2\log(2))^k(h+\delta)^{-k}}{k!}\right)^{-1} \\ &= \frac{1}{(h+\delta)^d}\left(\frac{1}{(h+\delta)^{d+1}}\sum_{k=0}^{d+1}\frac{(C_2\log(2))^k(h+\delta)^{(d+1)-k}}{k!}\right)^{-1} \\ &= (h+\delta)\left(\frac{(C_2\log(2))^{d+1}}{(d+1)!} + \sum_{k=0}^{d}\frac{(C_2\log(2))^k(h+\delta)^{(d+1)-k}}{k!}\right)^{-1}.\end{aligned}$$

Therefore, for $h < \frac{l_{\mathcal{C}}\log(2)}{(d+1)}$ we have that

$$\frac{2^{\frac{-C_2}{h+\delta}}}{(h+\delta)^d} \leq \frac{(dp+1)!}{(2l_{\mathcal{C}}\log(2))^{d+1}}(h+\delta),$$

Similarly, using the inequality $e^y \geq 1 + y$ and the change of variable $y = log(2)C_2/(h + \delta)$, we deduce that for $h < l_{\mathcal{C}} \log(2)$ we have that $2^{\frac{-C_2}{h+\delta}} \leq (l_{\mathcal{C}} \log(2))^{-1}(h + \delta)$. Consequently, from (B.9) we obtain the inequality

$$\|\phi^{\mathcal{L}}\|_{L^\infty(\mathcal{C};\mathbb{R}_+)} \leq C\,(1 + \delta(h + \delta) + h)\,,$$

with $C > 0$ a constant depending on $d$, $m_d(\mathcal{C})$, and $\|f\|_{L^p(\Omega;\mathbb{R}_+)}$. This concludes the proof. □

## Acknowledgments

The authors wish to express their gratitude to D. Ruiz-Balet and A. Alcalde for their insightful discussions and to A.Álvarez-López and T. Crin-Barat, for taking the time to critically review our manuscript.

M. Hernández has been funded by the Transregio 154 Project, Mathematical Modelling, Simulation, and Optimization Using the Example of Gas Networks of the DFG, project C07, and the fellowship "ANID-DAAD bilateral agreement".

E. Zuazua was funded by the European Research Council (ERC) under the European Union's Horizon 2030 research and innovation programme (grant agreement NO: 101096251-CoDeFeL), the Alexander von Humboldt-Professorship program, the ModConFlex Marie Curie Action, HORIZON-MSCA-2021-dN-01, the COST Action MAT-DYNNET, the Transregio 154 Project of the DFG, AFOSR Proposal 24IOE027 and grants PID2020-112617GB-C22/AEI/10.13039/501100011033 and TED2021131390B-I00/AEI/10.13039/501100011033 of MINECO (Spain), and Madrid GovernmentUAM Agreement for the Excellence of the University Research Staff in the context of the V PRICIT (Regional Programme of Research and Technological Innovation).

Both authors have been partially supported by the DAAD/CAPES Programs for Project-Related Personal, grant 57703041 'Control and numerical analysis of complex system'.

## References

[1] A. Agrachev and A. Sarychev. Control on the manifolds of mappings with a view to the deep learning. *Journal of Dynamical and Control Systems*, 28(4):989–1008, 2022.

[2] S. Alberti, N. Dern, L. Thesing, and G. Kutyniok. Sumformer: Universal approximation for efficient transformers, 2023.

[3] A. Álvarez-López, A. H. Slimane, and E. Zuazua. Interplay between depth and width for interpolation in neural odes. *Neural Networks*, 180:106640, 2024.

[4] A. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[5] E. B. Baum. On the capabilities of multilayer perceptrons. *Journal of complexity*, 4(3):193–215, 1988.

[6] K. Bhandari, J. Lemoine, and A. Münch. Exact boundary controllability of 1D semilinear wave equations through a constructive approach. *Math. Control Signals Systems*, 35(1):77–123, 2023.

[7] Y. Cai. Achieve the minimum width of neural networks for universal approximation. *arXiv preprint arXiv:2209.11395*, 2022.

[8] J. Cheng, Q. Li, T. Lin, and Z. Shen. Interpolation, approximation and controllability of deep neural networks, 2023.

[9] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, (3):326–334, 1965.

[10] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[11] O. Davydov. Algorithms and error bounds for multivariate piecewise constant approximation. In *Approximation Algorithms for Complex Systems: Proceedings of the 6th International Conference on Algorithms for Approximation, Ambleside, UK, 31st August-4th September 2009*, pages 27–45. Springer, 2010.

[12] R. DeVore, B. Hanin, and G. Petrova. Neural network approximation. *Acta Numerica*, 30:327–444, 2021.

[13] R. A. DeVore. Nonlinear approximation. In *Acta numerica, 1998*, volume 7 of *Acta Numer.*, pages 51–150. Cambridge Univ. Press, Cambridge, 1998.

[14] S. Ervedoza, J. Lemoine, and A. Münch. Exact controllability of semilinear heat equations through a constructive approach. *Evol. Equ. Control Theory*, 12(2):567–599, 2023.

[15] B. Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, 7(10):992, 2019.

[16] B. Hanin and M. Sellke. Approximating continuous functions by relu nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

[17] M. Hardt and T. Ma. Identity matters in deep learning. In *International Conference on Learning Representations*, 2017.

[18] K. Hornik, M. B. Stinchcombe, and H. L. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[19] G.-B. Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, 14(2):274–281, 2003.

[20] G.-B. Huang and H. A. Babri. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE transactions on neural networks*, 9(1):224–229, 1998.
[21] S.-C. Huang and Y.-F. Huang. Bounds on number of hidden neurons of multilayer perceptrons in classification and recognition. In *1990 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2500–2503. IEEE, 1990.
[22] M. Jankovic, R. Sepulchre, and P. V. Kokotovic. Constructive Lyapunov stabilization of nonlinear cascade systems. *IEEE Trans. Automat. Control*, 41(12):1723–1735, 1996.
[23] M. Jidou Khayar, A. Brouri, and M. Ouzahra. Exact controllability of the reaction-diffusion equation under bilinear control. *Nonlinear Dyn. Syst. Theory*, 22(5):538–549, 2022.
[24] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, Aug 2021.
[25] P. Kidger and T. Lyons. Universal approximation with deep narrow networks. In *Conference on learning theory*, pages 2306–2327. PMLR, 2020.
[26] N. Kim, C. Min, and S. Park. Minimum width for universal approximation using relu networks on compact domain. *arXiv preprint arXiv:2309.10402*, 2023.
[27] A. Kowalczyk. Estimates of storage capacity of multilayer perceptron with threshold logic hidden units. *Neural networks*, 10(8):1417–1433, 1997.
[28] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.
[29] L. Li, Y. Duan, G. Ji, and Y. Cai. Minimum width of leaky-relu neural networks for uniform universal approximation. In *International Conference on Machine Learning*, pages 19460–19470. PMLR, 2023.
[30] J. Lohéac and E. Zuazua. From averaged to simultaneous controllability. *Ann. Fac. Sci. Toulouse Math. (6)*, 25(4):785–828, 2016.
[31] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30, 2017.
[32] S. Park, J. Lee, C. Yun, and J. Shin. Provable memorization via deep neural networks using sub-linear parameters. In *Conference on Learning Theory*, pages 3627–3661. PMLR, 2021.
[33] S. Park, C. Yun, J. Lee, and J. Shin. Minimum width for universal approximation. *arXiv preprint arXiv:2006.08859*, 2020.
[34] A. Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.
[35] D. Ruiz-Balet and E. Zuazua. Neural ODE Control for Classification, Approximation, and Transport. *SIAM Rev.*, 65(3):735–773, 2023.
[36] M. Schönlein and U. Helmke. Controllability of ensembles of linear dynamical systems. *Math. Comput. Simulation*, 125:3–14, 2016.
[37] J. W. Siegel. Optimal approximation rates for deep relu neural networks on sobolev and besov spaces. *Journal of Machine Learning Research*, 24(357):1–52, 2023.
[38] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.
[39] C. h. Song, G. Hwang, J. h. Lee, and M. Kang. Minimal width for universal property of deep RNN. *J. Mach. Learn. Res.*, 24:Paper No. [121], 41, 2023.
[40] P. Tabuada and B. Gharesifard. Universal approximation power of deep residual neural networks through the lens of control. *IEEE Transactions on Automatic Control*, 68(5):2715–2728, 2023.
[41] M. Telgarsky. Benefits of depth in neural networks. In *Conference on learning theory*, pages 1517–1539. PMLR, 2016.
[42] M. Tucsnak and G. Weiss. Simultaneous exact controllability and some applications. *SIAM J. Control Optim.*, 38(5):1408–1427, 2000.
[43] G. Vardi, G. Yehudai, and O. Shamir. On the optimal memorization power of reLU neural networks. In *International Conference on Learning Representations*, 2022.
[44] P. Wang, R. Katz, and E. Fridman. Constructive finite-dimensional boundary control of stochastic 1D parabolic PDEs. *Automatica J. IFAC*, 148:Paper No. 110793, 16, 2023.
[45] M. Yamasaki. The lower bound of the capacity for a neural network with multiple hidden layers. In *ICANN'93: Proceedings of the International Conference on Artificial Neural Networks Amsterdam, The Netherlands 13–16 September 1993 3*, pages 546–549. Springer, 1993.
[46] C. Yun, S. Sra, and A. Jadbabaie. Small relu networks are powerful memorizers: a tight analysis of memorization capacity. *Advances in Neural Information Processing Systems*, 32, 2019.
[47] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

*Email address*: martin.hernandez@fau.de

*Email address*: enrique.zuazua@fau.de