

# **Towards a statistical foundation for machine learning methods**

Johannes Schmidt-Hieber

## **Abstract**

Recently, significant progress has been made regarding the theoretical understanding of deep artificial neural networks. One of the very promising directions is the statistical approach, which interprets deep learning as a statistical method and builds on existing techniques in mathematical statistics to derive theoretical error bounds and to understand phenomena such as overparametrization. The paper describes the contributions of the article [SH2020] "Nonparametric regression using deep neural networks with ReLU activation function" within a broader context.

## **Contents**

|          |                                                             |           |
|----------|-------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction to supervised learning</b>                  | <b>2</b>  |
| <b>2</b> | <b>Feedforward neural networks</b>                          | <b>3</b>  |
| <b>3</b> | <b>Deep learning and generalization</b>                     | <b>4</b>  |
| <b>4</b> | <b>Challenges for a statistical theory of deep learning</b> | <b>5</b>  |
| <b>5</b> | <b>Addressing non-linearity and highdimensionality</b>      | <b>8</b>  |
| <b>6</b> | <b>Avoiding the curse of dimensionality</b>                 | <b>9</b>  |
| <b>7</b> | <b>Statistical theory for algorithmic regularization</b>    | <b>11</b> |
| <b>8</b> | <b>Making other methods deep</b>                            | <b>12</b> |
| <b>9</b> | <b>Funding</b>                                              | <b>12</b> |

# 1 Introduction to supervised learning

Given data, a key problem in machine learning and statistics is to learn the functional relationship of how a number of variables affects another variable.

Supervised data are input-output pairs, such as images with their corresponding labels or measurements of medical data of individuals (input) together with a corresponding outcome, say the blood pressure, of these individuals.

The key idea in supervised learning is then to recover as good as possible the functional relationship between the inputs and the outputs in order to generalize to unseen data. For instance, given labeled images from cats and dogs, we aim to find a function that, given as input a new cat image, outputs 'cat', and given as input a new dog image, outputs 'dog'. Of course, learning a perfect separation from finitely many training data is hard, if not impossible. The quality of a learned function is therefore evaluated by the fraction of mistakes that it will make on unseen data (also called test data) that were not used for the learning.

Historically, statistics was first developed for parametric models. As function spaces do not allow for natural parametrizations with finitely many parameters, learning/estimating/recovering a function from data is referred to as a nonparametric estimation problem. Nonparametric statistics is the subfield that studies learning of functions.

Statistical theory requires to model the underlying data generating process. In the context of supervised learning, one typically considers the so called nonparametric regression model. [SH2020] considers the nonparametric regression model with random covariates in the unit hypercube. This means we observe  $n$  i.i.d. vectors  $\mathbf{X}_i \in [0, 1]^d$  with corresponding responses  $Y_i \in \mathbb{R}$  from the model

$$Y_i = f_0(\mathbf{X}_i) + \varepsilon_i, \quad i = 1, \dots, n, \quad (1.1)$$

where the unobserved noise variables  $\varepsilon_i$  are assumed to be i.i.d. standard normal and independent of  $(\mathbf{X}_i)_i$ . The statistical problem is to recover/learn the unknown function  $f_0 : [0, 1]^d \rightarrow \mathbb{R}$  from the sample  $(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n)$ .

Highlights of nonparametric statistics include a rigorous analysis of kernel smoothing techniques, spline fitting and wavelet methods [GKKW2002, W2006, T2009]. Enormous work has been invested in nonparametric statistics to answer questions related to the optimality of these methods. Here optimality is measured in a worst-case sense over a class of possible target functions. By using tools from information theory one often can determine the fastest possible rate at which the learned function can approximate (with respect to a given loss function) the true function in terms of the number of training samples  $n$ .

Next to machine learning and nonparametric statistics, a theoretical framework for learning of functions has been developed within the statistical learning theory. This approach relies on empirical process theory and can be applied when the learned functions are minimizers of a training loss and no explicit expression for the minimizer exists. This is the common scenario for machine learning methods.

Deep learning applied to a supervised training dataset is a specific procedure to fit a function - in this case a deep neural network - to a supervised dataset. While the method is rooted in the machine learning literature, its theoretical properties

can be studied by borrowing strength from the nonparametric statistics and the statistical learning literature.

To explain this, we will first briefly introduce feedforward neural networks.

## 2 Feedforward neural networks

In the computer science literature, neural networks are commonly introduced via the graph representation of a computational graph. To analyze the mathematical properties, it is more common to work with purely algebraic formulations. In this section, we closely follow the mathematical framework of representing neural networks as described in [SH2020].

To define neural networks, we need to fix in a first step an activation function  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$  and afterwards the so called network architecture. The default choice in deep learning is the ReLU activation function

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

although recently the SiLU  $\sigma(x) = x/(1 + e^{-x})$  became very popular in applications.

Regarding the network architecture, we need to introduce some suitable notation. For  $\mathbf{v} = (v_1, \dots, v_r) \in \mathbb{R}^r$ , define the shifted activation function  $\sigma_{\mathbf{v}} : \mathbb{R}^r \rightarrow \mathbb{R}^r$  as

$$\sigma_{\mathbf{v}} \begin{pmatrix} y_1 \\ \vdots \\ y_r \end{pmatrix} = \begin{pmatrix} \sigma(y_1 - v_1) \\ \vdots \\ \sigma(y_r - v_r) \end{pmatrix}.$$

[SH2020] introduces the network architecture  $(L, \mathbf{p})$  as a pair consisting of a positive integer  $L$  called the *number of hidden layers* or *depth* and a *width vector*  $\mathbf{p} = (p_0, \dots, p_{L+1}) \in \mathbb{N}^{L+2}$ . A neural network with network architecture  $(L, \mathbf{p})$  is then any function of the form

$$f : \mathbb{R}^{p_0} \rightarrow \mathbb{R}^{p_{L+1}}, \quad \mathbf{x} \mapsto f(\mathbf{x}) = W_L \sigma_{\mathbf{v}_L} W_{L-1} \sigma_{\mathbf{v}_{L-1}} \cdots W_1 \sigma_{\mathbf{v}_1} W_0 \mathbf{x}, \quad (2.1)$$

where  $W_i$  is a  $p_{i+1} \times p_i$  weight matrix and  $\mathbf{v}_i \in \mathbb{R}^{p_i}$  is a shift vector. The entries of weight matrices and the shift vectors are the free parameters that are learned from data. The number of free parameters can easily be in the millions or even much larger for modern network architectures. This is the network structure considered for regression tasks, where the response variables are real-valued. For classification tasks, one needs to map the outputs to a probability vector. This is typically done via the so called softmax function.

Without any hidden layers, the network is of the form  $f(\mathbf{x}) = W_0 \mathbf{x}$ . This is essentially the form of the standard linear model in statistic. This shows that neural networks can be viewed as nonlinear extensions of the linear model (or the logistic regression model for classification tasks). Deep network functions alternate between matrix-vector multiplications and the action of the non-linear activation function  $\sigma$ . If the inputs are  $d$ -dimensional vectors and the outputs are real valued such as in the  $d$ -variate nonparametric regression model (6.2), one has to choose  $p_0 = d$  and  $p_{L+1} = 1$  in the network architecture.

### 3 Deep learning and generalization

The question is now how to learn the network parameters from a supervised dataset  $(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n) \in \mathbb{R}^d \times \mathbb{R}$ . For convenience, we consider real-valued outputs. The framework can be adapted and extended to outputs that are either multivariate or encoded class labels.

Let  $\boldsymbol{\theta}$  be a vector containing all the network parameters (all entries of the weight matrices and all entries of the shift vectors). To indicate the dependence of the network function on the parameters, we denote the network function by  $f_{\boldsymbol{\theta}}$ , that is, (2.1) gets replaced by

$$\mathbf{x} \mapsto f_{\boldsymbol{\theta}}(\mathbf{x}) = W_L \sigma_{\mathbf{v}_L} W_{L-1} \sigma_{\mathbf{v}_{L-1}} \cdots W_1 \sigma_{\mathbf{v}_1} W_0 \mathbf{x}, \quad (3.1)$$

As mentioned above, we need to choose  $p_0 = d$  and  $p_{L+1} = 1$  in the architecture to ensure that the network functions map  $d$ -dimensional vectors into real numbers.

We now introduce a loss function  $\ell : \mathbb{R} \times \mathbb{R} \rightarrow [0, \infty)$  and consider the so called training loss

$$L(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n \ell(Y_i, f_{\boldsymbol{\theta}}(\mathbf{X}_i)), \quad (3.2)$$

measuring how well the responses  $f_{\boldsymbol{\theta}}(\mathbf{X}_i)$  that are predicted by the model if we choose the parameters to be  $\boldsymbol{\theta}$  match the true responses  $Y_1, \dots, Y_n$  in our training sample. As we consider the training sample to be fixed, we omit the dependence on the training sample in the notation  $L(\boldsymbol{\theta})$ .

A common loss function for regression tasks is the squared loss  $\ell(u, v) = (u - v)^2$ . In this case,  $L(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n (Y_i - f_{\boldsymbol{\theta}}(\mathbf{X}_i))^2$ .

Ideally, we want to find the parameter  $\boldsymbol{\theta}$  that minimizes  $L(\boldsymbol{\theta})$ . Finding the minimizer is, however, a non-convex problem. Deep learning therefore relies on (stochastic) gradient descent based methods that iteratively decrease the training loss.

Call the learned network parameters  $\hat{\boldsymbol{\theta}}$ . The performance of a trained neural network  $f_{\hat{\boldsymbol{\theta}}}$  is then evaluated on a validation data set  $(\mathbf{X}_{n+1}, Y_{n+1}), \dots, (\mathbf{X}_{n+m}, Y_{n+m})$  that was not used for training of the neural network. The empirical generalization error is then determined by the average loss of the trained neural network on the validation set, that is,

$$\frac{1}{m} \sum_{i=n+1}^{n+m} \ell(Y_i, f_{\hat{\boldsymbol{\theta}}}(\mathbf{X}_i)).$$

Under the standard assumption that the data  $(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_{n+m}, Y_{n+m})$  are independent and identically distributed, one can also define a population version of the generalization risk as follows: Assume that  $(\mathbf{X}, Y)$  is independent of the dataset but follows the same distribution. The risk is then defined as the expected value

$$E[\ell(Y, f_{\hat{\boldsymbol{\theta}}}(\mathbf{X}))],$$

where the expectation is taken over the randomness of the data and the randomness of  $(\mathbf{X}, Y)$ .

For the squared loss and working in the nonparametric regression model (6.2), we can rewrite the risk as

$$E[(Y - f_{\hat{\theta}}(\mathbf{X}))^2] = E[\varepsilon^2] + E[(f_0(\mathbf{X}) - f_{\hat{\theta}}(\mathbf{X}))^2] = 1 + E[(f_0(\mathbf{X}) - f_{\hat{\theta}}(\mathbf{X}))^2].$$

This means that minimizing the generalization risk on the left hand side can be linked to estimation of the regression function  $f_0$  by the neural network  $f_{\hat{\theta}}$  in the nonparametric regression model.

## 4 Challenges for a statistical theory of deep learning

In the previous section, we have seen that neural networks of a given architecture can be represented as function class parametrized in the network parameters. Thus, assuming supervised data generated from the nonparametric regression model (6.2), one can now fit a neural network function to the data and view the learned/estimated function as a nonparametric estimation method for the regression function  $f_0$ . It is then natural to ask why one cannot simply apply the machinery developed in nonparametric statistics to analyze the theoretical properties of this procedure.

However, there are several challenges that make this a non-standard problem.

- (i) **Non-linearity:** The map  $\theta \mapsto f_{\theta}$  is nonlinear. Whereas one can do the analysis explicit in linear models, dealing with non-linear methods is difficult. One way to address this is to study the statistical properties of the so called empirical risk minimizer of the training loss, that is,

$$\hat{\theta} \in \arg \min_{\theta} L(\theta),$$

where minimization is over the whole parameter space generated by the network parameters given a fixed network architecture. Even then, we still cannot obtain an explicit formula that links  $\hat{\theta}$  to the training set. Another way to state this is that even if we find a network configuration that gives us the global minimum, the method is still blackbox as we do not understand why the network parameters were selected in this way. Moreover, neural networks also have some internal symmetries that make the global minima non-unique.

For a mathematical analysis, the only information that we can rely on is that the selected neural network is a global minimum. To derive generalization bounds, this is, however, already sufficient to apply techniques from statistical learning theory.

- (ii) **Higdimensional parameter spaces:** Another challenge is that the number of network parameters in state-of-the-art architectures is of a larger order than the number of training samples. For instance, AlexNet has 60 million

network parameters and has been trained based on 1.2 million training samples. This means that from a statistical perspective, fitting a neural network to data has to be interpreted as high-dimensional estimation problem.

Given a training sample of size  $n$ , using  $d$  parameters, and taking the square loss, typically, the fastest possible convergence rate of the (generalization) risk is  $d/n$ . This means that in the high-dimensional regime  $d \gg n$ , the rate diverges which is a strong indication that the statistical method has poor generalization behavior.

In high-dimensional statistics, the standard way to overcome this is by assuming low-dimensional structure, such as sparsity of the parameter vector, or a low-rank constraint if parameter matrices are involved. Imposing sparsity, the number of parameters in the convergence rate typically becomes

$$\frac{\text{number of non-zero parameters}}{n} \times \text{some logarithmic factors.}$$

- (iii) **Avoiding the curse of dimensionality:** One of the mysteries underlying neural networks is that they perform very well for high-dimensional input spaces. An example are image classification tasks, where the number of inputs is the number of pixels in the image. This seems to contradict classical statistical wisdom that learning functions defined on high-dimensional input spaces necessarily leads to very slow convergence rates.

The rationale for the pessimistic theory is that in order to learn the regression function  $f_0$  from noisy measurements  $Y_i = f_0(\mathbf{X}_i) + \varepsilon_i$ ,  $i = 1, \dots, n$ , one has to somehow get rid of the unobserved noise variables  $\varepsilon_i$ . The only way to do this is by forming some average over responses  $Y_i$  that have similar inputs  $\mathbf{X}_i$ . There is a trade-off: if we average over a neighborhood that is too small, we only partially denoise the data, whereas if we take a neighborhood that is too large, we start to induce significant bias as the function  $f_0$  will also vary as we go to neighboring points. If the  $\mathbf{X}_i$  lie in a high-dimensional space, say in a hypercube  $[0, 1]^d$  with  $d$  large, then, even if the  $\mathbf{X}_i$  are uniformly distributed, the distance between the nearest neighbors scales with  $n^{-1/d}$  and thus points are further apart as  $d$  grows. This makes the necessary averaging over neighboring points extremely bad and leads to the slow convergence rates. Using the aforementioned concept of lower bounds in nonparametric statistics, it can be shown that the slow convergence rates are unavoidable and cannot be overcome by any method. This is also known as curse of dimensionality.

How to overcome then this mismatch between the practical success of deep learning and the pessimistic lower bounds in nonparametric statistics? As in the case of high-dimensional parameter spaces, there must be some additional structure that is present in the applications and can also be picked up by neural networks. To identify such structures is a key challenge in the

development of a deep learning theory.

In Section 6, we will sketch three possible mechanisms.

(iv) **Understand phenomena such as good generalization despite zero training loss, data augmentation, . . . :**

Empirical deep learning research has revealed several phenomena that contradict classical statistical wisdom. One of them is that if one takes an extremely overparametrized neural network and then runs stochastic gradient descent until the training loss eventually becomes zero, then, good generalization has been observed. Classical statistical theory would predict that we then overfit the training data and that this should lead to poor generalization performance.

Another phenomenon is data augmentation. Here one takes the training data, applies some transformations such as rotations and rescalings for image data, and then adds these transformed data to the original training sample pretending that those are fresh samples. This can improve generalization in machine learning although there is no additional information in the augmented data.

It is crucial to reconcile these empirical findings with the established theoretical framework.

(v) **algorithmic regularization:**

Neural networks are hard to train and require a substantial amount of hyperparameter tuning. There are various algorithmic regularization techniques that can be added to the deep learning algorithm such as dropout regularization or batch normalization. These methods have been introduced to overcome issues with exploding/vanishing gradients and the co-adaptation of the learned weights. Whether algorithmic regularization leads to a performance gain is case dependent. It is up to the practitioner to try these algorithmic regularization techniques and find the combination that works best. In some cases, algorithmic regularization can improve the generalization performance a little. While this seems not to be a big deal, increasing the misclassification error by 0.5 percent in a classification task can determine whether deep learning will eventually become state of the art or not.

Regularization is a central theme in statistical methodology and statistical theory. The difference is, however, that the regularization is most commonly chosen as a penalization term that is added to the loss function such as the  $\ell^1$ -penalty in the famous LASSO estimator. On the contrary, algorithmic regularization changes the update steps of the gradient descent method. To track how these changes propagate through the gradient descent iterates and how they affect the final outcome is non-standard and a largely open field.

## 5 Addressing non-linearity and highdimensionality

It is natural to focus on specific challenges. [SH2020] mainly deals with the non-linearity of the method, the highdimensional parameters, and the curse of dimensionality. To overcome the slow convergence that is intrinsic to highdimensional models, it is assumed that the networks are sparsely connected. In the algebraic formulation (2.1), this means that the weight matrices and the shift vectors are sparse. More precisely,  $s$ -sparse means that at most  $s$  parameters are non-zero. No constraints are imposed where these non-zero parameters are located.

Below, we will first state the generalization bound assuming sparsity and then discuss whether network sparsity is necessary.

### 5.1 Generalization bound

Given a network architecture  $(L, \mathbf{p})$  and a sparsity  $s$ , denote by  $\mathcal{F}(L, \mathbf{p}, s)$  the class of all neural network functions with this architecture and sparsity and all network weight bounded in absolute value by 1.

It has been mentioned that one way to deal with the non-linear parameter dependence is to consider the global minimum (also called the empirical risk minimizer) and to apply statistical learning techniques. One can generalize this.

Instead of only considering the empirical risk minimizer or a specific way to train the neural network, we consider an abstract learning procedure  $\hat{f}_n$  that, for a given sample consisting of  $n$  training data, spits out a neural network in the class  $\mathcal{F}(L, \mathbf{p}, s)$ .

We then associate to each procedure  $\hat{f}_n$  a so called optimization error. Informally speaking, for a given learning procedure  $\hat{f}_n$ , this error measures the gap between the expected training loss of  $\hat{f}_n$  and the expected training loss of the empirical risk minimizer taken over the network class  $\mathcal{F}(L, \mathbf{p}, s)$ . This difference is always non-negative. It is zero if  $\hat{f}_n$  is the empirical risk minimizer. The optimization error will be denoted by

$$\Delta_n(\hat{f}_n).$$

**Simplified informal version of the main theorem in [SH2020]:** Consider training data  $(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n)$  generated from the nonparametric regression model with target function belonging to some ball of  $\beta$ -smooth Hölder functions with radius  $R$  denoted by  $\mathcal{H}(\beta, R)$ . For sparsely connected neural networks (the sparsity  $s$  has to be chosen of the right order with respect to the sample size  $n$ ) of depth  $L \geq c \log n$ , where  $c = c(\beta, d)$  is a constant, an arbitrary procedure  $\hat{f}_n : \{(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n)\} \rightarrow \mathcal{F}(L, \mathbf{p}, s)$ , will have the generalization bound

$$\sup_{f_0 \in \mathcal{H}(\beta, R)} E[(f_0(\mathbf{X}) - \hat{f}_n(\mathbf{X}))^2] \asymp n^{-\frac{2\beta}{2\beta+d}} + \Delta_n(\hat{f}_n),$$

up to some  $\log n$ -factors that we omit here.

Differently speaking, here  $\asymp$  means that the two sequences can be bounded by each other up to a factor of the form  $\text{constant} \times \log^\kappa(n)$  for some  $\kappa$ . If the distribution of  $\mathbf{X}$  is sufficiently regular, the rate  $n^{-\frac{2\beta}{2\beta+d}}$  is known to be the optimal convergence rate for this class of target functions over all possible estimators. This already implies that the rate is a lower bound.

The result can be interpreted as follows: If the learning procedure sufficiently reduces the training loss such that the optimization error satisfies  $\Delta_n(\hat{f}_n) \lesssim n^{-\frac{2\beta}{2\beta+d}}$ , then, the deep neural network fit will achieve the fastest possible convergence rate. Note that the objective of deep learning is reduction of the training loss.

Because of the lower bound, the result also shows the opposite: If we do not bring the optimization error down, then, neural networks can only achieve suboptimal convergence rate.

While biological neural networks (aka the brain) are sparsely connected, artificial neural networks are typically trained on fully-connected neural networks. The current hardware does not allow for significant amounts of sparsity. However, sparsity can be added in a post-processing step borrowing ideas from the compression literature. While small compression does seem to slightly increase the performance, very high-compression rates have been shown to yield similar performance as uncompressed neural networks [HABNDP].

A key question is therefore whether the sparsity constraint in the theory can be avoided. [SH2020b] discusses this question in detail. It is shown that if the outer layer in an extremely overparametrized shallow network is trained, then, SGD will converge to a function fit that is inconsistent, that is, it does not converge to the true regression function even if  $n \rightarrow \infty$ .

Later work [KL2021, JSLH2023] has also shown that if one gives up on the high-dimensionality of the networks, one can achieve optimal convergence rates for fully-connected neural networks. The known parameter assignments in a neural network that hypothetically can do this task seem to be more difficult than in the sparsely connected case.

## 6 Avoiding the curse of dimensionality

The statistical rate  $n^{-\beta/(2\beta+d)}$  that occurs in the previous theorem becomes very slow as  $d$  grows and therefore suffers from the curse of dimensionality.

In the subsequent subsections, we will describe three ways, how neural networks can avoid the curse of dimensionality.

### 6.1 Compositional sparsity

Deep neural networks are build by stacking layers on top of each other. This suggests that they might be particularly suitable to learn compositional structure in the target function  $f_0$ .

The target function models how the responses/labels are computed from the inputs. Assumptions imposed on the target function are therefore assumptions on

the underlying data-generating mechanism.

Compositional sparsity assumes that the target function can be written as composition of vector-valued functions with a sparsity assumption that is imposed on the component functions. Citing the definition and discussion in [SH2020]: *It is assumed that*

$$f_0 = g_q \circ g_{q-1} \circ \dots \circ g_1 \circ g_0 \quad (6.1)$$

with  $g_i : [a_i, b_i]^{d_i} \rightarrow [a_{i+1}, b_{i+1}]^{d_{i+1}}$ . Denote by  $g_i = (g_{ij})_{j=1, \dots, d_{i+1}}^\top$  the components of  $g_i$ , assume that each component function  $g_{ij}$  lies in a Hölder ball of  $\beta_i$ -smooth functions, and let  $t_i$  be the maximal number of variables on which each of the  $g_{ij}$  depends on. Thus, each  $g_{ij}$  is a  $t_i$ -variate function. We always must have  $t_i \leq d_i$ .

*Example:* For  $f_0(x_1, x_2, x_3) = g_{11}(g_{01}(x_1, x_3), g_{02}(x_1, x_2))$  we have  $d_0 = 3$ ,  $t_0 = 2$ ,  $d_1 = t_1 = 2$ ,  $d_2 = 1$ .

For specific constraints such as additive models,  $t_i$  might be much smaller than  $d_i$ . The single components  $g_0, \dots, g_q$  and the pairs  $(\beta_i, t_i)$  are obviously not identifiable. As we are only interested in estimation of  $f_0$  this causes no problems.

Theorem 1 in [SH2020] shows that the slow convergence rate  $n^{-\beta/(2\beta+d)}$  can in this case be replaced by a potentially much faster rate only depending on  $(t_0, \beta_0), \dots, (t_q, \beta_q)$ . If  $\max_i t_i/\beta_i$  remains bounded as  $d \rightarrow \infty$ , then, neural networks can avoid the curse of dimensionality.

For classification, similar results were derived in [BSH2022].

Specific structural models such as generalized additive models, which were previously known to circumvent the curse of dimensionality, can be shown to be specific compositionally sparse functions and the framework allows to show that the optimal convergence rates can be achieved by deep neural networks, see Section 4 in [SH2020].

## 6.2 Input on a manifold

Images of a specific class such as cat images can be thought of as lying on a low-dimensional manifold embedded into the high-dimensional ambient space generated by all possible images [PZAGG2021].

In the nonparametric regression model (2.1) with inputs on a manifold  $\mathcal{M}$ , we observe  $n$  i.i.d. vectors  $\mathbf{X}_i \in \mathcal{M} \subset [0, 1]^d$  with corresponding responses  $Y_i \in \mathbb{R}$  from the model

$$Y_i = f_0(\mathbf{X}_i) + \varepsilon_i, \quad i = 1, \dots, n, \quad (6.2)$$

where the unobserved noise variables  $\varepsilon_i$  are assumed to be i.i.d. standard normal and independent of  $(\mathbf{X}_i)_i$ .

The manifold  $\mathcal{M}$  is unobserved. Still, a procedure should "see" the manifold structure in the input and perform as if the inputs were drawn from a  $D$  dimensional space, where  $D$  is the manifold dimension.

It has been shown that neural networks can adapt to the manifold structure, in the sense that if  $f_0$  lies in a ball of  $\beta$ -Hölder smooth functions, then there exist

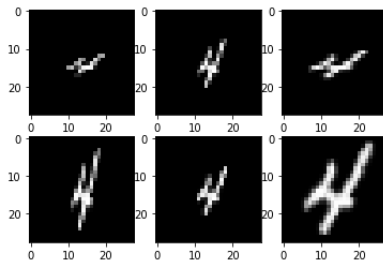


Figure 1: Data for one class generated by the proposed image deformation model in [CLSH2025].

sparingly connected neural network architectures based on which one can learn the function  $f_0$  with the convergence rate

$$n^{-\frac{\beta}{2\beta+D}}$$

up to logarithmic factors in the sample size  $n$ .

### 6.3 Image deformation models

Proposing more realistic data-generating models for state-of-the-art machine learning applications provides another way to show how neural networks can circumvent the curse of dimensionality.

In [CLSH2025], a new model for image classification has been proposed. In this model, a  $d \times d$  image is modeled as a  $d \times d$  matrix with the corresponding pixel values as entries. The authors then propose a general model for object deformation on images. Based on this, one can then formulate a binary image classification model. While this model simplifies image classification, it does capture key characteristics of this task. Figure 1 shows data generated from this model illustrating the variety in image deformation that the model is able to generate.

The nonparametric regression model adds noise to each entry, making the conditional distribution  $Y_i|X_i$  random. On the contrary, assuming correctly labeled images, all the noise in the image classification is in the inputs and the  $i$ -th label is then a deterministic function of  $X_i$ .

In this model, [CLSH2025] show that the obtained convergence rates for the misclassification error of convolutional neural networks only dependent polynomially on the input dimension  $d$ .

The whole framework seems to align much better with deep learning practice. In particular, one can naturally rely on fully-connected neural networks.

## 7 Statistical theory for algorithmic regularization

We give a brief account regarding the statistical understanding of algorithmic regularization.

The implicit regularization of (S)GD has been studied in various settings including the seminal work on benign overfitting [BLLT2020].

Dropout regularization does some random masking of units in the iterations [SHKSS2014]. From a statistical perspective, dropout has been analyzed in the linear model by [CLSH2024, LSHW2025].

Batch normalization is an important regularization technique that keeps the gradient stable during training. Despite its relevance, it seems not to have been analyzed as a statistical procedure.

[ISH2023] study the regularizing effect that is induced by the loss landscape on the gradient descent iterates.

Statistical properties of sharpness-aware minimization (SAM) were derived in [BLB2023, CLSH2025].

## 8 Making other methods deep

The mathematical analysis shows that depth can have favorable properties beyond neural networks.

Deep Gaussian priors [N1996, DL2013] are an emerging theme in Bayesian analysis. The theoretical insights above have contributed to a class of specific deep Gaussian process priors [FSH2023, GRSH2022].

The theory has also led to the development of Kolmogorov-Arnold networks that build a deep spline composition model based on the Kolmogorov-Arnold representation theorem [K1957, SH2021].

Kernel regression is another natural candidate for depth [GWHSX2016].

## 9 Funding

The research has been supported by the project "Statistical foundation for multilayer neural networks" (project number VI.Vidi.192.021 of the Vidi ENW programme) financed by the Dutch Research Council (NWO).

## References

- [BLB2023] P. L. Bartlett, P. M. Long, O. Bousquet. *The dynamics of sharpness-aware minimization: Bouncing across ravines and drifting towards wide minima*, Journal of Machine Learning Research **24** (2023), 1–36.
- [BLLT2020] P.L. Bartlett, P.M. Long, G. Lugosi, A. Tsigler, *Benign overfitting in linear regression*, Proc. Natl. Acad. Sci. U.S.A., **117** (2020), 30063–30070.
- [BSH2022] T. Bos and J. Schmidt-Hieber, *Convergence rates of deep ReLU networks for multiclass classification*, Electronic Journal of Statistics, **16** (2022), 2724–2773.
- [CLSH2025] J. Chen, S. Langer, J. Schmidt-Hieber *A novel statistical approach to analyze image classification* ArXiv preprint 2206.02151.

- [CLSH2025] G. Clara, S. Langer, and J. Schmidt-Hieber *Training diagonal linear networks with stochastic sharpness-aware minimization* ArXiv preprint 2503.11891.
- [CLSH2024] G. Clara, S. Langer, and J. Schmidt-Hieber *Dropout regularization versus  $\ell^2$ -penalization in the linear model*, Journal of Machine Learning Research (JMLR), **25** (2024), 1–48.
- [DL2013] A. Damianou and N. Lawrence. *Deep Gaussian processes*, In Artificial Intelligence and Statistics (2013), 207–215.
- [FSH2023] G. Finocchio, J. Schmidt-Hieber *Posterior contraction for deep Gaussian process priors*, Journal of Machine Learning Research, **24** (2023), 1–49.
- [GRSH2022] M. Giordano, K. Ray and J. Schmidt-Hieber, *On the inability of Gaussian process regression to optimally learn compositional functions*, 36th Conference on Neural Information Processing Systems (NeurIPS 2022), **35** (2022), 22341–22353.
- [GWHXS2016] A. Gordon Wilson, Z. Hu, R. Salakhutdinov, E. P. Xing, *Deep Kernel Learning*, Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, PMLR **51** (2016), 370–378.
- [GKKW2002] L. Györfi, M. Kohler, A. Krzyżak, and H. Walk, *A distribution-free theory of nonparametric regression*, Springer Series in Statistics. Springer-Verlag, New York (2002).
- [HABNDP] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, A. Peste, *Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks*, Journal of Machine Learning Research, **22** (2021), 1–124.
- [ISH2023] M. Imaizumi, J. Schmidt-Hieber *On generalization bounds for deep networks based on loss surface implicit regularization*, IEEE Transactions on Information Theory, **69** (2023), 1203–1223.
- [JSLH2023] Y. Jiao, G. Shen, Y. Lin, J. Huang, *Deep nonparametric regression on approximate manifolds: Nonasymptotic error bounds with polynomial prefactors*, The Annals of Statistics **51** (2023), 691–716.
- [KL2021] M. Kohler, S. Langer, *On the rate of convergence of fully connected deep neural network regression estimates*, The Annals of Statistics **49** (2021), 2231–2249.
- [K1957] A.N. Kolmogorov, *On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition* Doklady Akademii Nauk SSSR **114** (1957), 953–956.
- [LSHW2025] J. Li, J. Schmidt-Hieber, W. B. Wu, *Asymptotics of stochastic gradient descent with dropout regularization in linear models*, ArXiv preprint 2409.07434.
- [PZAGG2021] P. Pope, C. Zhu, A. Abdelkader, M. Goldblum, T. Goldstein, *The intrinsic dimension of images and its impact on learning*, International Conference on Learning Representations (2021).
- [N1996] R. M. Neal, *Bayesian Learning for Neural Networks*, volume 118 of Lecture Notes in Statistics. Springer, New York (1996).
- [SH2021] J. Schmidt-Hieber, *The Kolmogorov-Arnold representation theorem re-*

- visited*, Neural Networks, **137** (2021), 119–126.
- [SH2020] J. Schmidt-Hieber, *Nonparametric regression using deep neural networks with ReLU activation function*, Annals of Statistics **48** (2020), 1875–1897 (with discussion).
- [SH2020b] J. Schmidt-Hieber, *Rejoinder: "Nonparametric regression using deep neural networks with ReLU activation function"* Annals of Statistics, **48** (2020), 1916–1921.
- [SHKSS2014] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*. J. Mach. Learn. Res. **15** (2014), 1929–1958.
- [T2009] A. B. Tsybakov, *Introduction to nonparametric estimation*. Springer Series in Statistics. Springer, New York (2009).
- [W2006] L. Wasserman, *All of nonparametric statistics*. Springer Texts in Statistics. Springer, New York (2006).

DEPARTMENT OF APPLIED MATHEMATICS, UNIVERSITY OF TWENTE  
DRIENERLOLAAN 5,  
7522 NB ENSCHEDE, THE NETHERLANDS.  
*E-mail address:* `a.j.schmidt-hieber@utwente.nl`