Noname manuscript No.
(will be inserted by the editor)

Estimation of the Mean Function of Functional Data via Deep Neural Networks

Shuoyang Wang · Guanqun Cao · Zuofeng Shang

Received: date / Accepted: date

Abstract In this work, we propose a deep neural networks based method to perform nonparametric regression for functional data. The proposed estimators are based on sparsely connected deep neural networks with ReLU activation function. We provide the convergence rate of the proposed deep neural networks estimator in terms of the empirical norm. We discuss how to properly select of the architecture parameters by cross-validation. Through Monte Carlo simulation studies we examine the finite-sample performance of the proposed method. Finally, the proposed method is applied to analyze positron emission tomography images of patients with Alzheimer disease obtained from the Alzheimer Disease Neuroimaging Initiative database.

Keywords Functional data analysis · Neural networks · Nonparametric regression · Rate of convergence · ReLU activation function · ADNI database.

1 Introduction

Functional data refer to curves or functions, i.e. the data for each variable are viewed as smooth curves, surfaces, or hypersurfaces evaluated at a finite subset of some interval in 1D and 2D (e.g., some period of time, some range of pixels or voxels and

Wang's and Cao's research was partially supported by NSF award DMS 1736470. Shang's research was supported in part by NSF awards DMS 1764280 and 1821157.

Shuoyang Wang
Department Mathematics and Statistics, Auburn University, U.S.A.

Guanqun Cao
Department Mathematics and Statistics, Auburn University, U.S.A. Auburn University
E-mail: gzc0009@auburn.edu
Tel.: +1334-844-6563
Fax: +1334-844-6555

Zuofeng Shang
Department of Mathematical Sciences, New Jersey Institute of Technology, U.S.A.

so on). Functional data means intrinsically infinite-dimensional but are usually measured discretely. The high intrinsic dimensionality of these data poses challenges both for theory and computation. Functional data analysis (FDA) has been a topic of increasing interest in the statistics community for recent decades. [19] and [28] gave a comprehensive overview of FDA.

In FDA problems, estimation of mean functions is the fundamental first step; see [7, 20, 10] for example. Various methods exist that allow to estimate the regression function nonparametrically. [20] adopted the mixed effect models where the mean function and the eigenfunctions were represented with B-splines and the spline coefficients were estimated by the EM algorithm; [30] applied the local linear smoothers to estimate the mean and the covariance functions. [18] generalized the linear mixed model to the functional mixed model framework, with model fitting done by using a Bayesian wavelet-based approach. In [6], a polynomial spline estimator is proposed for the mean function of functional data together with a simultaneous confidence band. These nonparametric methods apply the pre-specified basis expansion, e.g., polynomial spline, local linear smoother, wavelet and so on, to fit the unknown mean function. The convergence rates achieve either optimal nonparametric rate or parametric rate dependents on how dense of the observed points for each subject.

Even though FDA has received considerable attention over the last decade, most approaches still focus on 1D functional data. The high intrinsic dimensionality of these data poses challenges both for theory and computation; these challenges vary with how the functional data were sampled. Hence, few are developed for general multi-dimensional functional data. Recently, several attempts have been made to extend these nonparametric methods for spatial and image data. [29] used bivariate splines over triangulations to handle an irregular domain of the images that is common in brain imaging studies. The proposed spline estimators of the mean functions are shown to be consistent and asymptotically normal. However, the triangularized bivariate splines are designed for 2D functions only. Extending spline basis functions for general d-dimensional data observed on an irregular domain is very sophisticated and becomes extremely complex as d increases. [27] proposed a regularized Haar wavelet-based approach for the analysis of 3D brain image data in the framework of functional linear regression model.

Another popular method is functional principal component analysis (FPCA) which is an extension of multivariate principal component analysis, see [11, 31] for example. Recently, there are a few studies on 2D FDA. [33] proposed a smooth FPCA for 2D functions on irregular planar domains; their approach is based on a mixed effects model that specifies the principal component functions as bivariate splines on triangulations and the principal component scores as random effects. [15] proposed a FPCA model that can handle real functions observable on a 2D manifold. [8] extended it to analyze functional/longitudinal data observed on a general d-dimensional domain. When applying FPCA, how to choose the number of eigenfunctions is an important practical issue without a satisfactory theoretical solution. Presumably, the larger the number of eigenfunctions, the more flexible the approximation would be, and hence, the closer to the true curve. However, a large number of eigenfunctions always result in a complex model which introduces difficulties to follow-up analysis.

For many years, the use of neural networks has been one of the most promising approaches in connection with applications related to approximation and estimation of multivariate functions (see, e.g., [2, 21]). Recently, the focus is on multilayer neural networks, which use many hidden layers, and the corresponding techniques are called *deep learning*. Under the nonparametric regression model, via sparsely connected deep neural networks, [22] and [17] showed that the L_2 risk of the least squares neural network regression estimator achieves the same minimax rate of convergence (up to a logarithmic factor) as proposed in [24]. Furthermore, this neural network estimator does not suffer the curse-of-dimensionality which is a classical drawback in the traditional nonparametric regression framework. [3] has also obtained the similar results under deep learning framework via a different activation function. [16] further removed the logarithmic factors to achieve exact optimal nonparametric rate.

Although considerable advances have been achieved in deep learning research, from the statistical perspective its application and theoretical research is still in its infancy stage [9]. There are many technical challenges left for statisticians. For example, the availability of scalable computing and stochastic optimization techniques are challenge for developing statistical asymptotic properties. Recently, there are some works proposed for deep learning algorithms from statistical point of view [25, 26]. Motivated by these desiderata, the main goal of this article is to provide a novel method of FDA in the neural network framework.

The contributions of the present paper are three-fold. First, to our best knowledge, this is the first work on proposing deep neural networks (DNN) based estimator for FDA. An R package "`FDADNN`" has been developed and is available from GitHub website. Second, we develop the convergence rate (in empirical norm) of the proposed neural networks estimator. It is well-known that when the observed points come from a hypercube, i.e., $[0,1]^d$, $d = 3$ for 3D imaging study, the nonparametric convergence rates are slower than the optimal nonparametric rate. This means that no statistical procedure can perfectly recover the signal pointwisely. However, by borrowing the advantage from the *deep learning* domain, the convergence rate of the proposed DNN estimator does not depend on the dimension d. Finally, we do not assume additional or complex structure for the true mean function, for example, additive models or single-index models. As in the deep learning domain, the true regression functions are assumed to be constructed in a modular form and the modularity of the system can be fairly complex, which resolve the misspecification issue.

Different from the existing neural network literature on nonparametric regression [3], [22] and [17] which only handle i.i.d. data, we focus on FDA, where each subject is a random curve in a hypercube. Because of this special data structure, the major challenge becomes to deal with the correlation among the N evaluation points in the framework of neural network, which has not been achieved in the existing works. It is not surprising that the convergence rate decreases with n (number of subjects) as well as N.

The paper is structured as follows. Section 2 provides the model setting in FDA and introduces multilayer feed-forward artificial neural networks and discusses mathematical modeling. The implementation on hyperparameter selections also be included in Section 2. The theoretical properties of the proposed DNN estimator can

be found in Section 4. In Section 5, it is shown that the finite sample performance of proposed neural network estimator. The proposed method is applied to the spatially normalized positron emission tomography (PET) data from Alzheimer Disease Neuroimaging Initiative (ADNI) in Section 6 and make some concluding remarks in Section 7. The proof of the main result together with additional discussion can be found in the supplementary file.

2 Models and DNN estimator

2.1 FDA model

Denote by Y_{ij} the j-th observation of the random curve $\xi_i(\cdot)$ at grid points $\mathbf{X}_{ij}$, $1 \leq i \leq n, 1 \leq j \leq N_i$. For simple notations, we examine the equally spaced design, in other words, $\mathbf{X}_{ij} = \mathbf{X}_j = j/N$. The main results can be extended to irregularly spaced design. Without loss of generality, let $\mathbf{X}_j = (X_{j1}, \ldots, X_{jd}) \in [0,1]^d$. For the i-th subject, its sample path $\{\mathbf{X}_j, Y_{ij}\}$ consists of the noisy realization of the Gaussian process $\xi_i(\mathbf{X})$ in the sense that $Y_{ij} = \xi_i(\mathbf{X}_j) + \epsilon_i(\mathbf{X}_j)$, and $\left\{\xi_i(\mathbf{X}), \mathbf{X} \in [0,1]^d\right\}$ are i.i.d. copies of the process $\left\{\xi(\mathbf{X}), \mathbf{X} \in [0,1]^d\right\}$ which is L^2, i.e., $E\int_{[0,1]^d} \xi^2(\mathbf{X}) d\mathbf{X} < +\infty$. The error term $\epsilon_i(\mathbf{X}_j)$ has mean zero and finite variance.

In this work, we consider the following classical FDA model:

$$\begin{aligned} Y_{ij} &= \xi_i(\mathbf{X}_j) + \epsilon_i(\mathbf{X}_j) \\ &= f_0(\mathbf{X}_j) + \eta_i(\mathbf{X}_j) + \epsilon_i(\mathbf{X}_j), \quad i = 1, 2, \ldots, n, j = 1, 2, \ldots, N, \end{aligned} \tag{1}$$

where $f_0 : \mathbb{R}^d \to \mathbb{R}$, $E(Y_{ij}) = f_0(\mathbf{X}_j)$, $\eta_i(\cdot)$ is a Gaussian process characterizing individual curve variations from $f_0(\cdot)$ with mean zero and $\mathrm{Cov}(\eta(\mathbf{X}_j), \eta(\mathbf{X}_{j'})) := G(\mathbf{X}_j, \mathbf{X}_{j'})$. Let $\epsilon_i(\mathbf{X}_j) = \tau(\mathbf{X}_j)\varepsilon_{ij}$, where ε_{ij}'s are independent normal random variables and $\tau(\mathbf{X})$ is the standard deviation function bounded above zero for any $\mathbf{X} \in [0,1]^d$. By Mercer's Theorem, covariance function $G(\mathbf{X}, \mathbf{X}')$ has the following spectrum decomposition

$$G(\mathbf{X}, \mathbf{X}') = \sum_{k=1}^{\infty} \lambda_k \psi_k(\mathbf{X}) \psi_k(\mathbf{X}'),$$

where $\{\lambda_k\}_{k=1}^{\infty}$ and $\{\psi_k(\mathbf{X})\}_{k=1}^{\infty}$ are the eigenvalues and eigenfunctions of $G(\mathbf{X}, \mathbf{X}')$, and $\{\psi_k(\mathbf{X})\}_{k=1}^{\infty}$ are orthonormal bases in $L_2([0,1]^d)$.

2.2 Deep Neural Networks

Before conducting the estimation of the mean function f_0 in (1) via the DNN, let us briefly introduce the necessary notations and terminologies used in the neural networks. From the high level, typical DNN use a composition of a series of simple nonlinear functions to model nonlinearity, i.e.,

$$\mathbf{h}_L = \mathbf{g}_L \circ \mathbf{g}_{L-1} \circ \ldots \circ \mathbf{g}_1(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d, \tag{2}$$

where $\circ$ denotes composition of two functions and L is called the number of hidden layers or depth of a DNN model. One can define $\mathbf{h}_l = \mathbf{g}_l(\mathbf{h}_{l-1})$ for each $1 \leq l \leq L$ recursively and $\mathbf{h}_0 = \mathbf{x}$. "Deep" in deep neural networks refers to the use of multiple layers in the network. In the feed-forward neural network, the information moves in only one direction-forward-from the input layers, through the hidden layers and to the output nodes layers. In this kind of neural nets, there is a specific choice of $\mathbf{g}_l$: $\mathbf{g}_l(\mathbf{h}_{l-1}) = \boldsymbol{\sigma}(\mathbf{W}_l\mathbf{h}_{l-1}), l = 1, \ldots, L$, where $\mathbf{W}_l$ is a $p_l \times p_{l+1}$ weight matrix in the l-th layer and $\mathbf{p} = (p_0, \ldots, p_{L+1})$ is the width vector. The nonlinear function $\boldsymbol{\sigma}$ is called the activation function. Here, we study the popular rectifier linear unit (ReLU) activation function applied element-wise $[\boldsymbol{\sigma}(\mathbf{x})]_j = \max(x_j, 0)$, $j = 1, \ldots, d$. For any vector $\mathbf{v} = (v_1, \ldots, v_d) \in \mathbb{R}^d$, define the shifted activation function $\sigma_{\mathbf{v}} : \mathbb{R}^d \to \mathbb{R}^d$ as: $[\sigma_{\mathbf{v}}(\mathbf{x})]_j = \sigma(x_j - v_j)$, $j = 1, \ldots, d$. We call $\mathbf{v}$ the activation vector. The DNN model is then any function of the form is defined as $f : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}$,

$$f(\mathbf{x}) = \mathbf{W}_L\sigma_{\boldsymbol{V}_L}W_{L-1}\sigma_{\boldsymbol{V}_{L-1}}\ldots\mathbf{W}_1\sigma_{\boldsymbol{V}_1}\mathbf{W}_0\mathbf{x}. \tag{3}$$

To fit networks with data generated from the d-dimensional hypercube functional data model, we must have $p_0 = d$ and $p_{L+1} = 1$. Without loss of generality, we assume for any $f \in \mathcal{F}$, its empirical norm is bounded, i.e., $\|f\|_N = \left(\frac{1}{N}\sum_{j=1}^N f^2(\mathbf{x}_j)\right)^{1/2} \leq C_f < \infty$.

Although the depth and width of the neural nets can be extremely deep and wide, overfitting and computational burden are serious problems in such networks. To overcome these issues, the networks are modeled by assuming that each unit will be active only for a small fraction of the data to avoid overfitting. Smaller weights in a neural network can result in a model that is more stable and less likely to overfit the training dataset, in turn having better performance when making a prediction on new data [23]. Therefore, we assume that there are only few non-zero network parameters. Equivalently, we define the sparse neural networks and add constrains on the maximum-entry norm and non-zero entries of weight matrix $\mathbf{W}_l$ and activation vector $\mathbf{v}_l$. The sparse neural networks for our functional data model are given by

$$\begin{aligned}&\mathcal{F}(L, \mathbf{p}, s) \\ &= \left\{ f(\cdot) \text{ of the form (3)} : \max_{l=0,\ldots,L} \|\mathbf{W}_l\|_\infty + \|\mathbf{v}_l\|_\infty \leq 1, \sum_{l=0}^{L} \|\mathbf{W}_l\|_0 + \|\mathbf{v}_l\|_0 \leq s \right\},\end{aligned} \tag{4}$$

where $s > 0$, $\|\cdot\|_\infty$ denotes the maximum-entry norm and $\|\cdot\|_0$ denotes the number of non-zero entries, respectively. Let $\mathbf{v}_0$ be a zero vector for simply notation. The selecting procedures of unknown tuning parameters $(L, \mathbf{p}, s)$ shall be given in the Section 3.

2.3 Deep neural network estimator

In the functional data regression model, the common objective is to find an optimal estimator by least-square loss function. In the neural network setting, this coincides

with training neural networks by minimizing the empirical risk over all the training data. In particularly, given the networks in (4) and denote $\mathcal{F} = \mathcal{F}(L, \mathbf{p}, s)$, the proposed DNN estimator is defined as

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{j=1}^{N} \left\{ \overline{Y}_{\cdot j} - f\left(\mathbf{X}_j\right) \right\}^2, \tag{5}$$

where $\overline{Y}_{\cdot j} = \frac{1}{n} \sum_{j=1}^{n} Y_{ij}$. Different from classical nonparametric estimators, $\widehat{f}$ has no analytical expression or basis expansion expression. Hence, to better understand the reasons that this DNN estimator has excellent performance, we first project f_0 onto the network space $\mathcal{F}$, namely, $f^* := \arg\min_{f \in \mathcal{F}} \|f_0 - f\|_\infty$. In other words, f^* is the best possible approximation of f_0 in $\mathcal{F}$. Note that

$$\frac{1}{N} \sum_{j=1}^{N} \left(\overline{Y}_{\cdot j} - \widehat{f}(\mathbf{X}_j) \right)^2 \leq \frac{1}{N} \sum_{j=1}^{N} \left(\overline{Y}_{\cdot j} - f^*(\mathbf{X}_j) \right)^2,$$

which is equivalent to

$$\frac{1}{N} \sum_{j=1}^{N} \left(f_0(\mathbf{X}_j) - \widehat{f}(\mathbf{X}_j) + \overline{\rho}_{\cdot j} \right)^2 \leq \frac{1}{N} \sum_{j=1}^{N} \left(f_0(\mathbf{X}_j) - f^*(\mathbf{X}_j) + \overline{\rho}_{\cdot j} \right)^2,$$

where $\overline{\rho}_{\cdot j} = \frac{1}{n} \sum_{i=1}^{n} \rho_{ij} = \frac{1}{n} \sum_{i=1}^{n} \eta_i\left(\mathbf{X}_j\right) + \frac{1}{n} \sum_{i=1}^{n} \epsilon_i\left(\mathbf{X}_j\right)$. Hence, we follow the conventional approximation-estimation decomposition (or bias-variance tradeoff) to decompose the empirical norm $\|\widehat{f} - f_0\|_N = \frac{1}{N} \sum_{j=1}^{N} \left(\widehat{f}(\mathbf{X}_j) - f_0(\mathbf{X}_j) \right)^2$ as

$$\|\widehat{f} - f_0\|_N \leq \underbrace{\frac{1}{N} \sum_{j=1}^{N} \left(f^*(\mathbf{X}_j) - f_0(\mathbf{X}_j) \right)^2}_{\text{approximation error}} + \underbrace{\frac{2}{N} \sum_{j=1}^{N} \left(\widehat{f}(\mathbf{X}_j) - f^*(\mathbf{X}_j) \right) \overline{\rho}_{\cdot j}}_{\text{estimation error}}. \tag{6}$$

The above equation indicates that the empirical norm of the estimator is bounded by two items. The first item is the approximation error and essentially determined by the distance between the network class $\mathcal{F}$ and true function class f_0, which can be arbitrarily small according to [32]. From statistical point of view, the second item is the estimation error and is a weighted average of a random process. It is affected by the parameters in $\mathcal{F}$, true mean function class, and the characteristic of the error terms.

3 Implementation

In this section, we discuss the detailed computational procedure for the proposed DNN estimator in (5). The following proposed computational procedure can be easily realized via R package "`FDADNN`" which is available at `https://github.com/FDASTATAUBURN/FDADNN`.

3.1 Neural network's architecture selection

Tuning parameters are crucial as they control the overall behavior of the proposed estimator and the learning process. In machine learning, those parameters are called network architecture parameters. A neural network's architecture can simply be defined as the number of layers L, and the number of hidden neurons within these layers $\mathbf{p}$. In our considered sparse neural network space $\mathcal{F}$, sparse parameter s should also be carefully selected. Note that in the practice, it's unrealistic to control the exact number of inactive nodes, so instead of using sparse parameter s, we add an L_1 penalty to control the number of active nodes in each layer during optimization procedure. Denote ζ as the L_1 regularization factor. In the following, we utilize ζ to replace sparse parameter s in the numerical analysis. The ultimate goal is to find an optimal combination of $(L, \mathbf{p}, \zeta)$ that minimizes a pre-defined loss function to give better results. There are fairly large numbers of literature discussing the optimization selection, such as grid search, random search, and Bayesian optimization. Considering the computational efficiency and statistical properties, we recommend the following data-adaptive selection procedure in the practical application. The further justification of the optimization algorithm is beyond the scope of this work and shall be anther interesting and challenge topic for the future work.

We set the same neuron numbers for each layer for simplicity, i.e. $\mathbf{p} = (p, \ldots, p)$, and we follow the rule that p is increasing as n and N are increasing. We use K-fold cross-validation to choose (L, p, ζ), i.e.,

$$(L_{opt}, p_{opt}, \zeta_{opt}) = \arg \min_{(L,p,s)\in\Theta} \sum_{k=1}^{K} \sum_{j=1}^{N} \left(\widehat{\overline{Y}}_{\cdot j}^{(-k)} (L, p, \zeta) - \overline{Y}_{\cdot j}^{(k)} \right)^2,$$

where Θ is a architecture parameter space which contains pre-selected choices of (L, p, ζ). Typically, $K = 5$ or 10. For the k-th cross-validation, at the j-th grid point, $\widehat{\overline{Y}}_{\cdot j}^{(-k)} (L, p, \zeta)$ denotes the estimated output given (L, p, ζ) and $\overline{Y}_{\cdot j}^{(k)}$ is the average of observations.

3.2 Training neural networks

The minimization in (5) is usually done via stochastic gradient descent (SDG). In a way similar to gradient descent, in each update, a small sub-sample called a *batch* which is typically of size $B = 32$ to 512, is randomly drawn and the gradient calculation is only on the sub-sample instead of the whole training dataset. This saves considerably the computational cost in calculation of gradient. By the law of large numbers, this stochastic gradient should be close to the full sample one, albeit with some random fluctuations. We choose $B = 32$ or 64 batches depending on the performance of convergence. A pass of the whole training set is called an *epoch*. Typical choices of epochs are 200, 300 and 500. The number of epochs which defines the number times that the learning algorithm works through the entire training data set.

There are certainly some challenges for SGD to train DNN. For example, albeit good theoretical guarantees for well-behaved problems, SGD might converge very

slowly; the learning rates are difficult to tune [1]. To address these challenges, several variants gradient-based optimization algorithms are introduced, such as `Adam`, `RMSprop` and `Adadelta`. Instead of the classical SGD procedure, `Adam` is a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement. Hence, it is well suited for problems when there are large sample size and parameters [12]. In our numerical studies, `Adam` provides the best results and is the most computationally efficient among these candidates. We recommend `Adam` in the real life applications for FDA.

4 Theoretical properties of the DNN estimator

In this section, we develop the convergence rate of the proposed DNN estimator in (5). For simple notations, $\log$ means the logarithmic function with base 2. For sequences $(a_n)_n$ and $(b_n)_n$, $a_n \asymp b_n$ means $a_n \leq c_1 b_n$ and $a_n \geq c_2 b_n$ where c_1 and c_2 are absolute constants for any n. Let $\mathbf{C}_N = [G(\mathbf{X}_j, \mathbf{X}_{j'})/N]_{j,j'=1}^N$ be the $N \times N$ kernel matrix corresponding to covariance function G. We now introduce the main assumptions:

(A1) The true regression function $f_0 \in \mathcal{G}(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, \mathbf{K})$. (The definition of $\mathcal{G}(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, \mathbf{K})$ is given in the supplementary file.)

(A2) The standard deviation function $\tau(\cdot)$ is bounded for any $\mathbf{x} \in [0,1]^d$.

(A3) The eigenvalues of $G(\cdot,\cdot)$ satisfy $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$ and $\sum_{k=1}^{\infty} \lambda_k < \infty$. Moreover, the maximal eigenvalue of the kernel matrix $\mathbf{C}_N$ satisfies $\lambda_{1,N} = O(N^{-\varrho})$ for some constant $\varrho \geq 0$.

(A4) The DNN estimator $\widehat{f} \in \mathcal{F}(L, \mathbf{p}, s)$, where $L \asymp \log(nN^{\varrho})$, $s \asymp (nN^{\varrho})^{\frac{1}{\theta+1}}$, $\min_{l=1,\ldots,L} p_l \asymp (nN^{\varrho})^{\frac{1}{\theta+1}}$, for $\theta = \min_{i=0,\ldots,q} \frac{2\beta_i^*}{t_i}$.

Assumption (A1) is a natural definition for neural network, which is fairly flexible and many well known function classes are contained in it. For example, the additive model $f_0(\mathbf{x}) = \sum_{i=1}^d f_i(x_i)$, can be written as a composition of two functions $f_0 = g_1 \circ g_0$, with $g_0(\mathbf{x}) = (f_1(x_1), \ldots, f_d(x_d))^\top$ and $g_1(\mathbf{x}) = \sum_{j=1}^d x_j$, such that $g_0 : [0,1]^d \to \mathbb{R}^d$ and $g_1 : \mathbb{R}^d \to \mathbb{R}$. Here $\mathbf{d} = (d, d, 1)$ and $\mathbf{t} = (1, d)$. The generalized additive model $f_0(\mathbf{x}) = h\left(\sum_{i=1}^d f_i(x_i)\right)$, it can be written as a composition of three functions $f_0 = g_2 \circ g_1 \circ g_0$, with g_0, g_1 described above, and $g_2 = h$.

Assumption (A2) is a standard assumption for the variance of measurement errors, which requires the bounded variance of measurement error over the whole space. This assumption has been widely used in functional data nonparametric regression literature, see [6, 30] for example. Assumption (A3) is a standard eigenvalue assumption for Mercer kernel and it is widely used assumption for covariance functions in FDA literature, see [5, 14] for example. We also provide two examples to demonstrate (A3) is a reasonable assumption in the supplementary file. By [4], Assumption (A3) trivially holds for $\varrho = 0$ (see Proposition 1 and 2), and may even hold for some positive ϱ as revealed by Examples 1 in Section A. Assumption (A4) depicts the architecture and parameters' setting in the network space.

We assume a natural compositional function class for the true mean function f_0:

$$f_0 = g_q \circ g_{q-1} \circ \ldots \circ g_1 \circ g_0,$$

where $g_i : [a_i, b_i]^{d_i} \to [a_{i+1}, b_{i+1}]^{d_{i+1}}$, $g_i = (g_{ij})^\top_{j=1,\ldots,d_{i+1}}$, $i = 1, \ldots, q$, with unknown parameters d_i and q.

The following theorem establishes the convergence rate of the DNN estimator $\widehat{f}$ under the empirical norm. Its proof and some technical lemmas will be provided in the supplementary file.

Theorem 1 *Under Assumptions (A1)-(A4), with probability greater than* $(1 - \frac{2}{nN^\varrho})^{\lceil \log(nN^\varrho) \rceil + 1} \to 1$*, we have*

$$\|\widehat{f} - f_0\|_N^2 \leq c(nN^\varrho)^{-\frac{\theta}{\theta+1}} \log^6(nN^\varrho), \tag{7}$$

where $\varrho \geq 0$*,* $\theta = \min_{i=0,\ldots,q} \frac{2\beta_i^*}{t_i}$*, c is a constant only depends on* $\mathbf{t}$*,* $\mathbf{d}$*,* $\boldsymbol{\beta}$ *which are defined in (1) in the supplementary file.*

5 Simulation

To illustrate how the introduced nonparametric regression estimators based on our proposed neural networks method behave in case of finite sample sizes, we conduct substantial simulations for both 2D and 3D functional data.

5.1 2D simulation

In this simulation, the 2D images are generated from the model:

$$Y_{ij} = f_0(\mathbf{X}_j) + \eta_i(\mathbf{X}_j) + \epsilon_i(\mathbf{X}_j), \tag{8}$$

where $\mathbf{X}_j = (X_{1j}, X_{2j}) = (j_1/N_2, j_2/N_2)$, $1 \leq j_1, j_2 \leq N_2$ are equally spaced grid points on the $[0,1]^2$ and $N_2^2 = N$. To demonstrate the practical performance of our theoretical results, we consider the following two mean functions:

- Case 1 : $f_0(x_{1j}, x_{2j}) = \frac{-8}{1+\exp\left(\cot(x_{1j}^2)\cos(2\pi x_{2j})\right)}$,
- Case 2 : $f_0(x_{1j}, x_{2j}) = \log\left(\sin(2\pi x_{1j}) + 2|\tan(2\pi x_{2j})| + 2\right)$,

and the corresponding images are shown in the first row of Figure 1. To simulate the within-subject dependence for each subject i, we generate $\eta_i(\cdot)$ from a Gaussian process, with mean 0, and covariance function $G_0(\mathbf{x}_j, \mathbf{x}_{j'}) = \sum_{k=1}^2 \cos(2\pi(x_{kj} - x_{kj'}))$, $j, j' = 1, \ldots, N$. We generate $\epsilon_i(\mathbf{x}_j) = \varepsilon_{ij} \sim^{\text{i.i.d.}} \mathcal{N}(0, \sigma^2)$ for $i = 1, \ldots, n$, $j = 1, \ldots, N$. The noise level is set to be $\sigma = 1, 2$. We consider sample size $n = 50, 100, 200$ and for each image, let $N_2 = 15$ or 25, which means for each 2D image, the number of observational points (pixels) is set to be $N = N_2^2 = 225$ or 625. The neural network (5) is trained through optimizer `Adam` with architecture parameters (L, p, ζ) selected from $L \in \{3, 4\}$, $p \in \{100, 300, 500, 1000, 2000\}$,

$\zeta \in \{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}$. 10-fold cross-validation method discussed in Section 3 is applied to select the optimal architecture parameters in each Monte Calro simulation. Epochs are selected from 300 to 500 and batch size is chosen as 32. We find the convergence of algorithms is promising.

The alternative approach for 2D case we considered is a 2D regression spline method (bivariate spline). With regard to the variety of modifications of this approach known in the literature, we focus on the version for 2D FDA in [13]. Let $\mathbf{B}^\top(\mathbf{x}) = \{B_m(\mathbf{x})\}_{m\in\mathcal{M}}$ be the set of bivariate Bernstein basis polynomials, where $\mathcal{M}$ stands for an index set of Bernstein basis polynomials. Then we can represent any bivariate function $f(\mathbf{x})$ by $f(\mathbf{x}) \approx \mathbf{B}^\top(\mathbf{x})\boldsymbol{\gamma}$ where $\boldsymbol{\gamma}^\top = (\gamma_m, m \in \mathcal{M})$ is the bivariate spline coefficient vector. The estimator $\widehat{f}_{BS}$ is implemented by the R package `BPST`, which was developed by the authors of [13].

The second and the third rows in Figure 1 depicts the proposed neural network estimator $\widehat{f}_{DNN}$ and bivariate spline estimator $\widehat{f}_{BS}$ when $n = 200$, $N = 625$ and $\sigma = 1$. Table 1 summarizes the empirical L_2 risk and standard deviation of estimators $\widehat{f}_{DNN}$ and $\widehat{f}_{BS}$ under 100 simulations for two different noise levels. From the above figures and table, one can see that our method and the bivariate spline method have fairly similar estimation performances. As the bivariate spline estimator is able to achieve the optimal nonparametric convergence rate [29], the comparable estimation results in Tables 1 and 2 also support the asymptotic convergence rate of our proposed estimator $\widehat{f}_{DNN}$ in Theorem 1.

Table 1 The average empirical L_2 risk and their standard deviations of f_0 (Case 1) across 100 simulation runs (2D case).

$f_0(x_{1j}, x_{2j}) = \frac{-8}{1+\exp\left(\cot(x_{1j}^2)\cos(2\pi x_{2j})\right)}$						
σ	N	n	DNN		bivariate spline	
			L_2 risk	SD	L_2 risk	SD
1	225	50	0.1327	0.1905	0.6030	0.0418
		100	0.0797	0.1244	0.5757	0.0249
		200	0.0432	0.0574	0.5584	0.0120
	625	50	0.0770	0.0497	0.1497	0.0462
		100	0.0535	0.0368	0.1136	0.0214
		200	0.0352	0.0295	0.0987	0.0098
2	225	50	0.1880	0.1521	0.6564	0.1009
		100	0.0918	0.0793	0.6035	0.0619
		200	0.0593	0.0529	0.5765	0.0316
	625	50	0.1594	0.1555	0.2241	0.1218
		100	0.0862	0.0755	0.1430	0.0557
		200	0.0420	0.0412	0.1098	0.0232

5.2 3D simulation

For 3D simulation, the images are generated from the model (8) in 2D case. The true mean function is $f_0(x_{1j}, x_{2j}, x_{3j}) = \exp\left(\frac{1}{3}x_{1j} + \frac{1}{3}x_{2j} + \sqrt{x_{3j} + 0.1}\right)$, where

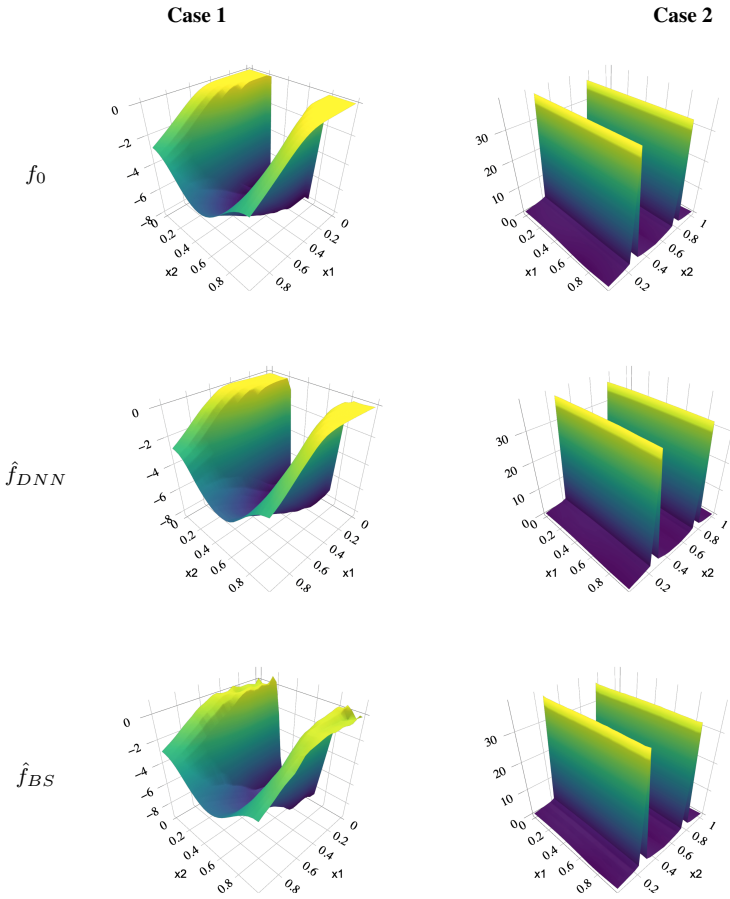

Fig. 1 2D simulation. Left: from the top to bottom, they are true function f_0 (Case 1) and its estimators $\widehat{f}_{DNN}$ and $\widehat{f}_{BS}$. Right: from the top to bottom, they are true function f_0 (Case 2) and its estimators $\widehat{f}_{DNN}$ and $\widehat{f}_{BS}$. ($n = 200$, $N = 625$ and $\sigma = 1$)

$(x_{1j}, x_{2j}, x_{3j}) = \left(\frac{j_1}{N_3}, \frac{j_2}{N_3'}, \frac{j_3}{N_3''}\right)$, $1 \leq j_1 \leq N_3$, $1 \leq j_2 \leq N_3'$, $1 \leq j_3 \leq N_3''$ are equally spaced grid points in each dimension on $[0,1]^3$ and $N_3 N_3' N_3'' = N$. Here, we mimic the number of voxels of the real data, which usually have different values for N_3, N_3' and N_3''. For each subject, the within-imaging dependence $\eta_i(\cdot)$ is generated from a Gaussian process with mean 0, and covariance function $G_0(\mathbf{x}_j, \mathbf{x}_{j'}) = \sum_{k=1}^{3} \cos(2\pi(x_{kj} - x_{kj'}))$, $j, j' = 1, \ldots, N$. Measurement errors $\epsilon_i(\cdot)$ are generated the same as 2D case. We consider sample size $n = 50, 100, 200$ and $N = 3,000$ ($20 \times 15 \times 10$) and $4,500$ ($30 \times 15 \times 10$). Results of each setting are based on 100 simulations. The training of neural networks architecture (L, p, s) follows the same procedures as in 2D case. Architecture parameters (L, p, ζ) selected from $L \in \{3, 4\}$, $p \in \{100, 300, 500, 1000, 2000, 5000\}$, $\zeta \in \{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}$. The triangu-

Table 2 The average empirical L_2 risk and their standard deviations of f_0 (Case 2) across 100 simulation runs (2D case).

$f_0(x_{1j}, x_{2j}) = \log(\sin(2\pi x_{1j}) + 2\|\tan(2\pi x_{2j})\| + 2)$						
σ	N	n	DNN		bivariate spline	
			L_2 risk	SD	L_2 risk	SD
1	225	50	0.0731	0.0446	0.0804	0.0382
		100	0.0437	0.0249	0.0517	0.0186
		200	0.0254	0.0217	0.0351	0.0100
	625	50	0.0560	0.0206	0.0751	0.0351
		100	0.0351	0.0128	0.0541	0.0254
		200	0.0245	0.0085	0.0383	0.0110
2	225	50	0.1190	0.0975	0.1290	0.0950
		100	0.0829	0.0681	0.0931	0.0597
		200	0.0348	0.0276	0.0464	0.0251
	625	50	0.0573	0.0264	0.1213	0.0859
		100	0.0331	0.0132	0.0827	0.0630
		200	0.0139	0.0059	0.0502	0.0251

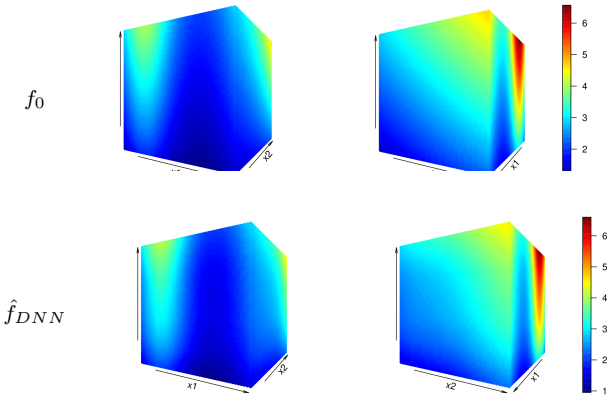

Fig. 2 Two different angles (Left and Right panels) to view the true mean function and the DNN estimator in 3D simulation case. ($n = 200$, $N = 4,500$, $\sigma = 1$)

larized bivariate splines method proposed in [29] are designed for 2D functions only. Extending spline basis functions for 3D functional data is very sophisticated and to our best knowledge, it is not available for 3D FDA yet. Hence, we only conduct 3D numerical analysis with our proposed DNN method. To exam the performance of the estimator $\widehat{f}$, we also summarizes the empirical L_2 risk and standard deviation of estimators $\widehat{f}_{DNN}$ in Table 3. It is clear to find that the empirical risk decrease when sample sizes or observed voxels numbers increase for both noise levels, which supports our theoretical findings. The mean function f_0 and its DNN estimator are presented in Figure 2. It is easy to conclude that the DNN estimator follows the the same pattern as the true mean function.

Table 3 The average empirical L_2 risk and their standard deviations of f_0 across 100 simulation runs (3D case).

σ	N	n	L_2 risk	SD
1	3000	50	0.0028	0.0020
		100	0.0011	0.0006
		200	0.0006	0.0004
	4500	50	0.0007	0.0007
		100	0.0005	0.0007
		200	0.0003	0.0004
2	3000	50	0.0030	0.0024
		100	0.0012	0.0007
		200	0.0007	0.0005
	4500	50	0.0009	0.0007
		100	0.0005	0.0008
		200	0.0003	0.0005

6 ADNI PET data analysis

The dataset used in the preparation of this article were obtained from the ADNI database (`adni.loni.usc.edu`). The ADNI is a longitudinal multicenter study designed to develop clinical, imaging, genetic, and biochemical biomarkers for the early detection and tracking of AD. From this database, we collect PET data from 79 patients in AD group. This PET dataset has been spatially normalized and post-processed. These AD patients have three to six times doctor visits and we only select the PET scans obtained in the third visits. Patients' age ranges from 59 to 88 and average age is 76.49. There are 33 females and 46 males among these 79 subjects. All scans were reoriented into $79 \times 95 \times 69$ voxels, which means each patient has 69 sliced 2D images with 79×95 pixels. For 2D case, it means each subject has $N = 7,505 = 79 \times 95$ observed pixels for each selected image slice. For 3D case, the observed number of voxels for each patient's brain sample is $N = 79 \times 95 \times 69$, which is more than 0.5 million.

For 2D case, we select the 20-th, 40-th and 60-th slices from 69 slices for each patient. We first take average across 79 patients for each slices (the first row in Figure 3). Then, based on the averaged images, we obtain the proposed DNN estimators for each slice (the second row in Figure 3). We also recover the image with higher resolutions 512×512 pixels, instead of the original 95×69 pixels for each slice (the third row in Figure 3). The neural network (5) is trained through optimizer `Adam` with architecture parameters (L, p, ζ) selected from $L \in \{3, 4\}$, $p \in \{500, 1000\}$, $\zeta \in \{10^{-5}, 10^{-6}, 10^{-7}\}$. 10-fold cross-validation method selects the optimal architecture parameters $L_{opt} = 3$, $p_{opt} = 1000$ and $\zeta_{opt} = 10^{-7}$. We used 300 to 500 epochs and 2 to 8 as batch size given different slices.

In 3D case, on 79 patients, and total $79 \times 95 \times 69$ voxels. Same as 2D case, we first average the total 79 3D scans into one 3D scan, and then perform neural network to train the model based on the averaged 3D image. In the bottom row of Figure 3, we break down the recovered 3D image and show the recovered 20-th, 40-th and 60-th slices. The neural network (5) is trained through optimizer `Adam` with architecture parameters (L, p, ζ) selected from $L \in \{3, 4\}$, $p \in \{1000, 1500\}$,

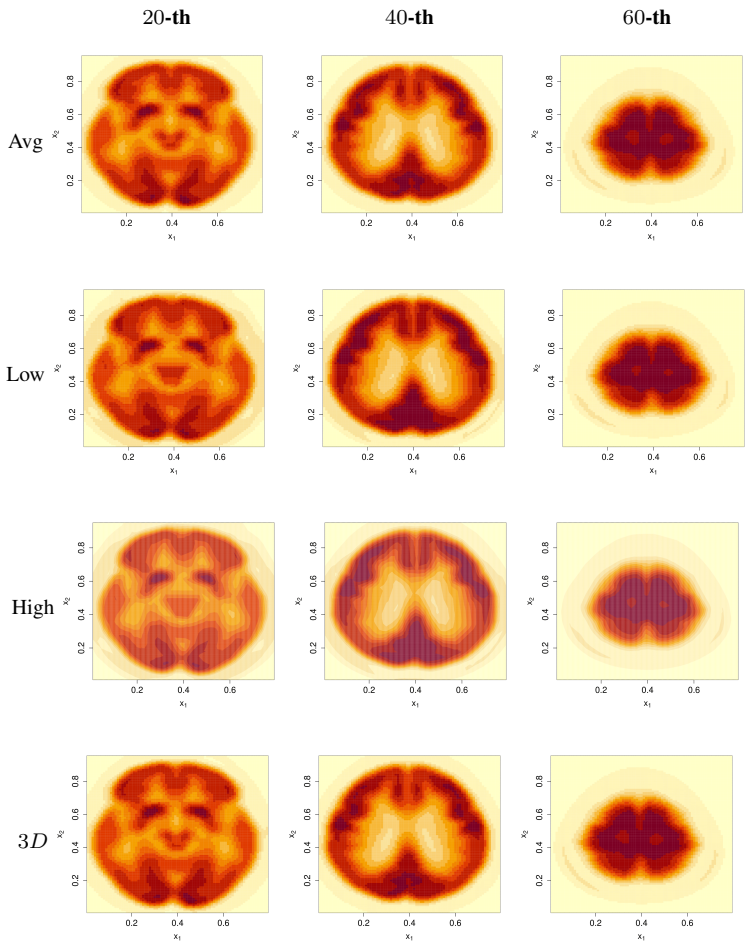

Fig. 3 From top to bottom are averaged images $\{\overline{Y}_{\cdot j}\}_{j=1}^{7505}$, recovered images $\widehat{f}(x_{1j}, x_{2j'})$, $j = 1, \ldots, 79$, $j' = 1, \ldots, 95$, recovered high resolution (128×128) images $\widehat{f}(x_{1j}, x_{2j})$, $j = 1, \ldots, 128$ and recovered images from $3D$ image. Left: The 20-th slices; Middle: The 40-th slices; Right: The 60-th slices.

$\zeta \in \{10^{-5}, 10^{-6}, 10^{-7}\}$. 10-fold cross-validation method selects the optimal architecture parameters $L_{opt} = 4$, $p_{opt} = 1500$ and $\zeta_{opt} = 10^{-7}$. According to our numerical experience, we find 300 epochs and 64 batch size providing the reasonable well results.

In Figure 4, we also recover the image in higher resolutions $128 \times 128 \times 128$ voxels, which means instead of the original $79 \times 95 \times 69$ voxels, we can provide the estimated image slices with higher resolution (128×128 pixels, instead of the original 79×95 pixels) at finer grid points (128 points, instead of the original 69 points).

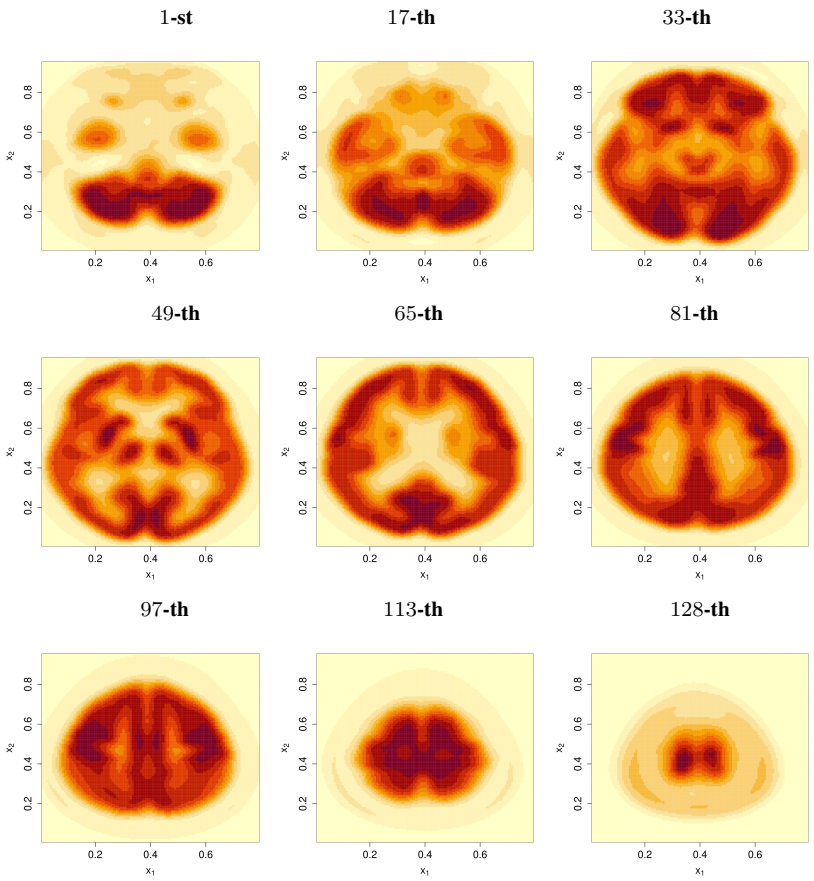

Fig. 4 Recovered higher resolutions of selected nine slices in 3D case.

7 Discussion

In this work, we resolve the model misspecification issue in multi-dimensional FDA via the promising technique from the deep learning domain. By properly choosing network architecture, our estimator achieves the optimal nonparametric convergence rate in empirical norm. To our best knowledge, this is the first piece of work in FDA, which yields attractive empirical convergence rate for multi-dimensional FDA, and at meanwhile is free from model misspecification. Numerical analysis demonstrates that our approach is useful in recovering the signal for imaging data. Some interesting future works may include the functional linear regression model and classification problems in the framework of DNN.

Acknowledgements

Data used in preparation of this article were obtained from the Alzheimers Disease Neuroimaging Initiative (ADNI) database (`adni.loni.usc.edu`). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at: `http://adni.loni.usc.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf`.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *Proceedings of the 36th International Conference on Machine Learning*, 97:242–252, 2019.
2. M. Anthony and P. Bartlett. *Neural Network Learning*. Cambridge University Press, Cambridge, 2009.
3. B. Bauer and M. Kohler. On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *The Annals of Statistics*, 47:2261–2285, 2019.
4. M. L. Braun. Accurate error bounds for the eigenvalues of the kernel matrix. *Journal of Machine Learning Research*, 7:2303–2328, 2006.
5. G. Cao, L. Wang, Y. Li, and L. Yang. Oracle-efficient confidence envelopes for covariance functions in dense functional data. *Statistica Sinica*, 26:359–383, 2016.
6. G. Cao, L. Yang, and D. Todem. Simultaneous inference for the mean function of dense functional data. *Journal of Nonparametric Statistics*, 24:359–377, 2012.
7. Hervé Cardot. Nonparametric estimation of smoothed principal components analysis of sampled noisy functions. *Journal of Nonparametric Statistics*, 12:503–538, 2000.
8. Lu-Hung Chen and Ci-Ren Jiang. Multi-dimensional functional principal component analysis. *Stat. Comput.*, 27(5):1181–1192, 2017.
9. Jianqing Fan, Cong Ma, and Yiqiao Zhong. A selective overview of deep learning. *arXiv preprint arXiv:1904.05526*, 2019.
10. Frédéric Ferraty and Philippe Vieu. *Nonparametric functional data analysis: Theory and practice*. Springer Series in Statistics. Springer, New York, 2006.
11. P. Hall, H. G. Müller, and J. L. Wang. Properties of principal component methods for functional and longitudinal data analysis. *The Annals of Statistics*, 34:1493–1517, 2006.
12. D. Kingma and J. Ba. Adam: A method for stochastic optimization. *In the 3rd International Conference on Learning Representations (ICLR)*, 2015.

13. M.J. Lai and L Wang. Bivariate penalized splines for regression. *Statistica Sinica*, 23:1399–1417, 2003.
14. Y. Li and T. Hsing. Uniform convergence rates for nonparametric regression and principal component analysis in functional/longitudinal data. *The Annals of Statistics*, 38:3321–3351, 2010.
15. E. Lila, J. A. D. Aston, and L. Sangalli. Smooth principal component analysis over two-dimensional manifolds with an application to neuroimaging. *The Annals of Applied Statistics*, 10(4):1854–1879, 2016.
16. R. Liu, B. Boukai, and Z. Shang. Optimal nonparametric inference via deep neural network. *arXiv Preprint arXiv:1902.01687*, 2019.
17. R. Liu, Z. Shang, and G. Cheng. On deep instrumental variables estimate. *arXiv preprint arXiv:2004.14954*, 2021.
18. J. S. Morris and R. J. Carroll. Wavelet-based functional mixed models. *Journal of the Royal Statistical Society, Series B*, 68:179–199, 2006.
19. J. O. Ramsay and B. W. Silverman. *Functional Data Analysis, Second Edition*. Springer Series in Statistics, New York, 2005.
20. J.A. Rice and C.O. Wu. Nonparametric mixed effects models for unequally sampled noisy curves. *Biometrics*, 57:253–259, 2001.
21. B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 2014.
22. J. Schmidt-Hieber. Nonparametric regression using deep neural networks with relu activation function. *The Annals of Statistics*, 48(4):1875–1897, 2020.
23. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.
24. Charles J. Stone. Optimal global rates of convergence for nonparametric regression. *The Annals of Statistics*, 10(4):1040–1053, 1982.
25. Barinder Thind, Kevin Multani, and Jiguo Cao. Deep learning with functional inputs. *arXiv preprint arXiv:2006.09590*, 2020.
26. Barinder Thind, Kevin Multani, and Jiguo Cao. Neural networks as functional classifiers. *arXiv preprint arXiv:2010.04305*, 2020.
27. B. Wang, B. Nan, J. Zhu, and R. Koeppe. Regulzarized 3d functional regression for brain image data via haar wavelets. *The Annals of Applied Statistics*, 8:1045–1064, 2014.
28. J.L. Wang, J. M. Chiou, and H. G. Müller. Functional data analysis. *Annual Review of Statistics and Its Application*, 3:257–295, 2016.
29. Y. Wang, G. Wang, L. Wang, and T. Ogden. Simultaneous confidence corridors for mean functions in functional data analysis of imaging data. *Biometrics*, 76:427–437, 2020.
30. F. Yao, H. G. Müller, and J. L. Wang. Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100:577–590, 2005.
31. F. Yao, H. G. Müller, and J. L. Wang. Functional linear regression analysis for longitudinal data. *The Annals of Statistics*, 33:2873–2903, 2005.
32. D. Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 27(94):103 –114, 2017.

33. Lan Zhou and Huijun Pan. Principal component analysis of two-dimensional functional data. *Journal of Computational and Graphical Statistics*, 23(3):779–801, 2014.