On the rate of convergence of image classifiers based on convolutional neural networks [1]

Michael Kohler[1], Adam Krzyżak[2,2] and Benjamin Walter[1]

[1] *Fachbereich Mathematik, Technische Universität Darmstadt, Schlossgartenstr. 7, 64289 Darmstadt, Germany, email: kohler@mathematik.tu-darmstadt.de, bwalter@mathematik.tu-darmstadt.de*

[2] *Department of Computer Science and Software Engineering, Concordia University, 1455 De Maisonneuve Blvd. West, Montreal, Quebec, Canada H3G 1M8, email: krzyzak@cs.concordia.ca*

October 14, 2020

Abstract

Image classifiers based on convolutional neural networks are defined, and the rate of convergence of the misclassification risk of the estimates towards the optimal misclassification risk is analyzed. Under suitable assumptions on the smoothness and structure of the aposteriori probability, the rate of convergence is shown which is independent of the dimension of the image. This proves that in image classification, it is possible to circumvent the curse of dimensionality by convolutional neural networks. Our classifiers are compared with various other classification methods using simulated data. Furthermore, the performance of our estimates is also tested on real images.

AMS classification: Primary 62G05; secondary 62G20.

Key words and phrases: Curse of dimensionality, convolutional neural networks, image classification, rate of convergence.

1 Introduction

1.1 Scope of this article

Deep neural networks are nowadays among the most successful and most widely used methods in machine learning, see, e.g., Schmidhuber (2015), Rawat and Wang (2017), and the literature cited therein. In many applications the most successful networks are deep convolutional networks, see, e.g., Krizhevsky, Sutskever and Hinton (2012) and Kim (2014) concerning applications in image classification or language recognition, respectively. These networks can be considered as a special case of the deep feedforward neural networks, where symmetry constraints are imposed on the weights of the networks. For general deep feedforward neural networks it was recently shown that under suitable

---

[1] Running title: *Rate of convergence of image classifiers*
[2] Corresponding author. Tel: +1-514-848-2424 ext. 3007, Fax:+1-514-848-2830

compository assumptions on the structure of the regression function these networks are able to achieve dimension reduction in estimation of high-dimensional regression functions (cf., Kohler and Krzyżak (2017), Bauer and Kohler (2019), Schmidt-Hieber (2019), Kohler and Langer (2019) and Suzuki and Nitanda (2019)). The purpose of this article is to characterize situations in image classification, where deep convolutional neural networks can achieve a similar dimension reduction.

1.2 Image classification

Let $d_1, d_2 \in \mathbb{N}$ and let $(\mathbf{X}, Y)$, $(\mathbf{X}_1, Y_1)$, ..., $(\mathbf{X}_n, Y_n)$ be independent and identically distributed random variables with values in

$$[0,1]^{\{1,\dots,d_1\} \times \{1,\dots,d_2\}} \times \{0,1\}.$$

Here we use the notation

$$[0,1]^J = \{(a_j)_{j \in J} : a_j \in [0,1] \quad (j \in J)\}$$

for a nonempty and finite index set J, and we describe a (random) image from (random) class $Y \in \{0,1\}$ by a (random) matrix X with d_1 columns and d_2 rows, which contains at position (i,j) the grey scale value of the pixel of the image at the corresponding position.

Let

$$\eta(\mathbf{x}) = \mathbf{P}\{Y = 1 | \mathbf{X} = \mathbf{x}\} \quad (\mathbf{x} \in [0,1]^{\{1,\dots,d_1\} \times \{1,\dots,d_2\}}) \tag{1}$$

be the so–called aposteriori probability. Then we have

$$\min_{f:[0,1]^{\{1,\dots,d_1\} \times \{1,\dots,d_2\}} \to \{0,1\}} \mathbf{P}\{f(\mathbf{X}) \neq Y\} = \mathbf{P}\{f^*(\mathbf{X}) \neq Y\},$$

where

$$f^*(\mathbf{x}) = \begin{cases} 1, & \text{if } \eta(\mathbf{x}) > \frac{1}{2} \\ 0, & \text{elsewhere} \end{cases}$$

is the so–called Bayes classifier (cf., e.g., Theorem 2.1 in Devroye, Györfi and Lugosi (1996)). Set

$$\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n)\}.$$

In the sequel we consider the problem of constructing a classifier

$$f_n = f_n(\cdot, \mathcal{D}_n) : [0,1]^{\{1,\dots,d_1\} \times \{1,\dots,d_2\}} \to \{0,1\}$$

such that the misclassification risk

$$\mathbf{P}\{f_n(\mathbf{X}) \neq Y | \mathcal{D}_n\}$$

of this classifier is as small as possible. Our aim is to derive a bound on the expected difference of the misclassification risk of f_n and the optimal misclassification risk, i.e., we want to derive an upper bound on

$$\mathbf{E}\left\{\mathbf{P}\{f_n(\mathbf{X}) \neq Y | \mathcal{D}_n\} - \min_{f:[0,1]^{\{1,\dots,d_1\} \times \{1,\dots,d_2\}} \to \{0,1\}} \mathbf{P}\{f(\mathbf{X}) \neq Y\}\right\}$$
$$= \mathbf{P}\{f_n(\mathbf{X}) \neq Y\} - \mathbf{P}\{f^*(\mathbf{X}) \neq Y\}.$$

1.3 Plug-in classifiers

We will use plug-in classifiers of the form

$$
f_n(\mathbf{x}) = \begin{cases} 1, & \text{if } \eta_n(\mathbf{x}) \geq \frac{1}{2} \\ 0, & \text{elsewhere} \end{cases}
$$

where

$$
\eta_n(\cdot) = \eta_n(\cdot, \mathcal{D}_n) : [0,1]^{\{1,\ldots,d_1\} \times \{1,\ldots,d_2\}} \to \mathbb{R}
$$

is an estimate of the aposteriori probability (1). It is well-known that such plug-in classifiers satisfy

$$
\mathbf{P}\{f_n(\mathbf{X}) \neq Y | \mathcal{D}_n\} - \mathbf{P}\{f^*(\mathbf{X}) \neq Y\} \leq 2 \cdot \int |\eta_n(\mathbf{x}) - \eta(\mathbf{x})| \, \mathbf{P}_{\mathbf{X}}(d\mathbf{x})
$$

(cf., e.g., Theorem 1.1 in Györfi et al. (2002)), which implies (via the Cauchy-Schwartz inequality)

$$
\mathbf{P}\{f_n(\mathbf{X}) \neq Y\} - \mathbf{P}\{f^*(\mathbf{X}) \neq Y\} \leq 2 \cdot \sqrt{\mathbf{E}\left\{\int |\eta_n(\mathbf{x}) - \eta(\mathbf{x})|^2 \, \mathbf{P}_{\mathbf{X}}(d\mathbf{x})\right\}}. \tag{2}
$$

Hence we can derive an upper bound on the difference between the expected misclassification risk of our estimate and the minimal possible value from a bound on the expected L_2 error of the estimate η_n of the aposteriori probability.

It is well-known that the bound in (2) is not tight, therefore classification is easier than regression estimation (cf., Devroye, Görfi and Lugosi (1996)). In the sequel we will nevertheless solve an image classification problem via regression estimation, because this will enable us to impose conditions on the underlying distribution by restricting the structure of the aposteriori probability. And, as we will see in the next subsection, it is easy to formulate such restrictions such that they seem to be natural assumptions in image classification applications.

1.4 A hierarchical max-pooling model for the aposteriori probability

In order to derive nontrivial rate of convergence results on the difference between the misclassification risk of any estimate and the minimal possible value it is necessary to restrict the class of distributions (cf., Cover (1968) and Devroye (1982)). In the sequel we will use assumptions on the structure and the smoothness of the aposteriori probability.

The basic idea behind the formulation of our structural constraint is the following: Consider an application where a human has to decide about a class of an image, e.g., the human has to decide whether a given image contains a specific traffic sign or not. Then the human will survey the whole image and look at each subpart of the image whether it contains the traffic sign or not. By looking at a subpart, the human can estimate a probability that this subpart contains the traffic sign. It is then natural to assume that the probability that the whole image contains a traffic sign is simply the maximum of

the probabilities for each subpart of the image. This idea leads to the definition of a max-pooling model for the aposteriori probability introduced below.

Furthermore, we take decision whether a given subpart of the image contains a traffic sign or not by taking several decisions whether the image contains parts of a traffic sign or not, and by combining these decisions about the different parts hierarchically. This idea leads to the hierarchical model introduced below.

Combining both ideas leads to the hierarchical max-pooling model introduced below.

Now consider an application in which a human has to classify an image by applying a function to the information about the existence of several objects, e.g., the human has to decide whether an image contains exactly three specific traffic signs out of a list of five specific traffic signs. Then, for each of these five traffic signs, the human estimates the probability that the image contains the traffic sign and then verifies whether exactly three probabilities are sufficiently large. This leads us to our main model, the generalized hierarchical max-pooling model, which we introduce next. In order to define this model we need the following notation: For $M \subseteq \mathbb{R}^d$ and $\mathbf{x} \in \mathbb{R}^d$ we define

$$\mathbf{x} + M = \{\mathbf{x} + \mathbf{z} : \mathbf{z} \in M\}.$$

For $I \subseteq \{1, \ldots, d_1\} \times \{1, \ldots, d_2\}$ and $\mathbf{x} = (x_i)_{i \in \{1,\ldots,d_1\} \times \{1,\ldots,d_2\}} \in [0,1]^{\{1,\ldots,d_1\} \times \{1,\ldots,d_2\}}$ we set

$$\mathbf{x}_I = (x_i)_{i \in I}.$$

Definition 1 *Let $d_1, d_2 \in \mathbb{N}$ with $d_1, d_2 > 1$ and $m : [0,1]^{\{1,\ldots,d_1\} \times \{1,\ldots,d_2\}} \to \mathbb{R}$.*
a) *We say that m satisfies a* **max-pooling model with index set**

$$I \subseteq \{0, \ldots, d_1 - 1\} \times \{0, \ldots, d_2 - 1\},$$

if there exist a function $f : [0,1]^{(1,1)+I} \to \mathbb{R}$ such that

$$m(\mathbf{x}) = \max_{(i,j) \in \mathbb{Z}^2 \, : \, (i,j)+I \subseteq \{1,\ldots,d_1\} \times \{1,\ldots,d_2\}} f\left(\mathbf{x}_{(i,j)+I}\right) \quad (\mathbf{x} \in [0,1]^{\{1,\ldots,d_1\} \times \{1,\ldots,d_2\}}).$$

b) *Let $I = \{0, \ldots, 2^l - 1\} \times \{0, \ldots, 2^l - 1\}$ for some $l \in \mathbb{N}$. We say that*

$$f : [0,1]^{\{1,\ldots,2^l\} \times \{1,\ldots,2^l\}} \to \mathbb{R}$$

satisfies a **hierarchical model of level** *l, if there exist functions*

$$g_{k,s} : \mathbb{R}^4 \to [0,1] \quad (k = 1, \ldots, l, s = 1, \ldots, 4^{l-k})$$

such that we have

$$f = f_{l,1}$$

for some $f_{k,s} : [0,1]^{\{1,\ldots,2^k\} \times \{1,\ldots,2^k\}} \to \mathbb{R}$ recursively defined by

$$
\begin{aligned}
f_{k,s}(\mathbf{x}) \;=\; & g_{k,s}\big(f_{k-1,4\cdot(s-1)+1}(\mathbf{x}_{\{1,\ldots,2^{k-1}\} \times \{1,\ldots,2^{k-1}\}}), \\
& f_{k-1,4\cdot(s-1)+2}(\mathbf{x}_{\{2^{k-1}+1,\ldots,2^k\} \times \{1,\ldots,2^{k-1}\}}),
\end{aligned}
$$

$$f_{k-1,4\cdot(s-1)+3}\big(\mathbf{x}_{\{1,\dots,2^{k-1}\}\times\{2^{k-1}+1,\dots,2^k\}}\big),$$
$$f_{k-1,4\cdot s}\big(\mathbf{x}_{\{2^{k-1}+1,\dots,2^k\}\times\{2^{k-1}+1,\dots,2^k\}}\big)\big)$$
$$\left(\mathbf{x}\in[0,1]^{\{1,\dots,2^k\}\times\{1,\dots,2^k\}}\right)$$

for $k=2,\dots,l$, $s=1,\dots,4^{l-k}$, and

$$f_{1,s}(x_{1,1},x_{1,2},x_{2,1},x_{2,2})=g_{1,s}(x_{1,1},x_{1,2},x_{2,1},x_{2,2})\quad(x_{1,1},x_{1,2},x_{2,1},x_{2,2}\in[0,1])$$

for $s=1,\dots,4^{l-1}$.

c) *We say that $m:[0,1]^{\{1,\dots,d_1\}\times\{1,\dots,d_2\}}\to\mathbb{R}$ satisfies a* **hierarchical max-pooling model of level** l *(where $2^l\leq\min\{d_1,d_2\}$), if m satisfies a max-pooling model with index set*
$$I=\left\{0,\dots,2^l-1\right\}\times\left\{0,\dots,2^l-1\right\}$$
and the function $f:[0,1]^{(1,1)+I}\to\mathbb{R}$ in the definition of this max-pooling model satisfies a hierarchical model with level l.

d) *Let $d^*\in\mathbb{N}$. We say $m:[0,1]^{\{1,\dots,d_1\}\times\{1,\dots,d_2\}}\to\mathbb{R}$ satisfies a* **generalized hierarchical max-pooling model of order** d^* **and level** l*, if there exist functions*
$$m_1,\dots,m_{d^*}:[0,1]^{\{1,\dots,d_1\}\times\{1,\dots,d_2\}}\to\mathbb{R},$$
which satisfy a hierarchical max-pooling model of level l, and if there exists a function $g:\mathbb{R}^{d^*}\to[0,1]$ such that
$$m(\mathbf{x})=g(m_1(\mathbf{x}),\dots,m_{d^*}(\mathbf{x})).$$

e) *Let $p_1,p_2\in(0,\infty)$. We say that a generalized hierarchical max-pooling model of order d^* and level l has smoothness constraints p_1 and p_2, if all functions $g_{k,s}$ in the definition of the functions m_i are (p_1,C_1)–smooth for some $C_1>0$ for any $i\in\{1,\dots,d^*\}$, and if the function g is (p_2,C_2)–smooth for some $C_2>0$ (see Subsection 1.7 for the definition of (p,C)–smoothness).*

Remark 1. In the definition of our generalized hierarchical max-pooling model we do not allow distinct levels $l_1,\dots,l_{d^*}$ for the functions $m_1,...,m_{d^*}$. This is a restriction of the more general case which we use because it makes our proofs much less technical (see Remark 5 for the generalization of our results).

1.5 Main results

The main contributions in this paper are as follows: First, we introduce the above setting for the mathematical analysis of an image classification problem. Here our main idea is to use plug-in classification estimates, which allows us to restrict the underlying class of distributions by imposing constraints on the structure and the smoothness of the aposteriori probability. The main advantage of this approach is that we can introduce in this setting with the above generalized hierarchical max-pooling model a natural condition

for applications. Second, we analyze the rate of convergence of the deep convolutional neural network classifiers (with ReLU activation function) in this context. Here we show in Theorem 1 below that in case that the aposteriori probability satisfies a generalized hierarchical max-pooling model of order d^* with smoothness constraints p_1 and p_2, the expected misclassification risk of the estimate converges toward the minimal possible value with rate

$$\max\left\{n^{-\frac{p_1}{2 \cdot p_1 + 4}}, n^{-\frac{p_2}{2 \cdot p_2 + d^*}}\right\}$$

(up to some logarithmic factor). Since this rate of convergence does not depend on the dimension $d_1 \cdot d_2$ of the image, this shows that under suitable assumptions on the structure of the aposteriori probability it is possible to circumvent the curse of dimensionality in image classification by using convolutional neural networks.

1.6 Discussion of related results

Convolutional neural networks, introduced by Le Cun et al. (1989), have become the leading techniques in pattern recognition applications, cf., e.g., Le Cun et al. (1998), LeCun, Bengio and Hinton (2015), Goodfellow, Bengio and Courville (2016), Rawat and Wang (2017), and the literature cited therein.

As mentioned by Rawat and Wang (2017), despite the empirical success of these methods the theoretical proof of why they succeed is lacking. In fact there are only a few papers addressing theoretical properties of these networks. Several papers used the idea that properly defined convolutional neural networks are able to mimic deep feedforward neural networks and obtained rate of convergence results for estimates based on convolutional neural networks similar to feedforward neural networks estimates (cf., e.g., Oono and Suzuki (2019) and the literature cited therein). The drawback of this approach is that in this way it is not possible to identify situations in which convolutional neural networks are superior to standard feedforward neural networks. Generalization bounds for convolutional neural networks have been analyzed in Lin and Zhang (2019). In several papers it was shown that gradient descent is able to find the global minimum of the empirical loss function in case of overparametrized convolutional neural networks, cf., e.g., Du et al. (2019). But, as was shown by a counterexample in Kohler and Krzyżak (2019), overparametrized deep neural networks do not, in general, generalize well. In an abstract setting, very interesting approximation properties of deep convolutional neural networks have been obtained by Yarotsky (2018). However, it is unclear how one can apply these results in statistical estimation problem.

Much more is known about standard deep feedforward neural networks. Here, it was recently shown that under suitable compository assumptions on the structure of the regression function these networks are able to achieve dimension reduction in estimation of high-dimensional regression functions (cf., Kohler and Krzyżak (2017), Bauer and Kohler (2019), Schmidt-Hieber (2019), Kohler and Langer (2019) and Suzuki and Nitanda (2019)). Imaizumi and Fukamizu (2019) derived results concerning estimation by neural networks of piecewise polynomial regression functions with partitions having rather general smooth boundaries. Eckle and Schmidt-Hieber (2019) and Kohler, Krzyżak and

Langer (2019) showed that the least squares neural network regression estimates based on deep neural networks can achieve the rate of convergence results similar to piecewise polynomial partition estimates where partition is chosen in an optimal way.

Classification theory has been intensively studied in statistics, see e.g., the book Devroye, Györfi and Lugosi (1996) which discusses probabilistic theory of pattern recognition in depth. This theory can of course be applied to image classification, but due to high dimensionality of the input in image classification, this will not lead to useful results. To the best of our knowledge there do not exist until now papers which analyze the rate of convergence of image classifiers and are able to achieve sufficient, and for some applications satisfactory, dimension reduction. Classification problem with standard deep feedforward neural networks has been analyzed in Kim, Ohn and Kim (2019).

Bayesian image analysis, which can be used, e.g., for feature extraction, can be found in Chang et al. (2017).

A related problem to image classification is image reconstruction or image denoising. Here, quite a few theoretical results exist, see, e.g., Korostelev and Tsybakov (1993) and the literature cited therein.

1.7 Notation

Throughout the paper, the following notation is used: The sets of natural numbers, natural numbers including 0, integers and real numbers are denoted by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Z}$ and $\mathbb{R}$, respectively. For $z \in \mathbb{R}$, we denote the smallest integer greater than or equal to z by $\lceil z \rceil$. Let $D \subseteq \mathbb{R}^d$ and let $f : \mathbb{R}^d \to \mathbb{R}$ be a real-valued function defined on $\mathbb{R}^d$. We write $\mathbf{x} = \arg\min_{\mathbf{z} \in D} f(\mathbf{z})$ if $\min_{\mathbf{z} \in \mathcal{D}} f(\mathbf{z})$ exists and if $\mathbf{x}$ satisfies $\mathbf{x} \in D$ and $f(\mathbf{x}) = \min_{\mathbf{z} \in \mathcal{D}} f(\mathbf{z})$. For $f : \mathbb{R}^d \to \mathbb{R}$

$$\|f\|_\infty = \sup_{\mathbf{x} \in \mathbb{R}^d} |f(\mathbf{x})|$$

is its supremum norm, and the supremum norm of f on a set $A \subseteq \mathbb{R}^d$ is denoted by

$$\|f\|_{A,\infty} = \sup_{\mathbf{x} \in A} |f(\mathbf{x})|.$$

Let $p = q + s$ for some $q \in \mathbb{N}_0$ and $0 < s \leq 1$. A function $f : \mathbb{R}^d \to \mathbb{R}$ is called (p, C)-smooth, if for every $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_d) \in \mathbb{N}_0^d$ with $\sum_{j=1}^d \alpha_j = q$ the partial derivative $\frac{\partial^q f}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}}$ exists and satisfies

$$\left| \frac{\partial^q f}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}}(\mathbf{x}) - \frac{\partial^q f}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}}(\mathbf{z}) \right| \leq C \cdot \|\mathbf{x} - \mathbf{z}\|^s$$

for all $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$.

Let $\mathcal{F}$ be a set of functions $f : \mathbb{R}^d \to \mathbb{R}$, let $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ and set $\mathbf{x}_1^n = (\mathbf{x}_1, \dots, \mathbf{x}_n)$. A finite collection $f_1, \dots, f_N : \mathbb{R}^d \to \mathbb{R}$ is called an ε-cover of $\mathcal{F}$ on $\mathbf{x}_1^n$ if for any $f \in \mathcal{F}$ there exists $i \in \{1, \dots, N\}$ such that

$$\frac{1}{n} \sum_{k=1}^n |f(\mathbf{x}_k) - f_i(\mathbf{x}_k)| < \varepsilon.$$

The ε–covering number of $\mathcal{F}$ on $\mathbf{x}_1^n$ is the size N of the smallest ε–cover of $\mathcal{F}$ on $\mathbf{x}_1^n$ and is denoted by $\mathcal{N}_1(\varepsilon, \mathcal{F}, \mathbf{x}_1^n)$.

For $z \in \mathbb{R}$ and $\beta > 0$ we define $T_\beta z = \max\{-\beta, \min\{\beta, z\}\}$. If $f : \mathbb{R}^d \to \mathbb{R}$ is a function and $\mathcal{F}$ is a set of such functions, then we set

$$(T_\beta f)(\mathbf{x}) = T_\beta (f(\mathbf{x})) \quad \text{and} \quad T_\beta \mathcal{F} = \{T_\beta f \quad : \quad f \in \mathcal{F}\}.$$

1.8 Outline of the paper

In Section 2 the convolutional neural network image classifiers used in this paper are defined. The main result is presented in Section 3 and proven in Section 4.

2 Convolutional neural network image classifiers

In the sequel we define a convolutional neural network architecture by computing several convolutional networks in parallel and by finally applying a fully connected standard feedforward network consisting of several layers to the results of these networks.

Firstly, we define a fully connected multilayer feedforward neural network with L hidden layers and k_r neurons in layer r $(r = 1, \ldots, L)$. The output of the network is produced by a function $g : \mathbb{R}^t \to \mathbb{R}$ of the form

$$g(\mathbf{x}) = \sum_{i=1}^{k_L} w_i^{(L)} g_i^{(L)}(\mathbf{x}) + w_0^{(L)}, \tag{3}$$

where $w_0^{(L)}, \ldots, w_{k_L}^{(L)} \in \mathbb{R}$ denote the output weights and for $i \in \{1, \ldots, k_L\}$ the $g_i^{(L)}$ are recursively defined by

$$g_i^{(r)}(\mathbf{x}) = \sigma \left(\sum_{j=1}^{k_{r-1}} w_{i,j}^{(r-1)} g_j^{(r-1)}(\mathbf{x}) + w_{i,0}^{(r-1)} \right)$$

for $w_{i,0}^{(r-1)}, \ldots, w_{i,k_{r-1}}^{(r-1)} \in \mathbb{R}$, $i \in \{1, \ldots, k_r\}$, $r \in \{1, \ldots, L\}$, $k_0 = t$ and

$$g_i^{(0)}(\mathbf{x}) = x_i$$

for $i \in \{1, \ldots, k_0\}$, where the function $\sigma : \mathbb{R} \to \mathbb{R}$ denotes the ReLU activation function

$$\sigma(x) = \max\{x, 0\}.$$

We define the function class of all real-valued functions on $\mathbb{R}^t$ of the form (3) with parameters L and $\mathbf{k} = (k_1, \ldots, k_L)$ by $\mathcal{G}_t(L, \mathbf{k})$.

Secondly, we define a convolutional neural network with $L \in \mathbb{N}$ convolutional layers, one linear layer and one max-pooling layer for a $[0, 1]^{\{1, \ldots, d_1\} \times \{1, \ldots, d_2\}}$–valued input, where $d_1, d_2 \in \mathbb{N}$. The network has $k_r \in \mathbb{N}$ channels (also called feature maps) in the convolutional layer r and the convolution in layer r is performed by a window of values of layer

$r - 1$ of size $M_r \in \{1, \ldots, \min\{d_1, d_2\}\}$, where $r \in \{1, \ldots, L\}$. We will denote the input layer as the convolutional layer 0 with $k_0 = 1$ channels. The network depends on the weight matrix (so–called filter)

$$\mathbf{w} = \left(w_{i,j,s_1,s_2}^{(r)} \right)_{1 \leq i,j \leq M_r, s_1 \in \{1,\ldots,k_{r-1}\}, s_2 \in \{1,\ldots,k_r\}, r \in \{1,\ldots,L\}},$$

the weights

$$\mathbf{w}_{bias} = \left(w_{s_2}^{(r)} \right)_{s_2 \in \{1,\ldots,k_r\}, r \in \{1,\ldots,L\}}$$

for the bias in each channel and each convolutional layer and the output weights

$$\mathbf{w}_{out} = (w_s)_{s \in \{1,\ldots,k_L\}}.$$

The output of the network is given by a real–valued function on $[0,1]^{\{1,\ldots,d_1\} \times \{1,\ldots,d_2\}}$ of the form

$$f_{\mathbf{w}, \mathbf{w}_{bias}, \mathbf{w}_{out}}(\mathbf{x}) \quad = \quad \max \left\{ \sum_{s_2=1}^{k_L} w_{s_2} \cdot o_{(i,j),s_2}^{(L)} : i \in \{1, \ldots, d_1 - M_L + 1\}, \right.$$

$$\left. j \in \{1, \ldots, d_2 - M_L + 1\} \right\},$$

where $o_{(i,j),s_2}^{(L)}$ is the output of the last convolutional layer, which is recursively defined as follows:

We start with

$$o_{(i,j),1}^{(0)} = x_{i,j} \quad \text{for } i \in \{1, \ldots, d_1\} \text{ and } j \in \{1, \ldots, d_2\}.$$

Then we define recursively

$$o_{(i,j),s_2}^{(r)} = \sigma \left(\sum_{s_1=1}^{k_{r-1}} \sum_{\substack{t_1,t_2 \in \{1,\ldots,M_r\} \\ (i+t_1-1, j+t_2-1) \in D}} w_{t_1,t_2,s_1,s_2}^{(r)} \cdot o_{(i+t_1-1, j+t_2-1), s_1}^{(r-1)} + w_{s_2}^{(r)} \right) \qquad (4)$$

for the index set $D = \{1, \ldots, d_1\} \times \{1, \ldots, d_2\}$, $(i,j) \in D$, $s_2 \in \{1, \ldots, k_r\}$ and $r \in \{1, \ldots, L\}$.

Let $\mathcal{F}(L, \mathbf{k}, \mathbf{M})$ be the set of all functions of the above form with parameters L, $\mathbf{k} = (k_1, \ldots, k_L)$ and $\mathbf{M} = (M_1, \ldots, M_L)$. With the definition of the index set D in (4) we use a so-called zero padding which is illustrated in Figure 1. Therefore, the size of a channel is the same as in the previous layer.

The function class that we will introduce here is then given by

$$\mathcal{F}_t \left(\mathbf{L}, \mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \mathbf{M} \right)$$
$$= \left\{ g \circ (f_1, \ldots, f_t) : f_1, \ldots, f_t \in \mathcal{F} \left(L_1, \mathbf{k}^{(1)}, \mathbf{M} \right), g \in \mathcal{G}_t \left(L_2, \mathbf{k}^{(2)} \right) \right\}.$$
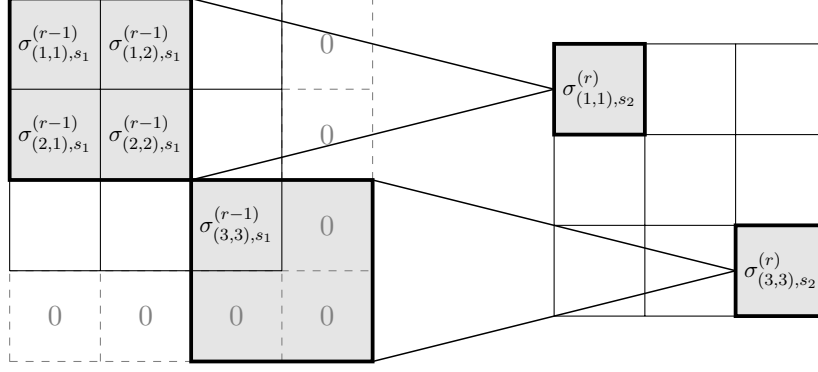
Figure 1: Illustration of the zero padding for $M_r = 2$ and $d_1 = d_2 = 3$.

It depends on the parameters

$$\mathbf{L} = (L_1, L_2), \ \mathbf{k}^{(1)} = \left(k_1^{(1)}, \ldots, k_{L_1}^{(1)}\right), \ \mathbf{k}^{(2)} = \left(k_1^{(2)}, \ldots, k_{L_2}^{(2)}\right), \ \mathbf{M} = (M_1, \ldots, M_{L_1})$$

and $t \in \mathbb{N}$. Let

$$\eta_n = \underset{f \in \mathcal{F}_t\left(\mathbf{L}, \mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \mathbf{M}\right)}{\arg\min} \frac{1}{n} \sum_{i=1}^n |Y_i - f(\mathbf{X}_i)|^2 \tag{5}$$

be the least squares estimate of $\eta(\mathbf{x}) = \mathbf{E}\{Y = 1 | \mathbf{X} = \mathbf{x}\}$. Then our estimate f_n is defined by

$$f_n(\mathbf{x}) = \begin{cases} 1, & \text{if } \eta_n(\mathbf{x}) \geq \frac{1}{2} \\ 0, & \text{elsewhere.} \end{cases}$$

3 Main results

Our main result is the following theorem, which presents an upper bound on the distance between the expected misclassification risk of our plug-in classifier and the optimal misclassification risk.

Theorem 1 *Let $d_1, d_2 \in \mathbb{N}$ with $d_1, d_2 > 1$. Let $(\mathbf{X}, Y)$, $(\mathbf{X}_1, Y_1)$, ..., $(\mathbf{X}_n, Y_n)$ be independent and identically distributed $[0, 1]^{\{1, \ldots, d_1\} \times \{1, \ldots, d_2\}} \times \{0, 1\}$-valued random variables with $n > 1$. Assume that the aposteriori probability $\eta(\mathbf{x}) = \mathbf{P}\{Y = 1 | \mathbf{X} = \mathbf{x}\}$ satisfies a generalized hierarchical max-pooling model of finite order d^* and level l with smoothness constraints $p_1, p_2 \in [1, \infty)$. Choose*

$$L_n = \max\left\{ \left\lceil c_1 \cdot n^{\frac{4}{2 \cdot (2 \cdot p_1 + 4)}} \right\rceil, \left\lceil c_1 \cdot n^{\frac{d^*}{2 \cdot (2 \cdot p_2 + d^*)}} \right\rceil \right\}$$

and set

$$\mathbf{L} = (L_1, L_2) = \left(\frac{4^l - 1}{3} \cdot L_n + l, L_n \right),$$

for $c_1 > 0$ sufficiently large. Furthermore, choose $t = d^*$,

$$k_s^{(1)} = \frac{2 \cdot 4^l + 4}{3} + c_2$$

for $s \in \{1, \dots, L_1\}$, $k_s^{(2)} = c_2$ for $s \in \{1, \dots, L_2\}$ and $c_2 \in \mathbb{N}$ sufficiently large and set

$$M_s = 2^{\pi(s)} \quad \text{for } s \in \{1, \dots, L_1\},$$

where $\pi : \{1, \dots, L_1\} \to \{1, \dots, l\}$ is an increasing function defined by

$$\pi(s) = \sum_{i=1}^{l} \mathbb{I}_{\left\{s \geq i + \sum_{r=l-i+1}^{l-1} 4^r \cdot L_n\right\}}.$$

We define the estimate f_n as in Section 2. Then

$$\mathbf{P}\{f_n(\mathbf{X}) \neq Y\} - \min_{f : [0,1]^{\{1,\dots,d_1\} \times \{1,\dots,d_2\}} \to \{0,1\}} \mathbf{P}\{f(\mathbf{X}) \neq Y\}$$
$$\leq c_3 \cdot \sqrt{\log(d_1 \cdot d_2)} \cdot (\log n)^2 \cdot \max\left\{ n^{-\frac{p_1}{2 \cdot p_1 + 4}}, n^{-\frac{p_2}{2 \cdot p_2 + d^*}} \right\},$$

for some constant $c_3 > 0$ which does not depend on d_1, d_2 and n.

Remark 2. The rate of convergence in Theorem 1 does not depend on the dimension $d_1 \cdot d_2$ of $\mathbf{X}$, hence the estimate is able to circumvent the curse of dimensionality under the above structural assumption on η.

Remark 3. In the proof of Theorem 1 we show that the expected L_2 error of our estimate of the aposteriori probability tends to zero with the rate of convergence

$$\max\left\{ n^{-\frac{2p_1}{2p_1 + 4}}, n^{-\frac{2p_2}{2p_2 + d^*}} \right\} \tag{6}$$

(up to some logarithmic factor). According to Stone (1982)

$$n^{-\frac{2p}{2p+t}}$$

is the optimal minimax rate of convergence for estimation of (p, C)–smooth functions defined on $\mathbb{R}^t$. We conjecture that (6) is in our setting the optimal rate of convergence for estimation of the aposteriori probability.

Remark 4. To show the above bound on the misclassification risk we bound the L_2– error of the estimate η_n of the aposteriori probability (see inequality (2)). So we solve our classification problem via regression estimation. Kohler and Langer (2019) present an upper bound for the expected L_2–error of least squares neural network regression estimates based on a set of fully connected neural networks. The upper bound is linear-dependent on the dimension of $\mathbf{X}$. This dependence on the dimension of $\mathbf{X}$ results from the VC dimension of the class of fully connected neural networks (see Subsection 6.3 for the definition of the VC dimension). In our result, however, the dimension $d_1 \cdot d_2$ of

11

X only occurs logarithmically, which gives us an indication of why convolutional neural networks could be able to outperform the standard feedforward neural networks in image classification.

Remark 5. The above result can also be shown for the more general case where the aposteriori probability satisfies a generalized hierarchical max-pooling model in which functions $m_1, ..., m_{d^*}$ have distinct levels $l_1, ..., l_{d^*}$. In this case we would choose the parameter l in the definition of the convolutional neural network as the maximum $\max\{l_1, ...l_{d^*}\}$. The biggest challenge would then be to modify the approximation result of Lemma 5. Here the idea of the proof would then be to represent the maximum $\max\{x_1, \ldots, x_4\}$ on $\mathbb{R}^4$ as a standard feedforward neural network and apply a modification of Lemma 6 to it. This would enable us to calculate the maximum of four positions of a channel. However, the proof would be much more technical.

4 Application to simulated data

In this section we illustrate how the introduced image classifier based on the convolutional neural networks behaves in case of finite sample sizes. Therefore, we apply it to the synthetic image data sets and compare the results with other classification methods using Python code. Firstly, we describe how the synthetic image data sets were generated. A data set consists of finitely many realizations $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots$ of a random variable

$$(\mathbf{X}, Y) \in [0, 1]^{\{1, \ldots, 32\} \times \{1, \ldots, 32\}} \times \{0, 1\},$$

where $\mathbf{X}$ is a random image with label Y and the image dimensions here correspond to $d_1 = d_2 = 32$. As described in Section 1, the matrix $\mathbf{X}$ contains at position (i, j) the grey scale value of the pixel of the image at the corresponding position. We consider two different classification problems, where our classifier is supposed to distinguish between two classes of geometric objects.

The first classification task is to detect whether an image contains a circle. Therefore our synthetic image data set consists of images that do not contain a circle and images that contain at least one circle. In the following we describe how such an image is created. Each image consists of three geometric objects. For each object we randomly and independently choose between a square, an equilateral triangle and a circle with fixed probabilities each. The circle is choosen with probability $p = 1 - 0.5^{\frac{1}{3}}$ and the square and the equilateral triangle with probability $q = \frac{1}{2} \cdot 0.5^{\frac{1}{3}}$, respectively. After an object has been defined as the square, triangle or circle we randomly choose its area, rotation and grey scale values. For each object, rotation and area are choosen independently and are uniformly distributed on a fixed interval. We determine the grey scale values of the three objects by randomly permuting the list $(\frac{1}{3}, \frac{2}{3}, 1)$ of three grey scale values. The positions of the objects are determined one after the other. For the first object, we generate its position from the uniform distribution on the restricted image area so that the object lies completely within the image. The position of the second object is chosen in the same way with the additional restriction that the second object only covers a maximum of one percent of the area of the first object. For the placement of the third object, we use the

corresponding restriction that the third image only covers a maximum of one percent of the area of the first and second object. With the above procedure, the label Y is discrete and is uniformly distributed on $\{0, 1\}$, since the probability that the image does not contain a circle is $(2 \cdot q)^3 = 0.5$.
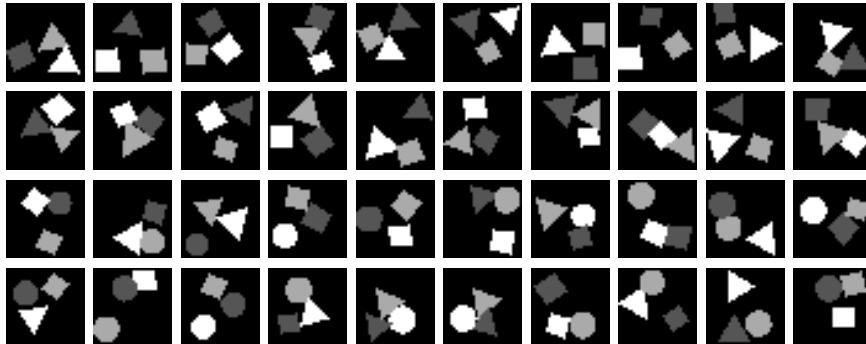

Figure 2: Some random images as realizations of the random variable $\mathbf{X}$ for the first classification task, where the first two rows show images of class 0 and the two lower rows show images of class 1.
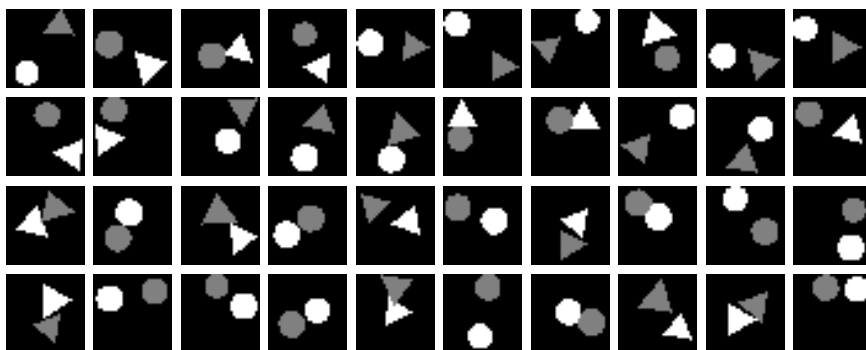

Figure 3: Some random images as realizations of the random variable $\mathbf{X}$ for the second classification task, where the first two rows show images of class 0 and the two lower rows show images of class 1.

In our second classification task we determine whether an image consists of two equal geometric objects. The first differences to the above problem is that only the two geometric objects circle and triangle are available and each image contains only two geometric objects. Apart from that, the images are generated in the same way as above with the difference that the two objects are choosen with the probability $p = 0.5$ each and the list of grey scale values only consists of the values $\frac{1}{2}$ and 1. Again, label Y is discrete and is uniformly distributed on $\{0, 1\}$, since the probability that the image does contain two identical objects is given by $2 \cdot p^2 = 0.5$.

We conjecture that the aposteriori probability of the first classification task satisfies our generalized hierarchical max-pooling model of order 1, since only one object has to be

detected. To solve our second classification task, we apply a function to the information about the existence of the two objects. Therefore, we conjecture that for the second classification task the aposteriori probability satisfies our generalized hierarchical max-pooling model of order 2.

Since all classifiers, i.e., ours and the classifiers we compare ourselves to, depend on parameters that influence their behavior, we choose some parameters in data-dependent manner by sample splitting. This means that we train the classifiers with a training set of $n_{train} = \lfloor \frac{4}{5} \cdot n \rfloor$ realizations several times with different choices for the parameters each time and test with a validation set consisting of $n_{val} = n - n_{train}$ realizations which parameters we should use. Then we train the classifiers with the selected parameters on the entire training set consisting of n realizations. First we describe how to choose the parameters in the convolutional part of our network, which depend on the level l and order d^* of the generalized hierarchical max-pooling model. We adaptively choose $l \in \{2, 3, 4\}$ and $t \in \{1, 2\}$. As in our theoretical result the filter sizes M_r have the values $2^1, 2^2, \ldots, 2^l$ for $r \in \{1, \ldots, L_1\}$, where the filter sizes grow with increasing r. To simplify the architecture of our classifier, each value of the filter sizes is repeated $L_n \in \{1, 2, 3\}$ times. The number of layers in the convolutional part is then given by $L_1 = L_n \cdot l$. We determine the number of channels in each convolutional layer from $k^{(1)} \in \{2, 4, 8\}$. Furthermore, we choose the number of layers in the dense part by L_n and the number of neurons by $k^{(2)} \in \{5, 10\}$ for each layer. To avoid overparameterization, we only use those parameter combinations for which the total number of trainable parameters of our model does not exceed the size of the training data set. To approximate the minimum of the least squares problem (5), we use the stochastic gradient descent method *Adam* from the Keras library.

We compare the results of our estimate (abbr. *neural-c*) with other conventional classification methods. Firstly, we consider a fully connected standard feedforward neural network (abbr. *neural-s*) with an adaptively chosen number of hidden layers and neurons per layer. We choose the number of hidden layers from $\{1, 2, \ldots, 8\}$ and the number of neurons per layer from $\{10, 20, 50, 100, 200\}$. We have implemented both the above approach and our convolutional neural network classifier, using the Keras library in Python. As a second alternative approach, we consider a support vector machine (abbr. *svm-rbf*) using a Gaussian radial basis function kernel and polynomial kernel (abbr. *svm-p*) with a degree adaptively choosen from $\{1, 2, 3, 4\}$. The parameter C which controls the importance of the regularization term and the kernel coefficient γ we adaptively choose from $\{10^{-2}, 10^{-1}, 1, 10\}$ and $\{10^{-2}, 10^{-1}, 1, 10\}$, respectively for both variants of the support vector machines approach. For its computation we use the function *SVC* integrated in the Python library scikit-learn. We also compare our estimate with a k_n- nearest neighbors classification estimate (abbr. *neighbor*) with an adaptively choosen k_n from $\{1, 2, 3\} \cup \{4, 8, 12, 16, \ldots, 4 \cdot \lfloor \frac{n_{train}}{4} \rfloor\}$, using the function *KNeighborsClassifier* from the scikit-learn library. Finally we compare our estimate to a random forest classifier (abbr. *rand-f*). We adaptively choose the maximum number of leaf nodes and the number of trees in the forest from $\{8, 16, 32\}$ and $\{50, 100, 200\}$, respectively and use the *RandomForestClassifier* function from the scikit-learn library to compute our classifier.

The quality of each estimate is measured by its empirical misclassification risk

$$\epsilon_N = \frac{1}{N} \sum_{k=1}^{N} \mathbb{I}_{\{f_n(\mathbf{x}_{n+k}) \neq y_{n+k}\}} \tag{7}$$

where f_n is the considered estimate based on the training set and

$$(\mathbf{x}_{n+1}, y_{n+1}), \ldots, (\mathbf{x}_{n+N}, y_{n+N})$$

are newly generated independent realizations of the random variable $(\mathbf{X}, Y)$, i. e. different from the n labeled training images. We choose $N = 10^5$. Since our results depend on randomly selected data, we calculate the estimators and their errors (7) based on 25 independently generated data sets $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n+N}, y_{n+N})\}$. Table 1 lists the median and interquartile range (IQR) of all runs.

	task 1		task 2	
sample size	$n = 1000$	$n = 2000$	$n = 1000$	$n = 2000$
approach	median (IQR)	median (IQR)	median (IQR)	median (IQR)
neural-c	**0.05 (0.02)**	**0.02 (0.01)**	**0.05 (0.05)**	**0.02(0.01)**
neural-s	0.46 (0.01)	0.45 (0.01)	0.50 (0.02)	0.50(0.01)
neighbor	0.48 (0.01)	0.46 (0.01)	0.50 (0.01)	0.50(0.01)
rand-f	0.46 (0.01)	0.45 (0.02)	0.50 (0.01)	0.50(0.01)
svm-p	0.42 (0.01)	0.39 (0.01)	0.50 (0.01)	0.50(0.01)
svm-rbf	0.50 (0.01)	0.49 (0.01)	0.50 (0.01)	0.50(0.01)

Table 1: Median and interquartile range of the empirical misclassification risk ϵ_N.

We observe that our convolutional neural network classifier (*neural-c*) outperforms the other approaches in both classification tasks. The errors of our classifier are 8 to 25 times smaller than the erros of the other approaches. The relative improvement of our classifier with increasing sample size is much larger than the relative improvements of the other approaches. This could indicate that our classifier also has a better rate of convergence. In the second classification task all approaches except our classifier, are not able to achieve satisfactory results, since the errors of these estimates corresponds to the expected error of a classifier which always estimates the same class.

5 Application to real images

In this section we test the different image classification methods on real data to show the practical relevance of our classifier. We consider the CIFAR-10 data set described in Krizhevsky (2009). It contains $60,000$ images, which consist of 10 different classes. We limit ourselves here to only two of these classes ($12,000$ images). One class contains images of cars and the other class contains images of ships. The size of each image is 32×32 pixels. Since the images are in color, we have converted them to grey scale.
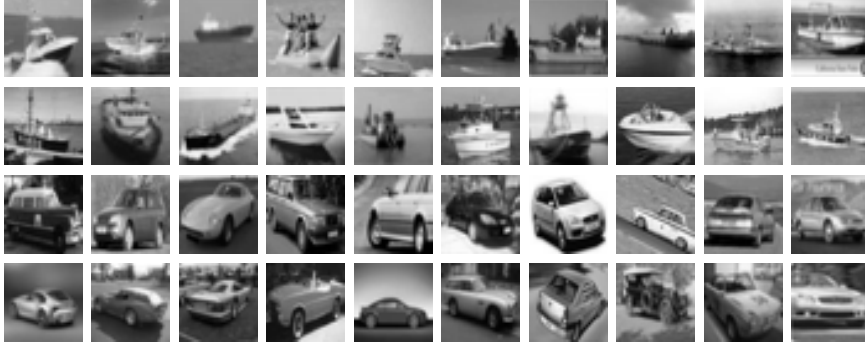
Figure 4: The first two rows show some images of the ships and the lower two rows show images of the cars of the grey scaled CIFAR-10 data set.

The different approaches of classification, as well as the parameter sets we use, are described in the simulation study in Section 4. We choose $n = 2,000$, $n_{train} = 1,600$ and $n_{val} = 400$ to train our classifiers. We calculate the empirical misclassification risk (7) using the remaining N=10,000 images. The results of all approaches are summarized in Table 2.

neural-c	neural-s	neighbor	rand-f	svm-p	svm-rbf
0.16	0.22	0.36	0.22	0.28	0.30

Table 2: The empirical misclassification risk ϵ_N for each estimate based on the presented grey scaled subset of the CIFAR-10 data set.

Again we observe that our estimate outperforms the others. This time, contrary to the synthetic image data sets, the error is only 1.4 times smaller than the error of the two second best approaches (the fully connected standard feedforward neural network and the random forest classifier). However, in the case of the real images we do not know for which model parameters the aposteriori probability could satisfy our generalized hierarchical max-pooling model. In particular, we have only tested the values $t \in \{1, 2\}$ which correspond to the orders $d^* \in \{1, 2\}$ of our generalized hierarchical max-pooling model. Moreover, the errors of the other approaches are much smaller than their errors in the two classification tasks of the synthetic image data sets.

6 Proofs

6.1 Auxiliary results

In this subsection we present several auxiliary results from the literature which we will use in the proof of Theorem 1. Our first result is a well-known bound on the misclassification risk of the plug-in classifiers.

Lemma 1 *Define* $(\mathbf{X}, Y)$, $(\mathbf{X}_1, Y_1)$, ..., $(\mathbf{X}_n, Y_n)$, *and* $\mathcal{D}_n$, η, f^* *and* f_n *as in Section 1. Then*

$$
\begin{aligned}
\mathbf{P}\{f_n(\mathbf{X}) \neq Y | \mathcal{D}_n\} - \mathbf{P}\{f^*(\mathbf{X}) \neq Y\} &\leq 2 \cdot \int |\eta_n(\mathbf{x}) - \eta(\mathbf{x})| \, \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \\
&\leq 2 \cdot \sqrt{\int |\eta_n(\mathbf{x}) - \eta(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x})}
\end{aligned}
$$

holds.

Proof. See Theorem 1.1 in Györfi et al. (2002). $\qquad\square$

Our next result is a bound on the expected L_2 error of the (truncated) least squares regression estimate.

Lemma 2 *Let* $(\mathbf{X}, Y)$, $(\mathbf{X}_1, Y_1)$, ..., $(\mathbf{X}_n, Y_n)$ *be independent and identically distributed* $\mathbb{R}^d \times \mathbb{R}$*-valued random variables. Assume that the distribution of* $(\mathbf{X}, Y)$ *satisfies*

$$
\mathbf{E}\{\exp(c_3 \cdot Y^2)\} < \infty
$$

for some constant $c_3 > 0$ *and that the regression function* $m(\cdot) = \mathbf{E}\{Y | \mathbf{X} = \cdot\}$ *is bounded in absolute value. Let* $\tilde{m}_n$ *be the least squares estimate*

$$
\tilde{m}_n(\cdot) = \arg \min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(\mathbf{X}_i)|^2
$$

based on some function space $\mathcal{F}_n$ *consisting of functions* $f : \mathbb{R}^d \to \mathbb{R}$ *and set* $m_n = T_{c_4 \cdot \log(n)} \tilde{m}_n$ *for some constant* $c_4 > 0$*. Then* m_n *satisfies*

$$
\begin{aligned}
&\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \\
&\leq \frac{c_5 \cdot (\log(n))^2 \cdot \sup_{\mathbf{x}_1^n \in (\mathbb{R}^d)^n} \left(\log \left(\mathcal{N}_1 \left(\frac{1}{n \cdot c_4 \log(n)}, T_{c_4 \log(n)} \mathcal{F}_n, \mathbf{x}_1^n \right) \right) + 1 \right)}{n} \\
&\quad + 2 \cdot \inf_{f \in \mathcal{F}_n} \int |f(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x})
\end{aligned}
$$

for $n > 1$ *and some constant* $c_5 > 0$*, which does not depend on* n *or the parameters of the estimate.*

Proof. This result follows in a straightforward way from the proof of Theorem 1 in Bagirov, Clausen and Kohler (2009). A complete proof can be found in the supplement of Bauer and Kohler (2019). $\qquad\square$

Our third auxiliary result is an approximation result for (p, C)–smooth functions by very deep feedforward neural networks.

Lemma 3 *Let $d \in \mathbb{N}$, let $f : \mathbb{R}^d \to \mathbb{R}$ be (p, C)–smooth for some $p = q + s$, $q \in \mathbb{N}_0$ and $s \in (0, 1]$, and $C > 0$. Let $M \in \mathbb{N}$ with $M > 1$ sufficiently large, where*

$$M^{2p} \geq c_5 \cdot \left(\max \left\{ 2, \sup_{\substack{\mathbf{x} \in [-2,2]^d \\ (l_1, \dots, l_d) \in \mathbb{N}^d \\ l_1 + \cdots + l_d \leq q}} \left| \frac{\partial^{l_1 + \cdots + l_d} f}{\partial^{l_1} x^{(1)} \dots \partial^{l_d} x^{(d)}} (\mathbf{x}) \right| \right\} \right)^{4(q+1)}$$

must hold for some sufficiently large constant $c_5 \geq 1$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be the ReLU activation function

$$\sigma(x) = \max\{x, 0\}$$

and let $L, r \in \mathbb{N}$ such that

(i)

$$
\begin{aligned}
L \geq &5M^d + \left\lceil \log_4 \left(M^{2p + 4 \cdot d \cdot (q+1)} \cdot e^{4(q+1) \cdot (M^d - 1)} \right) \right\rceil \\
&\cdot \lceil \log_2(\max\{d, q\} + 2) \rceil + \lceil \log_4(M^{2p}) \rceil
\end{aligned}
$$

(ii)

$$r \geq 132 \cdot 2^d \cdot \lceil e^d \rceil \cdot \binom{d + q}{d} \cdot \max\{q + 1, d^2\}$$

hold. Then there exists a feedforward neural network

$$f_{net} \in \mathcal{G}_d(L, \mathbf{k})$$

with $\mathbf{k} = (k_1, \dots, k_L)$ and $k_1 = \cdots = k_L = r$ such that

$$
\begin{aligned}
&\sup_{\mathbf{x} \in [-2,2]^d} |f(\mathbf{x}) - f_{net}(\mathbf{x})| \\
&\leq c_6 \cdot \left(\max \left\{ 2, \sup_{\substack{\mathbf{x} \in [-a,a]^d \\ (l_1, \dots, l_d) \in \mathbb{N}^d \\ l_1 + \cdots + l_d \leq q}} \left| \frac{\partial^{l_1 + \cdots + l_d} f}{\partial^{l_1} x^{(1)} \dots \partial^{l_d} x^{(d)}} (\mathbf{x}) \right| \right\} \right)^{4(q+1)} \cdot M^{-2p}.
\end{aligned}
$$

Proof. See Theorem 2 in Kohler and Langer (2019). An alternative proof of a closely related result can be found in Yarotsky and Zhevnerchuk (2019), see Theorem 4.1 therein.

$\square$

6.2 An approximation result for convolutional neural networks

In this subsection we describe in Lemma 5 below a connection between fully connected neural networks and convolutional neural networks, which will enbable us to derive in the proof of Theorem 1 an approximation result for the generalized hierarchical max-pooling models by the convolutional neural networks. Before we do this we present a bound on the error we make in case that we replace the functions $g_{k,s}$ in a hierarchical model by some approximations of them.

Lemma 4 *Let $d_1, d_2, t \in \mathbb{N}$ and $l \in \mathbb{N}$ with $2^l \leq \min\{d_1, d_2\}$. For $a \in \{1, \dots, t\}$, set $I = \{0, 1, \dots, 2^l - 1\} \times \{0, 1, \dots, 2^l - 1\}$ and define*

$$m_a(\mathbf{x}) = \max_{(i,j) \in \mathbb{Z}^2 \,:\, (i,j)+I \subseteq \{1,\dots,d_1\} \times \{1,\dots,d_2\}} f_a\left(x_{(i,j)+I}\right)$$

and

$$\bar{m}_a(\mathbf{x}) = \max_{(i,j) \in \mathbb{Z}^2 \,:\, (i,j)+I \subseteq \{1,\dots,d_1\} \times \{1,\dots,d_2\}} \bar{f}_a\left(x_{(i,j)+I}\right),$$

where f_a and $\bar{f}_a$ satisfy

$$f_a = f_{l,1}^{(a)} \quad and \quad \bar{f}_a = \bar{f}_{l,1}^{(a)}$$

for some $f_{k,s}^{(a)}, \bar{f}_{k,s}^{(a)} : \mathbb{R}^{\{1,\dots,2^k\} \times \{1,\dots,2^k\}} \to \mathbb{R}$ recursively defined by

$$
\begin{aligned}
f_{k,s}^{(a)}(\mathbf{x}) \;=\; g_{k,s}^{(a)}\big(& f_{k-1,4\cdot(s-1)+1}^{(a)}(x_{\{1,\dots,2^{k-1}\}\times\{1,\dots,2^{k-1}\}}), \\
& f_{k-1,4\cdot(s-1)+2}^{(a)}(x_{\{2^{k-1}+1,\dots,2^k\}\times\{1,\dots,2^{k-1}\}}), \\
& f_{k-1,4\cdot(s-1)+3}^{(a)}(x_{\{1,\dots,2^{k-1}\}\times\{2^{k-1}+1,\dots,2^k\}}), \\
& f_{k-1,4\cdot s}^{(a)}(x_{\{2^{k-1}+1,\dots,2^k\}\times\{2^{k-1}+1,\dots,2^k\}}))
\end{aligned}
$$

and

$$
\begin{aligned}
\bar{f}_{k,s}^{(a)}(\mathbf{x}) \;=\; \bar{g}_{k,s}^{(a)}\big(& \bar{f}_{k-1,4\cdot(s-1)+1}^{(a)}(x_{\{1,\dots,2^{k-1}\}\times\{1,\dots,2^{k-1}\}}), \\
& \bar{f}_{k-1,4\cdot(s-1)+2}^{(a)}(x_{\{2^{k-1}+1,\dots,2^k\}\times\{1,\dots,2^{k-1}\}}), \\
& \bar{f}_{k-1,4\cdot(s-1)+3}^{(a)}(x_{\{1,\dots,2^{k-1}\}\times\{2^{k-1}+1,\dots,2^k\}}), \\
& \bar{f}_{k-1,4\cdot s}^{(a)}(x_{\{2^{k-1}+1,\dots,2^k\}\times\{2^{k-1}+1,\dots,2^k\}}))
\end{aligned}
$$

for $k = 2, \dots, l, s = 1, \dots, 4^{l-k}$, and

$$f_{1,s}^{(a)}(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) = g_{1,s}^{(a)}(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2})$$

and

$$\bar{f}_{1,s}^{(a)}(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) = \bar{g}_{1,s}^{(a)}(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2})$$

for $s = 1, \dots, 4^{l-1}$, where

$$g_{k,s}^{(a)} : \mathbb{R}^4 \to [0,1] \quad and \quad \bar{g}_{k,s}^{(a)} : \mathbb{R}^4 \to \mathbb{R}$$

are functions for $a \in \{1, \ldots, t\}$, $k \in \{1, \ldots, l\}$ and $s \in \{1, \ldots, 4^{l-k}\}$. Furthermore, let $g : \mathbb{R}^t \to [0,1]$ and $\bar{g} : \mathbb{R}^t \to \mathbb{R}$ be functions. Assume that all restrictions $g_{k,s}^{(a)}|_{[-2,2]^4} : [-2,2]^4 \to [0,1]$ and $g|_{[-2,2]^t} : [-2,2]^t \to [0,1]$ are Lipschitz continuous regarding the Euclidean distance with Lipschitz constant $C > 0$ and for all $a \in \{1, \ldots, t\}$, $k \in \{1, \ldots, l\}$ and $s \in \{1, \ldots, 4^{l-k}\}$ we assume that

$$\left\| \bar{g}_{k,s}^{(a)} \right\|_{[-2,2]^4, \infty} \le 2. \tag{8}$$

Then for any $\mathbf{x} \in [0,1]^{\{1,\ldots,d_1\} \times \{1,\ldots,d_2\}}$ it holds:

$$\begin{aligned}
&|g(m_1(\mathbf{x}), \ldots, m_t(\mathbf{x})) - \bar{g}(\bar{m}_1(\mathbf{x}), \ldots, \bar{m}_t(\mathbf{x}))| \\
&\le \sqrt{t} \cdot (2C+1)^l \\
&\quad \cdot \max_{a \in \{1,\ldots,t\}, j \in \{1,\ldots,l\}, s \in \{1,\ldots,4^{l-j}\}} \left\{ \| g_{j,s}^{(a)} - \bar{g}_{j,s}^{(a)} \|_{[-2,2]^4, \infty}, \| g - \bar{g} \|_{[-2,2]^t, \infty} \right\}.
\end{aligned}$$

Proof. Firstly, we show for any $a \in \{1, \ldots, t\}$ that

$$|m_a(\mathbf{x}) - \bar{m}_a(\mathbf{x})| \le (2C+1)^{l-1} \cdot \max_{j \in \{1,\ldots,l\}, s \in \{1,\ldots,4^{l-j}\}} \| g_{j,s}^{(a)} - \bar{g}_{j,s}^{(a)} \|_{[-2,2]^4, \infty}. \tag{9}$$

If a_1, b_1, $\ldots$, a_n, $b_n \in \mathbb{R}$, then

$$\left| \max_{i=1,\ldots,n} a_i - \max_{i=1,\ldots,n} b_i \right| \le \max_{i=1,\ldots,n} |a_i - b_i|.$$

Indeed, in case $a_1 = \max_{i=1,\ldots,n} a_i \ge \max_{i=1,\ldots,n} b_i$ (which we can assume w.l.o.g.) we have

$$\left| \max_{i=1,\ldots,n} a_i - \max_{i=1,\ldots,n} b_i \right| = a_1 - \max_{i=1,\ldots,n} b_i \le a_1 - b_1 \le \max_{i=1,\ldots,n} |a_i - b_i|.$$

Consequently it suffices to show

$$\begin{aligned}
&\max_{(i,j) \in \mathbb{Z}^2 \,:\, (i,j)+I \subseteq \{1,\ldots,d_1\} \times \{1,\ldots,d_2\}} \left| f_a \left(x_{(i,j)+I} \right) - \bar{f}_a \left(x_{(i,j)+I} \right) \right| \\
&\le (2C+1)^{l-1} \cdot \max_{j \in \{1,\ldots,l\}, s \in \{1,\ldots,4^{l-j}\}} \| g_{j,s}^{(a)} - \bar{g}_{j,s}^{(a)} \|_{[-2,2]^4, \infty}.
\end{aligned}$$

This in turn follows from

$$|f_{k,s}^{(a)}(\mathbf{x}) - \bar{f}_{k,s}^{(a)}(\mathbf{x})| \le (2C+1)^{k-1} \cdot \max_{i \in \{1,\ldots,k\}, s \in \{1,\ldots,4^{l-i}\}} \| g_{i,s}^{(a)} - \bar{g}_{i,s}^{(a)} \|_{[-2,2]^4, \infty} \tag{10}$$

for all $k \in \{1, \ldots, l\}$, all $s \in \{1, \ldots, 4^{l-k}\}$ and all $\mathbf{x} \in [0,1]^{\{1,\ldots,2^k\} \times \{1,\ldots,2^k\}}$, which we show in the sequel by induction on k.

For $k = 1$ and $s \in \{1, \ldots, 4^{l-1}\}$ we have

$$\left| f_{1,s}^{(a)}(\mathbf{x}) - \bar{f}_{1,s}^{(a)}(\mathbf{x}) \right| = \left| g_{1,s}^{(a)}(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) - \bar{g}_{1,s}^{(a)}(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) \right|$$

$$\leq \ \left\|g_{1,s}^{(a)} - \bar{g}_{1,s}^{(a)}\right\|_{[0,1]^4,\infty}.$$

Assume now that (10) holds for some $k \in \{1, \ldots, l-1\}$. The definition of $\bar{f}_{k,s}^{(a)}$ and inequality (8) imply that

$$\left|\bar{f}_{k,s}^{(a)}(\mathbf{x})\right| \leq 2$$

for all $\mathbf{x} \in [0,1]^{\{1,\ldots,2^k\} \times \{1,\ldots,2^k\}}$ and $s \in \{1, \ldots, 4^{l-k}\}$. Then, the triangle inequality and the Lipschitz assumption on g imply

$$|f_{k+1,s}^{(a)}(\mathbf{x}) - \bar{f}_{k+1,s}^{(a)}(\mathbf{x})|$$

$$\leq \Big| g_{k+1,s}^{(a)}\big(f_{k,4\cdot(s-1)+1}^{(a)}(x_{\{1,\ldots,2^k\}\times\{1,\ldots,2^k\}}), f_{k,4\cdot(s-1)+2}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{1,\ldots,2^k\}}),$$

$$f_{k,4\cdot(s-1)+3}^{(a)}(x_{\{1,\ldots,2^k\}\times\{2^k+1,\ldots,2^{k+1}\}}), f_{k,4\cdot s}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{2^k+1,\ldots,2^{k+1}\}})\big)$$

$$-g_{k+1,s}^{(a)}\big(\bar{f}_{k,4\cdot(s-1)+1}^{(a)}(x_{\{1,\ldots,2^k\}\times\{1,\ldots,2^k\}}), \bar{f}_{k,4\cdot(s-1)+2}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{1,\ldots,2^k\}}),$$

$$\bar{f}_{k,4\cdot(s-1)+3}^{(a)}(x_{\{1,\ldots,2^k\}\times\{2^k+1,\ldots,2^{k+1}\}}), \bar{f}_{k,4\cdot s}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{2^k+1,\ldots,2^{k+1}\}})\big)\Big|$$

$$+\Big| g_{k+1,s}^{(a)}\big(\bar{f}_{k,4\cdot(s-1)+1}^{(a)}(x_{\{1,\ldots,2^k\}\times\{1,\ldots,2^k\}}), \bar{f}_{k,4\cdot(s-1)+2}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{1,\ldots,2^k\}}),$$

$$\bar{f}_{k,4\cdot(s-1)+3}^{(a)}(x_{\{1,\ldots,2^k\}\times\{2^k+1,\ldots,2^{k+1}\}}), \bar{f}_{k,4\cdot s}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{2^k+1,\ldots,2^{k+1}\}})\big)$$

$$-\bar{g}_{k+1,s}^{(a)}\big(\bar{f}_{k,4\cdot(s-1)+1}^{(a)}(x_{\{1,\ldots,2^k\}\times\{1,\ldots,2^k\}}), \bar{f}_{k,4\cdot(s-1)+2}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{1,\ldots,2^k\}}),$$

$$\bar{f}_{k,4\cdot(s-1)+3}^{(a)}(x_{\{1,\ldots,2^k\}\times\{2^k+1,\ldots,2^{k+1}\}}), \bar{f}_{k,4\cdot s}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{2^k+1,\ldots,2^{k+1}\}})\big)\Big|$$

$$\leq C \cdot \Big(|f_{k,4\cdot(s-1)+1}^{(a)}(x_{\{1,\ldots,2^k\}\times\{1,\ldots,2^k\}}) - \bar{f}_{k,4\cdot(s-1)+1}^{(a)}(x_{\{1,\ldots,2^k\}\times\{1,\ldots,2^k\}})|^2$$

$$+|f_{k,4\cdot(s-1)+2}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{1,\ldots,2^k\}}) - \bar{f}_{k,4\cdot(s-1)+2}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{1,\ldots,2^k\}})|^2$$

$$+|f_{k,4\cdot(s-1)+3}^{(a)}(x_{\{1,\ldots,2^k\}\times\{2^k+1,\ldots,2^{k+1}\}}) - \bar{f}_{k,4\cdot(s-1)+3}^{(a)}(x_{\{1,\ldots,2^k\}\times\{2^k+1,\ldots,2^{k+1}\}})|^2$$

$$+|f_{k,4\cdot s}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{2^k+1,\ldots,2^{k+1}\}}) - \bar{f}_{k,4\cdot s}^{(a)}(x_{\{2^k+1,\ldots,2^{k+1}\}\times\{2^k+1,\ldots,2^{k+1}\}})|^2 \Big)^{1/2}$$

$$+\|g_{k+1,s}^{(a)} - \bar{g}_{k+1,s}^{(a)}\|_{[-2,2]^4,\infty}$$

$$\leq (2 \cdot C) \cdot (2C+1)^{k-1} \cdot \max_{i \in \{1,\ldots,k\}, s \in \{1,\ldots,4^{l-i}\}} \|g_{i,s}^{(a)} - \bar{g}_{i,s}^{(a)}\|_{[-2,2]^4,\infty}$$

$$+\|g_{k+1,s}^{(a)} - \bar{g}_{k+1,s}^{(a)}\|_{[-2,2]^4,\infty}$$

$$\leq (2C+1)^k \cdot \max_{i \in \{1,\ldots,k+1\}, s \in \{1,\ldots,4^{l-i}\}} \|g_{i,s}^{(a)} - \bar{g}_{i,s}^{(a)}\|_{[-2,2]^4,\infty}$$

for all $\mathbf{x} \in [0,1]^{\{1,\ldots,2^{k+1}\} \times \{1,\ldots,2^{k+1}\}}$.

The definition of the functions $\bar{f}_{k,s}^{(a)}$ and inequality (8) imply that

$$|\bar{m}_a(\mathbf{x})| \leq 2$$

21

for all $\mathbf{x} \in [0,1]^{\{1,\ldots,d_1\} \times \{1,\ldots,d_2\}}$ and $a \in \{1,\ldots,t\}$. Then, the triangle inequality, the Lipschitz assumption on g and inequality (9) imply

$$
\begin{aligned}
&|g(m_1(\mathbf{x}),\ldots,m_t(\mathbf{x})) - \bar{g}(\bar{m}_1(\mathbf{x}),\ldots,\bar{m}_t(\mathbf{x}))| \\
&\leq |g(m_1(\mathbf{x}),\ldots,m_t(\mathbf{x})) - g(\bar{m}_1(\mathbf{x}),\ldots,\bar{m}_t(\mathbf{x}))| \\
&\quad + |g(\bar{m}_1(\mathbf{x}),\ldots,\bar{m}_t(\mathbf{x})) - \bar{g}(\bar{m}_1(\mathbf{x}),\ldots,\bar{m}_t(\mathbf{x}))| \\
&\leq C \cdot \left(|m_1(\mathbf{x}) - \bar{m}_1(\mathbf{x})|^2 + \cdots + |m_t(\mathbf{x}) - \bar{m}_t(\mathbf{x})|^2\right)^{1/2} \\
&\quad + \|g - \bar{g}\|_{[-2,2]^t, \infty} \\
&\leq \sqrt{t} \cdot C \cdot (2C+1)^{l-1} \cdot \max_{a \in \{1,\ldots,t\}, j \in \{1,\ldots,l\}, s \in \{1,\ldots,4^{l-j}\}} \|g_{j,s}^{(i)} - \bar{g}_{j,s}^{(i)}\|_{[-2,2]^4, \infty} \\
&\quad + \|g - \bar{g}\|_{[-2,2]^t, \infty} \\
&\leq \sqrt{t} \cdot (2C+1)^l \\
&\quad \cdot \max_{\substack{a \in \{1,\ldots,t\}, \\ j \in \{1,\ldots,l\}, s \in \{1,\ldots,4^{l-j}\}}} \left\{ \|g_{j,s}^{(a)} - \bar{g}_{j,s}^{(a)}\|_{[-2,2]^4, \infty}, \|g - \bar{g}\|_{[-2,2]^t, \infty} \right\}
\end{aligned}
$$

for all $\mathbf{x} \in [0,1]^{\{1,\ldots,d_1\} \times \{1,\ldots,d_2\}}$. $\qquad \square$

Lemma 5 *Let $d_1, d_2, l \in \mathbb{N}$ with $2^l \leq \min\{d_1, d_2\}$. For $k \in \{1,\ldots,l\}$ and $s \in \{1,\ldots,4^{l-k}\}$ let*

$$
\bar{g}_{net,k,s} : \mathbb{R}^4 \to \mathbb{R}
$$

be defined by a feedforward neural network with $L_{net} \in \mathbb{N}$ hidden layers and $r_{net} \in \mathbb{N}$ neurons per hidden layer and ReLU activation function. Set

$$
I = \left\{0,\ldots,2^l - 1\right\} \times \left\{0,\ldots,2^l - 1\right\}
$$

and define $\bar{m} : [0,1]^{\{1,\ldots,d_1\} \times \{1,\ldots,d_2\}} \to \mathbb{R}$ by

$$
\bar{m}(\mathbf{x}) = \max_{(i,j) \in \mathbb{Z}^2 \,:\, (i,j)+I \subseteq \{1,\ldots,d_1\} \times \{1,\ldots,d_2\}} \bar{f}\left(x_{(i,j)+I}\right),
$$

where $\bar{f}$ satisfies

$$
\bar{f} = \bar{f}_{l,1}
$$

for some $\bar{f}_{k,s} : [0,1]^{\{1,\ldots,2^k\} \times \{1,\ldots,2^k\}} \to \mathbb{R}$ recursively defined by

$$
\begin{aligned}
\bar{f}_{k,s}(\mathbf{x}) = \ \bar{g}_{net,k,s}\big(&\bar{f}_{k-1,4\cdot(s-1)+1}(\mathbf{x}_{\{1,\ldots,2^{k-1}\} \times \{1,\ldots,2^{k-1}\}}), \\
&\bar{f}_{k-1,4\cdot(s-1)+2}(\mathbf{x}_{\{2^{k-1}+1,\ldots,2^k\} \times \{1,\ldots,2^{k-1}\}}), \\
&\bar{f}_{k-1,4\cdot(s-1)+3}(\mathbf{x}_{\{1,\ldots,2^{k-1}\} \times \{2^{k-1}+1,\ldots,2^k\}}), \\
&\bar{f}_{k-1,4\cdot s}(\mathbf{x}_{\{2^{k-1}+1,\ldots,2^k\} \times \{2^{k-1}+1,\ldots,2^k\}})\big)
\end{aligned}
$$

for $k = 2,\ldots,l$, $s = 1,\ldots,4^{l-k}$, and

$$
\bar{f}_{1,s}(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) = \bar{g}_{net,1,s}(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2})
$$

for $s = 1, \ldots, 4^{l-1}$. Set

$$l_{net} = \frac{4^l - 1}{3} \cdot L_{net} + l,$$

$$k_s = \frac{2 \cdot 4^l + 4}{3} + r_{net} \quad (s = 1, \ldots, l_{net}),$$

and set

$$M_s = 2^{\pi(s)} \quad for \ s \in \{1, \ldots, l_{net}\},$$

where the function $\pi : \{1, \ldots, l_{net}\} \to \{1, \ldots, l\}$ is defined by

$$\pi(s) = \sum_{i=1}^{l} \mathbb{I}_{\left\{ s \geq i + \sum_{r=l-i+1}^{l-1} 4^r \cdot L_{net} \right\}}.$$

Then there exists some $m_{net} \in \mathcal{F}(l_{net}, \mathbf{k}, \mathbf{M})$ such that

$$\bar{m}(\mathbf{x}) = m_{net}(\mathbf{x})$$

holds for all $\mathbf{x} \in [0, 1]^{\{1, \ldots, d_1\} \times \{1, \ldots, d_2\}}$.

In order to prove Lemma 5 we will use the following auxiliary result.

Lemma 6 *Let $g_{net} : \mathbb{R}^4 \to \mathbb{R}$ be a standard feedforward neural network with $L_{net} \in \mathbb{N}$ hidden layers and $r_{net} \in \mathbb{N}$ neurons per hidden layer. Let $d_1, d_2 \in \mathbb{N}$ with $d_1, d_2 > 1$ and let $\sigma(x) = \max\{x, 0\}$ be the ReLU activation function. We assume that there is given a convolutional neural network $m_{net} \in \mathcal{F}(L, \mathbf{k}, \mathbf{M})$ with $L = r_0 + L_{net} + 1$ convolutional layers and $k_r = t + r_{net}$ channels in the convolutional layer r $(r = 1, \ldots, r_0 + L_{net} + 1)$ for $r_0, t \in \mathbb{N}$, and filter sizes $M_1, \ldots, M_{r_0 + L_{net} + 1} \in \mathbb{N}$ with*

$$M_{r_0 + 1} = 2^k \ for \ some \ k \in \mathbb{N} \ with \ 2^k \leq \min\{d_1, d_2\}.$$

The convolutional neural network m_{net} is given by its weight matrix

$$\mathbf{w} = \left(w_{i,j,s_1,s_2}^{(r)} \right)_{1 \leq i, j \leq M_r, s_1 \in \{1, \ldots, k_{r-1}\}, s_2 \in \{1, \ldots, k_r\} r \in \{1, \ldots, r_0 + L_{net} + 1\}}, \tag{11}$$

and its bias weights

$$\mathbf{w}_{bias} = \left(w_{s_2}^{(r)} \right)_{s_2 \in \{1, \ldots, k_r\}, r \in \{1, \ldots, r_0 + L_{net} + 1\}}. \tag{12}$$

Set $I^{(m)} = \{0, \ldots, 2^m - 1\} \times \{0, \ldots 2^m - 1\}$ for $m \in \mathbb{N}_0$. Furthermore, let $f_1, \ldots, f_4 : [0, 1]^{(1,1) + I^{(k-1)}} \to \mathbb{R}$ be functions and let $s_{2,1}, \ldots, s_{2,10} \in \{1, \ldots, t\}$. Assume that the given convolutional neural network m_{net} satisfies the following four conditions for all $(i_2, j_2) \in \{1, \ldots, d_1 - 2^k + 1\} \times \{1, \ldots, d_2 - 2^k + 1\}$:

$$o_{(i_2, j_2), s_{2,1}}^{(r_0)} - o_{(i_2, j_2), s_{2,2}}^{(r_0)} = f_1(x_{(i_2, j_2) + I^{(k-1)}}), \tag{13}$$

23

$$o^{(r_0)}_{(i_2+2^{k-1},j_2),s_{2,3}} - o^{(r_0)}_{(i_2+2^{k-1},j_2),s_{2,4}} = f_2\big(x_{(i_2+2^{k-1},j_2)+I^{(k-1)}}\big), \tag{14}$$

$$o^{(r_0)}_{(i_2,j_2+2^{k-1}),s_{2,5}} - o^{(r_0)}_{(i_2,j_2+2^{k-1}),s_{2,6}} = f_3\big(x_{(i_2,j_2+2^{k-1})+I^{(k-1)}}\big) \tag{15}$$

and

$$o^{(r_0)}_{(i_2+2^{k-1},j_2+2^{k-1}),s_{2,7}} - o^{(r_0)}_{(i_2+2^{k-1},j_2+2^{k-1}),s_{2,8}} = f_4\big(x_{(i_2+2^{k-1},j_2+2^{k-1})+I^{(k-1)}}\big). \tag{16}$$

Then we are able to modify the weights (11) *and* (12)

$$w^{(r)}_{t_1,t_2,s_1,s_2}, w^{(r)}_{s_2} \quad (s_1 \in \{1,\dots,t+r_{net}\}) \tag{17}$$

in layers $r \in \{r_0+1,\dots,r_0+L_{net}+1\}$ *and in channels* $s_2 \in \{s_{2,9}, s_{2,10}, t+1, \dots, t+r_{net}\}$
such that

$$o^{(r_0+L_{net}+1)}_{(i_2,j_2),s_{2,9}} - o^{(r_0+L_{net}+1)}_{(i_2,j_2),s_{2,10}}$$
$$= g_{net}\Big(f_1(x_{(i_2,j_2)+I^{(k-1)}}), f_2(x_{(i_2+2^{k-1},j_2)+I^{(k-1)}}),$$
$$f_3(x_{(i_2,j_2+2^{k-1})+I^{(k-1)}}), f_4(x_{(i_2+2^{k-1},j_2+2^{k-1})+I^{(k-1)}})\Big)$$

holds for all $(i_2,j_2) \in \{1,\dots,d_1-2^k+1\} \times \{1,\dots,d_2-2^k+1\}$.

Remark 6. In the proof of Lemma 6 we only modify in layers $r_0+1,\dots,r_0+L_{net}+1$ the filters and bias weights (17) in channels

$$t+1,\dots,t+r_{net}$$

and in layer $r_0 + L_{net} + 1$ the filters and bias weights in channels

$$s_{2,9}, s_{2,10}.$$

This means that the calculation only takes place in these channels. The filter and bias weights in the remaining channels can therefore be arbitrary.

Proof of Lemma 6. Let $(i_2,j_2) \in \{1,\dots,d_1-2^k+1\} \times \{1,\dots,d_2-2^k+1\}$ be arbitrary. We modify the weights (15) by using the weights of g_{net}. Here we assume that g_{net} is given by

$$g_{net}(\mathbf{x}) = \sum_{i=1}^{r_{net}} w^{(L_{net})}_{1,i} g^{(L_{net})}_i(\mathbf{x}) + w^{(L_{net})}_{1,0}$$

for $g^{(L_{net})}_i$'s recursively defined by

$$g^{(r)}_i(\mathbf{x}) = \sigma\left(\sum_{j=1}^{r_{net}} w^{(r-1)}_{i,j} g^{(r-1)}_j(\mathbf{x}) + w^{(r-1)}_{i,0}\right)$$

for $i \in \{1, \ldots, r_{net}\}$, $r \in \{2, \ldots, L_{net}\}$, and

$$g_i^{(1)}(\mathbf{x}) = \sigma\left(\sum_{j=1}^{4} w_{i,j}^{(0)} x^{(j)} + w_{i,0}^{(0)}\right) \quad (i \in \{1, \ldots, r_{net}\}).$$

In layer $r_0 + 1$ we modify the weights (17) in channel $t + i$ by setting

$$w_{t_1,t_2,s,t+i}^{(r_0+1)} = 0$$

for all $t_1, t_2 \notin \{1, 1 + 2^{k-1}\}$ and all $s \notin \{s_{2,1}, \ldots, s_{2,8}\}$ and choose the only nonzero weights by

$$
\begin{aligned}
&w_{1,1,s_{2,1},t+i}^{(r_0+1)} = w_{i,1}^{(0)}, && w_{1,1,s_{2,2},t+i}^{(r_0+1)} = -w_{i,1}^{(0)}, \\
&w_{1+2^{k-1},1,s_{2,3},t+i}^{(r_0+1)} = w_{i,2}^{(0)}, && w_{1+2^{k-1},1,s_{2,4},t+i}^{(r_0+1)} = -w_{i,2}^{(0)}, \\
&w_{1,1+2^{k-1},s_{2,5},t+i}^{(r_0+1)} = w_{i,3}^{(0)}, && w_{1,1+2^{k-1},s_{2,6},t+i}^{(r_0+1)} = -w_{i,3}^{(0)}, \\
&w_{1+2^{k-1},1+2^{k-1},s_{2,7},t+i}^{(r_0+1)} = w_{i,4}^{(0)}, && w_{1+2^{k-1},1+2^{k-1},s_{2,8},t+i}^{(r_0+1)} = -w_{i,4}^{(0)}
\end{aligned}
$$

and $w_{t+i}^{(r_0+1)} = w_{i,0}^{(0)}$ for $i \in \{1, \ldots, r_{net}\}$. Then we calculate with the modified weights and the assumptions (13)–(16)

$$
\begin{aligned}
o_{(i_2,j_2),t+i}^{(r_0+1)} =& \sigma\left(\sum_{s_1=1}^{t+r_{net}} \sum_{\substack{t_1,t_2 \in \{1,\ldots,M_{r_0+1}\} \\ (i_2+t_1-1,j_2+t_2-1) \in D}} w_{t_1,t_2,s_1,t+i}^{(r_0+1)} \cdot o_{(i_2+t_1-1,j_2+t_2-1),s_1}^{(r_0)} + w_{t+i}^{(r_0+1)}\right) \\
=& \sigma\left(w_{i,1}^{(0)} \cdot \left(o_{(i_2,j_2),s_{2,1}}^{(r_0)} - o_{(i_2,j_2),s_{2,2}}^{(r_0)}\right)\right. \\
& \quad + w_{i,2}^{(0)} \cdot \left(o_{(i_2+2^{k-1},j_2),s_{2,3}}^{(r_0)} - o_{(i_2+2^{k-1},j_2),s_{2,4}}^{(r_0)}\right) \\
& \quad + w_{i,3}^{(0)} \cdot \left(o_{(i_2,j_2+2^{k-1}),s_{2,5}}^{(r_0)} - o_{(i_2,j_2+2^{k-1}),s_{2,6}}^{(r_0)}\right) \\
& \quad \left. + w_{i,4}^{(0)} \cdot \left(o_{(i_2+2^{k-1},j_2+2^{k-1}),s_{2,7}}^{(r_0)} - o_{(i_2+2^{k-1},j_2+2^{k-1}),s_{2,8}}^{(r_0)}\right) + w_{i,0}^{(0)}\right) \\
=& \sigma\left(w_{i,1}^{(0)} f_1(x_{(i_2,j_2)+I^{(k-1)}}) + w_{i,2}^{(0)} f_2(x_{(i_2+2^{k-1},j_2)+I^{(k-1)}})\right. \\
& \quad \left. + w_{i,3}^{(0)} f_3(x_{(i_2,j_2+2^{k-1})+I^{(k-1)}}) + w_{i,4}^{(0)} f_4(x_{(i_2+2^{k-1},j_2+2^{k-1})+I^{(k-1)}}) + w_{i,0}^{(0)}\right) \\
=& g_i^{(1)}\left(f_1(x_{(i_2,j_2)+I^{(k-1)}}), f_2(x_{(i_2+2^{k-1},j_2)+I^{(k-1)}}),\right. \\
& \quad \left. f_3(x_{(i_2,j_2+2^{k-1})+I^{(k-1)}}), f_4(x_{(i_2+2^{k-1},j_2+2^{k-1})+I^{(k-1)}})\right)
\end{aligned}
$$

$$(18)$$

for $i \in \{1, \ldots, r_{net}\}$. In layers $r \in \{r_0 + 2, \ldots, r_0 + L_{net}\}$ in channel $t + i$ we modify the weights (17) by setting

$$w^{(r)}_{t_1,t_2,s,t+i} = 0$$

for all $(t_1, t_2) \neq (1, 1)$ and all $s \in \{1, \ldots, t\}$ and choose the only nonzero weights by

$$w^{(r)}_{1,1,t+j,t+i} = w^{(r-r_0-1)}_{i,j}, \quad w^{(r)}_{t+i} = w^{(r-r_0-1)}_{i,0} \quad (j \in \{1, \ldots, r_{net}\})$$

for $i \in \{1, \ldots, r_{net}\}$. Thus we obtain

$$o^{(r_0+r)}_{(i_2,j_2),t+i} = \sigma \left(\sum_{j=1}^{r_{net}} w^{(r-1)}_{i,j} o^{(r_0+r-1)}_{(i_2,j_2),t+j} + w^{(r-1)}_{i,0} \right)$$

for $i \in \{1, \ldots, r_{net}\}$ and $r \in \{2, \ldots, L_{net}\}$. Then we get by equation (18) and the definition of $g_i^{(r)}$ that

$$o^{(r_0+r)}_{(i_2,j_2),t+i} = g_i^{(r)} \Big(f_1(x_{(i_2,j_2)+I^{(k-1)}}), f_2(x_{(i_2+2^{k-1},j_2)+I^{(k-1)}}),$$
$$f_3(x_{(i_2,j_2+2^{k-1})+I^{(k-1)}}), f_4(x_{(i_2+2^{k-1},j_2+2^{k-1})+I^{(k-1)}}) \Big)$$

for $i \in \{1, \ldots, r_{net}\}$ and $r \in \{2, \ldots, L_{net}\}$. Now in layer $r_0 + L_{net} + 1$ in channels $s_{2,9}, s_{2,10} \in \{1, \ldots, t\}$ we modify the weights (17) by setting

$$w^{(r_0+L_{net}+1)}_{t_1,t_2,s,s_{2,9}} = w^{(r_0+L_{net}+1)}_{t_1,t_2,s,s_{2,10}} = 0$$

for all $(t_1, t_2) \neq (1, 1)$ and all $s \in \{1, \ldots, t\}$ and choose the only nonzero weights by

$$w^{(r_0+L_{net}+1)}_{1,1,t+i,s_{2,9}} = w^{(L_{net})}_{1,i}, \quad w^{(r_0+L_{net}+1)}_{1,1,t+i,s_{2,10}} = -w^{(L_{net})}_{1,i},$$
$$w^{(r_0+L_{net}+1)}_{s_{2,9}} = w^{(L_{net})}_{1,0}, \quad w^{(r_0+L_{net}+1)}_{s_{2,10}} = -w^{(L_{net})}_{1,0}$$

for $i \in \{1, \ldots, r_{net}\}$. Consequently, we get the following outputs:

$$o^{(r_0+L_{net}+1)}_{(i_2,j_2),s_{2,9}} = \sigma \left(\sum_{i=1}^{r_{net}} w^{(L_{net})}_{1,i} \cdot o^{(r_0+L_{net})}_{(i_2,j_2),t+i} + w^{(L_{net})}_{1,0} \right)$$

$$= \sigma \Bigg(g_{net} \Big(f_1(x_{(i_2,j_2)+I^{(k-1)}}), f_2(x_{(i_2+2^{k-1},j_2)+I^{(k-1)}}),$$

$$f_3(x_{(i_2,j_2+2^{k-1})+I^{(k-1)}}), f_4(x_{(i_2+2^{k-1},j_2+2^{k-1})+I^{(k-1)}}) \Big) \Bigg)$$

and

$$o^{(r_0+L_{net}+1)}_{(i_2,j_2),s_{2,10}} = \sigma \left(\sum_{i=1}^{r_{net}} -w^{(L_{net})}_{1,i} \cdot o^{(r_0+L_{net})}_{(i_2,j_2),t+i} - w^{(L_{net})}_{1,0} \right)$$

$$= \sigma\Bigg(-g_{net}\Big(f_1(x_{(i_2,j_2)+I^{(k-1)}}), f_2(x_{(i_2+2^{k-1},j_2)+I^{(k-1)}}),$$

$$f_3(x_{(i_2,j_2+2^{k-1})+I^{(k-1)}}), f_4(x_{(i_2+2^{k-1},j_2+2^{k-1})+I^{(k-1)}})\Big)\Bigg).$$

Finally, we obtain

$$o^{(r_0+L_{net}+1)}_{(i_2,j_2),s_{2,9}} - o^{(r_0+L_{net}+1)}_{(i_2,j_2),s_{2,10}}$$

$$= \max\Bigg\{ g_{net}\Big(f_1(x_{(i_2,j_2)+I^{(k-1)}}), f_2(x_{(i_2+2^{k-1},j_2)+I^{(k-1)}}),$$

$$f_3(x_{(i_2,j_2+2^{k-1})+I^{(k-1)}}), f_4(x_{(i_2+2^{k-1},j_2+2^{k-1})+I^{(k-1)}})\Big), 0\Bigg\}$$

$$- \max\Bigg\{ -g_{net}\Big(f_1(x_{(i_2,j_2)+I^{(k-1)}}), f_2(x_{(i_2+2^{k-1},j_2)+I^{(k-1)}}),$$

$$f_3(x_{(i_2,j_2+2^{k-1})+I^{(k-1)}}), f_4(x_{(i_2+2^{k-1},j_2+2^{k-1})+I^{(k-1)}})\Big), 0\Bigg\}$$

$$= g_{net}\Big(f_1(x_{(i_2,j_2)+I^{(k-1)}}), f_2(x_{(i_2+2^{k-1},j_2)+I^{(k-1)}}),$$

$$f_3(x_{(i_2,j_2+2^{k-1})+I^{(k-1)}}), f_4(x_{(i_2+2^{k-1},j_2+2^{k-1})+I^{(k-1)}})\Big).$$

$\square$

Proof of Lemma 5. In the proof we will use the network $f_{id} : \mathbb{R} \to \mathbb{R}$ defined by

$$f_{id}(x) = \sigma(x) - \sigma(-x) = \max\{x, 0\} - \max\{-x, 0\} = x,$$

which enables us to save a value computed in layer $r-1$ in channel s at position (i,j) by a difference of the outputs of two neurons in distinct channels s_1 and s_2 in layer r by

$$o^{(r)}_{(i,j),s_1} - o^{(r)}_{(i,j),s_2} = \sigma\Big(o^{(r-1)}_{(i,j),s}\Big) - \sigma\Big(-o^{(r-1)}_{(i,j),s}\Big) = o^{(r-1)}_{(i,j),s}. \tag{19}$$

Once a value has been saved in layer r by the difference of two neurons, it will be propagated analogously to the next layer $r+1$ by calculating

$$o^{(r+1)}_{(i,j),s_1} - o^{(r+1)}_{(i,j),s_2} = \sigma\Big(o^{(r)}_{(i,j),s_1} - o^{(r)}_{(i,j),s_2}\Big) - \sigma\Big(o^{(r)}_{(i,j),s_2} - o^{(r)}_{(i,j),s_1}\Big) = o^{(r)}_{(i,j),s_1} - o^{(r)}_{(i,j),s_2}. \tag{20}$$

In case we want to make use of equation (19) or equation (20), we have to choose the filters (and the bias weights) of the convolutional neural network in layer r in the channels s_1 and s_2 accordingly from the set $\{-1, 0, 1\}$.

With this approach of storing and propagating calculated values, the idea of our proof is to choose the filters (and the bias weights) such that our convolutional neural network

saves in channels corresponding to position (i,j) the values of $x_{i,j}$, $\bar{f}_{1,s}(x_{(i,j)+I^{(1)}})$ ($s = 1,\dots,4^{l-1}$), $\bar{f}_{2,s}(x_{(i,j)+I^{(2)}})$ ($s = 1,\dots,4^{l-2}$), ..., $\bar{f}_{l,s}(x_{(i,j)+I^{(l)}})$ ($s = 1$), where we set

$$I^{(m)} = \{0,\dots,2^m - 1\} \times \{0,\dots,2^m - 1\}$$

for $m \in \mathbb{N}_0$. To do this we need two neurons for each of the above values, so altogether

$$2 \cdot (1 + 4^{l-1} + 4^{l-2} + \dots + 4^0) = 2 \cdot \left(1 + \frac{4^l - 1}{4 - 1}\right) = \frac{2 \cdot 4^l + 4}{3}$$

channels or neurons for each position (i,j). Furthermore, we will need r_{net} additional channels to compute the networks $\bar{g}_{net,k,s}$. So altogether we need

$$\frac{2 \cdot 4^l + 4}{3} + r_{net} = k_r$$

many channels in each convolutional layer r.

The convolutional neural network $m_{net} \in \mathcal{F}(l_{net}, \mathbf{k}, \mathbf{M})$, which we will construct to prove the assertion, has the parameters $l_{net}, \mathbf{k}$ and $\mathbf{M}$ of Lemma 5. We make use of the above idea by choosing the filters (and bias weights) of the convolutional neural m_{net} network so that it has the following property for any $k \in \{1,\dots,l\}$:

For any $s \in \{1,\dots,4^{l-k}\}$, $(i,j) \in \{1,\dots,d_1 - 2^k + 1\} \times \{1,\dots,d_2 - 2^k + 1\}$ and any $r \in \{4^{l-1} \cdot L_{net} + \dots + 4^{l-k} \cdot L_{net} + k,\dots,l_{net}\}$ it holds that

$$
\begin{aligned}
&o^{(r)}_{(i,j),2+2\cdot 4^{l-1}+\dots+2\cdot 4^{l-k+1}+2\cdot s-1} - o^{(r)}_{(i,j),2+2\cdot 4^{l-1}+\dots+2\cdot 4^{l-k+1}+2\cdot s} \\
&= \bar{f}_{k,s}(x_{(i,j)+I^{(k)}}).
\end{aligned} \tag{21}
$$

Due to equation (20) is suffices to show equation (21) for $r = 4^{l-1} \cdot L_{net} + \dots + 4^{l-k} \cdot L_{net} + k$. To construct our convolutional neural network m_{net} so that the above property (21) is fulfilled, we use an induction on k.

We start with $k = 1$. First we note that

$$\bar{f}_{1,s}(x_{(i,j)+I^{(1)}}) = \bar{g}_{net,1,s}(x_{(i,j)}, x_{(i+1,j)}, x_{(i,j+1)}, x_{(i+1,j+1)})$$

for $s \in \{1,\dots,4^{l-1}\}$ and $(i,j) \in \{1,\dots,d_1 - 1\} \times \{1,\dots,d_2 - 1\}$. So we have to compute the networks $\bar{g}_{net,1,1},\dots,\bar{g}_{net,1,4^{l-1}}$ applied to the input of our convultional network. The idea is to use Lemma 6 for each network $\bar{g}_{net,1,s}$. Therefore, we first make sure that the assumptions (13)–(16) are fulfilled as we need them. In the first convolutional layer we copy $x_{i,j}$ in the first two channels using the weights as in equation (19), and we propagate these values in the successive layers using the weights as in equation (20). So after the first layer we have available the input in the first two channels in all convolutional layers, so that for all $r \in \{2,\dots,l_{net}\}$ and all $(i,j) \in \{1,\dots,d_1\} \times \{1,\dots,d_2\}$ it holds that

$$o^{(r)}_{(i,j),1} - o^{(r)}_{(i,j),2} = x_{(i,j)}.$$

For the filter size it holds that

$$M_r = 2 \quad (r \in \pi^{-1}(1) = \{1, 2, \ldots, 4^{l-1} \cdot L_{net} + 1\}).$$

Starting already in parallel in the first layer, we compute successively the networks $\bar{g}_{net,1,1}, \ldots, \bar{g}_{net,1,4^{l-1}}$ in layers

$$1, 2, \ldots, 4^{l-1} \cdot L_{net} + 1$$

in the channels

$$\frac{2 \cdot 4^l + 4}{3} + 1, \frac{2 \cdot 4^l + 4}{3} + 2, \ldots, \frac{2 \cdot 4^l + 4}{3} + r_{net}$$

for the computation of their hidden layers and the ouput layers in channels $2 + 1, \ldots, 2 + 2 \cdot 4^{l-1}$ by applying Lemma 6 4^{l-1} times. We now describe how to use Lemma 6 to compute $\bar{g}_{net,1,s}$ $(s = 1, \ldots, 4^{l-1})$. In particular, we specify how to choose the parameters $s_{2,1}, \ldots, s_{2,10}$ from Lemma 6. The computation of $\bar{g}_{net,1,s}$ takes place in layers

$$(s-1) \cdot L_{net} + 1, \ldots, s \cdot L_{net}$$

in channels

$$\frac{2 \cdot 4^l + 4}{3} + 1, \frac{2 \cdot 4^l + 4}{3} + 2, \ldots, \frac{2 \cdot 4^l + 4}{3} + r_{net}$$

for the computation of its hidden layers and its output layer is computed in layer $s \cdot L_{net} + 1$ in channels $s_{2,9} = 2 + 2s - 1$ and $s_{2,10} = 2 + 2s$. As input the network $\bar{g}_{net,1,s}$ uses the first two channels for $s > 1$ such that

$$s_{2,1} = s_{2,3} = s_{2,5} = s_{2,7} = 1 \text{ and } s_{2,2} = s_{2,4} = s_{2,6} = s_{2,8} = 2,$$

and in case $s = 1$ it selects its input from the input of the convolutional network and then use a simple variation of Lemma 6 by adapting the assumptions (13)–(16). The computed function value of $\bar{g}_{net,1,s}$ is then saved in the two channels $s_{2,9} = 2 + 2s - 1$ and $s_{2,10} = 2 + 2s$. Here we propagate again the value of these neurons successively to the next layer by using the weights as in equation (20). So after layer $4^{l-1} \cdot L_{net} + 1$ we have available the values of all $\bar{f}_{1,s}$ in the channels $2 + 1, \ldots, 2 + 2 \cdot 4^{l-1}$, so that for any $s \in \{1, \ldots, 4^{l-1}\}$ and any $(i,j) \in \{1, \ldots, d_1 - 1\} \times \{1, \ldots, d_2 - 1\}$ it holds that

$$o^{(4^{l-1} \cdot L_{net}+1)}_{(i,j),2+2\cdot s-1} - o^{(4^{l-1} \cdot L_{net}+1)}_{(i,j),2+2\cdot s} = \bar{g}_{net,1,s}(x_{(i,j)}, x_{(i+1,j)}, x_{(i,j+1)}, x_{(i+1,j+1)})$$
$$= \bar{f}_{1,s}(x_{(i,j)+I^{(1)}}).$$

Thus property (21) holds for $k = 1$.

Now we assume that equation (21) holds for $k \in \{1, \ldots, l-1\}$. We use the values $\bar{f}_{k,s}(x_{(i,j)+I^{(k)}})$, which are given by equation (21), to compute all values of

$$\bar{f}_{k+1,s}(x_{(i,j)+I^{(k+1)}}) = \bar{g}_{net,k+1,s}\Big(\bar{f}_{k,4\cdot(s-1)+1}(x_{(i_2,j_2)+I^{(k)}}), \bar{f}_{k,4\cdot(s-1)+2}(x_{(i_2+2^k,j_2)+I^{(k)}}),$$
$$\bar{f}_{k,4\cdot(s-1)+3}(x_{(i_2,j_2+2^k)+I^{(k)}}), \bar{f}_{k,4\cdot s}(x_{(i_2+2^k,j_2+2^k)+I^{(k)}})\Big)$$

for $s \in \{1, \ldots 4^{l-(k+1)}\}$ using Lemma 6. We proceed similarly to the above case of $k = 1$. For the filter size it holds that

$$M_r = 2^{k+1} \quad (r \in \pi^{-1}(k+1)),$$

where $\pi^{-1}(k+1)$ is given by

$$\{4^{l-1} \cdot L_{net} + \cdots + 4^{l-k} \cdot L_{net} + (k+1), \ldots, 4^{l-1} \cdot L_{net} + \cdots + 4^{l-(k+1)} \cdot L_{net} + (k+1)\}.$$

By applying Lemma 6 $4^{l-(k+1)}$ times we compute successively the networks $\bar{g}_{net,k+1,1}, \ldots, \bar{g}_{net,k+1,4^{l-(k+1)}}$, in the corresponding layers

$$4^{l-1} \cdot L_{net} + \cdots + 4^{l-k} \cdot L_{net} + k + 1, \ldots, 4^{l-1} \cdot L_{net} + \cdots + 4^{l-k} \cdot L_{net} + 4^{l-(k+1)} \cdot L_{net} + k + 1,$$

where the computation of their hidden layers takes place in channels

$$\frac{2 \cdot 4^l + 4}{3} + 1, \frac{2 \cdot 4^l + 4}{3} + 2, \ldots, \frac{2 \cdot 4^l + 4}{3} + r_{net}$$

and the computation of their ouput layers takes place in channels

$$2 + 2 \cdot 4^{l-1} + \cdots + 2 \cdot 4^{l-k} + 1, \ldots, 2 + 2 \cdot 4^{l-1} + \cdots + 2 \cdot 4^{l-(k+1)}.$$

As above we describe how to use Lemma 6 to compute $\bar{g}_{net,k+1,s}$ $(s = 1, \ldots, 4^{l-(k+1)})$ and specify how to choose the parameters $s_{2,1}, \ldots, s_{2,10}$ from Lemma 6. The computation of $\bar{g}_{net,k+1,s}$ $(s = 1, \ldots, 4^{l-(k+1)})$ takes place in layers

$$4^{l-1} \cdot L_{net} + \cdots + 4^{l-k} \cdot L_{net} + k + (s-1) \cdot L_{net} + 1, \ldots, 4^{l-1} \cdot L_{net} + \cdots + 4^{l-k} \cdot L_{net} + k + s \cdot L_{net}$$

in channels

$$\frac{2 \cdot 4^l + 4}{3} + 1, \frac{2 \cdot 4^l + 4}{3} + 2, \ldots, \frac{2 \cdot 4^l + 4}{3} + r_{net}$$

for the computation of its hidden layers and its output layer is computed in layer

$$4^{l-1} \cdot L_{net} + \cdots + 4^{l-k} \cdot L_{net} + k + s \cdot L_{net} + 1$$

in channels

$$s_{2,9} = 2 + 2 \cdot 4^{l-1} + \cdots + 2 \cdot 4^{l-k} + 2s - 1 \tag{22}$$

and

$$s_{2,10} = 2 + 2 \cdot 4^{l-1} + \cdots + 2 \cdot 4^{l-k} + 2s. \tag{23}$$

We choose

$$s_{2,m} = 2 + \left(\sum_{i=l-(k-1)}^{l-1} 2 \cdot 4^i \right) + 2 \cdot 4 \cdot (s-1) + m$$

for $m \in \{1, \ldots, 8\}$, because then we have

$$o_{(i,j),s_{2,2 \cdot m-1}}^{(r)} - o_{(i,j),s_{2,2 \cdot m}}^{(r)} = \bar{f}_{k,4 \cdot (s-1)+m}\left(x_{(i,j)+I^{(k)}}\right)$$

30

for $m \in \{1, \ldots, 4\}$ and any $r \in \{4^{l-1} \cdot L_{net} + \cdots + 4^{l-k} \cdot L_{net} + k, \ldots, l_{net}\}$ and any $(i,j) \in \{1, \ldots, d_1 - 2^k + 1\} \times \{1, \ldots, d_2 - 2^k + 1\}$ due to the induction hypothesis. Then Lemma 6 let us choose the corresponding weights of the network m_{net} such that

$$
o_{(i,j),s_{2,9}}^{(4^{l-1} \cdot L_{net} + \cdots + 4^{l-k} \cdot L_{net} + k + s \cdot L_{net} + 1)} - o_{(i,j),s_{2,10}}^{(4^{l-1} \cdot L_{net} + \cdots + 4^{l-k} \cdot L_{net} + k + s \cdot L_{net} + 1)}
$$

$$
= \bar{g}_{net,k+1,s}\Big(\bar{f}_{k,4 \cdot (s-1)+1}(x_{(i,j)+I^{(k)}}), \bar{f}_{k,4 \cdot (s-1)+2}(x_{(i+2^k,j)+I^{(k)}}),
$$

$$
\bar{f}_{k,4 \cdot (s-1)+3}(x_{(i,j+2^k)+I^{(k)}}), \bar{f}_{k,4 \cdot s}(x_{(i+2^k,j+2^k)+I^{(k)}})\Big)
$$

$$
= \bar{f}_{k+1,s}(x_{(i,j)+I^{(k+1)}}).
$$

for any $(i,j) \in \{1, \ldots, d_1 - 2^{k+1} + 1\} \times \{1, \ldots, d_2 - 2^{k+1} + 1\}$. By propagating again the values of these neurons successively to the next layer we have available the values of all $\bar{f}_{k+1,s}$ after layer

$$
4^{l-1} \cdot L_{net} + \cdots + 4^{l-k} \cdot L_{net} + 4^{l-(k+1)} \cdot L_{net} + k + 1
$$

in the channels

$$
2 + 2 \cdot 4^{l-1} + \cdots + 2 \cdot 4^{l-k} + 1, \ldots, 2 + 2 \cdot 4^{l-1} + \cdots + 2 \cdot 4^{l-(k+1)}
$$

so that for any $s \in \{1, \ldots, 4^{l-(k+1)}\}$ and any $(i,j) \in \{1, \ldots, d_1 - 2^{k+1} - 1\} \times \{1, \ldots, d_2 - 2^{k+1} - 1\}$ it holds that

$$
o_{(i,j),2+2 \cdot 4^{l-1}+\cdots+2 \cdot 4^{l-k}+2s-1}^{(4^{l-1} \cdot L_{net}+\cdots+4^{l-k} \cdot L_{net}+4^{l-(k+1)} \cdot L_{net}+k+1)} - o_{(i,j),2+2 \cdot 4^{l-1}+\cdots+2 \cdot 4^{l-k}+2s-1}^{(4^{l-1} \cdot L_{net}+\cdots+4^{l-k} \cdot L_{net}+4^{l-(k+1)} \cdot L_{net}+k+1)}
$$

$$
= \bar{f}_{k+1,s}(x_{(i,j)+I^{(k+1)}}).
$$

So property (21) holds for all $k \in \{1, \ldots, l\}$.

Hence in layer

$$
l_{net} = 4^{l-1} \cdot L_{net} + 4^{l-2} \cdot L_{net} + \cdots + 4^0 \cdot L_{net} + l = \frac{4^l - 1}{3} \cdot L_{net} + l
$$

we have by equation (21)

$$
o_{(i,j),2+2 \cdot 4^{l-1}+\cdots+2 \cdot 4+1}^{(l_{net})} - o_{(i,j),2+2 \cdot 4^{l-1}+\cdots+2 \cdot 4+2}^{(l_{net})} = \bar{f}_{l,1}(x_{(i,j)+I^{(l)}})
$$

for all $(i,j) \in \{1, \ldots, d_1 - 2^l + 1\} \times \{1, \ldots, d_2 - 2^l + 1\}$. Now we choose the outer weights $\mathbf{w}_{out}$ of our convolutional neural network m_{net} such that

$$
w_s = \begin{cases} 1, & \text{if } s = 2 + 2 \cdot 4^{l-1} + \cdots + 2 \cdot 4 + 1 \\ -1, & \text{if } s = 2 + 2 \cdot 4^{l-1} + \cdots + 2 \cdot 4 + 2 \\ 0, & \text{else.} \end{cases}
$$

This implies that the output of our network is given by

$$m_{net}(\mathbf{x}) = \max \left\{ o_{(i,j),2+2\cdot 4^{l-1}+\dots+2\cdot 4^{l-l+1}+1}^{(l_{net})} - o_{(i,j),2+2\cdot 4^{l-1}+\dots+2\cdot 4^{l-l+1}+2}^{(l_{net})} \quad : \right.$$

$$\left. (i,j) \in \{1,\dots,d_1 - 2^l + 1\} \times \{1,\dots,d_2 - 2^l + 1\} \right\}$$

$$= \max \left\{ \bar{f}(x_{(i,j)+I}) \quad : \quad (i,j) \in \mathbb{Z}_2, (i,j) + I \subseteq \{1,\dots,d_1\} \times \{1,\dots,d_2\} \right\}$$

$$= \bar{m}(\mathbf{x}).$$

$\square$

6.3 A bound on the covering number

The purpose of the subsection is to show the following bound on the covering number of $\mathcal{F}_t\left(\mathbf{L}, \mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \mathbf{M}\right)$.

Lemma 7 *Let $\sigma(x) = \max\{x, 0\}$ be the ReLU activation function, define*

$$\mathcal{F} := \mathcal{F}_t\left(\mathbf{L}, \mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \mathbf{M}\right)$$

as in Section 2 and set

$$k_{max} = \max\left\{ k_1^{(1)}, \dots, k_{L_1}^{(1)}, t, k_1^{(2)}, \dots, k_{L_2}^{(2)} \right\}, \quad M_{max} = \max\{M_1, \dots, M_L\}$$

and

$$L_{max} = \max\{L_1, L_2\}.$$

Assume $d_1 \cdot d_2 > 1$ and $c_4 \cdot \log n \geq 2$. Then we have for any $\epsilon \in (0,1)$:

$$\sup_{\mathbf{x}_1^n \in (\mathbb{R}^{\{1,\dots,d_1\} \times \{1,\dots,d_2\}})^n} \log\left(\mathcal{N}_1\left(\epsilon, T_{c_4 \cdot \log n}\mathcal{F}, \mathbf{x}_1^n\right)\right)$$

$$\leq c_7 \cdot L_{max}^2 \cdot \log(L_{max} \cdot d_1 \cdot d_2) \cdot \log\left(\frac{c_4 \cdot \log n}{\epsilon}\right)$$

for some constant $c_7 > 0$ which depends only on k_{max} and M_{max}.

With the aim of proving Lemma 7, we first have to study the VC dimension of our function class $\mathcal{F}_t\left(\mathbf{L}, \mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \mathbf{M}\right)$. For a class of subsets of $\mathbb{R}^d$, the VC dimension is defined as follows:

Definition 2 *Let $\mathcal{A}$ be a class of subsets of $\mathbb{R}^d$ with $\mathcal{A} \neq \emptyset$ and $m \in \mathbb{N}$.*

1. For $\mathbf{x}_1, \dots, \mathbf{x}_m \in \mathbb{R}^d$ we define

$$s(\mathcal{A}, \{\mathbf{x}_1, \dots, \mathbf{x}_m\}) := |\{A \cap \{\mathbf{x}_1, \dots, \mathbf{x}_m\} \ : \ A \in \mathcal{A}\}|.$$

*2. Then the mth **shatter coefficient** $S(\mathcal{A}, m)$ of $\mathcal{A}$ is defined by*

$$S(\mathcal{A}, m) := \max_{\{\mathbf{x}_1, ..., \mathbf{x}_m\} \subset \mathbb{R}^d} s(\mathcal{A}, \{\mathbf{x}_1, ..., \mathbf{x}_m\}).$$

*3. The **VC dimension** (Vapnik-Chervonenkis-Dimension) $V_{\mathcal{A}}$ of $\mathcal{A}$ is defined as*

$$V_{\mathcal{A}} := \sup\{m \in \mathbb{N} \ : \ S(\mathcal{A}, m) = 2^m\}.$$

For a class of real-valued functions, we define the VC dimension as follows:

Definition 3 *Let $\mathcal{H}$ denote a class of functions from $\mathbb{R}^d$ to $\{0, 1\}$ and let $\mathcal{F}$ be a class of real-valued functions.*

*1. For any non-negative integer m, we define the **growth function** of H as*

$$\Pi_{\mathcal{H}}(m) := \max_{\mathbf{x}_1, ..., \mathbf{x}_m \in \mathbb{R}^d} |\{(h(\mathbf{x}_1), \ldots, h(\mathbf{x}_m)) : h \in H\}|.$$

*2. The **VC dimension** (Vapnik-Chervonenkis-Dimension) of $\mathcal{H}$ we define as*

$$\mathrm{VCdim}(\mathcal{H}) := \sup\{m \in \mathbb{N} : \Pi_{\mathcal{H}}(m) = 2^m\}.$$

*3. For $f \in \mathcal{F}$ we denote $\mathrm{sgn}(f) := \mathbb{I}_{\{f \geq 0\}}$ and $\mathrm{sgn}(\mathcal{F}) := \{\mathrm{sgn}(f) : f \in \mathcal{F}\}$. Then the **VC dimension** of $\mathcal{F}$ is defined as*

$$\mathrm{VCdim}(\mathcal{F}) := \mathrm{VCdim}(\mathrm{sgn}(\mathcal{F})).$$

A connection between both definitions is given by the following lemma.

Lemma 8 *Suppose $\mathcal{F}$ is a class of real-valued functions on $\mathbb{R}^d$. Furthermore, we define*

$$\mathcal{F}^+ := \{\{(\mathbf{x}, y) \in \mathbb{R}^d \times \mathbb{R} : f(\mathbf{x}) \geq y\} : f \in \mathcal{F}\}$$

and define the class $\mathcal{H}$ of real-valued functions on $\mathbb{R}^d \times \mathbb{R}$ by

$$\mathcal{H} := \{h((\mathbf{x}, y)) = f(\mathbf{x}) - y : f \in \mathcal{F}\}.$$

Then, it holds that
$$V_{\mathcal{F}^+} = \mathrm{VCdim}(\mathcal{H}).$$

Proof. For all $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \in \mathbb{R}^d \times \mathbb{R}$ with $m \in \mathbb{N}$ it holds that

$$
\begin{aligned}
&s(\mathcal{F}^+, \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}) \\
&= \left|\left\{A \cap \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \ : \ A \in \mathcal{F}^+\right\}\right| \\
&= \left|\left\{\{(\mathbf{x}, y) \in \mathbb{R}^d \times \mathbb{R} : f(\mathbf{x}) \geq y\} \cap \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \ : \ f \in \mathcal{F}\right\}\right| \\
&= |\{\{(\mathbf{x}, y) \in \{(x_1, y_1), \ldots, (x_m, y_m)\} : f(\mathbf{x}) \geq y\} \ : \ f \in \mathcal{F}\}|
\end{aligned}
$$

$$= |\{\{i \in \{1, \dots, m\} : f(\mathbf{x}_i) \ge y_i\} \; : \; f \in \mathcal{F}\}|$$
$$= |\{(\mathrm{sgn}(f(\mathbf{x}_1) - y_1), \dots, \mathrm{sgn}(f(\mathbf{x}_m) - y_m)) : f \in \mathcal{F}\}|$$
$$= |\{(\mathrm{sgn}(h(\mathbf{x}_1, y_1)), \dots, \mathrm{sgn}(h(\mathbf{x}_m, y_m))) : h \in \mathcal{H}\}|.$$

It follows that
$$S(\mathcal{F}^+, m) = \Pi_{\mathcal{H}}(m)$$

holds for all $m \in \mathbb{N}$, which implies
$$V_{\mathcal{F}^+} = \mathrm{VCdim}(\mathcal{H}).$$

$\square$

In order to bound the VC dimension of our function class, we need the following two auxiliary results. The first one is also known as weighted AM-GM inequality.

Lemma 9 *Suppose $x_1, \dots, x_n > 0$ and $w_1, \dots, w_n > 0$. We denote $w := \sum_{i=1}^n w_i$. Then, it holds that*
$$\prod_{i=1}^n \left(\frac{x_i}{w_i}\right)^{w_i} \le \left(\frac{\sum_{i=1}^n x_i}{w}\right)^w. \tag{24}$$

Proof. Since the natural logarithm is a concave function on $\{x \in \mathbb{R} : x > 0\}$ and $\sum_{i=1}^n \frac{w_i}{w} = 1$, Jensen's inequality implies that

$$\log\left(\left(\frac{\sum_{i=1}^n x_i}{w}\right)^w\right) = w \cdot \log\left(\frac{\sum_{i=1}^n w_i\left(\frac{x_i}{w_i}\right)}{w}\right)$$
$$\ge w \cdot \sum_{i=1}^n \frac{w_i}{w} \log\left(\frac{x_i}{w_i}\right)$$
$$= \log\left(\prod_{i=1}^n \left(\frac{x_i}{w_i}\right)^{w_i}\right).$$

Furthermore, the inverse function of the logarithm $u \mapsto e^u$ ($u \in \mathbb{R}$) is an increasing function, hence we get
$$\prod_{i=1}^n \left(\frac{x_i}{w_i}\right)^{w_i} \le \left(\frac{\sum_{i=1}^n x_i}{w}\right)^w.$$

$\square$

The second auxiliary result is about the number of possible sign vectors attained by polynomials of bounded degree.

Lemma 10 *Suppose $W \le m$ and let $f_1, \dots, f_m$ be polynomials of degree at most D in W variables. Define*
$$K := |\{(\mathrm{sgn}(f_1(\mathbf{a})), \dots, \mathrm{sgn}(f_m(\mathbf{a}))) : \mathbf{a} \in \mathbb{R}^W\}|.$$

Then we have
$$K \le 2 \cdot \left(\frac{2 \cdot e \cdot m \cdot D}{W}\right)^W.$$

34

Proof. See Theorem 8.3 in Anthony and Bartlett (1999). □

To get an upper bound for the VC dimension of our function class $\mathcal{F}_t \left(\mathbf{L}, \mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \mathbf{M} \right)$ defined as in Section 2 we will use a modification of Theorem 6 in Bartlett et al. (2019).

Lemma 11 *Let $\sigma(x) = \max\{x, 0\}$ be the ReLU activation function, define*

$$\mathcal{F} := \mathcal{F}_t \left(\mathbf{L}, \mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \mathbf{M} \right)$$

as in Section 2, set

$$k_{max} = \max \left\{ k_1^{(1)}, \ldots, k_{L_1}^{(1)}, t, k_1^{(2)}, \ldots, k_{L_2}^{(2)} \right\}, \quad M_{max} = \max\{M_1, \ldots, M_{L_1}\}$$

and

$$L_{max} = \max\{L_1, L_2\}.$$

Assume $d_1 \cdot d_2 > 1$. Then, we have

$$V_{\mathcal{F}^+} \leq c_{10} \cdot L_{max}^2 \cdot \log_2(L_{max} \cdot d_1 \cdot d_2)$$

for some constant $c_{10} > 0$ which depends only on k_{max} and M_{max}.

Proof. We want to use Lemma 8 to bound $\mathcal{V}_{\mathcal{F}^+}$ by $\mathrm{VCdim}(\mathcal{H})$, where $\mathcal{H}$ is the class of real-valued functions on $[0, 1]^{\{1, \ldots, d_1\} \times \{1, \ldots, d_2\}} \times \mathbb{R}$ defined by

$$\mathcal{H} := \{h((\mathbf{x}, y)) = f(\mathbf{x}) - y : f \in \mathcal{F}\}.$$

Let $h \in \mathcal{H}$. Then h depends on t convolutional neural networks

$$f_1, \ldots, f_t \in \mathcal{F}(L_1, \mathbf{k}^{(1)}, \mathbf{M})$$

and one standard feedforward neural network $g \in \mathcal{G}_t(L_2, \mathbf{k}^{(2)})$ such that

$$h((\mathbf{x}, y)) = g \circ (f_1, \ldots, f_t)(\mathbf{x}) - y$$

Each one of the convolutional neural networks $f_1, \ldots, f_t$ depends on a weight matrix

$$\mathbf{w}^{(b)} = \left(w_{i,j,s_1,s_2}^{(b,r)} \right)_{1 \leq i,j \leq M_r, s_1 \in \{1, \ldots, k_{r-1}^{(1)}\}, s_2 \in \{1, \ldots, k_r^{(1)}\}, r \in \{1, \ldots, L_1\}},$$

the weights

$$\mathbf{w}_{bias}^{(b)} = \left(w_{s_2}^{(b,r)} \right)_{s_2 \in \{1, \ldots, k_r^{(1)}\}, r \in \{1, \ldots, L_1\}}$$

for the bias in each channel and each convolutional layer, the output weights

$$\mathbf{w}_{out}^{(b)} = (w_s^{(b)})_{s \in \{1, \ldots, k_{L_1}^{(1)}\}}$$

for $b \in \{1, \ldots, t\}$. The standard feedforward neural network $g \in \mathcal{G}_t(L_2, \mathbf{k}^{(2)})$ depends on the inner weigths

$$w_{i,j}^{(r-1)}$$

for $j \in \{0, \ldots, k_{r-1}^{(2)}\}$, $i \in \{1, \ldots, k_r^{(2)}\}$ and $r \in \{1, \ldots, L_2\}$ and the outer weights

$$w_i^{(L_2)}$$

for $i \in \{0, \ldots, k_{L_2}^{(2)}\}$ (where $k_0^{(2)} = t$).

We set
$$\mathbf{k} = (k_0, \ldots, k_{L_1+L_2+1}) = (1, k_1^{(1)}, \ldots, k_{L_1}^{(1)}, t, k_1^{(2)}, \ldots, k_{L_2}^{(2)})$$
and count the number of weights used up to layer $r \in \{1, \ldots, L_1\}$ in the convolutional part by

$$W_r := t \cdot \left(\sum_{s=1}^{r} M_s^2 \cdot k_s \cdot k_{s-1} + \sum_{s=1}^{r} k_s \right),$$

for $r \in \{1, \ldots, L_1\}$ (where we set $W_0 := 0$) and

$$W_{L_1+1} := W_{L_1} + t \cdot k_{L_1}.$$

We continue in the part of the standard feedforward neural network by counting the weights used up to layer $r \in \{1, \ldots, L_2\}$ by

$$W_{L_1+1+r} = W_{L_1+r} + (k_{L_1+r} + 1) \cdot k_{L_1+r+1}$$

and denote the total number of weights by

$$\begin{aligned}
W &= W_{L_1+L_2+2} \\
&= W_{L_1+L_2+1} + k_{L_1+L_2+1} + 1 \\
&\leq L_1 \cdot t \cdot \left(M_{max}^2 \cdot k_{max}^2 + k_{max} \right) + t \cdot k_{max} \\
&\quad + L_2 \cdot ((k_{max} + 1) \cdot k_{max}) + k_{max} + 1 \\
&\leq L_1 \cdot t \cdot \left(M_{max}^2 \cdot (k_{max} + 1) \cdot k_{max} \right) \qquad (25)\\
&\quad + L_2 \cdot ((k_{max} + 1) \cdot k_{max}) \\
&\quad + 2 \cdot t \cdot (k_{max} + 1) \\
&\leq (L_1 + L_2 + 2) \cdot t \cdot M_{max}^2 \cdot (k_{max} + 1) \cdot k_{max} \\
&\leq 2 \cdot (L_1 + L_2 + 2) \cdot t \cdot M_{max}^2 \cdot k_{max}^2.
\end{aligned}$$

We define $I^{(0)} = \emptyset$ and for $r \in \{1, \ldots, L_1 + L_2 + 2\}$ we define the index sets

$$I^{(r)} = \{1, \ldots, W_r\}.$$

Furthermore, we define a sequence of vectors containing the weights used up to layer $r \in \{1, \ldots, L_1\}$ in the convolutional part by

$$\begin{aligned}
\mathbf{a}_{I^{(r)}} := \Big(&\mathbf{a}_{I^{(r-1)}}, w_{1,1,1,1}^{(1,r)}, \ldots, w_{M_r,M_r,k_{r-1},k_r}^{(1,r)}, w_1^{(1,r)}, \ldots, w_{k_r}^{(1,r)}, \\
&\ldots, w_{1,1,1,1}^{(t,r)}, \ldots, w_{M_r,M_r,k_{r-1},k_r}^{(t,r)}, w_1^{(t,r)}, \ldots, w_{k_r}^{(t,r)} \Big) \in \mathbb{R}^{W_r}
\end{aligned}$$

(where $\mathbf{a}_\emptyset$ denotes the empty vector),

$$\mathbf{a}_{I(L_1+1)} := (\mathbf{a}_{I(L_1)}, w_1^{(1)}, \ldots, w_{k_{L_1}}^{(1)}, \ldots, w_1^{(t)}, \ldots, w_{k_{L_1}}^{(t)}) \in \mathbb{R}^{W_{L_1+1}},$$

and by continuing with the part of the standard feedforward neural network we get for $r \in \{1, \ldots, L_2\}$

$$\mathbf{a}_{I(r+L_1+1)} := \left(\mathbf{a}_{I(r+L_1)}, w_{1,0}^{(r-1)}, \ldots, w_{k_{r+L_1+1}, k_{r+L_1}}^{(r-1)}\right) \in \mathbb{R}^{W_{r+L_1+1}}$$

and

$$\mathbf{a} := \left(\mathbf{a}_{I(L_1+L_2+1)}, w_0^{(L_2)}, \ldots, w_{L_2}^{(L_2)}\right) \in \mathbb{R}^{W}.$$

With this notation we can write

$$\mathcal{H} = \{(\mathbf{x}, y) \mapsto h((\mathbf{x}, y), \mathbf{a}) : \mathbf{a} \in \mathbb{R}^W\}$$

and for $b \in \{1, \ldots, t\}$

$$\mathcal{F}(L_1, \mathbf{k}^{(1)}, \mathbf{M}) = \{\mathbf{x} \mapsto f_b(\mathbf{x}, \mathbf{a}) : \mathbf{a} \in \mathbb{R}^W\},$$

where the convolutional networks $f_1, \ldots, f_t \in \mathcal{F}(L_1, \mathbf{k}^{(1)}, \mathbf{M})$, as described above, each depends only on W_{L_1+1}/t variables of $\mathbf{a}$. To get an upper bound for the VC-dimension of $\mathcal{H}$, we will bound the growth function $\Pi_{\text{sgn}(\mathcal{H})}(m)$. In the following we assume that m is a positiv integer with

$$m \geq W \tag{26}$$

since this will allow us several uses of Lemma 10. To bound the growth function $\Pi_{\text{sgn}(\mathcal{H})}(m)$, we fix the input values

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \in [0,1]^{\{1,\ldots,d_1\} \times \{1,\ldots,d_2\}} \times \mathbb{R}$$

and consider $h \in \mathcal{H}$ as a function of the weight vector $\mathbf{a} \in \mathbb{R}^W$ of h

$$\mathbf{a} \mapsto h((\mathbf{x}_k, y_k), \mathbf{a}) = g \circ (f_1, \ldots, f_t)(\mathbf{x}_k, \mathbf{a}) - y_k = h_k(\mathbf{a})$$

for any $k \in \{1, \ldots, m\}$. Then, an upper bound for

$$K := |\{(\text{sgn}(h_1(\mathbf{a})), \ldots, \text{sgn}(h_m(\mathbf{a}))) : \mathbf{a} \in \mathbb{R}^W\}|$$

implies an upper bound for the growth function $\Pi_{\text{sgn}(\mathcal{H})}(m)$. For any partition

$$\mathcal{S} = \{S_1, \ldots, S_M\}$$

of $\mathbb{R}^W$ it holds that

$$K \leq \sum_{i=1}^M |\{(\text{sgn}(h_1(\mathbf{a})), \ldots, \text{sgn}(h_m(\mathbf{a})) : \mathbf{a} \in S_i\}|. \tag{27}$$

We will construct a partition $\mathcal{S}$ of $\mathbb{R}^W$ such that within each region $S \in \mathcal{S}$, the functions $h_k(\cdot)$ are all fixed polynomials of bounded degree for $k \in \{1, \ldots, m\}$, so that each summand of equation (27) can be bounded via Lemma 10. We do this in two steps.

In the first step we construct a partition $\mathcal{S}^{(1)}$ of $\mathbb{R}^W$ such that within each $S \in \mathcal{S}^{(1)}$ the t convolutional neural networks $f_{1,k}(\mathbf{a}), \ldots, f_{t,k}(\mathbf{a})$ are all fixed polynomials with dergee of at most $L_1 + 1$ for all $k \in \{1, \ldots, m\}$, where we denote

$$f_{b,k}(\mathbf{a}) = f_b(\mathbf{x}_k, \mathbf{a})$$

for $b \in \{1, \ldots, t\}$. We define

$$D = \{1, \ldots, d_1\} \times \{1, \ldots, d_2\}.$$

For $b \in \{1, \ldots, t\}$ we have

$$f_{b,k}(\mathbf{a}) = \max \left\{ \sum_{s=1}^{k_{L_1}} w_s^{(b)} \cdot o_{(i,j),b,s,\mathbf{x}_k}^{(L_1)}(\mathbf{a}_{I^{(L_1)}}) : (i,j) \in D \right\},$$

where $o_{(i,j),b,s_2,\mathbf{x}}^{(L_1)}(\mathbf{a}_{I^{(L_1)}})$ is recursively defined by

$$o_{(i,j),b,s_2,\mathbf{x}}^{(r)}(\mathbf{a}_{I^{(r)}})$$

$$= \sigma \left(\sum_{s_1=1}^{k_{r-1}} \sum_{\substack{t_1,t_2 \in \{1,\ldots,M_r\} \\ (i+t_1-1,j+t_2-1) \in D}} w_{t_1,t_2,s_1,s_2}^{(b,r)} \cdot o_{(i+t_1-1,j+t_2-1),b,s_1,\mathbf{x}}^{(r-1)}(\mathbf{a}_{I^{(r-1)}}) + w_{s_2}^{(b,r)} \right)$$

for $(i,j) \in D$ and $r \in \{1, \ldots, L_1\}$, and by

$$o_{(i,j),b,1,\mathbf{x}}^{(0)}(\mathbf{a}_{I^{(0)}}) = x_{i,j} \quad \text{for } (i,j) \in D.$$

Firstly, we construct a partition $\mathcal{S}_{L_1} = \{S_1, \ldots, S_M\}$ of $\mathbb{R}^W$ such that within each $S \in \mathcal{S}_{L_1}$

$$o_{(i,j),b,s,\mathbf{x}_k}^{(L_1)}(\mathbf{a}_{I^{(L_1)}})$$

is a fixed polynomial for all $k \in \{1, \ldots, m\}$, $s \in \{1, \ldots, k_L\}$, $b \in \{1, \ldots, t\}$ and $(i,j) \in D$ with degree of at most L_1 in the W_{L_1} variables $\mathbf{a}_{I^{(L_1)}}$ of $\mathbf{a} \in S$. We construct the partition $\mathcal{S}_{L_1}$ iteratively layer by layer, by creating a sequence $\mathcal{S}_0, \ldots, \mathcal{S}_{L_1}$, where each $\mathcal{S}_r$ is a partition of $\mathbb{R}^W$ with the following properties:

1. We have $|\mathcal{S}_0| = 1$ and, for each $r \in \{1, \ldots, L_1\}$,

$$\frac{|\mathcal{S}_r|}{|\mathcal{S}_{r-1}|} \leq 2 \left(\frac{2 \cdot e \cdot t \cdot k_r \cdot d_1 \cdot d_2 \cdot m \cdot r}{W_r} \right)^{W_r}, \tag{28}$$

38

2. For each $r \in \{0, \ldots, L_1\}$, and each element $S \in \mathcal{S}_r$, each $(i,j) \in D$, each $s \in \{1, \ldots, k_r\}$, each $k \in \{1, \ldots, m\}$, and each $b \in \{1, \ldots, t\}$ when $\mathbf{a}$ varies in S,

$$o_{(i,j),b,s,\mathbf{x}_k}^{(r)}(\mathbf{a}_{I^{(r)}})$$

is a fixed polynomial function in the W_r variables $\mathbf{a}_{I^{(r)}}$ of $\mathbf{a}$, of total degree no more than r.

We define $\mathcal{S}_0 := \{\mathbb{R}^W\}$. Since

$$o_{(i,j),b,s,\mathbf{x}_k}^{(0)}(\mathbf{a}_{I^{(0)}}) = (x_k)_{i,j}$$

is a constant polynomial, property 2 above is satisfied for $r = 0$. Now suppose that $\mathcal{S}_0, \ldots, \mathcal{S}_{r-1}$ have been defined, and we want to define $\mathcal{S}_r$. For $S \in \mathcal{S}_{r-1}$ let

$$p_{(i,j),b,s_1,\mathbf{x}_k,S}(\mathbf{a}_{I^{(r-1)}})$$

denote the function $o_{(i,j),b,s_1,\mathbf{x}_k}^{(r-1)}(\mathbf{a}_{I^{(r-1)}})$, when $\mathbf{a} \in S$. By induction hypothesis

$$p_{(i,j),b,s_1,\mathbf{x}_k,S}(\mathbf{a}_{I^{(r-1)}})$$

is a polynomial with total degree no more than $r-1$, and depends on the W_{r-1} variables $\mathbf{a}_{I^{(r-1)}}$ of $\mathbf{a}$ for any $b \in \{1, \ldots, t\}$, $k \in \{1, \ldots, m\}$, $(i,j) \in D$ and $s_1 \in \{1, \ldots, k_{r-1}\}$. Hence for any $b \in \{1, \ldots, t\}$ $k \in \{1, \ldots, m\}$, $(i,j) \in D$ and $s_2 \in \{1, \ldots, k_r\}$

$$\sum_{s_1=1}^{k_{r-1}} \sum_{\substack{t_1,t_2 \in \{1,\ldots,M_r\} \\ (i+t_1-1,j+t_2-1) \in D}} w_{t_1,t_2,s_1,s_2}^{(r)} \cdot p_{(i+t_1-1,j+t_2-1),b,s_1,\mathbf{x}_k,S}(\mathbf{a}_{I^{(r-1)}}) + w_{s_2}^{(b,r)}$$

is a polynomial in the W_r variables $\mathbf{a}_{I^{(r)}}$ of $\mathbf{a}$ with total degree no more than r. Because of condition (26) we have $t \cdot k_r \cdot m \cdot d_1 \cdot d_2 \geq W_r$. Hence, by Lemma 10, the collection of polynomials

$$\left\{ \sum_{s_1=1}^{k_{r-1}} \sum_{\substack{t_1,t_2 \in \{1,\ldots,M_r\} \\ (i+t_1-1,j+t_2-1) \in D}} w_{t_1,t_2,s_1,s_2}^{(b,r)} \cdot p_{(i+t_1-1,j+t_2-1),b,s_1,\mathbf{x}_k,S}(\mathbf{a}_{I^{(r-1)}}) + w_{s_2}^{(b,r)} : \right.$$

$$\left. b \in \{1, \ldots, t\}, k \in \{1, \ldots, m\}, (i,j) \in D, s_2 \in \{1, \ldots, k_r\} \right\}$$

attains at most

$$\Pi := 2 \left(\frac{2 \cdot e \cdot t \cdot k_r \cdot m \cdot d_1 \cdot d_2 \cdot r}{W_r} \right)^{W_r}$$

distinct sign patterns when $\mathbf{a}_{I^{(r)}} \in \mathbb{R}^{W_r}$ and therefore the above collection of polynomials also attains at most Π distinct sign patterns when $\mathbf{a}$ varies in $\mathbb{R}^W$ since the above polynomials depend only on the W_r variables $\mathbf{a}_{I^{(r)}}$ of $\mathbf{a}$. Therefore, we can partition $S \subset \mathbb{R}^W$ into Π subregions, such that all the polynomials don't change their signs within each subregion. Doing this for all regions $S \in \mathcal{S}_{r-1}$ we get our required partition $\mathcal{S}_r$ by assembling all of these subregions. In particular, property 1 (inequality (28)) is then satisfied.

Fix some $S' \in \mathcal{S}_r$. Notice that, when $\mathbf{a}$ varies in S', all the polynomials

$$
\left\{ \sum_{s_1=1}^{k_{r-1}} \sum_{\substack{t_1,t_2 \in \{1,\ldots,M_r\} \\ (i+t_1-1,j+t_2-1) \in D}} w_{t_1,t_2,s_1,s_2}^{(b,r)} \cdot p_{(i+t_1-1,j+t_2-1),b,s_1,\mathbf{x}_k,S}\big(\mathbf{a}_{I^{(r-1)}}\big) + w_{s_2}^{(b,r)} : \right.
$$

$$
\left. b \in \{1,\ldots,t\}, k \in \{1,\ldots,m\}, (i,j) \in D, s_2 \in \{1,\ldots,k_r\} \right\}
$$

don't change their signs, hence

$$
o_{(i,j),b,s_2,\mathbf{x}_k}^{(r)}\big(\mathbf{a}_{I^{(r)}}\big)
$$

$$
= \sigma \left(\sum_{s_1=1}^{k_{r-1}} \sum_{\substack{t_1,t_2 \in \{1,\ldots,M_r\} \\ (i+t_1-1,j+t_2-1) \in D}} w_{t_1,t_2,s_1,s_2}^{(b,r)} \cdot o_{(i+t_1-1,j+t_2-1),b,s_1,\mathbf{x}_k}^{(r-1)}\big(\mathbf{a}_{I^{(r-1)}}\big) + w_{s_2}^{(b,r)} \right)
$$

$$
= \max \left\{ \sum_{s_1=1}^{k_{r-1}} \sum_{\substack{t_1,t_2 \in \{1,\ldots,M_r\} \\ (i+t_1-1,j+t_2-1) \in D}} w_{t_1,t_2,s_1,s_2}^{(b,r)} \cdot o_{(i+t_1-1,j+t_2-1),b,s_1,\mathbf{x}_k}^{(r-1)}\big(\mathbf{a}_{I^{(r-1)}}\big) + w_{s_2}^{(b,r)}, 0 \right\}
$$

is either a polynomial of degree no more than r in the W_r variables $\mathbf{a}_{I^{(r)}}$ of $\mathbf{a}$ or a constant polynomial with value 0 for all $(i,j) \in D$, $b \in \{1,\ldots,t\}$, $s_2 \in \{1,\ldots,k_r\}$ and $k \in \{1,\ldots,m\}$. Hence, property 2 is also satisfied and we are able to construct our desired partition $\mathcal{S}_{L_1}$. Because of inequality (28) of property 1 it holds that

$$
|\mathcal{S}_{L_1}| \leq \prod_{r=1}^{L_1} 2 \left(\frac{2 \cdot e \cdot t \cdot k_r \cdot d_1 \cdot d_2 \cdot m \cdot r}{W_r} \right)^{W_r}.
$$

For any $(i,j) \in D$, $b \in \{1,\ldots,t\}$ and $k \in \{1,\ldots,m\}$, we define

$$
f_{(i,j),b,\mathbf{x}_k}\big(\mathbf{a}_{I^{(L_1+1)}}\big) := \sum_{s_2=1}^{k_{L_1}} w_{s_2}^{(b)} \cdot o_{(i,j),b,s_2,\mathbf{x}_k}^{(L_1)}\big(\mathbf{a}_{I^{(L_1)}}\big).
$$

For any fixed $S \in \mathcal{S}_{L_1}$, let $p_{(i,j),b,S,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}})$ denote the function $f_{(i,j),b,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}})$, when $\mathbf{a} \in S$. By construction of $\mathcal{S}_{L_1}$ this is a polynomial of degree no more than L_1+1 in the W_{L_1+1} variables $\mathbf{a}_{I^{(L_1+1)}}$ of $\mathbf{a}$. Because of condition (26) we have $t \cdot d_1^2 \cdot d_2^2 \cdot m \geq W_{L_1+1}$. Hence, by Lemma 8, the collection of polynomials

$$\Big\{ p_{(i_1,j_1),b,S,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}}) - p_{(i_2,j_2),b,S,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}}) :$$

$$(i_1,j_1),(i_2,j_2) \in D, (i_1,j_1) \neq (i_2,j_2), b \in \{1,\dots,t\}, k \in \{1,\dots,m\} \Big\}$$

attains at most

$$\Delta := 2 \left(\frac{2 \cdot e \cdot t \cdot d_1^2 \cdot d_2^2 \cdot m \cdot (L_1+1)}{W_{L_1+1}} \right)^{W_{L_1+1}}$$

distinct sign patterns when $\mathbf{a}_{I^{(L_1+1)}} \in \mathbb{R}^{W_{L_1+1}}$ and therefore the above collection of polynomials also attains at most Δ distinct sign patterns when $\mathbf{a}$ varies in $\mathbb{R}^W$ since the above polynomials depend only on the W_{L_1+1} variables $\mathbf{a}_{I^{(L_1+1)}}$ of $\mathbf{a}$. Therefore, we can partition $S \subset \mathbb{R}^W$ into Δ subregions, such that all the polynomials don't change their signs within each subregion. Doing this for all regions $S \in \mathcal{S}_{L_1}$ we get our required partition $\mathcal{S}^{(1)}$ by assembling all of these subregions. For the size of our partition $\mathcal{S}^{(1)}$ we get

$$|\mathcal{S}^{(1)}| \leq \prod_{r=1}^{L_1} 2 \cdot \left(\frac{2 \cdot t \cdot e \cdot k_r \cdot d_1 \cdot d_2 \cdot m \cdot r}{W_r} \right)^{W_r} \cdot 2 \cdot \left(\frac{2 \cdot e \cdot t \cdot d_1^2 \cdot d_2^2 \cdot m \cdot (L_1+1)}{W_{L_1+1}} \right)^{W_{L_1+1}} .$$

Fix some $S' \in \mathcal{S}^{(1)}$. Notice that, when $\mathbf{a}$ varies in S', all the polynomials

$$\Big\{ p_{(i_1,j_1),b,S,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}}) - p_{(i_2,j_2),b,S,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}}) :$$

$$(i_1,j_1),(i_2,j_2) \in D, (i_1,j_1) \neq (i_2,j_2), b \in \{1,\dots,t\}, k \in \{1,\dots,m\} \Big\}$$

don't change their signs. Hence, there is a permutation $\pi^{(b,k)}$ of the set

$$\{1,\dots,d_1 - M_{L_1} + 1\} \times \{1,\dots,d_2 - M_{L_1} + 1\}$$

for any $b \in \{1,\dots,t\}$ and $k \in \{1,\dots,m\}$ such that

$$f_{\pi^{(b,k)}((1,1)),b,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}}) \geq \cdots \geq f_{\pi^{(b,k)}((d_1-M_{L_1}+1,d_2-M_{L_1}+1)),b,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}})$$

for $\mathbf{a} \in S'$ and any $k \in \{1,\dots,m\}$ and $b \in \{1,\dots,t\}$. Therefore, it holds that

$$f_{b,k}(\mathbf{a}) = \max \Big\{ f_{(1,1),b,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}}), \dots, f_{(d_1-M_{L_1}+1,d_2-M_{L_1}+1),b,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}}) \Big\}$$

$$= f_{\pi^{(b,k)}((1,1)),b,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}}),$$

for $\mathbf{a} \in S'$. Since $f_{\pi^{(b,k)}((1,1)),b,\mathbf{x}_k}(\mathbf{a}_{I^{(L_1+1)}})$ is a polynomial within S', also $f_{b,k}(\mathbf{a})$ is a polynomial within S' with degree no more than $L_1 + 1$ and in the W_{L_1+1} variables $\mathbf{a}_{I^{(L_1+1)}}$ of $\mathbf{a} \in \mathbb{R}^W$.

In the second step we construct the partition $\mathcal{S}$ starting from partition $\mathcal{S}^{(1)}$ such that within each region $S \in \mathcal{S}$ the functions $h_k(\cdot)$ are all fixed polynomials of degree of at most $L_1 + L_2 + 2$ for $k \in \{1, \ldots, m\}$. We have

$$h_k(\mathbf{a}) = \sum_{i=1}^{k_{L_1+L_2+1}} w_i^{(L_2)} g_{i,k}^{(L_2)} \left(\mathbf{a}_{I(L_1+L_2+1)}\right) + w_0^{(L_2)} - y_k$$

where the $g_{i,k}^{(L_2)}$ are recursively defined by

$$g_{i,k}^{(r)}\left(\mathbf{a}_{I(L_1+r+1)}\right) = \sigma\left(\sum_{j=1}^{k_{L_1+r}} w_{i,j}^{(r-1)} g_{j,k}^{(r-1)}(\mathbf{a}_{I(L_1+r)})\right)$$

for $r \in \{1, \ldots, L_2\}$ and

$$g_{i,k}^{(0)}(\mathbf{a}_{I(L_1+1)}) = f_{i,k}(\mathbf{a})$$

for $i \in \{1, \ldots, k_{L_1+1}\}$ ($k_{L_1+1} = t$). As above we construct the partition $\mathcal{S}$ iteratively layer by layer, by creating a sequence $\mathcal{S}_0, \ldots, \mathcal{S}_{L_2}$, where each $\mathcal{S}_r$ is a partition of $\mathbb{R}^W$ with the following porperties:

1. We set $\mathcal{S}_0 = \mathcal{S}^{(1)}$ and, for each $r \in \{1, \ldots, L_2\}$,

$$\frac{|\mathcal{S}_r|}{|\mathcal{S}_{r-1}|} \leq 2\left(\frac{2 \cdot e \cdot k_{L_1+r+1} \cdot m \cdot (L_1 + r + 1)}{W_{L_1+r+1}}\right)^{W_{L_1+r+1}}, \tag{29}$$

2. For each $r \in \{0, \ldots, L_2\}$, and each element $S \in \mathcal{S}_r$, each $i \in \{1, \ldots, k_{L_1+r+1}\}$, and each $k \in \{1, \ldots, m\}$ when $\mathbf{a}$ varies in S,

$$g_{i,k}^{(r)}(\mathbf{a}_{I(L_1+r+1)})$$

is a fixed polynomial function in the W_{L_1+r+1} variables $\mathbf{a}_{I(L_1+r+1)}$ of $\mathbf{a}$, of total degree no more than $L_1 + r + 1$.

As we have already shown in step 1, property 2 above is satisfied for $r = 0$. Now suppose that $\mathcal{S}_0, \ldots, \mathcal{S}_{r-1}$ have been defined, and we want to define $\mathcal{S}_r$. For $S \in \mathcal{S}_{r-1}$ and $j \in \{1, \ldots, k_{L_1+r}\}$ let $p_{j,k,S}(\mathbf{a}_{I(L_1+r)})$ denote the function $g_{j,k}^{(r-1)}(\mathbf{a}_{I(L_1+r)})$, when $\mathbf{a} \in S$. By induction hypothesis $p_{j,k,S}(\mathbf{a}_{I(L_1+r)})$ is a polynomial with total degree no more than $L_1 + r$, and depends on the W_{L_1+r} variables $\mathbf{a}_{I(L_1+r)}$ of $\mathbf{a}$. Hence for any $k \in \{1, \ldots, m\}$ and $i \in \{1, \ldots, k_{L_1+r+1}\}$

$$\sum_{j=1}^{k_{L_1+r}} w_{(i,j)}^{(r-1)} p_{j,k,S}(\mathbf{a}_{I(L_1+r)}) + w_{i,0}^{(r-1)}$$

is a polynomial in the W_{L_1+r+1} variables $\mathbf{a}_{I(L_1+r+1)}$ variables of $\mathbf{a}$ with total degree no more than $L_1 + r + 1$. Because of condition (26) we have $k_{L_1+r+1} \cdot m \geq W_{L_1+r+1}$. Hence, by Lemma 8, the collection of polynomials

$$\left\{ \sum_{j=1}^{k_{L_1+r}} w_{(i,j)}^{(r-1)} p_{j,k,S}(\mathbf{a}_{I(L_1+r)}) + w_{i,0}^{(r-1)} : k \in \{1,\dots,m\}, i \in \{1,\dots,k_{L_1+r+1}\} \right\}$$

attains at most

$$\Pi := 2 \left(\frac{2 \cdot e \cdot k_{L_1+r+1} \cdot m \cdot (L_1 + r + 1)}{W_{L_1+r+1}} \right)^{W_{L_1+r+1}}$$

distinct sign patterns when $\mathbf{a}_{I(L_1+r+1)} \in \mathbb{R}^{W_{L_1+r+1}}$ and therefore the above collection of polynomials also attains at most Π distinct sign patterns when $\mathbf{a}$ varies in $\mathbb{R}^W$ since the above polynomials depend only on the W_{L_1+r+1} variables $\mathbf{a}_{I(L_1+r+1)}$ of $\mathbf{a}$. Therefore, we can partition $S \subset \mathbb{R}^W$ into Π subregions, such that all the polynomials don't change their signs within each subregion. Doing this for all regions $S \in \mathcal{S}_{r-1}$ we get our required partition $\mathcal{S}_r$ by assembling all of these subregions. In particular property 1 is then satisfied. In order to see that condition 2 is also satisfied, we can proceed analogously to step 1. Hence, when $\mathbf{a}$ varies in $S \in \mathcal{S}$ the function

$$h_k(\mathbf{a}) = \sum_{i=1}^{k_{L_1+L_2+1}} w_i^{(L)} g_{i,k}^{(L_2)} \left(\mathbf{a}_{I(L_1+L_2+1)} \right) + w_0^{(L)} - y_k$$

is a polynomial of degree no more than $L_1 + L_2 + 2$ in the W variables of $\mathbf{a} \in \mathbb{R}^W$ for any $k \in \{1,\dots,m\}$. For the size of our partition $\mathcal{S}$ we get

$$|\mathcal{S}| \leq \prod_{r=1}^{L_1} 2 \cdot \left(\frac{2 \cdot e \cdot t \cdot k_r \cdot d_1 \cdot d_2 \cdot m \cdot r}{W_r} \right)^{W_r} \cdot 2 \cdot \left(\frac{2 \cdot e \cdot d_1^2 \cdot d_2^2 \cdot m \cdot (L_1 + 1)}{W_{L_1+1}} \right)^{W_{L_1+1}}$$
$$\cdot \prod_{r=1}^{L_2} 2 \cdot \left(\frac{2 \cdot e \cdot k_{L_1+r+1} \cdot m \cdot (L_1 + r + 1)}{W_{L_1+r+1}} \right)^{W_{L_1+r+1}}$$
$$\leq \prod_{r=1}^{L_1+L_2+1} 2 \cdot \left(\frac{2 \cdot e \cdot t \cdot k_r \cdot d_1^2 \cdot d_2^2 \cdot m \cdot r}{W_r} \right)^{W_r}$$

By condition (26) and another application of Lemma 10 it holds for any $S' \in \mathcal{S}$ that

$$|\{(\mathrm{sgn}(h_1(\mathbf{a})),\dots,\mathrm{sgn}(h_m(\mathbf{a}))) : \mathbf{a} \in S'\}|$$
$$\leq 2 \cdot \left(\frac{2 \cdot e \cdot m \cdot (L_1 + L_2 + 2)}{W} \right)^W .$$

Now we are able to bound K via equation (27) and because K is an upper bound for the growth function we set $k_{L_1+L_2+2} = 1$ and get

$$\Pi_{\mathrm{sgn}(\mathcal{H})}(m) \leq \prod_{r=1}^{L_1+L_2+2} 2 \cdot \left(\frac{2 \cdot e \cdot t \cdot k_r \cdot d_1^2 \cdot d_2^2 \cdot r \cdot m}{W_r} \right)^{W_r}$$

$$\overset{(24)}{\leq} 2^{L_1+L_2+2} \cdot \left(\frac{\sum_{r=1}^{L_1+L_2+2} 2 \cdot e \cdot t \cdot k_r \cdot d_1^2 \cdot d_2^2 \cdot r \cdot m}{\sum_{r=1}^{L_1+L_2+2} W_r} \right)^{\sum_{r=1}^{L_1+L_2+2} W_r}$$

$$= 2^{L_1+L_2+2} \cdot \left(\frac{R \cdot m}{\sum_{r=1}^{L_1+L_2+2} W_r} \right)^{\sum_{r=1}^{L_1+L_2+2} W_r} , \qquad (30)$$

with $R := 2etd_1^2 d_2^2 \sum_{r=1}^{L_1+L_2+2} k_r \cdot r$. In the third row we used inequation (24) of Lemma 7. Without loss of generality, we can assume that $\mathrm{VCdim}(\mathcal{H}) \geq \sum_{r=1}^{L_1+L_2+2} W_r$ because in the case $\mathrm{VCdim}(\mathcal{H}) < \sum_{r=1}^{L_1+L_2+2} W_r$ we have

$$
\begin{aligned}
\mathrm{VCdim}(\mathcal{H}) \ &< (L_1 + L_2 + 2) \cdot W \\
&\overset{(25)}{\leq} 2 \cdot (L_1 + L_2 + 2)^2 \cdot t \cdot M_{max}^2 \cdot k_{max}^2 \\
&\leq c_{10} \cdot L_{max}^2
\end{aligned}
$$

for some constant $c_{10} > 0$ which only depends on M_{max} and k_{max} and get the assertion by Lemma (8). Hence we get by the definition of the VC–dimension and inequality (30) (which only holds for $m \geq W$)

$$2^{\mathrm{VCdim}(\mathcal{H})} = \Pi_{\mathrm{sgn}(\mathcal{H})}(\mathrm{VCdim}(\mathcal{H})) \leq 2^{L_1+L_2+2} \cdot \left(\frac{R \cdot \mathrm{VCdim}(\mathcal{H})}{\sum_{r=1}^{L_1+L_2+2} W_r} \right)^{\sum_{r=1}^{L_1+L_2+2} W_r} .$$

Since

$$R \geq 2 \cdot e \cdot t \cdot d_1^2 \cdot d_2^2 \cdot \sum_{r=1}^{1+1+2} r \geq 2 \cdot e \cdot t \cdot d_1^2 \cdot d_2^2 \cdot 10 \geq 16$$

Lemma 12 below (with parameters R, $m = \mathrm{VCdim}(\mathcal{H})$, $w = \sum_{r=1}^{L_1+L_2+2} W_r$ and $L = L_1 + L_2 + 2$) implies that

$$
\begin{aligned}
\mathrm{VCdim}(\mathcal{H}) &\leq (L_1 + L_2 + 2) + \left(\sum_{r=1}^{L_1+L_2+2} W_r \right) \cdot \log_2(2 \cdot R \cdot \log_2(R)) \\
&\leq (L_1 + L_2 + 2) + (L_1 + L_2 + 2) \cdot W \\
&\qquad \cdot \log_2(2 \cdot (2 \cdot e \cdot t \cdot d_1^2 \cdot d_2^2 \cdot (L_1 + L_2 + 2) \cdot k_{max})^2) \\
&\leq 2 \cdot (L_1 + L_2 + 2) \cdot W \cdot \log_2 \left((2 \cdot e \cdot t \cdot (L_1 + L_2 + 2) \cdot k_{max} \cdot d_1 \cdot d_2)^4 \right) \\
&\overset{(25)}{\leq} 16 \cdot t \cdot (L_1 + L_2 + 2)^2 \cdot k_{\max}^2 \cdot M_{\max}^2 \\
&\qquad \cdot \log_2 \left(2 \cdot e \cdot t \cdot (L_1 + L_2 + 2) \cdot k_{max} \cdot d_1 \cdot d_2 \right) \\
&\leq c_{10} \cdot L_{max}^2 \cdot \log_2(L_{max} \cdot d_1 \cdot d_2),
\end{aligned}
$$

for some constant $c_{10} > 0$ which only depends on $k_{\max}$ and $M_{\max}$. In the third row we used equation (25) for the total number of weights W. Now we make use of Lemma 8 and finally get

$$V_{\mathcal{F}^+} \leq c_{10} \cdot L_{max}^2 \cdot \log_2(L_{max} \cdot d_1 \cdot d_2).$$

□

Lemma 12 *Suppose that $2^m \leq 2^L \cdot (m \cdot R/w)^w$ for some $R \geq 16$ and $m \geq w \geq L \geq 0$. Then,*

$$m \leq L + w \cdot \log_2(2 \cdot R \cdot \log_2(R)).$$

Proof. See Lemma 16 in Bartlett et al. (2019). □

Proof of Lemma 7. Using Lemma 11 and

$$V_{T_{c_4 \cdot \log n} \mathcal{F}^+} \leq V_{\mathcal{F}^+},$$

we can conclude from this together with Lemma 9.2 and Theorem 9.4 in Györfi et al. (2002)

$$
\begin{aligned}
&\mathcal{N}_1 \left(\epsilon, T_{c_4 \cdot \log n} \mathcal{F}, \mathbf{x}_1^n \right) \\
&\leq 3 \cdot \left(\frac{4e \cdot c_4 \cdot \log n}{\epsilon} \cdot \log \frac{6e \cdot c_4 \cdot \log n}{\epsilon} \right)^{V_{T_{c_4 \cdot \log n} \mathcal{F}^+}} \\
&\leq 3 \cdot \left(\frac{6e \cdot c_4 \cdot \log n}{\epsilon} \right)^{2 \cdot c_{10} \cdot L_{max}^2 \cdot \log(L_{max} \cdot d_1 \cdot d_2)}.
\end{aligned}
$$

This completes the proof of Lemma 7. □

6.4 Proof of Theorem 1

W.l.o.g. we assume that n is so large that $c_4 \cdot \log n \geq 2$ holds. Then $z > 1/2$ holds if and only if $T_{c_4 \cdot \log n} z > 1/2$ holds, and consequently we have

$$
f_n(\mathbf{x}) = \begin{cases} 1, & \text{if } T_{c_4 \cdot \log n} \eta_n(\mathbf{x}) \geq \frac{1}{2} \\ 0, & \text{elsewhere} \end{cases}
$$

Hence Lemma 1 implies that it suffices to show

$$\mathbf{E} \int |T_{c_4 \cdot \log n} \eta_n(\mathbf{x}) - \eta(\mathbf{x})|^2 \mathbf{P}_\mathbf{X}(d\mathbf{x}) \leq c_{11} \cdot \log(d_1 \cdot d_2) \cdot (\log n)^4 \cdot \max \left\{ n^{-\frac{2 \cdot p_1}{2 \cdot p_1 + 4}}, n^{-\frac{2 \cdot p_2}{2 \cdot p_2 + d^*}} \right\}.$$

By Lemma 2 we know

$$
\begin{aligned}
&\mathbf{E} \int |T_{c_4 \cdot \log n} \eta_n(\mathbf{x}) - \eta(\mathbf{x})|^2 \mathbf{P}_\mathbf{X}(d\mathbf{x}) \\
&\leq \frac{c_{12} \cdot (\log n)^2 \cdot \sup_{\mathbf{x}_1^n} \left(\log \left(\mathcal{N}_1 \left(\frac{1}{n \cdot c_4 \log(n)}, T_{c_4 \log(n)} \mathcal{F}, \mathbf{x}_1^n \right) \right) + 1 \right)}{n} \\
&\quad + 2 \cdot \inf_{f \in \mathcal{F}} \int |f(\mathbf{x}) - \eta(\mathbf{x})|^2 \mathbf{P}_\mathbf{X}(d\mathbf{x}),
\end{aligned}
$$

where $\mathcal{F} := \mathcal{F}_t\left(\mathbf{L}, \mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \mathbf{M}\right)$. Application of Lemma 7 yields

$$\frac{c_{12} \cdot (\log n)^2 \cdot \sup_{\mathbf{x}_1^n}\left(\log\left(\mathcal{N}_1\left(\frac{1}{n \cdot c_4 \log(n)}, T_{c_4 \log(n)}\mathcal{F}, \mathbf{x}_1^n\right)\right) + 1\right)}{n}$$

$$\leq c_{13} \cdot \frac{\log(d_1 \cdot d_2) \cdot (\log n)^3 \cdot L_{max}^2 \cdot \log L_{max}}{n}$$

$$\leq c_{14} \cdot \log(d_1 \cdot d_2) \cdot \log(n)^4 \cdot \max\left\{n^{-\frac{2 \cdot p_1}{2 \cdot p_1 + 4}}, n^{-\frac{2 \cdot p_2}{2 \cdot p_2 + d^*}}\right\},$$

where $L_{max} = \max\{L_1, L_2\}$. Next we derive a bound on the approximation error

$$\inf_{f \in \mathcal{F}} \int |f(\mathbf{x}) - \eta(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}).$$

Because of the assumption on η, we have

$$\eta(\mathbf{x}) = g(m_1(\mathbf{x}), \ldots, m_{d^*}(\mathbf{x}))$$

such that m_a satisfies a max-pooling with index set

$$I = \{0, \ldots, 2^l - 1\} \times \{0, \ldots, 2^l - 1\}$$

for any $a \in \{1, \ldots, d^*\}$ and m_a satisfies a hierarchical model with level l with functions

$$g_{k,s}^{(a)} : \mathbb{R}^4 \to [0, 1].$$

for $a \in \{1, \ldots, d^*\}$, $k \in \{1, \ldots, l\}$ and $s \in \{1, \ldots, 4^{l-k}\}$. Then, for any $a \in \{1, \ldots, d^*\}$, $k \in \{1, \ldots, l\}$ and any $s \in \{1, \ldots, 4^{l-k}\}$ let $\bar{g}_{net,k,s}^{(a)} \in \mathcal{G}_4\left(L_n, \mathbf{k}^{(2)}\right)$ and $\bar{g}_{net} \in \mathcal{G}_{d^*}\left(L_n, \mathbf{k}^{(2)}\right)$ be the neural networks from Lemma 3 which satisfies

$$\|g_{k,s}^{(a)} - \bar{g}_{net,k,s}^{(a)}\|_{[-2,2]^4, \infty} \leq c_{14} \cdot L_n^{-\frac{2 \cdot p_1}{4}} \leq c_{15} \cdot n^{-\frac{p_1}{2 \cdot p_1 + 4}}, \tag{31}$$

and

$$\|g - \bar{g}_{net}\|_{[-2,2]^{d^*}, \infty} \leq c_{14} \cdot L_n^{-\frac{2 \cdot p_2}{d^*}} \leq c_{15} \cdot n^{-\frac{p_2}{2 \cdot p_2 + d^*}}. \tag{32}$$

Then Lemma 5 let us choose

$$\bar{m}_1, \ldots, \bar{m}_{d^*} \in \mathcal{F}(L_1, \mathbf{k}^{(1)}, \mathbf{M}, \tilde{\mathbf{d}}) \tag{33}$$

such that

$$\bar{m}_a(\mathbf{x}) = \max_{(i,j) \in \mathbb{Z}^2 : (i,j) + I \subset \{1, \ldots, d_1\} \times \{1, \ldots, d_2\}} \bar{f}(x_{(i,j)+I}),$$

where $\bar{f}$ satisfies

$$\bar{f} = \bar{f}_{I,1}$$

for some $\bar{f}_{k,s} : [0,1]^{\{1, \ldots, 2^k\} \times \{1, \ldots, 2^k\}} \to \mathbb{R}$ recursively defined by

$$\bar{f}_{k,s}(\mathbf{x}) = \bar{g}_{net,k,s}\left(\bar{f}_{k-1, 4 \cdot (s-1)+1}(\mathbf{x}_{\{1, \ldots, 2^{k-1}\} \times \{1, \ldots, 2^{k-1}\}}),\right.$$

$$\bar{f}_{k-1,4\cdot(s-1)+2}\big(\mathbf{x}_{\{2^{k-1}+1,\ldots,2^k\}\times\{1,\ldots,2^{k-1}\}}\big),$$
$$\bar{f}_{k-1,4\cdot(s-1)+3}\big(\mathbf{x}_{\{1,\ldots,2^{k-1}\}\times\{2^{k-1}+1,\ldots,2^k\}}\big),$$
$$\bar{f}_{k-1,4\cdot s}\big(\mathbf{x}_{\{2^{k-1}+1,\ldots,2^k\}\times\{2^{k-1}+1,\ldots,2^k\}}\big)\big)$$

for $k = 2, \ldots, l, s = 1, \ldots, 4^{l-k}$, and

$$\bar{f}_{1,s}(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) = \bar{g}_{net,1,s}(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2})$$

for $s = 1, \ldots, 4^{l-1}$. Due to property (33) it holds that

$$\bar{g}_{net} \circ (\bar{m}_1, \ldots, \bar{m}_{d^*}) \in \mathcal{F}. \tag{34}$$

Since the functions $g_{k,s}^{(a)}$ are $[0,1]$–valued, inequalities (31) and (32) let us choose c_1 in the definition of L_n sufficiently large such that the triangle inequality implies that

$$\|\bar{g}_{net,k,s}^{(a)}\|_{[-2,2]^4,\infty} \leq 2$$

for all $a \in \{1, \ldots, d^*\}$, $k \in \{1, \ldots, l\}$ and $s \in \{1, \ldots, 4^{l-k}\}$. Then Lemma 4 implies

$$\inf_{f \in \mathcal{F}} \int |f(\mathbf{x}) - \eta(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x})$$

$$\overset{(34)}{\leq} \int |\bar{g}_{net}(\bar{m}_1(\mathbf{x}), \ldots, \bar{m}_{d^*}(\mathbf{x})) - g(m_1(\mathbf{x}), \ldots, m_{d^*}(\mathbf{x}))|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x})$$

$$\leq c_{16} \cdot \left(\max_{a \in \{1,\ldots,d^*\}, k \in \{1,\ldots,l\}, s \in \{1,\ldots,4^{l-j}\}} \left\{ \|g_{k,s}^{(a)} - \bar{g}_{net,k,s}^{(a)}\|_{[-2,2]^4,\infty}, \|g - \bar{g}_{net}\|_{[-2,2]^{d^*},\infty} \right\} \right)^2$$

$$\overset{(31),(32)}{\leq} c_{17} \cdot \max \left\{ n^{-\frac{2\cdot p_1}{2\cdot p_1+4}}, n^{-\frac{2\cdot p_2}{2\cdot p_2+d^*}} \right\}.$$

Summarizing the above results, the proof is complete. $\qquad\square$

7 Acknowledgment

References

[1] Anthony, M., and Bartlett, P. L. (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press.

[2] Bagirov, A. M., Clausen, C., and Kohler, M. (2009). Estimation of a regression function by maxima of minima of linear functions. *IEEE Transactions on Information Theory*, **55**, pp. 833-845.

[3] Bartlett, P. L., Harvey, N., Liaw, C., and Mehrabian, A. (2019). Nearly-tight VC-dimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, **20**, pp. 1–17.

[4] Bauer, B., and Kohler, M. (2019). On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *Annals of Statistics*, **47**, pp. 2261-2285.

[5] Chang, L.-B., Borenstein, E., Zhang, W., and Geman, Stuart (2017). Maximum likelihood features for generative image models. *The Annals of Applied Statistics*, **11**, pp. 1275-1308.

[6] Cover, T. M. (1968). Rates of convergence of nearest neighbor procedures. In *Proceedings of the Hawaii International Conference on Systems Sciences*, pp. 413-415, Honolulu, HI.

[7] Devroye, L. (1982). Necessary and sufficient conditions for the almost everywhere convergence of nearest neighbor regression function estimates. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, **61**, pp. 467-481.

[8] Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory Of Pattern Recognition*. Springer.

[9] Devroye, L., and Wagner, T. J. (1980). Distribution-free consistency results in nonparametric discrimination and regression function estimation. *Annals of Statistics*, **8**, pp. 231-239.

[10] Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. arXiv: 1811.03804.

[11] Eckle, K., and Schmidt-Hieber, J. (2019). A comparison of deep networks with ReLU activation function and linear spline-type methods. *Neural Networks*, **110**, pp. 232-242.

[12] Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*. MIT press.

[13] Györfi, L., Kohler, M., Krzyżak, A., and Walk, H. (2002). *A Distribution–Free Theory of Nonparametric Regression*. Springer.

[14] Imaizumi, M., and Fukamizu, K. (2019). Deep neural networks learn non-smooth functions effectively. *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS 2019)*. Naha, Okinawa, Japan.

[15] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. arXiv: 1408.5882.

[16] Kim, Y., Ohn, I, and Kim, D. (2019). Fast convergence rates of deep neural networks for classification. arXiv: 1812.03599.

[17] Korostelev, A., and Tsybakov, A. (1993). *Minimax Theory of Image Reconstruction.* Springer, New York.

[18] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira et al. (Eds.), *Advances In Neural Information Processing Systems* **25**, pp. 1097-1105. Red Hook, NY: Curran.

[19] Kohler, M., and Krzyżak, A. (2017). Nonparametric regression based on hierarchical interaction models. *IEEE Transaction on Information Theory*, **63**, pp. 1620-1630.

[20] Kohler, M., and Krzyżak, A. (2019). Over-parametrized deep neural networks do not generalize well. arXiv:1912.03925.

[21] Kohler, M., Krzyżak, A., and Langer, S. (2019). Deep learning and MARS: a connection. arXiv: 1908.11140.

[22] Kohler, M., and Langer, S. (2019). On the rate of convergence of fully connected very deep neural network regression estimates. arXiv: 1908.11133

[23] Krizhevsky, A.(2009). Learning Multiple Layers of Features from Tiny Images. University of Toronto.

[24] LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature*, **521**, pp.436-444.

[25] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, **1**, pp. 541-551.

[26] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. ()1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**, pp.2278-2324.

[27] Lin, S., and Zhang, J. (2019). Generalization bounds for convolutional neural networks. arXiv: 1910.01487.

[28] Oono, K., and Suzuki, T. (2019) Approximation and Non-parametric Estimation of ResNet-type Convolutional Neural Networks. arXiv: 1903.10047

[29] Rawat, W., and Wang, Z. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, **29**, pp. 2352-2449.

[30] Schmidhuber, J. (2015). Deep learning in neural networks: an overview. *Neural Networks*, **61**, pp. 85-117.

[31] Schmidt-Hieber, J. (2019). Nonparametric regression using deep neural networks with ReLU activation function. To appear in *Annals of Statistics* (with discussion). arXiv:1708.06633v2.

[32] Stone, C. J. (1982). Optimal global rates of convergence for nonparametric regression. *Annals of Statistics*, **10**, pp. 1040-1053.

[33] Suzuki, T., and Nitanda, A. (2019). Deep learning is adaptive to intrinsic diemsnionality of model smoothness in anisotropic Besov space. arXiv: 1910.12799.

[34] Yarotsky, D. (2018). Universal approximations of invariant maps by neural networks. arXiv: 1804.10306.

[35] Yarotsky, D., and Zhevnerchuk, A. (2019). The phase diagram of approximation rates for deep neural networks. arXiv: 1906.09477.