Approximation properties of shallow quadratic neural networks and clustering applications

Leon Frischauf¹ leon.frischauf@univie.ac.at

Otmar Scherzer^{1,2,3} otmar.scherzer@univie.ac.at Cong Shi^{1,4*} cong.shi@univie.ac.at

University of Vienna Oskar-Morgenstern-Platz 1 A-1090 Vienna, Austria

¹Faculty of Mathematics

² Johann Radon Institute for Computational and Applied Mathematics (RICAM) Altenbergerstraße 69 A-4040 Linz, Austria

³Christian Doppler Laboratory for Mathematical Modeling and Simulation of Next Generations of Ultrasound Devices (MaMSi) Oskar-Morgenstern-Platz 1 A-1090 Vienna, Austria ⁴School of Mathematics (Zhuhai) Sun Yat-Sen University Hanlin Rd, 519082 Zhuhai Guangdong Province, China

Abstract

In this paper we study *shallow* neural network functions which are linear combinations of compositions of activation and *quadratic* functions, replacing standard *affine linear* functions, often called neurons. We show the universality of this approximation and prove convergence rates results based on the theory of wavelets and statistical learning. We show for simple test cases that this ansatz requires a smaller numbers of neurons than standard affine linear neural networks. Moreover, we investigate the efficiency of this approach for clustering tasks with the MNIST data set. Similar observations are made when comparing *deep (multi-layer)* networks.

MSC: 41A30, 65XX, 68TXX

Keywords: Generalized neural network; universal approximation; convergence rates; numerical implementation and algorithm

1. Introduction

Approximation of functions with shallow (single-layer) neural networks is a classical topic of machine learning and in approximation theory. The basic mathematical problem consists in approximating a function $g: \mathbb{R}^n \to \mathbb{R}$ by neural network functions of the form

$$G(\vec{x}) := \sum_{j=1}^{N} \alpha_j \sigma\left(p_j(\vec{x})\right) \text{ where } p_j(\vec{x}) = \mathbf{w}_j^T \vec{x} + \theta_j \text{ with } \alpha_j, \theta_j \in \mathbb{R} \text{ and } \vec{x}, \mathbf{w}_j \in \mathbb{R}^n.$$
 (1.1)

Here $\sigma : \mathbb{R} \to \mathbb{R}$ is a given function, called the *activation function* and $\mathbf{w}_j \in \mathbb{R}^n$, $\theta_j \in \mathbb{R}$ and $\alpha_j \in \mathbb{R}$, j = 1, ..., N are parameters. We name functions of the form in Equation 1.1 affine

^{*}Cong Shi is the corresponding author.

linear neural networks. This approximation problem has been well studied in the literature already in the 80ties and 90ties, see for instance [21, 2, 6, 14, 17, 19], leading to the *universal approximation property* of affine linear neural networks. Later on the universal approximation property has been established for different classes of neural networks: Examples are dropout neural networks (see [25, 18]), convolutional neural networks (CNN) (see for example [27, 28]), recurrent neural networks (RNN) (see [22, 13]), networks with random nodes (see [26]), with random weights and biases (see [20, 15]) and with fixed neural network topology (see [12]).

Two classes of neural network are of particular importance for this work: In [24], the authors introduced paraboloid neurons and illustrate their efficiency in comparison with conventional affine linear neural networks in a number of applications. In [11], the authors proposed circular neurons and deep quadratic networks. The approaches of [24, 11] are conceptually similar to the idea of this paper, where we replace the affine linear functions $\{p_j\}$ by quadratic polynomials, leading to quadratic neural network functions of the form

$$G(\vec{x}) := \sum_{j=1}^{N} \alpha_j \sigma \left(\vec{x}^T A_j \vec{x} + \mathbf{w}_j^T \vec{x} + \theta_j \right) \text{ with } \alpha_j, \theta_j \in \mathbb{R}, \mathbf{w}_j \in \mathbb{R}^n \text{ and } A_j \in \mathbb{R}^{n \times n}.$$
 (1.2)

In comparison, neural networks considered here are shallow (meaning that they have only a few numbers of layers) the networks from [11] can, theoretically, have an infinite number of layers. The paraboloid neurons from [24] are a subset of the quadratic neurons.

Clearly, the functions from Equation 1.2 represent a more general class of function then shallow affine linear neural networks, and therefore it might be expected that the number of nodes N for an approximation of a function g might be lower than for an affine linear neural network as in Equation 1.1, which is indeed true as we show numerically in Section 5. In particular we show numerically that a shallow quadratic neural network can even be as efficient as a deep affine linear neural network. We essentially base our convergence (rates) analysis of approximation properties of quadratic neural networks on the fundamental results of [19, 8, 23]. In [23] they concentrate on analyzing 4-layer affine linear neural networks (which is already considered deep): For comparison purposes, in our numerical examples, we therefore concentrate mainly on 3-layer (these are actually termed shallow) quadratic neural networks. 1

Particular achievements of our paper are as follows:

- We highlight that quadratic neural networks can be implemented relative easily in *Tensor-Flow* [1] and *Keras* [4] by *customized* layers.
- Compared with [23] the number of both layers and neurons in our case is lower because of the quadratic neurons: they used a 4-layer network with a total of at least (8d + 2)N ordinary linear neurons in it for the same approximation level.
- Furthermore, the original version of Theorem 4.4 from [23] has been applied to prove convergence for 4-layer networks. Their network deals with a manifold setting and the first layer is responsible to determine compact atlas maps; see Figure 2, where the left image corresponds to [23, Figure 3]. However, the compact atlas is essential in their analysis, which is related to the fact that affine linear neurons of the form $\vec{x} \to \varphi(\vec{x}) := C_d \sigma(\omega^T \vec{x} + \theta)$ cannot satisfy item (iii) in Definition A.1, which is $\int_{\mathbb{R}^n} \varphi(\vec{x}) d\vec{x} = 1$, and in turn the results from [8] cannot be applied in free space \mathbb{R}^n , but of course it applies, when it is constrained to a compact set, which is the case for some quadratic functions.

The paper presents a proof of concept and thus we restrict attention only to quadratic neural networks although generalizations to higher order neural networks (such as cubic) is quite straightforward.

¹In this paper we make a count of numbers of layers as in [23]: Analogously we refer to an affine linear L-layer network when it consists of input and output layers and L-2 hidden (inner) layers.

2. Generalized universal approximation theorem

In this section we review the *universal approximation theorem* as formulated by [6] and prove a generalization. To this end we also need to introduce some elementary definitions and notation:

Notation 2.1 (Vectors) For two integer numbers $m, n \in \mathbb{N}$ we always assume that $m \geq n$. Line vectors in \mathbb{R}^m and \mathbb{R}^n are denoted by

$$\mathbf{w} = (w^{(1)}, w^{(2)}, \dots, w^{(m)})^T$$
 and $\vec{x} = (x_1, x_2, \dots, x_n)^T$, respectively.

The same notation will apply to functions: \mathbf{f}, \vec{f} are m, n-dimensional vector valued functions, respectively.

Notation 2.2 (\mathcal{L}^1 space) Define the norm following from Equation (1.10) in [3]

$$||f||_{\mathcal{L}^1} := \inf\{\sum_{g \in D} |c_g||f = \sum_{g \in D} c_g g\},\$$

where c_g are the coefficients of the wavelet expansion and D is the set of wavelet functions. Notice that the notation \mathcal{L}^1 does not refer to the common L^1 -function space and depends on the choice of the wavelet system. For more properties and details on this space see [23, Remark 3.11].

Definition 2.3 (Discriminatory function) Let $\mathcal{I}_n = [0,1]^n$ denote the closed *n*-dimensional unit-cube. A function $\sigma : \mathbb{R} \to \mathbb{R}$ is called *discriminatory* if every measure μ on \mathcal{I}_n , which satisfies

$$\int_{\mathcal{I}_n} \sigma(\vec{w}^T \vec{x} + \theta) \, d\mu(\vec{x}) = 0 \quad \text{ for all } \vec{w} \in \mathbb{R}^n \text{ and } \theta \in \mathbb{R}$$

implies that $\mu \equiv 0$.

Note that every non-polynomial function is discriminatory (this follows from the results in [17]).

Example 2.4 The sigmoid function, defined by $\sigma(t) = \frac{1}{1+e^{-t}}$ for all $t \in \mathbb{R}$, is discriminatory for the Lebesgue-measure.

With these basic concepts we are able to recall Cybenko's universal approximation result.

Theorem 2.5 ([6]) Let $\sigma : \mathbb{R} \to \mathbb{R}^+$ be a continuous discriminatory function. Then, for every function $g \in C(\mathcal{I}_n)$ and every $\epsilon > 0$, there exists a function

$$G_{\epsilon}(\vec{x}) = \sum_{j=1}^{N} \alpha_{j} \sigma(\vec{w}_{j}^{T} \vec{x} + \theta_{j}) \qquad \text{with } N \in \mathbb{N}, \alpha_{j}, \theta_{j} \in \mathbb{R}, \vec{w}_{j} \in \mathbb{R}^{n},$$
 (2.1)

satisfying

$$|G_{\epsilon}(\vec{x}) - g(\vec{x})| < \epsilon \text{ for all } \vec{x} \in \mathcal{I}_n.$$

In the following we formulate and prove a generalization of Cybenko's result, which requires again some elementary definitions:

Definition 2.6 (*m*-dimensional universal approximation functions) Let $f^{(1)}$, $f^{(2)}$, \cdots , $f^{(m)} \in C(\mathcal{I}_n)$, and denote $\mathbf{f}^T := (f^{(1)}, f^{(2)}, ..., f^{(m)})$. Then we call

$$\mathcal{D} := \mathcal{D}(\mathbf{f}) := \left\{ \vec{x} \to \mathbf{w}^T \mathbf{f}(\vec{x}) + \theta : \mathbf{w} \in \mathbb{R}^m, \theta \in \mathbb{R} \right\}$$
(2.2)

the set of decision functions associated to \mathbf{f}^T .

Theorem 2.7 (Generalized universal approximation theorem) Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuous discriminatory function and assume that $\mathbf{f} : \mathcal{I}_n \to \mathbb{R}^m$ is injective (this in particular means that $n \leq m$) and continuous.

Then for every $g \in C(\mathcal{I}_n)$ and every $\epsilon > 0$ there exists some function

$$G_{\epsilon}^{\mathbf{f}}(\vec{x}) := \sum_{j=1}^{N} \alpha_{j} \sigma \left(\mathbf{w}_{j}^{T} \mathbf{f}(\vec{x}) + \theta_{j} \right) \text{ with } \alpha_{j}, \theta_{j} \in \mathbb{R} \text{ and } \mathbf{w}_{j} \in \mathbb{R}^{m}$$

$$(2.3)$$

satisfying

$$\left| G_{\epsilon}^{\mathbf{f}}(\vec{x}) - g(\vec{x}) \right| < \epsilon \text{ for all } \vec{x} \in \mathcal{I}_n.$$

Proof We begin the proof by noting that since $\vec{x} \to \mathbf{f}(\vec{x})$ is injective (The injectivity of the continuous function \mathbf{f} follows from invariance of domain, see e.g. [7, Theorem 4.3].), the inverse function on the range of \mathbf{f} is well-defined, and we write $\mathbf{f}^{-1}: \mathbf{f}(\mathcal{I}_n) \subseteq \mathbb{R}^m \to \mathcal{I}_n \subseteq \mathbb{R}^n$.

The proof that \mathbf{f}^{-1} is continuous relies on the fact that the domain $[0,1]^n$ of \mathbf{f} is compact, see for instance [10, Chapter XI, Theorem 2.1]. Then applying the Tietze-Urysohn-Brouwer extension theorem (see [16]) to the continuous function $g \circ \mathbf{f}^{-1} : \mathbf{f}(\mathcal{I}_n) \to \mathbb{R}$, this can be extended continuously to \mathbb{R}^m . This extension will be denoted by $g^* : \mathbb{R}^m \to \mathbb{R}$.

We apply Theorem 2.5 to conclude that there exist $\alpha_j, \theta_j \in \mathbb{R}$ and $\mathbf{w}_j \in \mathbb{R}^m$, j = 1, ..., N such that

$$G^*(\mathbf{z}) := \sum_{j=1}^N \alpha_j \sigma(\mathbf{w}_j^T \mathbf{z} + \theta_j) \text{ for all } \mathbf{z} \in \mathbb{R}^m, \theta_j \in \mathbb{R},$$

which satisfies

$$|G^*(\mathbf{z}) - g^*(\mathbf{z})| < \epsilon \text{ for all } \mathbf{z} \in \mathbb{R}^m.$$
 (2.4)

Then, because **f** maps into \mathbb{R}^m we conclude, in particular, that

$$G^*(\mathbf{f}(\vec{x})) = \sum_{j=1}^N \alpha_j \sigma(\mathbf{w}_j^T \mathbf{f}(\vec{x}) + \theta_j) \text{ and } |G^*(\mathbf{f}(\vec{x})) - g(\vec{x})| = |G^*(\mathbf{f}(\vec{x})) - g^*(\mathbf{f}(\vec{x}))| < \epsilon.$$

Therefore $G_{\epsilon}^{\mathbf{f}}(\cdot) := G^{*}(\mathbf{f}(\cdot))$ satisfy the claimed assertions.

3. Universal approximation theorem with quadratic functions

In the following we introduce several classes of universal approximation functions as defined in Definition 2.6.

First, we observe that the definition of decision functions from Definition 2.6 generalizes the affine linear decision functions from [6] (see Theorem 2.5):

Example 3.1 (Affine linear decision functions) Let n = m and $f^{(i)}(\vec{x}) = x_i$ for all i = 1, ..., n. Then the set of decision functions is given by

$$\mathcal{D}_l := \left\{ \vec{x} \in \mathcal{I}_n \to \vec{w}^T \vec{x} + \theta : \vec{w} \in \mathbb{R}^n, \theta \in \mathbb{R} \right\}.$$

Note, that in this case our notation gives $\mathbf{w} = \vec{w}$ and $\mathbf{x} = \vec{x}$.

In the following we consider different kinds of quadratic functions:

Definition 3.2 (Quadratic decision functions) Let m = n + 1 and let

$$A = U \operatorname{diag}(\sigma_1, \dots, \sigma_n) V^T \in \mathbb{R}^{n \times n}$$

the singular value decomposition of A. The functions

$$f^{(i)}(\vec{x}) = x_i \text{ for } i = 1, \dots, n \quad \text{ and } \quad f^{(n+1)}(\vec{x}) = \vec{x}^T A \vec{x}$$
 (3.1)

define the quadratic decision functions associated to A. The set of such is denoted by

$$\mathcal{D} := \mathcal{D}(A) := \left\{ \vec{x} \to \mathbf{w}^T \mathbf{f}(\vec{x}) + \theta : \mathbf{w} \in \mathbb{R}^{n+1}, \theta \in \mathbb{R} \right\}. \tag{3.2}$$

Then if

- for all $i, \sigma_i \geq 0$ or for all $i, \sigma_i \leq 0$, then \mathcal{D} is called the set of **elliptic** decision functions. In particular, if for all $i, \sigma_i = 1$ and U = V = I, the unitary matrix, then \mathcal{D} is called the set of **circular** decision functions.
- If all but one σ_i have the same sign, and are all not equal to 0, then \mathcal{D} is called the set of **hyperbolic** decision functions, and
- if all $\sigma_i \neq 0$ and more than two σ_i have positive and negative signs, respectively, then \mathcal{D} is called the set of **ultrahyperbolic** decision functions.
- If exactly one $\sigma_i = 0$, and all others have the same sign, then \mathcal{D} is called the set of **parabolic** decision functions.

Remark 1 • Let A = 0 be the null-matrix, then the quadratic decision functions associated to A are the affine linear decision functions.

• For every matrix $A \in \mathbb{R}^{n \times n}$ we have

$$\mathcal{D}_l \subseteq \mathcal{D}(A). \tag{3.3}$$

• Consider a quadratic decision function with

$$A = \operatorname{diag}(a_1, a_2, ..., a_n) \quad \text{where } a_i \in \mathbb{R}, a_i \neq 0.$$
(3.4)

Note that m = n + 1. If $w^{(m)} \neq 0$ we define $\zeta_i := -\frac{w^{(i)}}{2a_i^2w^{(m)}}$, for $i = 1, \ldots, n$, $\zeta = (\zeta_1, \ldots, \zeta_n)^T$ and $\nu := w^{(m)}\zeta^T A \zeta - \theta$. Consequently, the decision function can be written as

$$\mathbf{w}^{T}\mathbf{f}(\vec{x}) + \theta = w^{(m)}\vec{x}^{T}A\vec{x} + \sum_{i=1}^{n} w^{(i)}x_{i} + \theta$$

$$= w^{(m)} \left(\vec{x}^{T}A\vec{x} - 2\sum_{i=1}^{n} \zeta_{i}a_{i}^{2}x_{i} + \zeta^{T}A\zeta\right) - w^{(m)}\zeta^{T}A\zeta + \theta$$

$$= w^{(m)} \|\vec{x} - \zeta\|_{A}^{2} - \nu.$$
(3.5)

Since the set of affine linear decision functions is always a subset of the quadratic decision functions (see Equation 3.3) the following result follows from an application of Theorem 2.7 taking into account that the function **f** defined in Equation 3.1 is injective.

Corollary 3.3 (Universal approximation of quadratic decision functions) Let m = n + 1, $A \in \mathbb{R}^{n \times n}$ and let \mathbf{f} be as defined in Equation 3.1. Suppose that the discriminatory function $\sigma : \mathbb{R} \to \mathbb{R}_+$ is Lipschitz continuous with Lipschitz constant λ . Then for every $g \in C(\mathcal{I}_n)$ and every $\epsilon > 0$ there exists some $N \in \mathbb{N}$ and some function

$$\vec{x} \in \mathcal{I}_n \to G_{\epsilon}^{\mathbf{f}}(\vec{x}) := \sum_{j=1}^N \alpha_j \sigma\left(\mathbf{w}_j^T \mathbf{f}(\vec{x}) + \theta_j\right) \text{ with } \alpha_j \in \mathbb{R}, \mathbf{w}_j \in \mathbb{R}^m \text{ and } \theta_j \in \mathbb{R}$$
 (3.6)

satisfying

$$\left| G_{\epsilon}^{\mathbf{f}}(\vec{x}) - g(\vec{x}) \right| < \epsilon \text{ for all } \vec{x} \in \mathcal{I}_n.$$

Note that the assumption that σ is Lipschitz continuous is needed in the proof of Corollary 3.3.

As it is presented here, the universal approximation Theorem 2.5 and Corollary 3.3 provide the existence of an approximating sequence for increasing N. The proof is not quantitative and only applicable for functions $g \in C(\mathcal{I}_n)$, that is for functions defined on the n-dimensional unit cube. The following section provides convergence rates results for the best approximation function with N coefficients. On a technical level, it allows for approximating functions $g \in L^1(\mathbb{R}^n)$, that is in free space.

4. Convergence rates for universal approximation of circular decision functions

In the following we prove convergence rates of circular decision functions of the form $G_{\epsilon}^{\mathbf{f}}$ in the \mathcal{L}^1 -norm. We recall that by construction, circular decision functions form a superset of the affine linear decision functions, and this generalization allows for more efficient approximations.

We follow the proof of convergence rates results from [23] for affine linear decision functions and extend it to *circular decision functions* in the following way:

- (i) We construct a wavelet frame from the set of circular decision functions $\mathcal{D}(I)$;
- (ii) We apply general convergence rates for wavelet expansions to prove convergence rates of the **best approximation** with respect to the circular frame of an arbitrary function $g \in \mathcal{L}^1(\mathbb{R}^n)$.

For the sake of simplicity of presentation we avoid a presentation of general elliptic decision functions.

Definition 4.1 (Circular wavelet frame) Let r > 0 and σ is a discriminatory function as defined in Definition 2.3 such that $\int_{\mathbb{R}^n} \sigma(r^2 - \|\vec{x}\|^2) d\vec{x} < \infty$. Then let

$$\vec{x} \in \mathbb{R}^n \to \varphi(\vec{x}) := C_d \sigma(r^2 - x_1^2 - x_2^2 - \dots - x_n^2),$$
 (4.1)

where C_d is a normalizing constant such that $\int_{\mathbb{R}^n} \varphi(\vec{x}) d\vec{x} = 1$.

Then we define for all $\vec{x}, \vec{y} \in \mathbb{R}^n$ and $k \in \mathbb{Z}$

$$S_k(\vec{x}, \vec{y}) := 2^k \varphi(2^{\frac{k}{n}}(\vec{x} - \vec{y})) \text{ and } \psi_{k, \vec{y}}(\vec{x}) := 2^{-\frac{k}{2}} (S_k(\vec{x}, \vec{y}) - S_{k-1}(\vec{x}, \vec{y})). \tag{4.2}$$

Remark 2 We abbreviate $\psi := \psi_{0,0}$. With this notation we see that

$$\psi_{k,\vec{y}}(\vec{x}) = 2^{-\frac{k}{2}} (S_k(\vec{x}, \vec{y}) - S_{k-1}(\vec{x}, \vec{y}))$$

$$= 2^{-\frac{k}{2}} \left(2^k \varphi(2^{\frac{k}{n}} (\vec{x} - \vec{y})) - 2^{k-1} \varphi(2^{\frac{k-1}{n}} (\vec{x} - \vec{y})) \right)$$

$$= 2^{\frac{k}{2}} \left(\varphi\left(2^{\frac{k}{n}} \vec{x} - 2^{\frac{k}{n}} \vec{y}\right) - 2^{-1} \varphi\left(2^{-\frac{1}{n}} \left(2^{\frac{k}{n}} \vec{x} - 2^{\frac{k}{n}} \vec{y}\right)\right) \right)$$

$$= 2^{\frac{k}{2}} \psi(2^{\frac{k}{n}} (\vec{x} - \vec{y})) \quad \text{for all } k \in \mathbb{Z}, \vec{y} \in \mathbb{R}^n.$$

For proving that \mathcal{F} satisfies the frame properties (see for instance [5]) and approximation properties of the best approximation with respect to the frames expansion we apply some general results from the literature, which are reviewed in the Appendix A.

4.1. Convergence rates of nets of circular decision function. We show that the circular wavelet frame is an *Approximation of the identity* (AtI (see Definition A.1)). For this purpose we use the following basic inequality.

Lemma 4.2 Let $h : \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function, which can be expressed in the following way:

$$h(\vec{x}) = h_s(\|\vec{x}\|^2)$$
 for all $\vec{x} \in \mathbb{R}^n$.

Then the spectral norm of the Hessian of h can be estimated as follows:²

$$\|\nabla^{2}h(\vec{x})\| \leq \max\left\{ \left| 4\|\vec{x}\|^{2} h_{s}''(\|\vec{x}\|^{2}) + 2h_{s}'(\|\vec{x}\|^{2}) \right|, \left| 2h_{s}'(\|\vec{x}\|^{2}) \right| \right\}. \tag{4.3}$$

Proof Since $\nabla^2 h(\vec{x})$ is a symmetric matrix, its operator norm is equal to its spectral radius, namely the largest absolute value of an eigenvalue. By routine calculation we can see that

$$\nabla_{x_i x_j} h(\vec{x}) = 4x_i x_j h_s''(\|\vec{x}\|^2) + 2\delta_{ij} h_s'(\|\vec{x}\|^2).$$

Let $C = (x_i x_j)$ and I the identity matrix, then λ is an eigenvalue with eigenvector \vec{z} of $\nabla^2 h(\vec{x})$ if and only if

$$4h_s''(\|\vec{x}\|^2)C\vec{z} = (-2h_s'(\|\vec{x}\|^2) + \lambda)\vec{z}.$$

Or in other words $\frac{-2h_s'(\|\vec{x}\|^2)+\lambda}{4h_s''(\|x\|^2)}$ is an eigenvalue of C. Moreover, $C=\vec{x}\vec{x}^T$ is a rank one matrix and thus the spectral values are 0 with multiplicity (n-1) and $\|\vec{x}\|^2$. This in turn shows that the eigenvalues of the Hessian are $+2h_s'(\|\vec{x}\|^2)$ (with multiplicity n-1) and $4\|\vec{x}\|^2h_s''(\|\vec{x}\|^2)+2h_s'(\|\vec{x}\|^2)$, which proves Equation 4.3.

In the following lemma, we will prove that the kernels $(S_k)_{k\in\mathbb{Z}}$ are an AtI (Approximation to the identity [8]). This is a streamlined assumption from Definition 3.4 in the book [8].

Lemma 4.3 Suppose that the activation function $\sigma : \mathbb{R} \to \mathbb{R}_+$ is monotonically increasing and satisfies for the *i*-th derivative (i = 0, 1, 2)

$$\left|\sigma^{i}(r^{2}-t^{2})\right| \leq C_{\sigma}(1+\left|t\right|^{n})^{-1-\frac{2i+1}{n}} \text{ for all } t \in \mathbb{R},$$
 (4.4)

where r is the same as in Definition 4.1. Then the kernels $(S_k)_{k\in\mathbb{Z}}$ as defined in Equation 4.2 form an AtI as defined in Definition A.1 that also satisfy Equation A.4.

Proof We verify the three conditions from Definition A.1 as well as Equation A.4. First of all, we note that

$$\left| \sigma^{i}(r^{2} - \|\vec{x}\|^{2}) \right| \le C_{\sigma}(1 + \|\vec{x}\|^{n})^{-1 - \frac{2i+1}{n}} \text{ for all } \vec{x} \in \mathbb{R}^{n}.$$
 (4.5)

• Verification of item (i) in Definition A.1: Equation 4.1 and Equation 4.4 imply that

$$0 \le \varphi(\vec{x} - \vec{y}) = C_d \sigma(r^2 - \|\vec{x} - \vec{y}\|^2) \le C_\sigma C_d (1 + \|\vec{x} - \vec{y}\|^n)^{-1 - \frac{1}{n}} \text{ for all } \vec{x}, \vec{y} \in \mathbb{R}^n.$$
 (4.6)

Therefore

$$S_k(\vec{x}, \vec{y}) = 2^k \varphi(2^{\frac{k}{n}} (\vec{x} - \vec{y})) \le C_\sigma C_d 2^k (1 + 2^k ||\vec{x} - \vec{y}||^n)^{-1 - \frac{1}{n}}$$
$$= C_\sigma C_d 2^{-\frac{k}{n}} (2^{-k} + ||\vec{x} - \vec{y}||^n)^{-1 - \frac{1}{n}}.$$

Thus item (i) in Definition A.1 holds with $\epsilon = 1/n$ and $C_{\rho} = 1$ and $C = C_d C_{\sigma}$.

²In the following ∇ and ∇ ² (without subscripts) always denote derivatives with respect to an *n*-dimensional variable such as \vec{x} . ' and " denotes derivatives of a one-dimensional function.

• Verification of item (ii) in Definition A.1 with $C_{\rho} = 1$ and $C_A = 2^{-n}$: Because σ is monotonically increasing it follows from Equation 4.1 and the fact that $S_0(\vec{x}, \vec{y}) = \varphi(\vec{x} - \vec{y})$ (see Equation 4.2) and Definition 4.1 that

$$F_{\vec{y}}(\vec{x}) := \|\nabla_{\vec{x}}(S_0(\vec{x}, \vec{y}))\| = 2C_d \|\vec{x} - \vec{y}\| \sigma'(r^2 - \|\vec{x} - \vec{y}\|^2) \text{ for all } \vec{y} \in \mathbb{R}^n.$$

Then Equation 4.5 implies that

$$F_{\vec{y}}(\vec{x}) \le 2C_d C_\sigma (1 + \|\vec{x} - \vec{y}\|^n)^{-1 - \frac{3}{n}} \|\vec{x} - \vec{y}\| \le 2C_d C_\sigma (1 + \|\vec{x} - \vec{y}\|^n)^{-1 - \frac{3}{n}} (1 + \|\vec{x} - \vec{y}\|^n)^{\frac{1}{n}}$$

$$= 2C_d C_\sigma (1 + \|\vec{x} - \vec{y}\|^n)^{-1 - \frac{2}{n}}.$$

From the definition of $S_k(\vec{x}, \vec{y})$, it follows

$$\|\nabla_{\vec{x}}(S_{k}(\vec{x}, \vec{y}))\| = \|\nabla_{\vec{x}}(2^{k}\varphi(2^{\frac{k}{n}}(\vec{x} - \vec{y})))\| = 2^{k} \|\nabla_{\vec{x}}S_{0}(2^{\frac{k}{n}}\vec{x}, 2^{\frac{k}{n}}\vec{y})\| = 2^{k + \frac{k}{n}}F_{2^{\frac{k}{n}}\vec{y}}(2^{\frac{k}{n}}\vec{x})$$

$$\leq 2^{-\frac{k}{n}}C_{d}C_{\sigma}(2^{-k} + \|\vec{x} - \vec{y}\|^{n})^{-1 - \frac{2}{n}}.$$

$$(4.7)$$

From the mean value theorem it therefore follows from Equation 4.7 and Equation A.6 that

$$\frac{|S_{k}(\vec{x}, \vec{y}) - S_{k}(\vec{x}', \vec{y})|}{\|\vec{x} - \vec{x}'\|} \leq \max_{\{\vec{z} = t\vec{x}' + (1-t)\vec{x}: t \in [0,1]\}} \|\nabla_{\vec{x}}(S_{k}(\vec{z}, \vec{y}))\|
\leq 2^{-\frac{k}{n}} C_{d} C_{\sigma} \max_{\{\vec{z} = \vec{x} + t(\vec{x}' - \vec{x}): t \in [0,1]\}} (2^{-k} + \|\vec{z} - \vec{y}\|^{n})^{-1 - \frac{2}{n}}
= 2^{-\frac{k}{n}} C_{d} C_{\sigma} \left(2^{-k} + \min_{\{\vec{z} = \vec{x} + t(\vec{x}' - \vec{x}): t \in [0,1]\}} \|\vec{z} - \vec{y}\|^{n}\right)^{-1 - \frac{2}{n}}.$$
(4.8)

Then application of Equation A.6 and noting that $\frac{1-2^{-n}}{2^{-n}} \ge 1$ gives

$$\frac{|S_k(\vec{x}, \vec{y}) - S_k(\vec{x}', \vec{y})|}{\|\vec{x} - \vec{x}'\|} \le 2^{-\frac{k}{n}} C_d C_\sigma \left((1 - 2^{-n}) 2^{-k} + 2^{-n} \|\vec{x} - \vec{y}\|^n \right)^{-1 - \frac{2}{n}}
\le 2^{-\frac{k}{n}} \left(2^{-n} \right)^{-1 - \frac{2}{n}} C_d C_\sigma \left(\frac{(1 - 2^{-n})}{2^{-n}} 2^{-k} + \|\vec{x} - \vec{y}\|^n \right)^{-1 - \frac{2}{n}}
\le 2^{-\frac{k}{n}} 2^{n+2} C_d C_\sigma \left(2^{-k} + \|\vec{x} - \vec{y}\|^n \right)^{-1 - \frac{2}{n}}.$$

Therefore item (ii) is satisfied with $C_{\rho} = 1$, $\zeta = 1/n$, $\epsilon = 1/n$ and $C = 2^{n+2}C_dC_{\sigma}$.

• Verification of item (iii) in Definition A.1: From the definition of S_k (see Equation 4.2) it follows that for every $k \in \mathbb{Z}$ and $\vec{y} \in \mathbb{R}^n$

$$1 = \int_{\mathbb{R}^n} S_k(\vec{x}, \vec{y}) d\vec{x} = \int_{\mathbb{R}^n} 2^k \varphi(2^{\frac{k}{n}} (\vec{x} - \vec{y})) d\vec{x}.$$

• Verification of the double Lipschitz condition Equation A.4 in Definition A.1: By using the integral version of the mean value theorem, we have

$$\begin{split} S_{k}(\vec{x}, \vec{y}) - S_{k}(\vec{x}', \vec{y}) - S_{k}(\vec{x}, \vec{y}') + S_{k}(\vec{x}', \vec{y}') \\ &= S_{k}(\vec{x}, \vec{y}) - S_{k}(\vec{x}', \vec{y}) - (S_{k}(\vec{x}, \vec{y}') - S_{k}(\vec{x}', \vec{y}')) \\ &= \int_{0}^{1} \langle \nabla_{\vec{y}} S_{k}(\vec{x}, \vec{y}' + t(\vec{y} - \vec{y}')), \vec{y} - \vec{y}' \rangle dt - \int_{0}^{1} \langle \nabla_{\vec{y}} S_{k}(\vec{x}', \vec{y}' + t(\vec{y} - \vec{y}')), \vec{y} - \vec{y}' \rangle dt \\ &= \int_{0}^{1} \int_{0}^{1} \langle \nabla_{\vec{x}, \vec{y}} S_{k}(\vec{x}' + s(\vec{x} - \vec{x}'), \vec{y}' + t(\vec{y} - \vec{y}'))(\vec{x} - \vec{x}'), \vec{y} - \vec{y}' \rangle dt ds. \end{split}$$

Following this identity, we get

$$\frac{|S_{k}(\vec{x}, \vec{y}) - S_{k}(\vec{x}', \vec{y}) + S_{k}(\vec{x}, \vec{y}') - S_{k}(\vec{x}', \vec{y}')|}{\|\vec{x} - \vec{x}'\| \|\vec{y} - \vec{y}'\|} \le \max_{\mathbf{I}} \left\| \nabla_{\vec{y}} \left(\frac{S_{k}(\vec{x}, \vec{z}) - S_{k}(\vec{x}', \vec{z})}{\|\vec{x} - \vec{x}'\|} \right) \right\| \\
\le \max_{\mathbf{I}} \left\| \nabla_{\vec{x}\vec{y}}^{2} S_{k}(\vec{z}', \vec{z}) \right\|, \tag{4.9}$$

where

$$\begin{split} \mathbf{I} &:= \left\{ \vec{z} = t \vec{y} + (1-t) \vec{y}' : t \in [0,1] \right\}, \\ \mathbf{Q} &:= \left\{ (\vec{z} = t_{\vec{y}} \vec{y} + (1-t_{\vec{y}}) \vec{y}', \vec{z}' = t_{\vec{x}} \vec{x} + (1-t_{\vec{x}}) \vec{x}') : t_{\vec{y}} \in [0,1], t_{\vec{x}} \in [0,1] \right\}, \end{split}$$

and $\left\|\nabla_{\vec{x}\vec{y}}^2 S_k(\vec{z}', \vec{z})\right\|$ denotes again the spectral norm of $\nabla_{\vec{x}\vec{y}}^2 S_k(\vec{z}', \vec{z})$.

Now, we estimate the right hand side of Equation 4.9: From the definition of S_k , Equation 4.2, and the definition of φ , Equation 4.1, it follows with the abbreviation $\vec{\omega} = 2^{\frac{k}{n}} (\vec{z}' - \vec{z})$:

$$\left\| \nabla^2_{\vec{x}\vec{y}} S_k(\vec{z}', \vec{z}) \right\| = 2^k \left\| \nabla^2_{\vec{x}\vec{y}} (\varphi \circ (2^{\frac{k}{n}} \cdot)) (\vec{z}' - \vec{z})) \right\| = 2^{k + 2\frac{k}{n}} \left\| \nabla^2 \varphi(\vec{\omega}) \right\|$$

Applications of Lemma 4.2 with $\vec{x} \to h(\vec{x}) = \varphi(\vec{x})$ and $t \to h_s(t) = C_d \sigma(r^2 - t)$ shows that (note that $h'_s(t) = -C_d \sigma'(r^2 - t)$)

$$\|\nabla^{2}\varphi(\vec{\omega})\| \leq C_{d} \max \left\{ \left| 4 \|\vec{\omega}\|^{2} \sigma''(r^{2} - \|\vec{\omega}\|^{2}) - 2\sigma'(r^{2} - \|\vec{\omega}\|^{2}) \right|, \left| 2\sigma'(r^{2} - \|\vec{\omega}\|^{2}) \right| \right\}$$

$$\leq 2^{2} C_{d} \max \left\{ 2 \|\vec{\omega}\|^{2} \left| \sigma''(r^{2} - \|\vec{\omega}\|^{2}) \right|, \left| \sigma'(r^{2} - \|\vec{\omega}\|^{2}) \right| \right\}.$$

$$(4.10)$$

Thus from Equation 4.5 it follows that

$$\begin{split} \left\| \nabla_{\vec{x}\vec{y}}^{2} S_{k}(\vec{z}', \vec{z}) \right\| &\leq 2^{2} 2^{k+2\frac{k}{n}} C_{d} C_{\sigma} \max \left\{ 2 \|\vec{\omega}\|^{2} \left(1 + \|\vec{\omega}\|^{n} \right)^{-1 - \frac{5}{n}}, \left(1 + \|\vec{\omega}\|^{n} \right)^{-1 - \frac{3}{n}} \right\} \\ &\leq 2^{3} 2^{k+2\frac{k}{n}} C_{d} C_{\sigma} (1 + \|\vec{\omega}\|^{n})^{-1 - \frac{3}{n}} \\ &\leq 2^{-\frac{k}{n}} 2^{3} C_{d} C_{\sigma} (2^{-k} + \|\vec{z}' - \vec{z}\|^{n})^{-1 - \frac{3}{n}}. \end{split}$$

In the next step we note that from Equation A.7 it follows that

$$\|\vec{z}' - \vec{z}\|^n \ge 3^{-n} \|\vec{x} - \vec{y}\|^n - 3^{-n}2^{1-k}.$$
 (4.11)

Thus we get because $\frac{1-3^{-n}2}{3^{-n}} \ge 1$

$$\frac{|S_{k}(\vec{x}, \vec{y}) - S_{k}(\vec{x}', \vec{y}) - S_{k}(\vec{x}, \vec{y}') + S_{k}(\vec{x}', \vec{y}')|}{\|\vec{x} - \vec{x}'\| \|\vec{y} - \vec{y}'\|} \\
\leq 2^{-\frac{k}{n}} 2^{3} C_{d} C_{\sigma} \left((1 - 3^{-n} 2) 2^{-k} + 3^{-n} \|\vec{x} - \vec{y}\|^{n} \right)^{-1 - \frac{3}{n}} \\
\leq 2^{-\frac{k}{n}} 2^{3} (3^{-n})^{-1 - \frac{3}{n}} C_{d} C_{\sigma} \left(\frac{1 - 3^{-n} 2}{3^{-n}} 2^{-k} + \|\vec{x} - \vec{y}\|^{n} \right)^{-1 - \frac{3}{n}} \\
\leq 2^{-\frac{k}{n}} 2^{3} 3^{n+3} C_{d} C_{\sigma} \left(2^{-k} + \|\vec{x} - \vec{y}\|^{n} \right)^{-1 - \frac{3}{n}}.$$

Therefore item (iii) is satisfied with $C_{\rho} = 1$, $\tilde{C} = 2^3 3^{n+3} C_d C_{\sigma}$, $\zeta = 1/n$, and $\epsilon = 1/n$.

Remark 3 One typical activation function which satisfies Equation 4.4 is the sigmoid function. For different orders of its derivative, Figure 1 shows this inequality in logarithmic coordinates. The orange curve corresponds to the left hand side of Equation 4.4, and the blue curve corresponds to the right hand side of it.

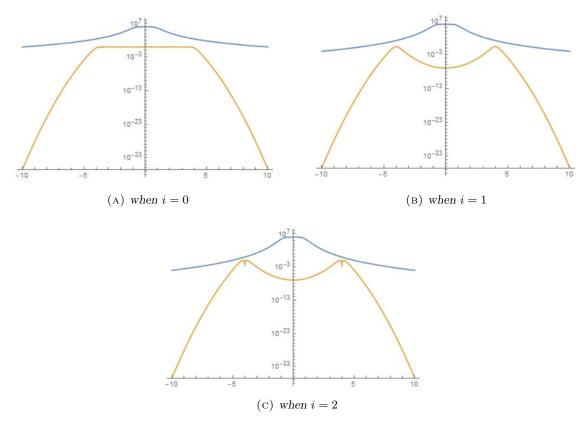


FIGURE 1. In Equation 4.4, we take $\sigma(t) = \frac{1}{1+e^{-t}}$, $C_{\sigma} = 10^{6}$, n = 5, r = 4. The x-axis is variable t, the y-axis is the logarithmic value of both sides of Equation 4.4.

By combining Theorem A.3 and Lemma 4.3, we get the following theorem:

Theorem 4.4 (\mathcal{L}^1 -convergence) Let σ be an activation function that satisfies the conditions in Lemma 4.3, and let $\psi_{k,b}$ be a frame constructed from σ by Definition 4.1. For any function $f \in \mathcal{L}^1(\mathbb{R}^n)$ and any positive integer N, there exists a function

$$f_N \in span_N(\mathcal{F}) \subseteq \mathcal{L}^1(\mathbb{R}^n) \text{ where } \mathcal{F} := \left\{ \vec{x} \in \mathbb{R}^n \to \psi_{k,b}(\vec{x}) : (k,b) \in \mathbb{Z}, b \in 2^{-\frac{k}{n}} \mathbb{Z} \right\}$$

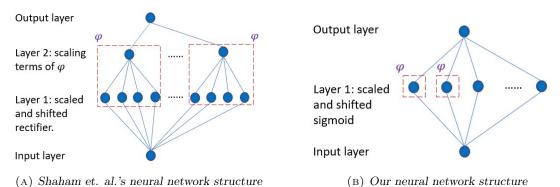
and $span_N$ denotes linear combinations of at most N terms in the set, such that

$$||f - f_N||_{L^2} \le ||f||_{\mathcal{L}^1} (N+1)^{-1/2}.$$
 (4.12)

Proof First, we note that the functions $\{S_k : k \in \mathbb{Z}\}$ are an AtI, which satisfies the double Lipschitz condition (see Definition A.1). Thus associated to Theorem A.3 \mathcal{F} is a wavelet frame. Moreover, let f_N be the wavelet approximation specified in Theorem A.3, then it satisfies Equation 4.12.

Remark 4 An original version of Theorem 4.4 has been applied to prove convergence for four layer networks (note, this means two hidden, one input and one output layer). Their network deals with a manifold setting and the first layer is responsible to determine **compact atlas maps**; see Figure 2, where the left image corresponds to [23, Figure 3]. However, the **compact atlas**

is essential in their analysis, which is related to the fact that affine linear neurons of the form $\vec{x} \to \varphi(\vec{x}) := C_d \sigma(\omega^T \vec{x} + \theta)$ cannot satisfy item *(iii)* in Definition A.1, which is $\int_{\mathbb{R}^n} \varphi(\vec{x}) d\vec{x} = 1$, and in turn the results from [8] cannot be applied in free space \mathbb{R}^n , but of course it applies, when it is constrained to a compact set.



In Lemma 4.5 in [23] also an L^{∞} approximation result is proven, which can be carried over to our setting as well.

Figure 2. Comparison and improvement

5. Numerical experiments

In this section we study numerically the approximation of functions with linear combinations of quadratic decision functions, in particular circular and hyperbolic ones, as defined in Definition 3.2. We compare the numerical results with results produced by approximation with affine linear decision functions.

Moreover, we compare deep affine linear neural networks with quadratic neural networks with shallow layer structure, that is a three-layer network (one hidden layer) (cf. right image of Figure 2).

We also compare numerically the approximation properties of deep quadratic neural networks, which is not considered theoretically here. We have chosen two simple test cases of one-dimensional functions (see Figure 3) to approximate, for which we analyze the approximation properties of different neural networks numerically.

Finally, our goal is to extend the basic proof-of-concept examples to some clusterization problems and discuss some real world application examples.

5.1. Proof-of-concept example.

5.1.1 Ground Truth Data

We study the numerical approximation of two simple test-data, which are analytically given by

$$f_1(x) = \begin{cases} 0 & x \le 0 \\ x & 0 < x \le 3 \\ 3 & 3 < x \le 5 \\ -0.4x + 5 & 5 < x \end{cases} \qquad f_2(x) = \begin{cases} 0 & x \le 0 \\ x^2 & 0 < x \le 3 \\ 9 & 3 < x \le 5 \\ 9e^{-(x-5)} & 5 < x \end{cases}$$
(5.1)

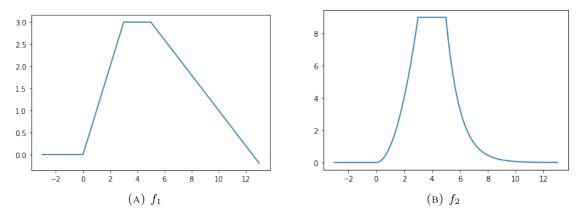


FIGURE 3. The two functions that are approximated via different types of neural networks in our numerical experiments below.

5.1.2 Generation of training data and initialization

The functions from above were evaluated in 1600 uniformly distributed points in [-3, 13]. The pairs (x, y) of generated data were then randomly split into 1072 training pairs and 528 test pairs. In the rest of the paper f_2 is observed, but the analysis has also been conducted for f_1 , where the results are similar.

5.1.3 Implementation details

The implementation is based on already implemented methods of TensorFlow [1] and Keras [4], with adaptations - where necessary - to resemble the structure of the decision functions from (3.2). The following pseudocode illustrates the general procedure and highlights custom implemented features.

The implementation in our specific setting has been done for one, three and four hidden layers, but, as the pseudocode demonstrates, it can easily be adapted to neural networks with any number of hidden layers. The Adam optimizer with a learning rate of 0.001 has been selected as the optimization method of choice. If we determine a bad initialization, it randomly chooses another one and starts again.

The following results shown in the next subsections have all been performed on a 2,4 GHz 8-Core Intel Core i9 processor with 32 GB RAM.

5.1.4 Convergence rates of shallow elliptic networks

We have varied the number of neurons N to evaluate the convergence rates proven in Theorem 4.4, which shows the predicted convergence rates of elliptic neural networks (generalization of circular decision functions, which lead to more stable results). The results can be observed in Figure 4.

Structure of an elliptical layer

```
Inputs:
   inputs ← function values of ground truth data
   input dimension \leftarrow numer of neurons of input
   output dimension \leftarrow number of neurons of output
Variables:
   \sigma_i \leftarrow weights for the 'elliptical' part (dim of matrix: input dimension \times number of neurons)
   \mathbf{w}_{1..n} \leftarrow \text{weights for the 'affine linear' part (dim of matrix: input dimension} \times \text{number of}
neurons)
   \mathbf{w}_{n+1} \leftarrow \text{weight that is multiplied with 'elliptical part' for each neuron (dim of matrix: 1)}
\times number of neurons)
   \theta \leftarrow bias for each neuron to add to 'affine linear part' (dim of matrix: 1 \times number of
neurons)
Initialize:
   \sigma_i, \mathbf{w}_{1..n}, \mathbf{w}_{n+1} \leftarrow \text{random gaussian distribution (mean=1.0, stddev=0, seed=None)}
class Elliptical Layer(input dimension, output dimension)
    function CALL(INPUTS)
        outputs \leftarrow \mathbf{w}_{1..n} \cdot \text{inputs} + \mathbf{w}_{n+1} \cdot \sigma_i^2 \cdot \text{inputs}^2 + \theta
        store outputs
        return outputs
    end function
    function BACKPROP(LABELS):
        update weights with Adam optimizer
    end function
end class
```

Stacking together different layers and training (Example for a deep neural network)

Initialize:

```
x_{train}, y_{train} \leftarrow \text{training data}
test_{epochs} \leftarrow \text{number of complete pass throughs of the training data}
layer_in \leftarrow \text{Object of Elliptic class with 1 input, } X_1 \text{ outputs}
layer_hidden_1 \leftarrow \text{Object of Elliptic class with } X_1 \text{ input, } Y_n \text{ outputs}
\vdots
layer_hidden_n \leftarrow \text{Object of Elliptic class with } X_n \text{ input, } Y_n \text{ outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 1 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 1 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 1 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 1 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 1 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 1 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 2 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 2 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 2 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 2 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 2 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 2 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 2 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 2 outputs}
layer_out \leftarrow \text{Object of Elliptic class with } Y_n \text{ input, 2 outputs}
```

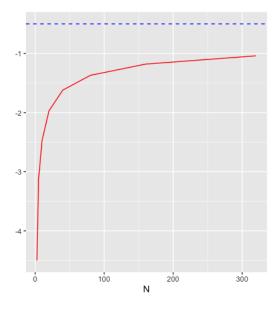


FIGURE 4. The graph depicts the convergence rates of $\frac{\log(\frac{\|S\|_{L^{2}}{\|f\|_{L^{1}}})}{\|\log(N+1)\|}$ in our numerical experiments, which are always below the upper bound given by the theoretical results. This means that actually the estimate Equation 4.12 seems to be too conservative.

In a next step, the convergence rates of elliptic neural networks should be compared with those of affine linear and hyperbolic ones. The following graph shows the development of the reconstruction each adding up 50 epochs. One epoch describes one complete pass through the training data, as in the **Keras** library. During each epoch, the weights (including σ_i) are updated.

The images in Figure 5 show us, that when one considers training for 50 epochs, only approximation via elliptic (generalization of circular) decision functions performs reasonably well. When using more epochs, hyperbolic and affine linear catches up with elliptic and have a similar approximation error (see also Table 1). Figure 6 provides us with the error function, where it is clearly observable that the elliptic layers converge faster than the affine linear and hyperbolic ones.

The MSE err	or corresponds	with the L^2	² -norm from	the theoretic	al section.

Type of network	Training epochs	Hidden layers	Units per hidden layer	Test MSE	Test MAE
Elliptic	250	1	[5]	0.0731	0.2592
	140	1	[5]	0.08795	0.25975
Affine Linear	250	1	[5]	0.2008	0.2102
	140	1	[5]	0.47645	0.30545
Hyperbolic	250	1	[5]	0.0574	0.2977
	140	1	[5]	0.2164	0.3281

Table 1. Overview of the result of the numerical experiments for shallow networks.

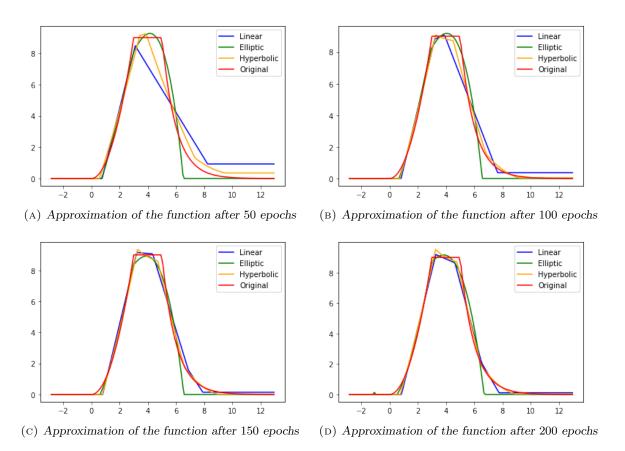


FIGURE 5. The graphs each show the original function and the approximations obtained with shallow neural networks with each 5 units per hidden layer, with gradual increase of 50 epochs each.

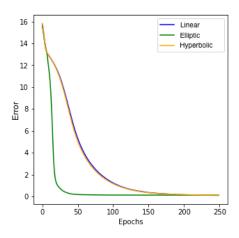


Figure 6. Here we compare the error functions (MSE) of affine linear, elliptic and hyperbolic shallow neural networks. Elliptic layers clearly converge faster than affine linear or hyperbolic layers in this setting.

Please note, that the approximation with hyperbolic and affine linear layers do not satisfy all proposed conditions on σ (see Lemma 4.3).

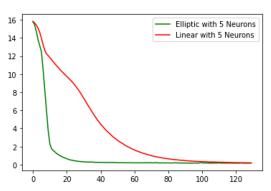
Type of network	Training epochs	Hidden layers	Units per hidden layer	Test MSE	Test MAE
Elliptic	140	3	[5, 5, 5]	0.0106	0.1752
	140	3	[30, 30, 30]	0.0014	0.0404
	140	4	[5, 5, 5, 5]	0.0528	0.0767
Affine Linear	140	4	[5,5,5,5]	0.0105	0.2074
	140	4	[30, 30, 30, 30]	0.0453	0.0618

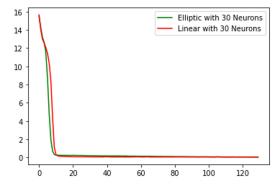
Table 2. Overview of the result of the numerical experiments for deep networks.

5.1.5 Deep networks

In this section, we present the error functions for deep neural networks with multiple hidden layers, associated to the pseudocode presented before.

The following results (both in Figure 7 and Figure 8) confirm our hypothesis, that the deep elliptic neural networks converge faster than the affine linear ones. Full results can be observed in Table 2.





- (A) Each hidden layer has 5 neurons included, here we compare a 3-layer elliptic network with a 4-layer affine linear one.
- (B) Each hidden layer has 30 neurons included, here we compare a 3-layer elliptic network with a 4-layer affine linear one.

Figure 7. A 3 hidden layer elliptic neural network converges faster than a 4 hidden layer affine linear one. In the x-axis it shows the number of epochs of the training, in the y-axis the mean squared error of the resulting approximation.

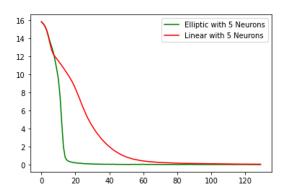


Figure 8. A 4 hidden layer elliptic neural network converges faster than a 4 hidden layer affine linear one. Each hidden layer has 5 neurons included, here we compare a 4-layer elliptic network with a 4-layer affine linear one. As before, it shows in the x-axis the number of epochs of the training, in the y-axis the mean squared error of the resulting approximation.

5.2. Clusterization examples. In this subsection, we are going to look at a more "applied setting", in detail classification problems.

We will start with the well-known MNIST dataset (see [9]), which is a database consisting out of 60,000 examples of training data as well as 10,000 examples as test data of handwritten digits. The images which were used are size-normalized and centered in a fixed-size image.

Out of these images a t-distributed stochastic neighbour embedding (t-SNE) was generated. This procedure is a machine learning algorithm for dimensionality reduction and often used for visualization purposes. t-SNE preserves local structures of the dataset by letting the distances between points stay the same.

The use case for the quadratic neural networks would be the following: By only having the different clusters, the goal would now be to correctly classify MNIST images of which only the t-SNE embedding is known.

5.2.1 Generation of training data and initialization

The training data is generated via the t-distributed stochastic neighbour of the 60,000 examples in the following one can see a visualization of it. One sees the clear groups of the different digits. For the test data set, we have generated 10,000 data points, where the correct classification shall be determined. For means of simplicity, this paper observes the correct clustering of e.g. digit 8.

To determine the correct clustering for the other digits as well, one simply generates additional networks for the classification of the other numbers. When having conducted these experiments, this has increased the performance for the other digits as well. For the generation of the neighbour clustering, we have used code available on Kaggle for the t-sne-visualization.

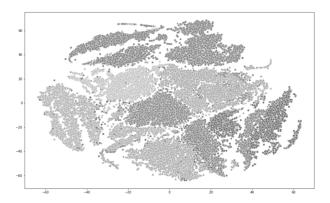


Figure 9. t-distributed stochastic neighbour clustering

5.2.2 Results

The following table indicates the advantage of using elliptic (generalization of circular) decision functions and not linear ones. When using more epochs, hyperbolic and affine linear catches up with elliptic and have a similar approximation error (see also Table 3).

We have used different combinations to observe the different outcomes (see Figure 10).

Type	Epochs	H. layers	Units per hidden layer	S. C. Crossentropy
Elliptic	10	1	[5]	0.9515
Elliptic	10	3	[5,5,5]	0.9757
Elliptic	30	3	[20,20,20]	0.9770
Affine Linear	10	1	[5]	0.9033
Affine Linear	10	3	[5,5,5]	0.9570
Affine Linear	30	3	[20, 20, 20]	0.973

Table 3. Overview of the result of the numerical experiments for deep networks.

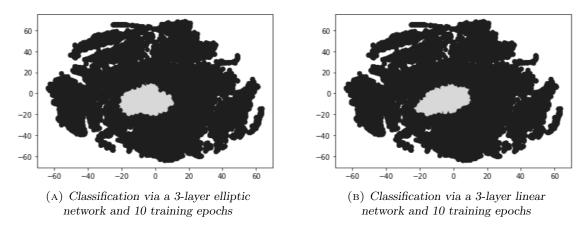


Figure 10. Comparison of elliptic with linear affine layers on MNIST dataset

5.2.3 Summary and possible extensions

Having also performed other experiments, we can clearly state that there are several cases, under which the use of elliptical layers practically makes sense. Especially, when one is interested into convergence rates, one is obliged to use a small neural network structure (or even shallow neural networks). In such cases, these specially constructed decision functions clearly have an advantage.

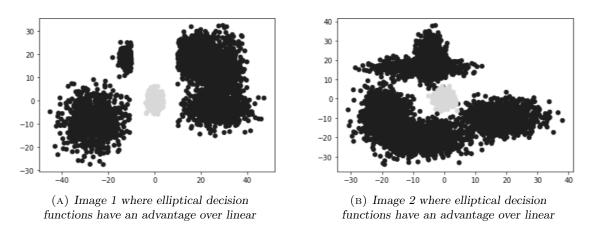


Figure 11. Goal is to correctly identify the yellow subspecies out of a bigger population

In the following analysis, we will discuss the classification on Figure 11 (B) when one sole layer (shallow neural network structure) is used. Via those two "relatively simple" examples, it lets us understand which effect the different layers particularly have from a geometrical point of view.

This we will compare with the usage of e.g. classical linear layers. The setting was done similarly as above, we have used for the experiments a shallow neural network structure with 20 epochs of training.

We remember: The subpopulation should be identified out of a much bigger population. As already performed in the previous experiment, one has specific data points given in both the populations.

One sees that the classification via an elliptical layer leads to a loss of 0.004 and an accuracy of 0.998. Therefore this structure can very well classify elliptical data points.

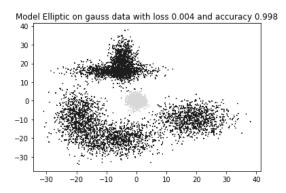


FIGURE 12. Classification via one elliptical layer and 20 epochs of training

When one e.g. observes the classification via one parabolic layer, the following phenomena occurs:

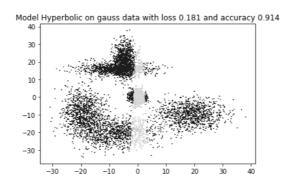


FIGURE 13. Classification via one hyperbolic layer and 20 epochs of training

This is why it makes definite sense to choose a specific layer for a specific application case. This also comes into play when one uses a shallow linear network, which does not lead to reasonably well results.

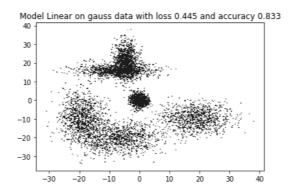


FIGURE 14. Classification via one linear layer and 20 epochs of training

When using more epochs and more hidden layers, hyperbolic and affine linear catches up with elliptic and have a similar approximation error. A similar result has been observed in the previous experiment.

Figure 15 provides us with the accuracy of the elliptic model and Figure 16 the development of the loss function.

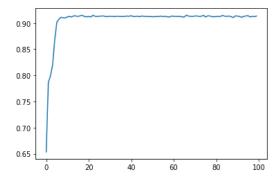


FIGURE 15. Development of the accuracy when using one shallow affine linear layer and increasing the number of epochs

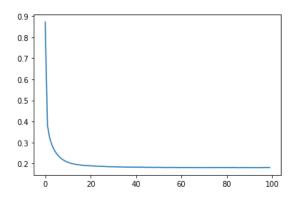


Figure 16. Development of the loss when using one shallow affine linear layer and increasing the number of epochs

6. CONCLUSION

In this paper we suggested the use of new types of neurons in fully connected neural networks. We proved a universal approximation theorem, which is applicable to such novel networks. Furthermore, we give the explicit convergence rates for the case of circular neurons, which has the

same order as the classical affine linear neurons, but with vastly reduced number of neurons (in particular is spares one layer).

Additionally, the numerical results have confirmed the improved convergence for quadratic neural network functions when compared with affine linear ones, not only for three-layer networks, but also for deep neural networks. Potential next steps involve similar analysis for higher order polynomials, as only quadratic neural network functions have been covered in this paper. Furthermore, higher dimension problems are a potential matter of interest in a follow-up paper.

A. Approximation to the identity (AtI)

Definition A.1 (Approximation to the identity [8]) A sequence of symmetric kernel functions $(S_k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R})_{k \in \mathbb{Z}}$ is said to be an approximation to the identity (AtI) if there exist a quintuple $(\epsilon, \zeta, C, C_\rho, C_A)$ of positive numbers satisfying the additional constraints

$$0 < \epsilon \le \frac{1}{n}, 0 < \zeta \le \frac{1}{n} \text{ and } C_A < 1 \tag{A.1}$$

the following three conditions are satisfied for all $k \in \mathbb{Z}$:

(i)
$$|S_k(\vec{x}, \vec{y})| \le C \frac{2^{-k\epsilon}}{(2^{-k} + C_\rho ||\vec{x} - \vec{y}||^n)^{1+\epsilon}}$$
 for all $\vec{x}, \vec{y} \in \mathbb{R}^n$;

(ii)
$$|S_k(\vec{x}, \vec{y}) - S_k(\vec{x}', \vec{y})| \le C \left(\frac{C_\rho ||\vec{x} - \vec{x}'||^n}{2^{-k} + C_\rho ||\vec{x} - \vec{y}||^n}\right)^{\zeta} \frac{2^{-k\epsilon}}{\left(2^{-k} + C_\rho ||\vec{x} - \vec{y}||^n\right)^{1+\epsilon}}$$
 for all triples $(\vec{x}, \vec{x}', \vec{y}) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n$ which satisfy

$$C_{\rho} \|\vec{x} - \vec{x}'\|^n \le C_A \left(2^{-k} + C_{\rho} \|\vec{x} - \vec{y}\|^n\right);$$
 (A.2)

(iii) $\int_{\mathbb{D}^n} S_k(\vec{x}, \vec{y}) d\vec{y} = 1$ for all $\vec{x} \in \mathbb{R}^n$.

Moreover, we say that the AtI satisfies the **double Lipschitz condition** if there exist a triple $(\tilde{C}, \tilde{C}_A, \zeta)$ of positive constants satisfying

$$\tilde{C}_A < \frac{1}{2},\tag{A.3}$$

such that for all $k \in \mathbb{Z}$

$$\left| S_{k}(\vec{x}, \vec{y}) - S_{k}(\vec{x}', \vec{y}) - S_{k}(\vec{x}, \vec{y}') + S_{k}(\vec{x}', \vec{y}') \right| \\
\leq \tilde{C} \left(\frac{C_{\rho} \|\vec{x} - \vec{x}'\|^{n}}{2^{-k} + C_{\rho} \|\vec{x} - \vec{y}\|^{n}} \right)^{\zeta} \left(\frac{C_{\rho} \|\vec{y} - \vec{y}'\|^{n}}{2^{-k} + C_{\rho} \|\vec{x} - \vec{y}\|^{n}} \right)^{\zeta} \frac{2^{-k\epsilon}}{(2^{-k} + C_{\rho} \|\vec{x} - \vec{y}\|^{n})^{1+\epsilon}}$$
(A.4)

for all quadruples $(\vec{x}, \vec{x}', \vec{y}, \vec{y}') \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n$ which satisfy

$$C_{\rho} \max \left\{ \left\| \vec{x} - \vec{x}' \right\|^{n}, \left\| \vec{y} - \vec{y}' \right\|^{n} \right\} \leq \tilde{C}_{A} \left(2^{-k} + C_{\rho} \left\| \vec{x} - \vec{y} \right\|^{n} \right).$$
 (A.5)

The conditions item (ii) and Equation A.5 are essential for our analysis. We characterize now geometric properties of these constrained sets:

Lemma A.2 Let $C_{\rho} = 1$, $C_{A} = 2^{-n}$ and $\tilde{C}_{A} = 3^{-n}$.

Then set of triples $(\vec{x}, \vec{x}', \vec{y})$ which satisfy Equation A.2 and for which $||\vec{x} - \vec{y}||^n \ge 2^{-k}$ and all $t \in [0, 1]$ satisfy

$$\|\vec{x} + t(\vec{x}' - \vec{x}) - \vec{y}\|^n \ge 2^{-n} \|\vec{x} - \vec{y}\|^n - 2^{-n}2^{-k}.$$
 (A.6)

• The set of quadrupels $(\vec{x}, \vec{x}', \vec{y}, \vec{y}')$ which satisfy Equation A.5 and for which $||\vec{x} - \vec{y}||^n \ge 2^{-k}$ satisfy

 $\|\vec{x} + t_{\vec{x}}(\vec{x}' - \vec{x}) - \vec{y} - t_{\vec{y}}(\vec{y}' - \vec{y})\|^{n} \ge 3^{-n} \|\vec{x} - \vec{y}\|^{n} - 3^{-n}2^{1-k}$ (A.7)

for all $t_{\vec{x}}, t_{\vec{y}} \in [0, 1]$.

Proof • With the concrete choice of parameters C_A , C_ρ Equation A.2 reads as follows

$$\|\vec{x} - \vec{x}'\|^n \le 2^{-n} \left(2^{-k} + \|\vec{x} - \vec{y}\|^n\right).$$
 (A.8)

Since we assume that $\|\vec{x} - \vec{y}\|^n \ge 2^{-k}$ it follows from Equation A.8 that

$$\|\vec{x} - \vec{x}'\| \le 2^{-1} \|\vec{x} - \vec{y}\|.$$

In particular $\|\vec{x} - \vec{y}\| - \|\vec{x} - \vec{x}'\| \ge 0$.

We apply Jensen's inequality, which states that for $a, b \ge 0$

$$a^n + b^n \ge 2^{1-n}(a+b)^n. \tag{A.9}$$

We use $a = \|\vec{x} + t(\vec{x}' - \vec{x}) - \vec{y}\|$ and $b = \|t(\vec{x}' - \vec{x})\|$, which then (along with the triangle inequality) gives

$$\|\vec{x} + t(\vec{x}' - \vec{x}) - \vec{y}\|^n + \|t(\vec{x}' - \vec{x})\|^n \ge 2^{1-n} (\|\vec{x} - \vec{y}\|)^n.$$

In other words, it follows from Equation A.8 that

$$\begin{split} \left\| \vec{x} + t(\vec{x}' - \vec{x}) - \vec{y} \right\|^n &\geq 2^{1-n} \left\| \vec{x} - \vec{y} \right\|^n - t^n \left\| \vec{x}' - \vec{x} \right\|^n \\ &\geq 2^{1-n} \left\| \vec{x} - \vec{y} \right\|^n - \left\| \vec{x}' - \vec{x} \right\|^n \\ &\geq 2^{1-n} \left\| \vec{x} - \vec{y} \right\|^n - 2^{-n} \left(2^{-k} + \left\| \vec{x} - \vec{y} \right\|^n \right) \\ &= 2^{-n} \left\| \vec{x} - \vec{y} \right\|^n - 2^{-n} 2^{-k}. \end{split}$$

• With the concrete choice of parameters \tilde{C}_A , C_ρ Equation A.5 reads as follows

$$\max\{\|\vec{x} - \vec{x}'\|^n, \|\vec{y} - \vec{y}'\|^n\} \le 3^{-n} \left(2^{-k} + \|\vec{x} - \vec{y}\|^n\right). \tag{A.10}$$

Since we assume that $\|\vec{x} - \vec{y}\|^n \ge 2^{-k}$ it follows from Equation A.10 that

$$\max \{ \|\vec{x} - \vec{x}'\|, \|\vec{y} - \vec{y}'\| \} \le 3^{-1} \|\vec{x} - \vec{y}\|.$$

This in particular shows that

$$\|\vec{x} - \vec{y}\| - \|\vec{x} - \vec{x}'\| - \|\vec{y} - \vec{y}'\| \ge 0.$$

We apply Jensen's inequality, which states that for $a,b,c\geq 0$

$$a^{n} + b^{n} + c^{n} \ge 3^{1-n}(a+b+c)^{n}.$$
 (A.11)

We use $a = \|\vec{x} + t_{\vec{x}}(\vec{x}' - \vec{x}) - \vec{y} - t_{\vec{y}}(\vec{y}' - \vec{y})\|$, $b = \|t_{\vec{x}}(\vec{x}' - \vec{x})\|$ and $c = \|t_{\vec{y}}(\vec{y}' - \vec{y})\|$, which then (along with the triangle inequality) gives

$$\|\vec{x} + t_{\vec{x}}(\vec{x}' - \vec{x}) - \vec{y} - t_{\vec{y}}(\vec{y}' - \vec{y})\|^n + \|t_{\vec{x}}(\vec{x}' - \vec{x})\|^n + \|t_{\vec{y}}(\vec{y}' - \vec{y})\|^n \ge 3^{1-n} \|\vec{x} - \vec{y}\|^n.$$

In other words, it follows from Equation A.10 that

$$\begin{aligned} \|\vec{x} + t_{\vec{x}}(\vec{x}' - \vec{x}) - \vec{y} - t_{\vec{y}}(\vec{y}' - \vec{y})\|^{n} &\geq 3^{1-n} \|\vec{x} - \vec{y}\|^{n} - t_{\vec{x}}^{n} \|\vec{x}' - \vec{x}\|^{n} - t_{\vec{y}}^{n} \|\vec{y}' - \vec{y}\|^{n} \\ &\geq 3^{1-n} \|\vec{x} - \vec{y}\|^{n} - \|\vec{x}' - \vec{x}\|^{n} - \|\vec{y}' - \vec{y}\|^{n} \\ &\geq 3^{1-n} \|\vec{x} - \vec{y}\|^{n} - 3^{-n} 2 \left(2^{-k} + \|\vec{x} - \vec{y}\|^{n}\right) \\ &= 3^{-n} \|\vec{x} - \vec{y}\|^{n} - 3^{-n} 2^{1-k}. \end{aligned}$$

The approximation to the identity in Definition A.1 can be used to construct wavelet frames that can approximate arbitrary functions in $\mathcal{L}^1(\mathbb{R}^n)$, as shown in the theorem below.

Remark 5 When $\|\vec{x} - \vec{y}\|^n < 2^{-k}$, Equation A.6 also holds since the right hand side of Equation A.6 is negative, so this inequality is trivial.

Theorem A.3 ([23]) Let $(S_k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R})_{k \in \mathbb{Z}}$ be a symmetric AtI which satisfies the double Lipschitz condition (see Equation A.4). Let

$$\psi_{k,\vec{y}}(\vec{x}) := 2^{-\frac{k}{2}} \left(S_k(\vec{x}, \vec{y}) - S_{k-1}(\vec{x}, \vec{y}) \right) \text{ for all } \vec{x}, \vec{y} \in \mathbb{R}^n \text{ and } k \in \mathbb{Z}.$$
 (A.12)

The the set of functions

$$\mathcal{F} := \left\{ \vec{x} \to \psi_{k,b}(\vec{x}) : k \in \mathbb{Z}, b \in 2^{-\frac{k}{d}} \mathbb{Z} \right\},\tag{A.13}$$

is a frame and for every function $f \in \mathcal{L}^1(\mathbb{R}^n)$ there exists a linear combination of N elements of \mathcal{F} , denoted by f_N , satisfying

$$||f - f_N||_{L^2} \le ||f||_{\mathcal{L}^1} (N+1)^{-1/2}.$$

Acknowledgements. OS is supported by the Austrian Science Fund (FWF), with SFB F68, project F6807-N36 - Tomography with Uncertainties. LF is supported by the Austrian Science Fund (FWF) with SFB F68, project F6807-N36 (Tomography with Uncertainties). CS is supported by the Austrian Science Fund (FWF), with project AT0116011 (Photoacoustic tomography: analysis and numerics). The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

REFERENCES

- [1] M. Abadi et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems". Preprint on ArXiv 1603.04467. 2021 (cited on pages 2, 12).
- [2] A. R. Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information Theory* 39.3 (1993), pp. 930–945. ISSN: 0018-9448. DOI: 10.1109/18.256500 (cited on page 2).
- [3] A. R. Barron, A. Cohen, W. Dahmen, and R. A. DeVore. "Approximation and learning by greedy algorithms". In: *Annals of Statistics* 36.1 (2008). ISSN: 0090-5364. DOI: 10.1214/009053607000000631 (cited on page 3).
- [4] F. Chollet et al. "Keras". In: (2015) (cited on pages 2, 12).
- [5] O. Christensen. "An Introduction to Frames and Riesz Bases". Springer International Publishing, 2016. DOI: 10.1007/978-3-319-25613-9 (cited on page 7).
- [6] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* 2.4 (1989), pp. 303–314. DOI: 10.1007/bf02551274 (cited on pages 2–4).
- [7] K. Deimling. "Nonlinear Functional Analysis". 1985. DOI: 10.1007/978-3-662-00547-7 (cited on page 4).
- [8] D. Deng and Y. Han. "Harmonic Analysis on Spaces of Homogeneous Type". Springer Berlin Heidelberg, 2009. DOI: 10.1007/978-3-540-88745-4 (cited on pages 2, 7, 11, 21).
- [9] L. Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142 (cited on page 17).

- [10] J. Dugundji. "Topology". 2nd ed. Allyn and Bacon, Inc., 1978 (cited on page 4).
- [11] F. Fan, J. Xiong, and G. Wang. "Universal approximation with quadratic deep networks". In: *Neural Networks* 124 (2020), pp. 383–392. DOI: 10.1016/j.neunet.2020.01.007 (cited on page 2).
- [12] E. Gelenbe, Z.-W. Mao, and Y.-D. Li. "Approximation by random networks with bounded number of layers". In: Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468). 1999. DOI: 10.1109/nnsp. 1999.788135 (cited on page 2).
- [13] B. Hammer. "Learning with Recurrent Neural Networks". Lecture Notes in Control and Information Sciences. Springer London, 2000. ISBN: 9781852333430 (cited on page 2).
- [14] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8 (cited on page 2).
- [15] B. Igelnik and Y. Pao. "Stochastic choice of basis functions in adaptive function approximation and the functional-link net". In: *IEEE Transactions on Neural Networks* 6.6 (1995), pp. 1320–1329. DOI: 10.1109/72.471375 (cited on page 2).
- [16] J. L. Kelley. "General Topology". Toronto-New York-London: D. Van Nostrand Company, 1955 (cited on page 4).
- [17] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". In: *Neural Networks* 6.6 (1993), pp. 861–867. DOI: 10.1016/s0893-6080(05)80131-5 (cited on pages 2, 3).
- [18] O. Manita, M. Peletier, J. Portegies, J. Sanders, and A. Senen-Cerda. "Universal Approximation in Dropout Neural Networks". In: *Journal of Machine Learning Research* 23.19 (2022), pp. 1–46 (cited on page 2).
- [19] H. N. Mhaskar. "Approximation properties of a multilayered feedforward artificial neural network". In: *Advances in Computational Mathematics* 1.1 (1993), pp. 61–80. ISSN: 1019-7168. DOI: 10.1007/bf02070821 (cited on page 2).
- [20] Y.-H. Pao, G.-H. Park, and D. J. Sobajic. "Learning and generalization characteristics of the random vector functional-link net". In: *Neurocomputing* 6.2 (1994), pp. 163–180. DOI: 10.1016/0925-2312(94)90053-1 (cited on page 2).
- [21] A. Pinkus. "Approximation theory of the MLP model in neural networks". In: *Acta Numerica* 8 (1999), pp. 143–195. ISSN: 0962-4929. DOI: 10.1017/s0962492900002919 (cited on page 2).
- [22] A. Schäfer and H. Zimmermann. "Recurrent Neural Networks are universal approximators". In: *International Journal of Neural Systems* 17.04 (2007), pp. 253–263. DOI: 10.1142/S0129065707001111 (cited on page 2).
- [23] U. Shaham, A. Cloninger, and R. R. Coifman. "Provable approximation properties for deep neural networks". In: *Applied and Computational Harmonic Analysis* 44.3 (2018), pp. 537–557. ISSN: 1063-5203. DOI: 10.1016/j.acha.2016.04.003 (cited on pages 2, 3, 6, 10, 11, 23).
- [24] N. Tsapanos, A. Tefas, N. Nikolaidis, and I. Pitas. "Neurons With Paraboloid Decision Boundaries for Improved Neural Network Classification Performance". In: *IEEE Transactions on Neural Networks and Learning Systems* 30.1 (2019), pp. 284–294. DOI: 10.1109/TNNLS.2018.2839655 (cited on page 2).
- [25] S. Wager, S. Wang, and P. Liang. "Dropout Training as Adaptive Regularization". In: Proceedings of the 26th International Conference on Neural Information Processing Systems Volume 1. Curran Associates Inc., 2013, pp. 351–359 (cited on page 2).

- [26] White. "An additional hidden unit test for neglected nonlinearity in multilayer feedforward networks". In: *International Joint Conference on Neural Networks*. 1989. DOI: 10.1109/ijcnn.1989.118281 (cited on page 2).
- [27] D.-X. Zhou. "Deep distributed convolutional neural networks: Universality". In: *Analysis and Applications* 16.6 (2018), pp. 895–919. DOI: 10.1142/s0219530518500124 (cited on page 2).
- [28] D.-X. Zhou. "Universality of deep convolutional neural networks". In: Applied and Computational Harmonic Analysis 48.2 (2020), pp. 787–794. ISSN: 1063-5203. DOI: 10.1016/j.acha.2019.06.004 (cited on page 2).