Annals of the Institute of Statistical Mathematics (2022) 74:1085–1108
https://doi.org/10.1007/s10463-022-00828-4

# On the rate of convergence of image classifiers based on convolutional neural networks

**Michael Kohler[1] · Adam Krzyżak[2] · Benjamin Walter[1]**

Received: 20 May 2021 / Revised: 16 March 2022 / Accepted: 17 March 2022 /
Published online: 27 April 2022


## 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 a posteriori 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. Furthermore, the obtained result gives an indication why convolutional neural networks are able to outperform the standard feedforward neural networks in image classification. Our classifiers are compared with various other classification methods using simulated data. Furthermore, the performance of our estimates is also tested on real images.

**Keywords** Curse of dimensionality · Convolutional neural networks · Image classification · Rate of convergence

Michael Kohler was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Projektnummer 449102119.
Adam Krzyżak gratefully acknowledges the support from the Natural Sciences and Engineering Research Council of Canada Under Grant RGPIW-2015-06412.

✉ Benjamin Walter
bwalter@mathematik.tu-darmstadt.de

Michael Kohler
kohler@mathematik.tu-darmstadt.de

Adam Krzyżak
krzyzak@cs.concordia.ca

[1] Fachbereich Mathematik, Technische Universität Darmstadt, Schlossgartenstr. 7, 64289 Darmstadt, Germany

[2] Department of Computer Science and Software Engineering, Concordia University, 1455 de Maisonneuve Blvd. West, Montreal, QC H3G 1M8, Canada

# 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 et al. (2012) and Kim (2014) concerning applications in image classification or language recognition, respectively. These networks can be considered as a special case of the standard deep feedforward neural networks, where symmetry constraints are imposed on the weights of the networks. For general standard deep feedforward neural networks 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, 2020; Kohler and Langer, 2021; 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

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

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 gray 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 a posteriori 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 et al. (1996)). Set

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

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

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

### 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

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

is an estimate of a posteriori 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–Schwarz inequality)

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

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 $\eta_n$ of a posteriori probability.

It is well-known that the bound in (2) is not tight, therefore classification is easier than regression estimation (cf., Devroye et al., 1996).

In the sequel, we will nevertheless solve an image classification problem via regression estimation by estimating the a posteriori probability and defining the corresponding plug-in classifier. We do this because, as described below, it is necessary to impose conditions on the underlying distribution and since this can be done by restricting the structure of a posteriori 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. Moreover, by considering the corresponding least squares estimate of the a posteriori probability, no further assumptions on the distribution of $(\mathbf{X}, Y)$ are required. In other papers that consider a classification problem and where a different loss function is used, such Kim et al. (2021) and Liu et al. (2021), they have made additional assumptions on the distribution of $(\mathbf{X}, Y)$.

### 1.4 A hierarchical max-pooling model for a posteriori 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; Devroye, 1982). Therefore, in Definition 1, we will introduce a model for a posteriori probability which makes assumptions on its structure and smoothness. The new model introduced here is specifically designed for applications in image classification.

For the structural assumptions in our model, consider an application where a human has to decide about a class of an image, e.g., the human has to decide whether an image contains a certain speed limit traffic sign, but not a dead-end or a no-entry traffic sign (an example image is shown in Fig. 1). Then, for each of the three traffic signs, the human will survey the whole image and look at each subpart of the image whether it contains the particular 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 the traffic sign is simply the maximum of the probabilities for each subpart of the image. Furthermore, the human takes decision whether a given subpart of the image contains the traffic sign or not by taking several decisions whether the image contains parts of the traffic sign or not, and by combining these decisions about the different parts hierarchically. To decide whether the image belongs to the described class, the human finally checks whether its estimated probability for the speed limit sign is sufficiently large and the probabilities for the other two traffic signs are correspondingly small. To do this, e.g., the human could use the function $g : \mathbb{R}^3 \to \mathbb{R}$, which is given by $g((p_1, p_2, p_3)) = \max\{p_1, 1 - p_2, 1 - p_3\}$.

For the structural assumptions in our model, we summarize the ideas above:

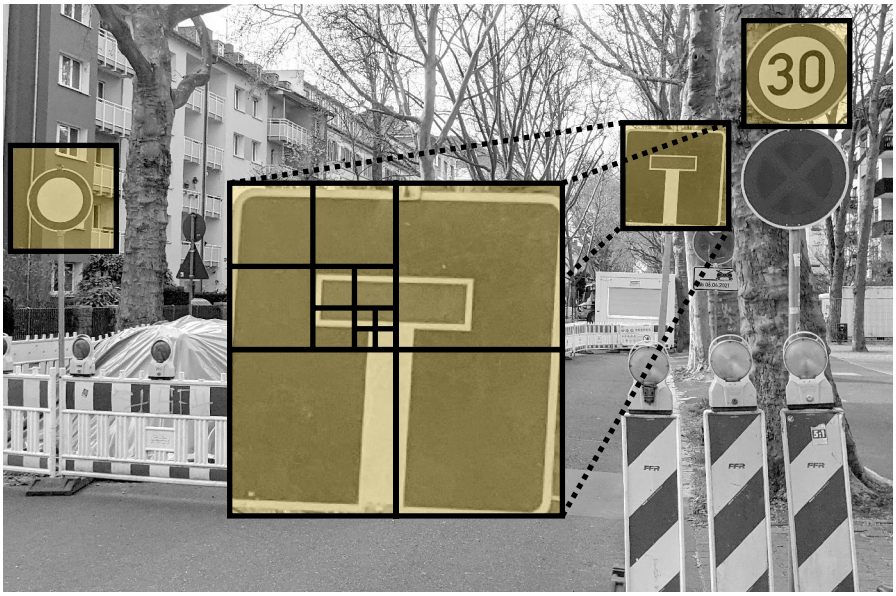


**Fig. 1** The shown image does not belong to the class described above, because it contains both a dead-end traffic sign and a no-entry traffic sign. We have marked the subparts with the corresponding traffic signs and have indicated how the decisions about the subparts are composed of the decisions of smaller and smaller subparts

1. We assume that the probability whether an image belongs to a class is obtained by applying a function to the probabilities of the existence of several objects.
2. The probability whether the image contains a certain object is obtained by looking at each subpart of the image whether it contains the object or not. Here, we assume that the probability that the whole image contains the object is the maximum of the probabilities for each subpart of the image.
3. Furthermore, we assume that the probability whether a given subpart contains a particular object is hierarchically composed of the decisions of smaller and smaller subparts.

Combining these three ideas leads us to the generalized hierarchical max-pooling model introduced 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 $I \subseteq \{1, \dots, d_1\} \times \{1, \dots, d_2\}$ and $\mathbf{x} = (x_i)_{i \in \{1,\dots,d_1\} \times \{1,\dots,d_2\}} \in [0,1]^{\{1,\dots,d_1\} \times \{1,\dots,d_2\}}$ 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 $m \, : \, [0,1]^{\{1,\dots,d_1\} \times \{1,\dots,d_2\}} \to \mathbb{R}$.

(a) We say that $m$ satisfies a max-pooling model with index set

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

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

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

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

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

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

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

such that we have

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

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

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

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 Sect. 1.7 for the definition of $(p, C)$–smoothness).

***Remark 1*** The generalized hierarchical max-pooling model with smoothness constraints is a special case of the function class considered in Schmidt-Hieber (2020) and also a special case of the more general hierarchical composition model with order and smoothness constraint from Kohler and Langer (2021).

***Remark 2*** In our definition, the function $g_{k,s}$ is mapping from $\mathbb{R}^4$ for simplicity, since this lets us easily implement the above idea of hierarchical decisions.

***Remark 3*** The parameter $d^*$ in our model describes the number of objects that are relevant for the classification of a random image. In our above example from Fig. 1, we assume that the a posteriori probability satisfies a generalized hierarchical max-pooling model of order $d^* = 3$, since the correct classification depends only on the existence of the three specific traffic signs. In our example, as already mentioned, the function $g : \mathbb{R}^3 \to \mathbb{R}$ could be given by $g(\mathbf{x}) = \max\{x_1, 1 - x_2, 1 - x_3\}$. The function $g$ is then $(1, \sqrt{3})$-smooth with respect to the Euclidean norm (see Sect. 1.7 for the definition of $(p, C)$–smoothness). The level $l$ of our model is derived from the minimum size of a square of $2^l \times 2^l$ pixels, which is sufficiently large to detect all relevant objects within a random image.

***Remark 4*** 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 7 for the generalization of our results).

### 1.5 Main results

The main contributions of 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 a posteriori 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 that in case that a posteriori 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 a posteriori probability it is possible to circumvent the curse of dimensionality in image classification by using convolutional neural networks.

In the proof, we use standard bounds from empirical process theory (cf., Lemma 10 in the supplement). The main technical novelty is the bound on the approximation error. Here, a connection between standard feedforward neural networks and convolutional neural networks is made (cf., Lemma 2 in the supplement) which, using the approximation result from Theorem 2 b) in Kohler and Langer (2021) for fully connected standard deep feedforward neural networks, yields us an approximation result for the generalized hierarchical max-pooling model by convolutional neural networks. Here, the difference with existing works such as Oono and Suzuki (2019) and Petersen and Voigtlaender (2020), which use the approximation capability of fully connected neural networks, is that we do not use the approximation capability of the fully connected neural networks to bound the overall approximation error of our convolutional neural network. Indeed we, on the other hand, have adapted the convolutional neural network architecture to the generalized hierarchical max-pooling function class and use the fully connected neural networks only as approximations of the functions $g_{k,s}^{(a)}$ and $g^{(a)}$ of the generalized hierarchical max-pooling model. For the estimation error, our technical novelty is a bound on the VC-dimension for convolutional neural networks (cf., Lemma 7 in the supplement), which is a modification of Theorem 6 in Bartlett et al. (2019). Specifically, we adapted the proof therein to our architecture of convolutional neural networks, which additionally includes max-pooling layers.

### 1.6 Discussion of related results

Convolutional neural networks, introduced by LeCun et al. (1989), have become the leading techniques in pattern recognition applications, cf., e.g., Lecun et al. (1998), LeCun et al. (2015), Goodfellow et al. (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 standard deep feedforward neural networks and obtained rate of convergence results for estimates based on convolutional neural networks similar to standard 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. More specifically, in Oono and Suzuki (2019), it is shown that an arbitrary block-sparse fully connected neural network, which consists of $M$ parallel blocks, can be realized by a convolutional residual neural network (i.e., a convolutional neural network that contains skip layers connections). Here, the skip layers connections are used to sum up the scaled outputs of each block in order to compute the output of the overall network, while in the proof of our result we successively compute several fully connected neural networks with small input dimension that share the same input by storing computed values in corresponding channels of the convolutional neural network. In Liu et al. (2021), they consider convolutional residual neural network architectures similar to Oono and Suzuki (2019) in a statistical setting for binary classification. Their idea is to exploit low-dimensional geometric structures in the input data to achieve dimensionality reduction. Instead of directly using the approximation capability of fully connected neural networks, cardinal B-splines and the geometric structure of the input data are used. 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 overparameterized convolutional neural networks, cf., e.g., Du et al. (2018). But, as was shown by a counterexample in Kohler and Krzyżak (2021), overparameterized deep neural networks minimizing the empirical loss do not, in general, generalize well. Remarkable approximation properties were obtained in Zhou (2020). Here, the network architecture has only one channel per layer, with the size of this channel increasing linearly with the number of layers, leading to extremely wide network architectures. 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, 2020; Kohler and Langer, 2021; Suzuki and Nitanda, 2019). As already mentioned in Remark 1, especially the results from Schmidt-Hieber (2020) and Kohler and Langer (2021) are closely related to our analysis. In Schmidt-Hieber (2020) the neural networks must be sparse, that is, the number of weights that are nonzero is specified, but an exact architecture is not provided. In Kohler and Langer (2021), it is shown that a similar convergence rate holds for fully connected standard deep feedforward neural networks, which makes their network architecture easier to apply.

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 et al. (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 et al. (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.

Classification problem with standard deep feedforward neural networks has been analyzed in Kim et al. (2021) and Hu et al. (2020). In Hu et al. (2020) they derive sharp convergence rates for standard deep feedforward neural networks in a teacher-student setting. However, in this setting we obtain only limited hints on how these results can be applied or how they can be used to identify good network architectures in applications.

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 $\min_{\mathbf{z} \in D} f(\mathbf{z})$ exists and if $\mathbf{x}$ satisfies $\mathbf{x} \in D$ and $f(\mathbf{x}) = \min_{\mathbf{z} \in D} f(\mathbf{z})$. 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 $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ with $\sum_{j=1}^d \alpha_j = q$ the partial derivative $\frac{\partial^q f}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}$ exists and satisfies

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

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

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

The $\varepsilon$–covering number of $\mathcal{F}$ on $\mathbf{x}_1^n$ is the size $N$ of the smallest $\varepsilon$–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

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

By $\mathbf{P}_X$ we denote the distribution of a random variable $X$, i.e., the measure associated with the random variable $X$.

We denote by $c_i > 0$ for $i \in \mathbb{N}$ constants that do not depend on the image dimensions $d_1, d_2 \in \mathbb{N}$ and the sample size $n \in \mathbb{N}$.

### 1.8 Outline of the paper

In Sect. 2, the convolutional neural network image classifiers used in this paper are defined. The main result is presented in Sect. 3 and proven in Sect. 6. The finite sample size behavior of our classifier is analyzed by applying it to simulated and real data in Sects. 4 and 5 respectively.

## 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 neural network consisting of several layers to the results of these networks.

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

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

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

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

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

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

for $i \in \{1, \dots, k_0\}$, 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 form (3) with parameters $L$ and $\mathbf{k} = (k_1, \dots, k_L)$ 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 $[0,1]^{\{1,\dots,d_1\}\times\{1,\dots,d_2\}}$–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 $M_r \in \{1, \dots, \min\{d_1, d_2\}\}$, where $r \in \{1, \dots, L\}$. 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)

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

the weights

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

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

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

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

$$f_{\mathbf{w},\mathbf{w}_{bias},\mathbf{w}_{out}}(\mathbf{x}) = \max\left\{ \sum_{s_2=1}^{k_L} w_{s_2} \cdot o^{(L)}_{(i,j),s_2} \;:\; i \in \{1, \dots, d_1 - M_L + 1\}, \right.$$
$$\left. j \in \{1, \dots, d_2 - M_L + 1\} \right\},$$

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

We start with

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

Then we define recursively

$$o^{(r)}_{(i,j),s_2} = \sigma\left( \sum_{s_1=1}^{k_{r-1}} \sum_{\substack{t_1,t_2\in\{1,\dots,M_r\} \\ (i+t_1-1, j+t_2-1)\in D}} w^{(r)}_{t_1,t_2,s_1,s_2} \cdot o^{(r-1)}_{(i+t_1-1,j+t_2-1),s_1} + w^{(r)}_{s_2} \right) \quad (4)$$

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

Let $\mathcal{F}(L, \mathbf{k}, \mathbf{M})$ be the set of all functions of the above form with parameters $L$, $\mathbf{k} = (k_1, \dots, k_L)$ and $\mathbf{M} = (M_1, \dots, M_L)$. With the definition of the index set $D$ in (4) we use a so-called zero padding which is illustrated in Fig. 2. 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

$$\begin{aligned}&\mathcal{F}_t\big(\mathbf{L}, \mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \mathbf{M}\big)\\ &= \Big\{ g \circ (f_1, \dots, f_t) : f_1, \dots, f_t \in \mathcal{F}\big(L^{(1)}, \mathbf{k}^{(1)}, \mathbf{M}\big), g \in \mathcal{G}_t\big(L^{(2)}, \mathbf{k}^{(2)}\big) \Big\}.\end{aligned}$$

It depends on the parameters

$$\begin{aligned}&\mathbf{L} = (L^{(1)}, L^{(2)}),\ \mathbf{k}^{(1)} = \Big(k_1^{(1)}, \dots, k_{L^{(1)}}^{(1)}\Big),\ \mathbf{k}^{(2)} = \Big(k_1^{(2)}, \dots, k_{L^{(2)}}^{(2)}\Big),\\ &\mathbf{M} = (M_1, \dots, M_{L^{(1)}})\end{aligned}$$

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

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

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

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

For simplicity, we assume here that the minimum of the empirical $L_2$ risk (5) exists. If this is not the case, we can choose an estimator whose empirical $L_2$ risk is close enough to the infimum. In addition, using the approach of Schmidt-Hieber (2020), it is possible to adjust our main result such that it is valid for an estimate whose empirical $L_2$ risk is not equal to the minimal possible value. To do this, it suffices to use

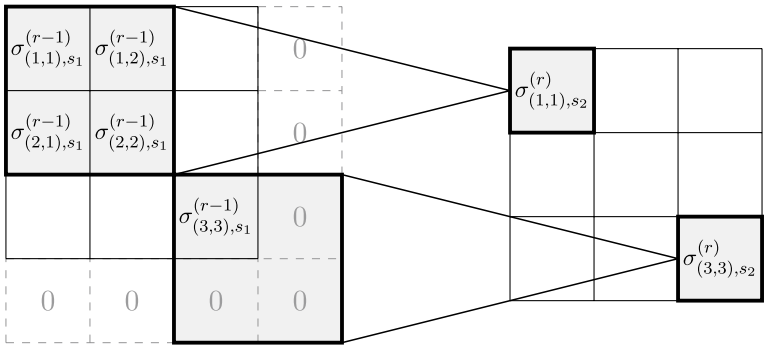


**Fig. 2** Illustration of the zero padding for $M_r = 2$ and $d_1 = d_2 = 3$

empirical process theory in order to adjust Lemma 10 from the supplement such that it contains a corresponding additional error term. In this case, our upper bound in Theorem 1 will contain an additional error term depending on the distance between the empirical $L_2$ risk and its minimal value. In this work, we ignore the optimization error that occurs in practical applications.

## 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 $[0,1]^{\{1,\dots,d_1\}\times\{1,\dots,d_2\}} \times \{0,1\}$-valued random variables with $n > 1$. Assume that a posteriori 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 plug-in classifier based on $\mathcal{F}_t\left(\mathbf{L}, \mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \mathbf{M}\right)$ as in Sect. 2. Then we have*

$$\begin{aligned}&\mathbf{P}\{f_n(\mathbf{X}) \neq Y\} - \min_{f:[0,1]^{\{1,\dots,d_1\}\times\{1,\dots,d_2\}}\to\{0,1\}} \mathbf{P}\{f(\mathbf{X}) \neq Y\} \\ &\leq c_3 \cdot \sqrt{\log(d_1 \cdot d_2)} \cdot (\log n)^2 \cdot \max\left\{ n^{-\frac{p_1}{2\cdot p_1+4}}, n^{-\frac{p_2}{2\cdot p_2+d^*}} \right\},\end{aligned} \tag{6}$$

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

***Remark 5*** The rate of convergence $\max\left\{ n^{-\frac{p_1}{2\cdot p_1+4}}, n^{-\frac{p_2}{2\cdot p_2+d^*}} \right\}$ (up to some logarithmic factor) 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 $\eta$. We have focused on the rate of convergence here and have made no attempt to choose the constants in the definitions of $\mathbf{L}$, $\mathbf{k}^{(1)}$ and $\mathbf{k}^{(2)}$ as small as possible. Since the constant $c_3$ in (6) depends on the level $l$, the dimension $d_1 \cdot d_2$ in (6) occurs logarithmically only for the case where $2^l \ll \min\{d_1, d_2\}$.

***Remark 6*** In the proof of Theorem 1 we show that the expected $L_2$ error for a regression estimation problem, where the regression function $\mathbf{E}\{Y|\mathbf{X} = \mathbf{x}\}$ satisfies a generalized hierarchical max-pooling model, tends to zero with the rate of convergence

$$\max\left\{ n^{-\frac{2p_1}{2p_1+4}}, n^{-\frac{2p_2}{2p_2+d^*}} \right\} \tag{7}$$

(up to some logarithmic factor). Therefore, our result implies a corresponding result for regression estimation. In Section D of the supplement, we show that the rate of convergence of (7) is the optimal minimax rate for the corresponding regression estimation problem. Consequently, the least squares regression estimator introduced in the proof of Theorem 1, after a suitable truncation, achieves the optimal rate of convergence up to some logarithmic factor.

***Remark 7*** The above result can also be shown for the more general case where a posteriori 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 2 from the supplement. Here, the idea of the proof would then be to represent the maximum $\max\{x_1, \dots, x_4\}$ on $\mathbb{R}^4$ as a standard feedforward neural network and apply a modification of Lemma 3 from the supplement to it. This would enable us to calculate the maximum of four positions of a channel. However, the proof would be much more technical.

***Remark 8*** In Section E of the supplement, we explain how the network architecture proposed by our theory can be applied in practice. Here, we rely on the practical applications from Sects. 4 and 5.

***Remark 9*** General convolutional neural network architectures such as Alex-net or VGG-net (cf., Krizhevsky et al., 2012; Simonyan and Zisserman, 2014) are

constructed for multi-label image classification and have additional modules, such as pooling layers, convolutional strides, response normalization layers and skip connections. Extending our theory to an even more realistic model, which would include further concepts of image recognition (e.g., translation invariance or rotation invariance) could yield results that could be even better transferred to such general architectures. Nevertheless, our analysis explains some properties of these convolutional neural networks and gives hints for good network architectures.

## 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 Sect. 1, the matrix $\mathbf{X}$ contains at position $(i, j)$ the gray 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 objects are defined theoretically on $[0, 32]^2$, then downsampled to the $\{1, \ldots, 32\} \times \{1, \ldots, 32\}$ grid using the Python package *Pillow*. The synthetic image datasets described below are inspired by Glorot and Bengio (2010).

The first classification task is to detect whether an image contains a circle (see Fig. 3). 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 chosen 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 gray scale values. For each object, rotation and area are chosen independently and are uniformly distributed on a fixed interval. We determine the gray scale values of the three objects by randomly permuting the list $(\frac{1}{3}, \frac{2}{3}, 1)$ of three gray 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

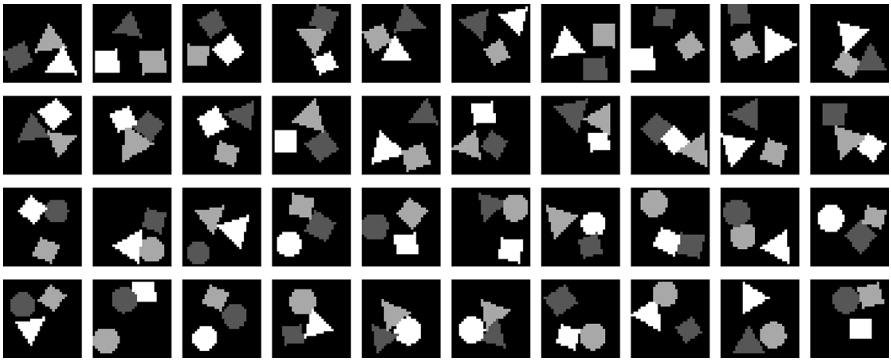

**Fig. 3** Some random images as realizations of the random variable $\mathbf{X}$ for the first classification task, where the first two rows show images of class 0 and the two lower rows show images of class 1

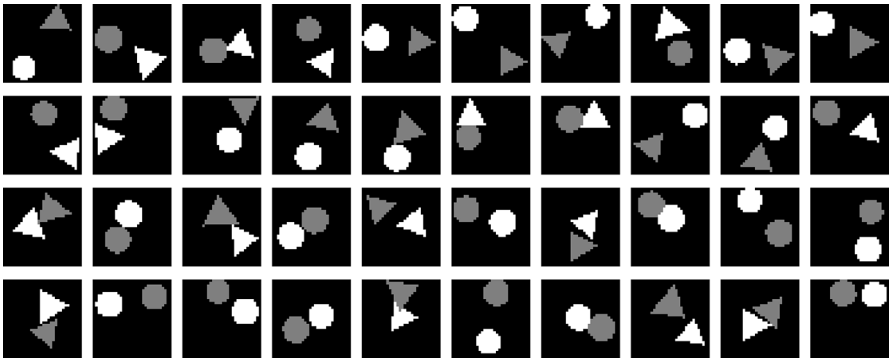

**Fig. 4** Some random images as realizations of the random variable $\mathbf{X}$ for the second classification task, where the first two rows show images of class 0 and the two lower rows show images of class 1

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 $(2 \cdot q)^3 = 0.5$.

In our second classification task, we determine whether an image consists of two equal geometric objects (see Fig. 4). The first difference 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 chosen with the probability $p = 0.5$ each and the list of gray 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 a posteriori 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 a posteriori 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 split our sample into a learning sample of size $n_l = \left\lfloor \frac{4}{5} \cdot n \right\rfloor$ and a testing sample of size $n_t = n - n_l$. We use the learning sample to train our classifiers several times with the different choices for the parameters and use the testing sample to select the classifiers that minimize the empirical misclassification risk. We then train the selected classifiers on the entire training set, consisting of the $n$ data points.

The detailed network architecture of our convolutional neural network image classifiers, i.e., the adaptive choice of the parameters $t$, $\mathbf{L} = (L^{(1)}, L^{(2)})$, $\mathbf{k}^{(1)}$, $\mathbf{k}^{(2)}$ and $\mathbf{M}$ of the network architecture we introduced in Sect. 2, can be found in Section E of the supplement. 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*). Here, we also use least squares estimation as in (5) and the plug-in method described in Sect. 1.3, with fully connected standard feedforward neural networks of the form (3). 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 using a Gaussian radial basis function kernel (abbr. *svm-rbf*) and polynomial kernel (abbr. *svm-p*). 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*) and a random forest classifier (abbr. *rand-f*), where we use the *RandomForestClassifier* function from the scikit-learn library. A description of the choice of the hyperparameters of all approaches is provided in Section E of the Supplement.

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

$$\epsilon_N = \frac{1}{N} \sum_{k=1}^{N} \mathbb{I}_{\{f_n(\mathbf{x}_{n+k}) \neq y_{n+k}\}} \tag{8}$$

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 (8) 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.

**Table 1** Median and interquartile range of the empirical misclassification risk $\epsilon_N$

| | 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) |

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 errors 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 correspond to the expected error of a classifier which always estimates the same class.

## 5 Application to real images

Although it is a well-known fact that convolutional neural networks are among the most successful methods in object classification using real images, in this section we will test the different image classification methods on real image datasets. On the one hand, we want to show the practical relevance of our particular network architecture, since it has some differences to convolutional neural networks used in practice, such as the asymmetric one-sided zero padding or the specific choice of filter sizes. On the other hand, by applying these methods to real images, we also want to show that the assumption of a generalized max-pooling model seems plausible, since the network architecture of our convolutional neural network image classifier was specifically designed for this model.

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 $32 \times 32$ pixels. Since the images are in color, we have converted them to gray scale (see Fig. 5).

The different approaches of classification, as well as the parameter sets we use, are described in the simulation study in Section 4 and in Section E of the supplement, respectively. We choose $n = 2000$, $n_l = 1600$ and $n_t = 400$ to train our

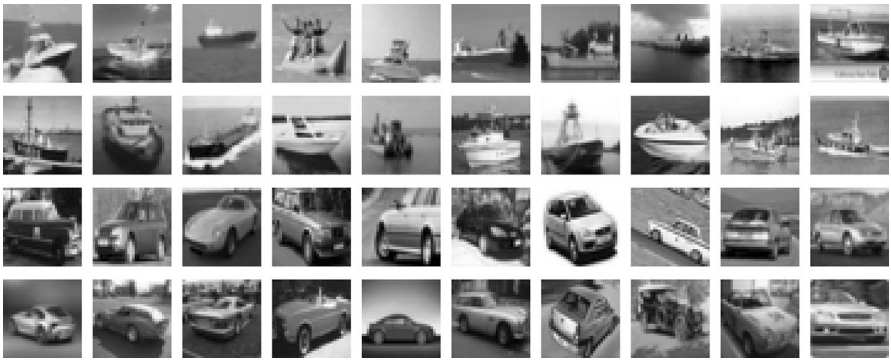

**Fig. 5** The first two rows show some images of the ships and the lower two rows show images of the cars of the gray scaled CIFAR-10 data set

classifiers. We calculate the empirical misclassification risk (8) using the remaining $N = 10{,}000$ images. The results of all approaches are summarized in Table 2.

Again, we observe that our estimate outperforms the others. This time, unlike in the case of 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 a posteriori 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.

## 6 Proof of theorem 1

Let $c_4 > 0$ be 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, inequality (2) implies that it suffices to show

**Table 2** The empirical misclassification risk $\epsilon_N$ for each estimate based on the presented gray scaled subset of the CIFAR-10 data set

| Neural-c | Neural-s | Neighbor | rand-f | svm-p | svm-rbf |
|---|---|---|---|---|---|
| 0.16 | 0.22 | 0.36 | 0.22 | 0.28 | 0.30 |

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

Using standard results from empirical process theory (cf., Lemma 9 in the supplement) we get

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

where $\mathcal{F}:=\mathcal{F}_t\left(\mathbf{L}, \mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \mathbf{M}\right)$. Our bound on the covering number (cf., Lemma 4 in the supplement) yields

$$\begin{aligned}
&\frac{c_{12}\cdot(\log n)^2\cdot\sup_{\mathbf{x}_1^n}\left(\log\left(\mathcal{N}_1\left(\frac{1}{n\cdot c_4\cdot\log n}, T_{c_4\cdot\log n}\mathcal{F}, \mathbf{x}_1^n\right)\right)+1\right)}{n}\\
&\le c_{13}\cdot\frac{\log(d_1\cdot d_2)\cdot(\log n)^3\cdot L_{max}^2\cdot\log L_{max}}{n}\\
&\le c_{14}\cdot\log(d_1\cdot d_2)\cdot(\log n)^4\cdot\max\left\{n^{-\frac{2\cdot p_1}{2\cdot p_1+4}}, n^{-\frac{2\cdot p_2}{2\cdot p_2+d^*}}\right\},
\end{aligned}$$

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 $\eta$, we have

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

such that $m_a$ satisfies a max-pooling model 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 $\bar{g}_{k,s}^{(a)}\in\mathcal{G}_4\left(L_n, \mathbf{k}^{(2)}\right)$ and $\bar{g}\in\mathcal{G}_{d^*}\left(L_n, \mathbf{k}^{(2)}\right)$ be the neural network from Kohler and Langer (2021) in Theorem 2 (cf., Lemma 10 in the supplement) which satisfies

$$\|g_{k,s}^{(a)} - \bar{g}_{k,s}^{(a)}\|_{[-2,2]^4,\infty} \leq c_{15} \cdot L_n^{-\frac{2 \cdot p_1}{4}} \leq c_{16} \cdot n^{-\frac{p_1}{2 \cdot p_1 + 4}}, \tag{9}$$

and

$$\|g - \bar{g}_{net}\|_{[-2,2]^{d^*},\infty} \leq c_{15} \cdot L_n^{-\frac{2 \cdot p_2}{d^*}} \leq c_{16} \cdot n^{-\frac{p_2}{2 \cdot p_2 + d^*}}. \tag{10}$$

Then Lemma 2 from the supplement let us choose

$$\bar{m}_1, \ldots, \bar{m}_{d^*} \in \mathcal{F}(L^{(1)}, \mathbf{k}^{(1)}, \mathbf{M}) \tag{11}$$

such that

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

where $\bar{f}^{(a)}$ satisfies

$$\bar{f}^{(a)} = \bar{f}_{l,1}^{(a)}$$

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

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

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

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

for $s = 1, \ldots, 4^{l-1}$. Due to property (11) it holds that

$$\bar{g} \circ (\bar{m}_1, \ldots, \bar{m}_{d^*}) \in \mathcal{F}. \tag{12}$$

Since the functions $g_{k,s}^{(a)}$ are $[0, 1]$–valued, inequalities (9) and (10) let us choose $c_1$ in the definition of $L_n$ sufficiently large such that the triangle inequality implies that

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

for all $a \in \{1, \ldots, d^*\}$, $k \in \{1, \ldots, l\}$ and $s \in \{1, \ldots, 4^{l-k}\}$. Then the application of Lemma 1 from the supplement yields

$$
\begin{aligned}
&\inf_{f\in\mathcal{F}} \int |f(\mathbf{x}) - \eta(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \\
&\overset{(12)}{\leq} \int |\bar{g}(\bar{m}_1(\mathbf{x}), \dots, \bar{m}_{d^*}(\mathbf{x})) - g(m_1(\mathbf{x}), \dots, m_{d^*}(\mathbf{x}))|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \\
&\leq c_{17} \cdot \left( \max_{a\in\{1,\dots,d^*\}, k\in\{1,\dots,l\}, s\in\{1,\dots,4^{l-j}\}} \left\{ \|g_{k,s}^{(a)} - \bar{g}_{k,s}^{(a)}\|_{[-2,2]^4,\infty}, \|g - \bar{g}\|_{[-2,2]^{d^*},\infty} \right\} \right)^2 \\
&\overset{(9),(10)}{\leq} c_{18} \cdot \max \left\{ n^{-\frac{2\cdot p_1}{2\cdot p_1+4}}, n^{-\frac{2\cdot p_2}{2\cdot p_2+d^*}} \right\}.
\end{aligned}
$$

Summarizing the above results, the proof is complete. □

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s10463-022-00828-4.

**Acknowledgements** The authors would like to thank Luc Devroye for a fruitful discussion on the topic of this paper. Furthermore, the authors would like to thank the AE and two anonymous referees for the helpful comments and suggestions to improve an early version of this manuscript.

## References

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

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

Chang, L.-B., Borenstein, E., Zhang, W., Geman, S. (2017). Maximum likelihood features for generative image models. *The Annals of Applied Statistics, 11*, 1275–1308.

Cover, T. M. (1968). Rates of convergence of nearest neighbor procedures. In *Proceedings of the Hawaii International Conference on Systems Siences*, 413–415. Honolulu, HI.

Devroye, L. (1982). Necessary and sufficient conditions for the pointwise convergence of nearest neighbor regression function estimates. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete, 61*, 467–481.

Devroye, L., Györfi, L., Lugosi, G. (1996). *A probabilistic theory of pattern recognition*. New York: Springer.

Du, S. S., Lee, J. D., Li, H., Wang, L., Zhai, X. (2018). Gradient descent finds global minima of deep neural networks. arXiv: 1811.03804.

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

Glorot, X., Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*: *Proceedings Track*, *9*, 249–256.

Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep learning*. London: MIT Press.

Györfi, L., Kohler, M., Krzyzak, A., Walk, H. (2002). *A distribution-free theory of nonparametric regression*. New York: Springer.

Hu, T., Shang, Z., Cheng, G. (2020). Sharp rate of convergence for deep neural network classifiers under the teacher-student setting. arXiv: 2001.06892.

Imaizumi, M., Fukamizu, K. (2019). Deep neural networks learn non-smooth functions effectively. In *Proceedings of the 22nd international conference on artificial intelligence and statistics*. Naha, Okinawa, Japan.

Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv: 1408.5882.

Kim, Y., Ohn, I., Kim, D. (2021). Fast convergence rates of deep neural networks for classification. *Neural Networks, 138*, 179–197.

Kohler, M., Krzyżak, A. (2017). Nonparametric regression based on hierarchical interaction models. *IEEE Transactions on Information Theory, 63*, 1620–1630.
Kohler, M., Krzyżak, A. (2021). Over-parametrized deep neural networks minimizing the empirical risk do not generalize well. *Bernoulli, 27*, 2564–2597.
Kohler, M., Langer, S. (2021). On the rate of convergence of fully connected very deep neural network regression estimates. *Annals of Statistics, 49*, 2231–2249.
Kohler, M., Krzyzak, A., Langer, S. (2019). Estimation of a function of low local dimensionality by deep neural networks. arXiv: 1908.11140.
Korostelev, A. P., Tsybakov, A. B. (1993). *Minimax theory of image reconstruction*. Number 82 in Lecture notes in statistics. New York: Springer.
Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images*. Technical report, Department of Computer Science, University of Toronto.
Krizhevsky, A., Sutskever, I., Hinton, G. E., et al. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira (Ed.), *Advances in neural information processing systems* (pp. 1097–1105). Red Hook, NY: Curran.
LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation, 1*, 541–551.
Lecun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*, 2278–2324.
LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. *Nature, 521*, 436–444.
Lin, S., Zhang, J. (2019). Generalization bounds for convolutional neural networks. arXiv: 1910.01487.
Liu, H., Chen, M., Zhao, T., Liao, W. (2021). Besov function approximation and binary classification on low-dimensional manifolds using convolutional residual networks. *Proceedings of the 38th international conference on machine learning* (*PMLR*), 139, 6770–6780.
Oono, K., Suzuki, T. (2019). Approximation and non-parametric estimation of resnet-type convolutional neural networks. In: *International conference on machine learning*, (pp. 4922–4931).
Petersen, P., Voigtlaender, F. (2020). Equivalence of approximation by convolutional neural networks and fully-connected networks. *Proceedings of the American Mathematical Society, 148*, 1567–1581.
Rawat, W., Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation, 29*, 2352–2449.
Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.
Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with relu activation function. *Annals of Statistics, 48*, 1875–1897.
Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv: 1409.1556.
Suzuki, T., Nitanda, A. (2019). Deep learning is adaptive to intrinsic dimensionality of model smoothness in anisotropic besov space. arXiv: 1910.12799.
Yarotsky, D. (2018). Universal approximations of invariant maps by neural networks. arXiv: 1804.10306.
Zhou, D.-X. (2020). Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis, 48*, 787–794.