Neural Networks for Scalar Input and Functional Output

Sidi Wu[1], Cédric Beaulac[2,3] and Jiguo Cao[1]

[1]Department of Statistics and Actuarial Science, Simon Fraser University, Burnaby, BC, Canada.
[2]Department of Mathematics and Statistics, University of Victoria, Victoria, BC, Canada.
[3]School of Engineering Science, Simon Fraser University, Burnaby, BC,Canada.

Abstract

The regression of a functional response on a set of scalar predictors can be a challenging task, especially if there is a large number of predictors, these predictors have interaction effects, or the relationship between those predictors and the response is nonlinear. In this work, we propose a solution to this problem: a feed-forward neural network (NN) designed to predict a functional response using scalar inputs. First, we transform the functional response to a finite-dimension representation and then we construct a NN that outputs this representation. We proposed different objective functions to train the NN. The proposed models are suited for both regularly and irregularly spaced data and also provide multiple ways to apply a roughness penalty to control the smoothness of the predicted curve. The difficulty in implementing both those features lies in the definition of objective functions that can be back-propagated. In our experiments, we demonstrate that our model outperforms the conventional function-on-scalar regression model in multiple scenarios while computationally scaling better with the dimension of the predictors.

Keywords: Neural network, Functional data analysis, Functional response, Functional principal component analysis

1 Introduction

Functional data analysis (FDA) is a rapidly developing branch of statistics which targets at the theory and analysis of functional variables. As the atom of FDA, functional variables or functional data refer to curves, surfaces and any random variables taking values in an infinite dimensional space, such as time and spatial space [14, 24]. A basic and commonly recognized framework in FDA is to regard the functional data as realization of an underlying stochastic process [34], and indeed, a large fraction of data coming from different fields can be characterized as functional data. Figure 1, for example, illustrates the fertility rates over the age of females for 92 countries. Each curve is treated as a smooth function of age and can be viewed as functional data. The fertility curves were measured at several time points, each of which represents an age group in increasing order. This age-specific fertility rate (ASFR) data was collected from the United Nations Gender Information (UNGEN) database by Mehrotra and Maity [31]. They estimated the observations based on the surveys conducted between 2000 and 2005. The data set is publicly available at https://github.com/suchitm/fosr_clust.

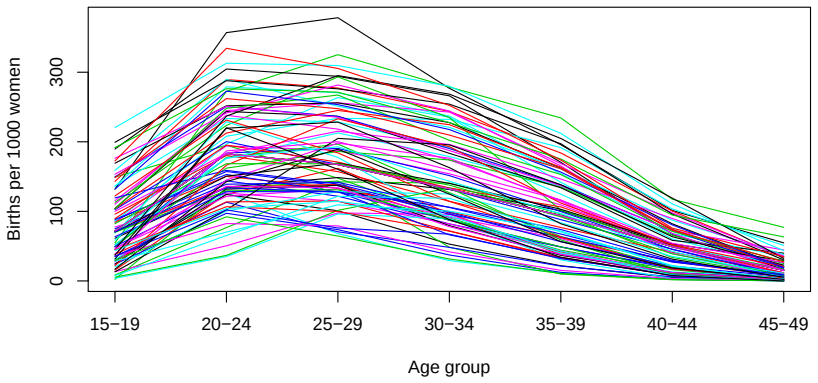

Fig. 1 Age-specific fertility rates for 92 countries around the world.

Among different aspects of FDA, functional regression has received the most attention in applications and methodological development [21]. Functional regression, in general, can be divided into three types: (1) scalar-on-function regression (regression analysis of scalar responses on functional predictors), (2) function-on-function regression (predictors and responses of the regression are both functional), and (3) function-on-scalar (FoS) (regression of functional responses on scalar predictors, also named as the varying-coefficient model). While plenty of work has been done for the first two scenarios, studies discussing functional response regression remain scarce. Functional linear

models (FLM), an extension of the classic linear models, are the most conventional and widely applied methods in dealing with regression problems. A general form of FLM for function-on-scalar regression can be written as

$$Y(t) = \boldsymbol{X}\boldsymbol{\beta}(t) + \epsilon(t), \tag{1}$$

where $Y(t)$ is the functional response, and $\boldsymbol{X}$ is a vector of scalar covariates. The unknown vector $\beta(t)$ consists of parameter functions varying over t, and $\epsilon(t)$ stands for the random error term. In FDA, the consensual approach is to represent functional data with a linear combination of a finite number of known basis functions. Ramsay & Silverman [24] introduced projecting functional response with basis functions like B-splines for fitting the function-on-scalar regression model. Chiou et al. [5, 6] proposed to use Functional Principal Component Analysis (FPCA) for dimensional reduction and represent the functional response with eigenfunctions obtained via spectral decomposition of the covariance function of $Y(t)$. Both approaches summarize the information carried by the functional variable $Y(t)$ to a finite-dimensional scalar representation (B-spline basis coefficients or FPC scores). In solving linear functional regression problems, taking FoS regression as an example, the linear relation between the scalar predictors and functional response is indeed captured by fitting a linear relationship between the predictors and the scalar representation.

Some nonlinear approaches has been developed to handle more complicated regression settings. Zhang and Wang [40] combined the FoS and additive models to gain the varying-coefficient additive model for functional data which do not require the linearity assumption between the scalar predictors and functional response. Afterwards, an extension to this varying-coefficient additive model, named as functional additive mixed model, was established by Scheipl et al. [12] with a general form

$$Y(t) = \sum_{r=1}^{R} f_r\left(\boldsymbol{X}_r, t\right) + \epsilon(t), \tag{2}$$

for functional response $Y(t)$. Each term in the additive predictor $f_r(\boldsymbol{X}_r, t)$ is a function of t and a subset $\boldsymbol{X}_r$ of the complete predictor set $\boldsymbol{X}$ including scalar and functional covariates. This extensive framework can consist of both linear and nonlinear effects of functional and scalar covariates that may vary smoothly over the index of the functional response. Naturally, this model was further extended, by Scheipl et al [13], to the generalized functional additive mixed model in order to take care of non-Gaussian functional response. Although several nonlinear attempts have been made, the existing regression methods for FDA have been predominantly linear [34]. Considering the emerging trend of complicatedly structured functional data, there is a increasing demand in developing more nonlinear approaches to FDA.

In this work, we propose a solution to the FoS problem which is able to handle a large number of predictors and the non-linear relation between the scalar predictors and the functional response. Our solution borrows from the machine learning literature, which follows a trend in adapting machine learning techniques to known statistical problems such as survival analysis [3, 17, 19] or causal inference [20, 29]. We adapt the neural network (NN) architecture for functional data.

In some existing works, efforts have been made to combine deep neural networks to the field of functional data analysis. For instance, Rossi et al.[26, 27] firstly explored the idea of applying neural networks to functional data by constructing a functional neural network (FNN) with functional neurons in the first hidden layer for functional inputs. FNN was then extended by Thind et al. [32, 33] to feed both functional and scalar coviarates as input and outputs a scalar response. Rao et al. [25] equipped FNN with geographically weighted regression and spatial autoregressive technique to handle regression problems with spatially correlated functional data. Meanwhile, Wang et al.[36] proposed a non-linear function-on-function model using a fully connected neural network. Previously, Yao et al. [39] developed a neural network with a new basis layer whose hidden neurons are micro neural networks, to perform a parsimonious dimension reduction for functional inputs using information relevant to the scalar target.

Most of the mentioned works are focused on building neural networks with functional inputs and scalar outputs. In this work, we consider the other side of the coin. We design a neural network meant to predict a functional response using scalar inputs. And to the best of our knowledge, this study is the first attempt to solve the function-on-scalar regression problem using artificial neural networks. Because the standard machine learning techniques are designed for finite-dimensional feature vectors, we propose to encode the information contained in the intrinsically infinite-dimensional functional response to some finite-dimensional scalar representation. Different from other nonlinear approaches targeting at function-on-scalar regression problems, our methods, in particular, focus on studying the nonlinear association between the predictors and the scalar representations of the functional response, to further reveal the relation between the scalar predictors and functional response. In this way, we maintain the interpretability of the relation between the scalar predictors and the functional response, despite of the usage of neural networks.

Challenged by the age-specific fertility rate data introduced previously, in this work, we are interested in conducting a function-on-scalar analysis to predict the functional curves using scalar predictors accurately and also reveal any potential nonlinear relation between the predictors and response. For each of the 92 observations (countries), the functional curve of fertility rates is associated with 15 demographic and socioeconomic variables averaging over the information available on Gapminder in a country-level manner from 2000 to 2005 [31]. These 15 scalar covariates consist of age at first marriage, under-5 mortality, maternity deaths per 1000 women, cervical cancer deaths per

100,000 women, female labor force participation, male to female ratio ((women % men women aged from 15-49), contraception prevalence (women aged from 15-49), life expectancy, mean years of school (women aged from 15-34), female's BMI, the number of births attended by trained birth staff (% of total), GDP per capita, the proportion of dollar billionaires per 1 million people, health expenditures (% of GDP) and the amount of alcohol its populace consumes. All the covariates downloaded from the listed data source have been previously standardized, and many predictor pairs among them are found highly correlated.

Due to the difficulty in directly determining whether there is a nonlinear relation between a scalar variable and a random function, we pay more attention to the association between scalar covariates and the finite-dimensional scalar representation summarizing the information of the functional response. Regarding the scalar representation, we choose to use the basis coefficients which are estimated by approximating the response functions with a linear expansion of the most common B-spline basis functions. Besides the initial choice of the basis coefficients, we also attempted to implant the FPC scores as the scalar representation in real application.

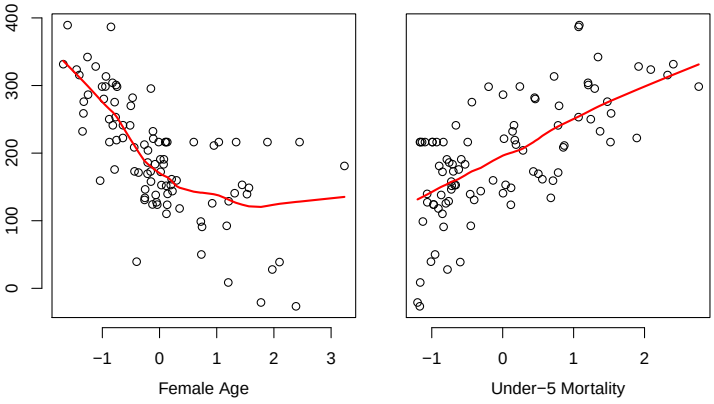

Fig. 2 Plots of the second B-spline basis coefficients (estimated using $Y(t)$) versus two scalar covariates, including female age and under-5 mortality, separately. The red line is the smoothing curve estimated using `R` function `lowess()`.

Accordingly, with the ASFR data set, instead of picturing the relation between the infinite-dimensional response curve and the 15 scalar covariates, we obtain visualizations of one B-spline basis coefficient against one predictor for all possible combinations. The representation process is purely based on the fertility trajectories, while no information from the 15 predictors are implanted. Six B-spline basis functions are employed in implementation, which

is recommended by the result of cross validation. Figure 2 displays the association for some selected basis-coefficient-predictor pairs. We incorporated a locally-weighted smoothing curve in red to help visualizing if the association is linear or not. The existence of nonlinear associations between some coefficient-predictor pairs is endorsed, as shown with the female age predictor in Figure 2. On the other hand, the association patterns of basis coefficients to under-5 mortality are linear as depicted in Figure 2. As both linear and nonlinear patterns are detected among all possible mapping-coefficient-predictor sets, we would like to set up a procedure with the ability to include every available covariate to precisely predict the response trajectory, while simultaneously capturing the true relationships between the predictors and the scalar representatives of the functional response.

Three main contributions are made in this paper. First, we propose a general framework of NN with functional response which aims to predict curves with scalar inputs through a feed-forward NN architecture. Second, we pay attention to the behaviour of functional outputs during the network training process and propose to hold their natural smoothness property by adding a roughness penalty in the objective function for training NN. Lastly, we make numerous comparisons under various scenarios to analyze and conclude the conditions of use for different versions of our model.

The biggest challenge to the implementation of the aforementioned contributions is to ensure the NN models can be trained using backpropagation [28]. Theoretically, we need to define the flow of information from the input to the objective function as a sequence of operations differentiable with respect to the weights of the NN. However, in practice, because we rely on an auto-differentiation package, such as Keras [7], to train the NN, we need to define the objective function as a sequence of operations provided by that package. The available differential operations are incapable of dealing with derivatives of the integral of the infinite-dimensional functional response with respect to NN weights. Besides, some common features when fitting functional data, such as roughness penalties, were obviously not designed with that consideration in mind. For example, it is common to penalize the integrated second-order derivative of a functional curve for smoothing in FDA. But a roughness penalty cannot simply be integrated into the objective function when training the NN. Overcoming that challenge is a central part of these contributions mentioned above and to do so, we needed to be creative and adapt the objective functions in various ways detailed later in this manuscript.

The remainder of this paper is organized as follows. In section 2, we detail the proposed neural networks with functional response, with an additional discussion on how to control the smoothness of the predicted curves. A brief discussion about the computational costs of neural networks and the FLM with functional response is provided in Section 3. Section 4 contains the prediction results on real applications using different methods. In section 5, we conduct simulation studies, under linear and nonlinear scenarios, and compare the proposed models and the existing function-on-scalar model in both the prediction

accuracy for discrete observed timestamps and the ability to reconstruct the true response trajectories over a continuum. Lastly, some concluding remarks are given in section 6.

2 Methodology

Suppose we have N subjects and for the i-th subject, the input is a set of numerical variables $\boldsymbol{X}_i = \{X_{i1}, X_{i2}, ..., X_{iP}\}$, and the output is a functional variable $Y_i(t), t \in \mathcal{T}$ in the $L^2(t)$ space. In reality, functional variables $Y_i(t)$ are usually measured in a discrete manner, for instance, at m_i time points or locations. Therefore, instead of observing the full trajectory of $Y_i(t)$, for the i-th subject, we obtain m_i pairs of observations $\{t_{ij}, Z_i(t_{ij})\}$, $j = 1, 2, ...m_i$ and $Z_i(t_{ij}) = Y_i(t_{ij}) + \epsilon_i(t_{ij})$, where $\epsilon(t_{ij})$ is the i.i.d. observation error. To simplify the situation, in the following discussion, we assume that for all subjects, the functional term $Y(t)$'s are observed at the same m dense and equally-spaced time points, In other words, we assume that $m_1 = m_2 = \cdots = m_N = m$ and $m_{1j} = m_{2j} = \cdots = m_{Nj}$ for $j = 1, 2, ..., m$.

2.1 Neural network with functional response: mapping to basis coefficients (NNBB)

To overcome the difficulty in modeling infinite-dimensional data, a common pipeline for FDA is to summarize the information of functions $\{Y_i(t)\}_{i=1}^N$ into a set of finite-dimensional vectors of coefficients using some basis representation method, and consequently,

$$Y_i(t) = \sum_{k=1}^{K_b} c_{ik}\theta_k(t) = \boldsymbol{\theta}'\boldsymbol{C}_i, \tag{3}$$

where K_b is some predefined truncation integer that determines the number of basis and $\boldsymbol{\theta}$ is the vector containing the basis functions $\theta_1(t), ..., \theta_{K_b}(t)$ from a selected basis system, such as the Fourier basis system and B-spline system (In our study, we mainly used the latter). $\boldsymbol{C_i}$ is a K_b-dimensional vector of basis coefficients $\{c_{ik}\}_{k=1}^{K_b}$ which need to be determined, they serve as parameters of this model. With smooth underlying functions $Y_i(t)$, the actual discrete observations $Z(t_{ij})$ are naturally used to estimate $\{c_{ik}\}_{k=1}^{K_b}$ by fitting $Z_i(t_{ij}) = \sum_{k=1}^{K_b} c_{ik}\theta_k(t_{ij}) + \epsilon_i(t_{ij})$. A least square estimator of c_{ik}, denoted by c_{ik}° is obtained by minimizing $\text{SSE}(Z_{ik}|c_{ik}) = \sum_{j=1}^{m} \left(Z_{ij} - \sum_{k=1}^{K_b} c_{ik}\theta_k(t_{ij})\right)^2$.

Eq.(3) implies that $Y(t)$ can be approximated by a linear combination of the basis functions $\theta(t)$, which carry the fixed modes of variations over $\mathcal{T}$ with real-valued basis coefficients. Learning how the predictors X's regress on the variation of $Y(t)$, as a result, can be naturally replaced with learning how the predictors X's regress on the set of unknown basis coefficients c_{ik}. Therefore we propose to set the basis coefficients c's as a function of X's. Let $F(\cdot)$ be

a mapping function from $\mathbb{R}^P$ to $\mathbb{R}^{K_b}$ so that $\boldsymbol{X}$ can be mapped to the basis coefficient matrix as

$$\boldsymbol{C}_i = F(\mathbf{X}_i), \tag{4}$$

which in turn can be used to map $\boldsymbol{X}$ to the functional response $Y(t)$ with $Y_i(t) = \boldsymbol{\theta}' F(\boldsymbol{X}_i)$. Consequently, we can learn the nonlinear relation between $\boldsymbol{X}_i$ and $Y_i(t)$ by setting $F(\cdot)$ to be a nonlinear function. Here we propose a dense feed-forward NN as the mapping function $F(\cdot)$, where the basis coefficients $[c_{i1}, c_{i2}, ..., c_{iK_b}]$ are the outputs of the NN, then the model can be expressed as

$$\boldsymbol{C}_i = \mathrm{NN}_\eta(\boldsymbol{X}_i) = g_L\left(\cdots g_1\left(\sum_{p=1}^{P} w_{1p}X_{ip} + b_1\right)\right), \tag{5}$$

where $g_1, ..., g_L$ are the activation functions at each layer, and η denote the NN parameter set consisting of weights $\{w_{\ell p}\}_{\ell=1}^{L}$ and bias $\{b_\ell\}_{\ell=1}^{L}$ of all hidden layers. The NN is optimized by minimizing the standard objective function calculating the mean-squared-error (MSE) difference as $L_{\boldsymbol{C}}(\eta) = 1/n_{\text{train}} \sum_{i=1}^{n_{\text{train}}} \sum_{k=1}^{K_b} (\hat{c}_{ik} - c_{ik})^2$, where n_{train} is the number of samples in the training set, and in application the *true* basis coefficients c_{ik}'s are replaced with $c^\circ{}_{ik}$.

We named this model NNBB has it is a Neural Network (NN), with B-spline coefficient output (B) and trained by minimizing the MSE of those B-spline coefficients (B). We summarize the training process of NNBB in Algorithm 1. The optimized NN is used for prediction where it takes the new scalar inputs in the test set and then outputs the predicted basis coefficients $\hat{\mathbf{C}}$. The predicted functional response $\hat{Y}(t)$ is further constructed as $\hat{Y}(t) = \boldsymbol{\theta}'\hat{\mathbf{C}}$.

2.2 Neural network with functional response: mapping to FPC scores (NNSS)

Apart from basis expansion, the other popular approach for dimension reduction is FPCA. To be specific, let $\mu(t)$ and $K(t, t') = \mathrm{cov}(Y(t), Y(t'))$ be the mean and covariance functions of the underlying function $Y(t)$, and accordingly, the spectral decomposition of the covariance functions is $K(t, t') = \sum_{k=1}^{\infty} \gamma_k \phi_k(t)\phi_k(t')$, where $\{\gamma_k, k \geq 1\}$ are the eigenvalues in decreasing order with $\sum_k \gamma_k < \infty$ and ϕ_k's are corresponding eigenfunctions with restriction of $\int \phi_k^2(t)dt = 1$. Following the Karhunen-Loéve expansion, the i-th observed functional response $Y_i(t)$ can be represented as $Y_i(t) = \mu(t) + \sum_{k=1}^{\infty} \xi_{ik}\phi_k(t)$. Denote $\tilde{Y}_i(t) = Y_i(t) - \mu(t)$ as the centered functional response we can express the expansion as

$$\tilde{Y}_i(t) = \sum_{k=1}^{\infty} \xi_{ik}\phi_k(t), \tag{6}$$

Algorithm 1: Training NNBB

Input: $\boldsymbol{X}_i = \{X_{i1}, X_{i2}, ..., X_{ip}\}$, $\{Z_i(t_{ij})\}_{j=1}^m$ in the training set n_{train}
Output: NN with optimized parameter set $\hat{\eta}$

set up hyper-parameters, including:
- for smoothing $Z_i(t_{ij})$: basis functions vector $\boldsymbol{\theta} = [\theta_1(t), ..., \theta_{K_b}(t)]$, number of basis functions K_b
- for training NN: number of hidden layers (L), number of nodes per hidden layer, activation functions ($g_1, ..., g_L$), number of epochs (E), batch size, NN optimizer (with a learning rate ϱ), etc.

forall $i \in n_{\text{train}}$ **do** obtain $\{c_{ik}^{\circ}\}_{k=1}^{K_b}$ by projecting $\{Z_i(t_{ij})\}_{j=1}^m$ to the selected set of basis functions $\boldsymbol{\theta}$

randomly initialize NN parameter set and get $\eta = \eta_{\text{initial}}$

train a fully-connected NN with inputs $\boldsymbol{X}_i$ and outputs $\{c_{ik}^{\circ}\}_{k=1}^{K_b}$

for $e = 1$ **to** E **do**

I. *forward propagation*
(i) pass $\boldsymbol{X}_i$ through the NN and get $\{\hat{c}_{ik}\}_{k=1}^{K_b}$ for all $i \in n_{\text{train}}$
(ii) calculate $L(\eta) = 1/n_{\text{train}} \sum_{i=1}^{n_{\text{train}}} \sum_{k=1}^{K_b} (\hat{c}_{ik} - c_{ik}^{\circ})^2$
II. *backward propagation* (i) update NN parameter set as $\eta^* = \eta - \varrho \frac{\partial L(\eta)}{\partial \eta}$
III. **if** $e < E$ **then** set $\eta = \eta^*$, and then repeat I. & II.
else return η^*

end

return NN the estimated parameter set $\hat{\eta} = \eta^*$

where $\xi_{ik} = \int \{Y_i(t) - \mu(t)\}\phi_k(t)dt$ is the k-th functional principal component score (FPC score) for $Y_i(t)$ with zero mean and $\text{cov}(\xi_{ik}, \xi_{il}) = \gamma_k 1(k = l)$. Representing functions with eigenfucntions eigen-decomposed from $\text{cov}(Y(t), Y(t'))$ and the corresponding FPC scores can be considered as a special case of basis expansion, where eigenfunctions play the role of basis functions which, however, is unknown and need to be numerically calculated using the observed functional data. Given a desired proportion of variance explained (τ) by the functional principle components (FPCs), $\tilde{Y}_i(t)$ can be approximated arbitrarily well by a finite number of the leading FPCs, where

$$\tilde{Y}_i(t) \approx \sum_{k=1}^{K_\tau} \xi_{ik}\phi_k(t), \tag{7}$$

and K_τ is a valued parameter truncating the FPCs and reducing the dimension of $Y(\cdot)$. K_τ is determined by τ as the smallest integer satisfying $\sum_{k=1}^{K_\tau} \gamma_k \geq \tau$. Because of the fast decay rate of γ_k, K_τ is usually an small integer. In

pratice, FPCA can be easily performed in `R` with the help of some FDA-related packages, e.g., **fda** and **fdapace**. The **fda** package requires the users to firstly smooth the discrete observations $Z(t_{ij})$ and then apply FPCA on the smoothed functional data for $\{\xi_{ik}^{\circ}\}_{k=1}^{K_\tau}$, an estimator of $\{\xi_{ik}\}_{k=1}^{K_\tau}$, while the **fdapace** package is able to estimate $\{\xi_{ik}\}_{k=1}^{K_\tau}$ with observed data $Z(t_{ij})$ directly.

It is clearly shown in Eq.(7) that $\tilde{Y}_i(t)$ can be effectively approximated by a linear combination of the top eigenfunctions $\{\phi_k(t)\}_{k=1}^{K_\tau}$, with the major modes of variations among $\tilde{Y}_i(t)$ captured. Following the idea in Wang & al. [37], under the regression setting, in order to learn the impact of different values of $\boldsymbol{X}_i$ on the variation of $\tilde{Y}_i(t)$, we propose to set the coefficients $\boldsymbol{\xi}_i = \{\xi_{i1}, \xi_{i2}, ..., \xi_{iK_\tau}\}$ to be a function of $\boldsymbol{X}_i$, by formulation, $\boldsymbol{\xi}_i = F(\mathbf{X}_i)$ and consequently we have

$$\tilde{Y}_i(t) = \boldsymbol{\phi}' F(\boldsymbol{X}_i), \tag{8}$$

where $\boldsymbol{\phi}$ is a K_τ-dimensional vector cataloging the K leading FPCs. To model the non-linear relation, similarly, we proposed to set the mapping function $F(\cdot)$ as a dense feed-forward NN, and then we can write

$$\boldsymbol{\xi}_i = \mathrm{NN}_\eta(\boldsymbol{X}_i) = g_L\left(\cdots g_1\left(\sum_{p=1}^{P} w_{1p}X_{ip} + b_1\right)\right), \tag{9}$$

where $g_1, ..., g_L$ are the activation functions at each layer, and η denote the NN parameter set consisting of weights $\{w_{\ell p}\}_{\ell=1}^{L}$ and bias $\{b_\ell\}_{\ell=1}^{L}$ of all hidden layers. Likewise, $\mathrm{NN}(\boldsymbol{X}_i)$ is trained by minimizing the MSE loss function $L_{\boldsymbol{\xi}}(\eta) = 1/n_{\text{train}} \sum_{i=1}^{n_{\text{train}}} \sum_{k=1}^{K_\tau} (\hat{\xi}_{ik} - \xi_{ik})^2$, where n_{train} stands for the number of training set subjects, and the *true* FPC scores ξ_{ik} is replaced by ξ_{ik}°.

Following the same naming convention, we refer to this model has NNSS because it is a NN model build with output being the FPC Scores (S) trained on the Scores (S) themselves. The complete training process of NNSS is summarized in Algorithm 2. Similarly, the returned NN with optimized parameters set $\hat{\eta}$ is used for prediction where it takes the new scalar inputs in the test set and then outputs the predicted basis coefficients $\hat{\boldsymbol{\xi}}$. The predicted functional response $\hat{Y}(t)$ is further recovered as $\hat{Y}(t) = \hat{\mu}(t) + \hat{\boldsymbol{\phi}}'\hat{\boldsymbol{\xi}} = \hat{\mu}(t) + \hat{\boldsymbol{\phi}}'\mathrm{NN}(\boldsymbol{X}_{\text{new}}|\hat{\eta})$.

2.3 Modification to the objective function (NNBR and NNSR)

In the two previous sections, we define two NNs outputting basis coefficients or FPC scores, and those outputs are further used to construct the predicted response variable. Let us discuss a way to build an objective function that directly uses the response variable in order to estimate the parameters. For simplicity, we only discuss the B-spline model of section 2.1 but know that a similar concept can also be applied on the FPCA model describe in section

Algorithm 2: Training NNSS

Input: $\boldsymbol{X}_i = \{X_{i1}, X_{i2}, ..., X_{ip}\}$, $\{Z_i(t_{ij})\}_{j=1}^m$ in the training set n_{train}
Output: NN with optimized parameter set $\hat{\eta}$

set up hyper-parameters, including:
- for performing FPCA: basis functions vector $\boldsymbol{\theta} = [\theta_1(t), ..., \theta_{K_b}(t)]$, number of basis functions K_b, the desired proportion of variance explained τ for determining K_τ
- for training NN: number of hidden layers (L), number of nodes per hidden layer, activation functions ($g_1, ..., g_L$), number of epochs (E), batch size, NN optimizer (with a learning rate τ), etc.

forall $i \in n_{\text{train}}$ **do**
- Estimate the mean function $\mu(t)$ and the covariance function $K(t, t')$ using $\{Z_i(t_{ij})\}_{j=1}^m$
- Perform eigen-decomposition on $\hat{K}(t, t')$ and get $\{\hat{\phi}_k(t)\}_{k=1}^{K_\tau}$ and the corresponding FPC scores $\{\xi_k^\circ\}_{k=1}^{K_\tau}$

end

randomly initialize NN parameter set and get $\eta = \eta_{\text{initial}}$

train a fully-connected NN with inputs $\boldsymbol{X}_i$ and outputs $\{\xi_{ik}^\circ\}_{k=1}^{K_\tau}$

for $e = 1$ **to** E **do**
- I. *forward propagation*
  - (i) pass $\boldsymbol{X}_i$ through the NN and get $\{\hat{\xi}_{ik}\}_{k=1}^{K_\tau}$ for all $i \in n_{\text{train}}$
  - (ii) calculate $L(\eta) = 1/n_{\text{train}} \sum_{i=1}^{n_{\text{train}}} \sum_{k=1}^{K_\tau} (\hat{\xi}_{ik} - \xi_{ik}^\circ)^2$
- II. *backward propagation*
  - (i) update NN parameter set as $\eta^* = \eta - \varrho \frac{\partial L(\eta)}{\partial \eta}$
- III. **if** $e < E$ **then** set $\eta = \eta^*$, and then repeat I. & II.
  else return η^*

end

return NN the estimated parameter set $\hat{\eta} = \eta^*$

2.2. In brief, in section 2.1, we first fit a B-spline model on the observed response variables to estimate a set of basis coefficients for the responses in the training set and then we train a NN to predict these basis coefficients using the predictors.

In this section, we propose to modify the objective function in order to bypass the initial basis coefficient estimation. The key idea of the new objective function is to directly minimize the prediction error of the response variable. In other words, instead of minimizing the MSE between $\hat{c}_{ik}$ and c_{ik}, here we first transform the NN output, the B-splines basis coefficients, into the predicted response $\hat{Y}_i(t)$ and then minimizes the MSE between $\hat{Y}_i(t)$ and $Y_i(t)$. What supports us to implement such objective function is the fact that we rely on differentiable operations to build this new objective function. Doing so

guarantees that we can rely on the back-propagation algorithm to train the NN using readily available packages.

Suppose that we want to fit the functional response with a B-spline made of K_b basis functions $\boldsymbol{\theta}$ and K_b associated basis coefficients $\boldsymbol{C}$. Then the predict response $\hat{Y}_i(t)$ at time t is the vector product between $\boldsymbol{\theta}$ and $\boldsymbol{C}$ as illustrated in Eq. (3). This means that the relation between the predicted response $\hat{Y}_i(t)$ and the predicted basis coefficient $\hat{\boldsymbol{C}}$ is linear, thus we can easily compute the derivative of $\hat{Y}_i(t)$ with respect to the coefficients. This means that if we observe the response at time t then we are able to compute the gradient of $(Y_i(t) - \hat{Y}_i(t))^2$ with respect to the basis coefficients and therefore we can also compute the gradient with respect to the parameters η of the NN that outputs those basis coefficients.

More generally, assuming the response is observed at m time points, $j_1, j_2, ..., j_m$ for every observation, then we generate a matrix of basis functions evaluated at those time points, denoted as Θ, which is a K_b by m matrix where each row represents a basis function with each entry taking the values of that basis function at each of the m time points. Consequently, we can obtain the predicted response $\hat{Y}_{(n\times m)}$ for every m observed time points and for all n observations by doing a simple matrix multiplication

$$\hat{Y}_{(n\times m)} = \mathbf{C}_{(n\times K_b)}\Theta_{(K_b\times m)}, \tag{10}$$

where $\mathbf{C}$ is a matrix where a row contains the K_b coefficients for a single observation. In our proposed model, $\mathbf{C}$ is the output produce by our NN function as explained in section 2.1. However, we now modify the objective function to train the NN to minimizes the MSE between the observed response and the predicted one

$$L_{\boldsymbol{Y}}(\eta) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \sum_{j=1}^{m} (Y_i(t_{ij}) - \hat{Y}_i(t_{ij}))^2. \tag{11}$$

Note that the discrete observation $Z_i(t_{ij}) = Y_i(t_{ij}) = \epsilon_i(t_{ij})$ will replace the underlying *true* functional curve $Y_i(t_{ij})$ in implementation. In this way, we entirely bypass the need to estimate the basis coefficient first and fit the parameters η with respected to the response directly. We call this variant NNBR because its architecture is a Neural Network (NN) with Basis coefficients output (B) fitted by minimizing the MSE of the Response variable (R). The entire training process of NNBR is detailed in Algorithm 3. Similarly, we can get a prediction for a new input $\boldsymbol{X}_{\text{new}}$ as $\hat{Y}(t) = \boldsymbol{\theta}'\hat{\boldsymbol{C}} = \boldsymbol{\theta}'\text{NN}(\boldsymbol{X}_{\text{new}}|\hat{\eta})$.

A similar process could be conducted with a NN that outputs FPC scores as described in section 2.2, where this time it predicts response $\hat{Y}$ using those scores. We named that model NNSR and describe the training process in Algorithm 4.

Algorithm 3: Training NNBR

Input: $\boldsymbol{X}_i = \{X_{i1}, X_{i2}, ..., X_{ip}\}$, $\{Z_i(t_{ij})\}_{j=1}^m$ in the training set n_{train}
Output: NN with optimized parameter set $\hat{\eta}$

set up hyper-parameters, including:
- for smoothing $Z_i(t_{ij})$: basis functions vector $\boldsymbol{\theta} = [\theta_1(t), ..., \theta_{K_b}(t)]$, number of basis functions K_b
- for training NN: number of hidden layers (L), number of nodes per hidden layer, activation functions $(g_1, ..., g_L)$, number of epochs (E), batch size, NN optimizer (with a learning rate ϱ), etc.

Evaluate $[\theta_1(t), ..., \theta_K(t)]$ at all observed timestamps $\{t_j\}_{j=1}^m$ and form the matrix $\Theta_{(K_b \times m)}$

randomly initialize NN parameter set and get $\eta = \eta_{\text{initial}}$

train a fully-connected NN with inputs $\boldsymbol{X}_i$ and outputs $\{Z_i(t_{ij})\}_{j=1}^m$

for $e = 1$ **to** E **do**

I. *forward propagation*

(i) pass $\boldsymbol{X}_i$ through the NN and get $\{\hat{c}_{ik}\}_{k=1}^{K_b}$ for all $i \in n_{\text{train}}$

(ii) multiply $\{\hat{c}_{ik}\}_{k=1}^{K_b}$ with $\Theta_{(K_b \times m)}$ to get $\{\hat{Y}(t_{ij})\}_{j=1}^m$

(iii) calculate $L(\eta) = 1/n_{\text{train}} \sum_{i=1}^{n_{\text{train}}} \sum_{j=1}^m (Z_i(ij) - \hat{Y}_i(t_{ij}))^2$

II. *backward propagation*

(i) update NN parameter set as $\eta^* = \eta - \varrho \frac{\partial L(\eta)}{\partial \eta}$

III. **if** $e < E$ **then** set $\eta = \eta^*$, and then repeat I. & II.
else return η^*

end

return NN the estimated parameter set $\hat{\eta} = \eta^*$

2.4 Irregularly-spaced functional data

Earlier in this section we made the common assumption that the functional response $Y(t)$ is observed at the same m equally-spaced time points. While this was a useful assumption to make in order to explain how to train the model, it is actually not a necessary condition to fit any of the four models described above and we can train these models even with irregularly spaced functional response.

For the first two models we introduced, NNBB and NNSS, in order to train the NN, we simply need an estimate for the basis coefficients or an estimate for the FPC scores. We can rely on some existing FDA literature to get those estimates [24, 38] for irregularly-spaced functional data.

For both models that utilize the modified objective function, NNBR and NNSR, it is a bit more complicated but not so much. Once again, let us focus on NNBR to simplify the explanations. The goal is to train the model using only the observed time points, and to achieve this we generate the matrix Θ

Algorithm 4: Training NNSR

Input: $\boldsymbol{X}_i = \{X_{i1}, X_{i2}, ..., X_{ip}\}$, $\{Z_i(t_{ij})\}_{j=1}^m$ in the training set n_{train}
Output: NN with optimized parameter set $\hat{\eta}$

set up hyper-parameters, including:
- for performing FPCA: basis functions vector $\boldsymbol{\theta} = [\theta_1(t), ..., \theta_{K_b}(t)]$, number of basis functions K_b, the desired proportion of variance explained τ for determining K_τ
- for training NN: number of hidden layers (L), number of nodes per hidden layer, activation functions $(g_1, ..., g_L)$, number of epochs (E), batch size, NN optimizer (with a learning rate τ), etc.

forall $i \in n_{\text{train}}$ **do**
- Estimate the mean function $\mu(t)$ and the covariance function $K(t, t')$ using $\{Z_i(t_{ij})\}_{j=1}^m$
- Perform eigen-decomposition on $\hat{K}(t, t')$ and get $\{\hat{\phi}_k(t)\}_{k=1}^{K_\tau}$
- Form $\hat{\Phi}_{(K\times m)}$, a matrix of eigenfunctions $[\hat{\phi}_1(t), ..., \hat{\phi}_{K_\tau}(t)]$ evaluated at all observed time points $\{t_j\}_{j=1}^m$

end

randomly initialize NN parameter set and get $\eta = \eta_{\text{initial}}$

train a fully-connected NN with inputs $\boldsymbol{X}_i$ and outputs $\{Z_i(t_{ij})\}_{j=1}^m$

for $e = 1$ **to** E **do**
- I. *forward propagation*
  - (i) pass $\boldsymbol{X}_i$ through the NN and get $\{\hat{\xi}_{ik}\}_{k=1}^K$ for all $i \in n_{\text{train}}$
  - (ii) multiply $\{\hat{\xi}_{ik}\}_{k=1}^{K_\tau}$ with $\hat{\Phi}_{(K_\tau\times m)}$ to get $\{\hat{Y}(t_{ij})\}_{j=1}^m$
  - (iii) calculate $L(\eta) = 1/n_{\text{train}} \sum_{i=1}^{n_{\text{train}}} \sum_{j=1}^m (Z_i(ij) - \hat{Y}_i(t_{ij}))^2$
- II. *backward propagation*
  - (i) update NN parameter set as $\eta^* = \eta - \varrho \frac{\partial L(\eta)}{\partial \eta}$
- III. **if** $e < E$ **then** set $\eta = \eta^*$, and then repeat I. & II.
  else return η^*

end

return NN the estimated parameter set $\hat{\eta} = \eta^*$

containing all K chosen basis functions evaluated at all time points with at least one observation within the training set. When computing the objective function, we simply eliminate the contribution of those unobserved subject-time point pairs by multiplying them by 0:

$$L_{\boldsymbol{Y}_{\text{irr}}}(\eta) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \sum_{j=1}^{m_i} \left(Y_i(t_{ij}) - hatY_i(t_{ij})\right)^2 \cdot \mathbb{1}\left(Y_i(t_{ij}) \text{ is observed}\right), \tag{12}$$

which assures that the unobserved subject-time point pairs do not contribute to the gradient of the objective function.

Because of the common strategy used in FDA, which represents the infinite-dimensional curve using a finite basis set, we are able to construct the curve over the entire interval $\mathcal{T}$. This allows us to compute the prediction accuracy even if the test set contains observations at time points previously unseen, as long as they are within the interval of time points observed in the training set.

2.5 Roughness penalty

When predicting the functional response using neural networks, we need to pay attention not only to the prediction accuracy but also the smoothness of the predicted trajectories, as it is standard in FDA. In practice, we observe that the performance of the neural network mapping with basis expansion is influenced by the number of basis functions selected. Usually, the more basis functions we use to re-express the functional response, the higher accuracy cane be when predicting the response at a set of given timestamps.

In FDA, it is common to set the number of basis functions to be less than the number of the observed time/location points with enough valid observations. However, in the cases where the observed timestamps are limited, we can consider increasing the number of basis functions to benefit the prediction accuracy. This benefit brought by more basis functions comes at the sacrifice of smoothness of the fitted functional trajectories, as they introduce in more variations. This is to be expected; because the prediction response for a specific time point $\hat{Y}_i(t)$ is the inner product of the basis coefficients and basis functions that are non-zero at t, then the more non-zero basis functions we have the more flexible that point estimate is. To control the smoothness of fitted curves without limiting the number of basis functions, we follow the common idea in FDA and propose to add some classic roughness penalty to the objective function of the neural network. This simple addition ensures the smoothness of the predicted functional curves without putting any burden to the differentialable operation (since the roughness penalty terms we consider also are linearly related the NN outputs).

There are multiple types of roughness penalties that can be applied to smooth the functional curves. In our implementation, we focus on two conventional and popular approaches: penalizing with the second derivative of the function, and penalizing directly the basis coefficients, to relieve the roughness concern with the NN-fitted functional response. Notice that we do not need to define roughness penalties for NNBB and NNSS as the smoothing is done when fitting the curve a priori.

2.5.1 Penalize the second-order derivative of $Y(t)$

When the smoothness of fit becomes a concern and the roughness penalty turns out to be a necessity, the most common and popular method would be penalizing the i-th order derivative of the function $\hat{Y}(t)$. The squared second

derivative of a function $\hat{Y}(t)$ at t reveals its curvature of at t, therefore it is natural to measure the roughness of the $\hat{Y}(t)$ by taking the integrated square of its second derivative [24]. Adding this penalty term to the objective function of the NN leads to

$$L_{\boldsymbol{pen}}(\eta) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \left(\sum_{j=1}^{m} \left(Y_i(t_{ij}) - \hat{Y}_i(t_{ij}) \right)^2 + \lambda \int_{\mathcal{T}} \left(\frac{d^2 \hat{Y}(t)}{dt^2} \right)^2 dt \right). \tag{13}$$

The parameter λ acts as a smoothness controller for balancing the fit of the data and the variability of the predicted function. The selection of the optimal λ can be achieved by using cross validation, where either the subjects or the time points are randomly divided into sub-samples (In practice, we partition the time points).

Unfortunately, we cannot back-propagate the gradient through the integral of Eq. (13), and thus we will approximate the integral with a summation over the domain $\mathcal{T}$. Because we are able to generate $\hat{Y}_i(t)$, we can approximate this integral with as many points as we deemed necessary:

$$L_{\boldsymbol{pen}}(\eta) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \left(\sum_{j=1}^{m} (Y_i(t_{ij}) - \hat{Y}_i(t_{ij}))^2 + \frac{\lambda T}{J-1} \sum_{j=2}^{J} \left(\frac{d^2 \hat{Y}_i(t_{ij})}{dt_{ij}^2} \right)^2 \right), \tag{14}$$

where $t_{ij} = t_j$ for all i and $\{t_j\}_{j=1}^{J}$ are equally spaced time points covering the entire domain $\mathcal{T}$ with length T . Notice that we do not need to actually compute those second order derivatives, because:

$$\begin{aligned} \hat{Y}_i(t) &= \sum_{k=1}^{K} \hat{c}_{ik} \theta_k(t) \\ \Rightarrow \frac{d^2 \hat{Y}_i(t)}{dt^2} &= \sum_{k=1}^{K} \hat{c}_{ik} \frac{d^2 \theta_k(t)}{dt^2}, \end{aligned} \tag{15}$$

then what we really need are the second order derivatives of the basis functions, which are much easier to compute and readily available in the fda package [23]. Therefore, we can back-propagate the gradient of the objective function of Eq. (14) with respect to η without any problems.

2.5.2 Penalize the basis coefficients C_i

Penalizing directly on the basis coefficients was firstly introduced by Eilers & Marx [11] when adjacent B-splines is used to re-express the functional variable in a regression problem. Compared to penalizing the derivative, this idea

reduces the dimensionality of the smoothing problem to K, the number of the basis functions, instead of n, the number of observations ([11]).

The basic structure of this penalty is the difference of a set of consecutive basis coefficients $\Delta^2 c_k = c_k - 2c_{k-1} + c_{k-2}$. This difference has a strong connection with the second derivative of the fitted function, which is revealed by the simple formula for derivatives of B-splines given by De Boor [9]. The summation of the squared differences for all $c_k, k = 1, ..., K$ would be the main component of the penalty term, with the strength of the further controlled by a tuning parameter λ. Like previously, the penalty term is added to the objective function of NN as:

$$L_{\boldsymbol{pen}}(\eta) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \sum_{j=1}^{m} \left(Y_i(t_{ij}) - \hat{Y}_i(t_{ij}) \right)^2 + \lambda \sum_{k=3}^{K} (\Delta c_k)^2. \tag{16}$$

Similarly, the optimal hyperparameter λ is selected using k-folds cross validation.

Empirically, when fitting one of the B-spline models, especially the NNBR model, including a large number of basis functions, more than m for example, lead to a better prediction accuracy, but also lead to a very rough predicted curve. When $K > m$ the predicted curve quickly becomes *wiggly*, and thus we would increase the number of basis such that $K >> m$ and impose a roughness penalty at the same time to keep the curve smooth. In some cases, this lead to the top performer in terms of prediction accuracy. Unfortunately, using $K >> m$ and a roughness penalty could make the complicated hyperparameter tuning process even more difficult. Nonetheless, we believe it is important to provide roughness penalties and justifications for those when providing a technique to fit functional data.

We implemented the four models and the two roughness penalties in R [22]. The functional component of our implementation relies on the fda package [23] and the NN component uses the R implementation of Keras [7]. Our implementation of those models along the real data example are available online on the second author's GitHub page.

3 Computational complexity

An advantage of using a NN as the link function instead of a linear combination is that it might have a lower computational cost. More specifically, when using a standard training procedure for both models, the NN approach scales better with the number of predictors. Indeed, we usually find an exact solution for the FoS model which involves the inversion of a matrix, resulting in worse scaling with respect to the number of predictors. On the flip side, the algorithm used to train NN scales linearly with the number of predictors.

Start by looking at the FoS model, we rely on the formulation established in Ramsay & Silverman [24], chapter 14. This book contains various information about the computational details of the solutions of the FoS optimization

problem under various parameterizations. However, its simplest form is quite similar to the least square solution of a traditional regression where:

$$\hat{\mathbf{B}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y, \tag{17}$$

which involves the inversion of a $p \times p$ matrix, which requires a number of operations proportional to p^3, say $O(p^3)$ in big O notation, using Gaussian elimination. The Strassen algorithm [30] manages to get that order down to $O(p^{2.8})$ and the more recent Coppersmith-Winograd algorithm [8] getting the order of matrix multiplication down to $O(p^{2.37})$ for a matrix of dimension p by p.

The fda package [23] uses the `solve()` function included in the default `R` language [22] which in turn employs the Linear Algebra PACKage (LAPACK) [1] that relies one Gaussian elimination to inverse matrices. Other than the matrix inversion, the matrix multiplication component of the solution involves a $p \times p$ matrix, a $p \times n$ matrix and a $n \times m$ matrix resulting in a number of operations of order $O(pn)$. Including the matrix multiplication part of the solution the whole FoS estimation requires a number of operations of order $O(p^3 n)$. The more complete solutions that consider the smoothness of the predicted curves involve the inversion of a matrix of dimension Kp by Kp where K is the number of basis of the functional parameters $\beta(t)$. It means scaling polynomial with power 3 with the number of predictors p when using Gaussian elimination. Hence, we consider that the number of operations needed to obtain the exact solution to the function-on-scalar problem requires a number of operations of order $O(p^3 n)$.

As far as NNs are concerned, according to Hastie & al. [15], the computational cost of training a NN is of order $O(npwe)$ when back-propagation is adopted [28] to estimate the gradient and when a gradient-based approach is applied to fit the NN. In the formulation above, n is the number of observations, p is the number of predictors, w is the number of weights (hidden nodes) in the NN and e the number of training epochs.

Therefore, if we strictly focus on how a NN run time scales with n and p, we are looking at a linear scaling in both cases, $O(pn)$. Consequently, when comparing computational costs, NN model has a pretty significant advantage when it comes to its scaling with respect to the number of predictors. Because of this better scaling with respect to the number of predictors, we can claim that our proposed models are better equipped than the FoS regression to deal with data sets containing lots of predictors.

However, let us be a little more nuanced. It is clear that under some circumstances we can fit extremely large NNs which could lead to a slow fitting process. Additionally, the main difference between the traditional ways to fit these models is that we pursue an exact solution for the linear problem and on the contrary we utilize a gradient-based approach to find a solution in the case of NNs. As a result, for a linear model, such as FoS, we do have worse scaling with respect to p but we have guarantees that the optimal solution is reached. In theory, it would be possible to fit a linear model, such as FoS, with

a gradient-based approach to improve how its computational cost scales with respect to p, though this is certainly not the standard optimization approach.

4 Real data application

We evaluate the prediction performance of the proposed methods, along with the conventional FoS model and FAM model on the age-specific fertility rates (ASFR) data set introduced in section 1. For each of the models, we proceed with 20 repetitions of random subsampling validation: randomly dividing the data set into a training set and a test set, with 80% and 20% of the total samples assigned to them, respectively. Hyper-parameters of all models (except FAM) in comparison are firstly tuned using 5-fold cross validation in order to fairly improve their performance during actual training. To reduce the computational time consumed by the tuning processes of the NN-based models, the number of basis functions selected by FoS model is consistently used by the rest of the models. 99% proportion of variance explained is recommended by cross-validation as the threshold to truncate the FPCs scores in NNSS and NNSR. For NNBR with penalized objective function (NNBR(P)), we apply 10 basis functions and the second-order derivative roughness penalty, together with smoothing parameter $\lambda = 10^{-7}$, which is similarly suggested by the 5-fold cross-validation. The FoS model used in this experiment is coming from the R-package **fda**, and FAM model is trained with the help of function `pffr()` in the **refund** package.

Table 1 Table of MSE for various models in test set for ASFR data set.

Methods	Mean	Std. Dev.	p-value of t-test
FoS	1177.98	513.58	-
NNBB	1031.87	294.87	0.18
NNSS	**992.50**	296.68	**0.01**
NNBR	1060.50	210.63	0.20
NNBR(P)	1059.07	**208.27**	0.19
NNSR	1047.46	277.37	0.11
FAM	1317.33	919.62	0.28

Table 1 shows the prediction performance of each model using the mean squared prediction error over the 20 replications. The prediction accuracy is measured by the mean squared prediction error (MSE_p) averaged across the number of samples and the number of observed time points in the test set. A two-sided paired t-test is later conducted to compare the MSE_p of the 20 replicates of our models to the 20 MSE_p of the FoS model. It is clearly shown that our proposed models consistently outperformed FoS and FAM models in predicting the fertility curves, with both smaller means and standard deviations of the prediction errors of 20 random test sets, demonstrating their advantages in capturing both linear and nonlinear effects of all predictors on

the functional response. NNSS model, as indicated in table 1, has the best prediction performance, which interestingly implies that the FPC scores are more informative than the basis coefficients in representing $Y(t)$ with this data set, and consequently the effect of covariates on functional response are better described by the relation between the predictors and FPC scores instead of basis coefficients. In addition, the prediction error of NNBR optimized by penalized objective function is lower than the non-penalized NNBR, indicating that including a relatively large number of basis functions, along with a roughness penalty term added to the objective function, can help improve the prediction without loss on the smoothness of the predicted curve.

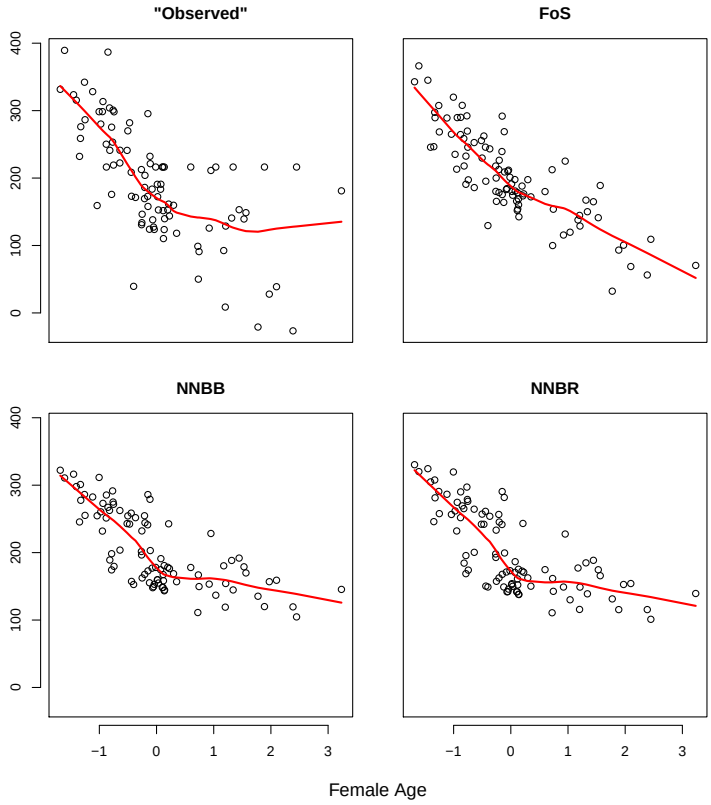

Fig. 3 A comparison of relations between covariate female age and the second $Y(t)$-estimated basis coefficients ($\hat{c}_{2,Y(t)}$), FoS-predicted basis coefficients ($\hat{c}_{2,\mathrm{FoS}}$), NNBB-predicted basis coefficients ($\hat{c}_{2,\mathrm{NNBB}}$) and NNBR-predicted basis coefficients ($\hat{c}_{2,\mathrm{NNBR}}$), respectively.

As previously mentioned in section 1, several covariates, such as female age and GDP per capita, are seemingly not linearly related to many of the basis coefficients representing the fertility trajectory. Therefore we focus on the basis coefficients $\hat{\boldsymbol{C}}_{\mathrm{FoS}}$, $\hat{\boldsymbol{C}}_{\mathrm{NNBB}}$ and $\hat{\boldsymbol{C}}_{\mathrm{MMBR}}$ predicted by FoS, NNBB

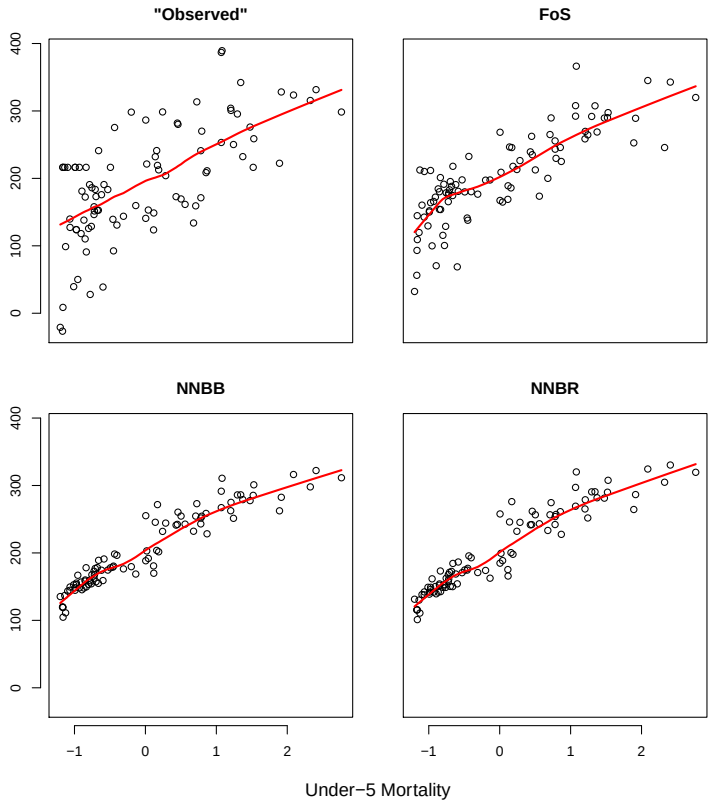

Fig. 4 A comparison of relations between covariate Under-5 Mortality and the second $Y(t)$-estimated basis coefficients ($\hat{c}_{2,Y(t)}$), FoS-predicted basis coefficients ($\hat{c}_{2,\text{FoS}}$), NNBB-predicted basis coefficients ($\hat{c}_{2,\text{NNBB}}$) and NNBR-predicted basis coefficients ($\hat{c}_{2,\text{NNBR}}$), respectively.

and NNBR model (they are the models using basis coefficients for regression) respectively. The relations between predictor female age and the second basis coefficient, including $Y(t)$-estimated second basis coefficient $\hat{c}_{2,Y(t)}$ and model-predicted ones, are displayed in Figure 3. We can see that the nonlinear pattern for female age and the second basis coefficient is better recovered by NNBB and NNBR, while the FoS-predicted basis coefficient $\hat{c}_{2,\text{FoS}}$ is more likely linearly related to female age. In contract, FoS performs the best in capturing the nonlinear relation between female age and the first, the third and the fourth basis coefficient. Likewise, we select under-5 mortality out of those covariates that have highly likely linear associations with some of the basis coefficients, and Figure 4 reveals how under-5 mortality relates to different model-predict second basis coefficient $\hat{c}_2$. It is noticed there are plainly linear trends between under-5 mortality and the second basis coefficients predicted by each of the three models mentioned, while under-5 mortality has been shown linearly related to the second basis coefficient estimated by $Y(t)$.

It is not surprising that FoS reconstructs the most similar pattern to the one built on $Y(t)$-estimated basis coefficients, but the proposed models also master the linear information when recovering the associations, which proves that the NN-based models have the ability to take good care of both the non-linear and linear relations between the predictors and scalar representation simultaneously. Due to the relatively small number of observations, it is worth mentioning that the relation between covariates and basis coefficients may not be fully revealed and visual misunderstanding could occur.

5 Simulation studies

5.1 Generating data

We are interested in data sets that have multiple predictors and a potentially non-linear relation between the scalar predictors and the functional response. We were inspired by the variable selection literature to design our data generators. More precisely, our data generator is similar than those proposed by Wang & al. [35] and Barber & al. [2]. The basic concept is to build the functional response using a linear combination of a set of K random curves $\psi_k(t)$ and K associated coefficients $\zeta_k(\mathbf{X})$:

$$Y(t) = \sum_{k=1}^{K} \zeta_k(\mathbf{X})\phi_k(t). \tag{18}$$

and we are going to explore multiple approaches to generate the random curves $\psi_k(t)$ and multiple ways to use the predictors $\mathbf{X}$ to build coefficients ζ.

Firstly, we concentrate on how to generate the curves $\psi_k(t)$. Wang & al. [35] employ spline functions constructed with B-splines. Due to the requirement of their simulation, they use rather simple order 4 B-splines with 1 interior knots corresponding to 5 basis functions. To create a simple scenario for visualization purpose, we consider a configuration where a random curve $\psi_k(t)$'s is set to be a single B-spline basis function $B_k(t)$:

$$\psi_k(t) = B_k(t). \tag{19}$$

Here, the applied B-spline basis system is of order of 4, while the number of interior knots are calculated using the number of predictors (# of predictors -4). The total number of basis functions, in this scenario, is equivalent to the number of scalar covariates. The main purpose of this configuration is to visualize if models are able to recover the coefficient ζ_k. Because in this example, a single covariates only affect a single B-spline basis function, we can fix $\zeta_k(\mathbf{X})$ to be a non-linear function of the coefficients $\mathbf{X}$ and to see if the trained model captured that non-linear effect. We notably use that configuration for our **Design 1**.

The second configuration is a somewhat more realistic case where the covariates affect the response over the entire time interval. In this configuration, each random curve $\psi_k(t)$ will be built using B-splines of order 4 with

9 interiors knots corresponding to 13 basis function $B_l(t)$. The 13 associated coefficients β_l are randomly generated from a Normal distribution. Thus, for the B-splines configuration we have:

$$\psi_k(t) = \sum_{l=1}^{13} \beta_{k,l} B_l(t). \tag{20}$$

This configuration is used in our **Design 2**.

In terms of the coefficients $\zeta_k(\mathbf{X})$ we will look at three configurations. Because NNs are known to capture non-linear relationships, it is important to design non-linear functions ζ_k. First, for visualization, we apply a polynomial function to the continuous predictors. Specifically, a customized proportion of continuous covariates, denoted by $\mathbf{X}_{\text{poly}}$, are randomly selected and then transformed by a polynomial function with either second or third degree (half of the selected covariates are processed through the quadratic function and the rest by the cubic function), and accordingly $\zeta_k(\mathbf{X}) = \text{polynomial}(X_k)$ for $X_k \in \mathbf{X}_{\text{poly}}$, otherwise $\zeta_k(\mathbf{X}) = X_k$. This configuration is used for a visualization example; **Design 1**. Our second configuration is also non-linear but this time it is a bit more complex. We define a 3-hidden-layer NN with sigmoid (which is non-linear) activation functions and random weights. For this configuration, $\zeta_k(\mathbf{X})$ is the k-th output of a NN taking $\mathbf{X}$ as input. These coefficients are used for **Design 3**. Finally, we consider the case where $Y(t)$ is a linear combination of curves with $\mathbf{X}$ being the coefficients directly. In other words, the k-th coefficient is simply the k-th predictor: $\zeta_k(\mathbf{X}) = X_k$. This is a scenario where we expect the FoS to outperform the models we propose, but we want to visit this example regardless. We call this the linear configuration and use it for **Design 4**.

Finally, we add a random noise, which is Normally distributed with zero mean and a variance of 2 to every data points $Y_i(t)$.

5.2 Results

Four different simulations designs are considered to illustrate the advantages of the proposed methods. For each design, we generated data sets of 2000 observations and randomly sampled 1800 training observations and 200 testing observations. This random subsampling procedure was repeated 20 times. We opted to experiment with approximately 1/3 categorical (half of them are ordinary) and 2/3 continuous predictors, along with different amount of predictors, either 10 or 20, different random curve configurations and different relations between the coefficients and predictors, either linear or non-linear, all with some random noise. To mimic the real-world scenario, the actual observations for $Y(t)$ are simulated discretely at 40 equally spaced time points $\{t_j\}_{j=1}^{40}, t \in [0, 1]$. Similar to the real application, we compare the proposed NN-based methods, including NNBB, NNSS, NNBR and NNSR, to the novel FoS model (We exclude FAMM because its training time is tremendously high and based on the results of the initial replicates, it performs poorly compared

to the other models). The predictive accuracy was evaluated using the MSE_p on the test set. We also report the p-value of the two-sided paired t-test of the MSE_p of our methods to that of the FoS.

Design 1 (nonlinear scenario): We generate 20 random predictors ($K = 20$), where X_k are i.i.d. uniform random variables from $[a, b]$ with $a \in \{-4, -3, -2, -1, 0\}, b \in \{3, 4, 5, 6, 7\}$ for all k. We apply the polynomial transformation to 50% of the continuous variables by setting $\zeta_k(X) = \text{polynomical}(X_k)$ with degree of 2 and 3 for $k = 8, 10, 12, 13, 14$ and $k = 1, 3, 4, 7, 9$, respectively, and $\zeta_k(X) = X_k$ otherwise. The random curves $\psi_k(t)$, in this case, is set to be the B-spline basis functions as $\psi_k = B_k(t)$ for all k, and accordingly the functional response is simulated as $Y(t) = \sum_{k=1}^{20} \zeta_k(\mathbf{X}) B_k(t)$.

Methods	FoS	NNBB	NNSS	NNBR	NNSR
Mean	49.20	5.14	5.98	4.95	5.91
Std. Dev.	1.64	4.11	0.23	0.11	0.19
p-value	-	<2.2e-16	<2.2e-16	<2.2e-16	<2.2e-16

Table 2 Table of MSE for various models when the data is generated using a direct combination of B-spline and polynomial-transformed covariates to construct the response variable.

Table 2 summarizes the mean and SD of the MSE_p on the same testing observations achieved by different models for Design 1. In the nonlinear scenario, we observe that the prediction performances of all the NN-based models surpass that of the FoS Model under 1% significance level. Two proposed models using basis coefficients as the output of NN exhibit the best performances in predicting the response curve, with approximate 3.5 times less on the means of MSE_p, along with the smaller standard deviations of MSE_p compared to those of the NN-based models outputting FPC scores. This is expected because the functional response in this design setting is generated as a directly linear combination of the polynomial-transformed predictors and B-spline basis functions, and given this special setting, we are able to visualize the relation as some describable patterns for some selected coefficient-predictor pairs. This design highlights a situation where using FoS would be extremely problematic and where any of our proposed NN models would provide a significant improvement.

Next, we compare NNBB, NNBR with FoS for their abilities in recovering the underlying relationship between different pair of basis coefficient and predictor. The relations between predictors and basis coefficients achieved by each model are visualized through the scatter-plots of a basis coefficient against a predictor. Remember that we designed this simulation specifically for to visualize how NN models learn $\boldsymbol{C} = \text{NN}_\eta(\boldsymbol{X})$ and compare it to the $\zeta_k(\mathbf{X})$ we designed.

We randomly select X_{12}, one of the polynomial-transformed covariates, and c_{12}, the basis coefficient corresponding to the 12-th B-spline basis function,

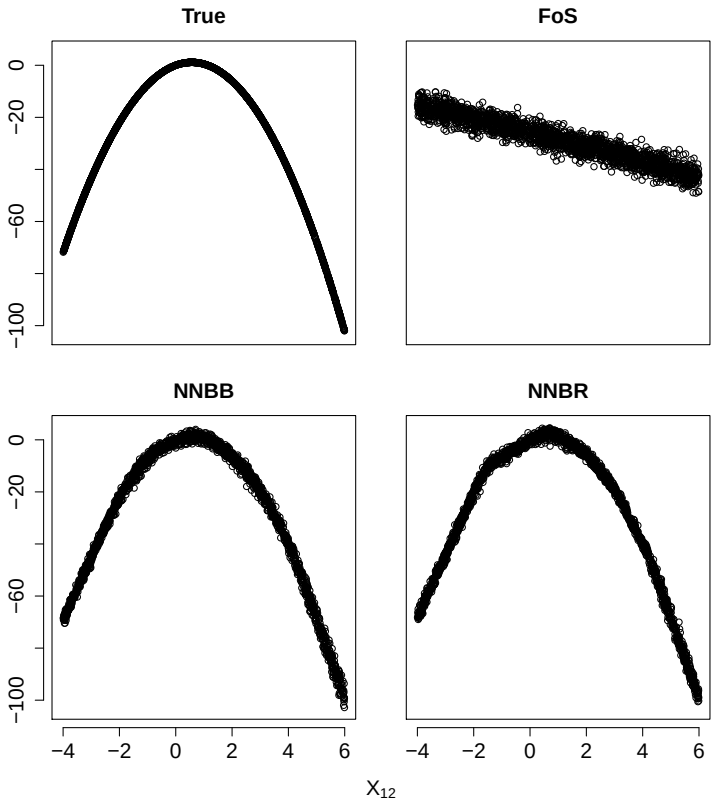

Fig. 5 Scatter plots of the true c_{12} (ζ_{12} as per the generator), FoS-predicted $\hat{c}_{12,\text{FoS}}$, NNBB-predicted $\hat{c}_{12,\text{NNBB}}$, and NNBR-predicted $\hat{c}_{12,\text{NNBR}}$ against X_{12}, from left to right respectively.

and show their true relation, as $\zeta_{12}(X_{12})$, together with the relation reconstructed by the three mentioned approaches using scatter plots of true c_{12} and $\hat{c}_{12}$ predicted by each of the model against X_{12} in Figure 5. Undoubtedly, NNBR and NNBB both precisely capture the $\cap$-shape between the true c_{12} and X_{12}, while FoS replace the nonlinear trend with a downwards linear pattern. Likewise, the associations between X_5 and c_5's by multiple models are revealed in Figure 6, but differently, the true relation for this X_5-c_5 pair is linear. We can observe, NNBB and NNBR perform similarly as FoS in retrieving the linear patterns but with comparably thicker bandwidth, indicating that our models can also successfully detect the linear relation for some coefficient-predictor pairs but with higher variance. Figure 5 and 6 highlight the utility of our methods in recovering the true underlying relation, especially the nonlinear relation, between the basis coefficients and predictors, and the success in learning the true relationships contributes to their fabulous performances on predicting the response trajectories. The ability to recover the true underlying nonlinear relation is the main benefit of our proposed approaches.

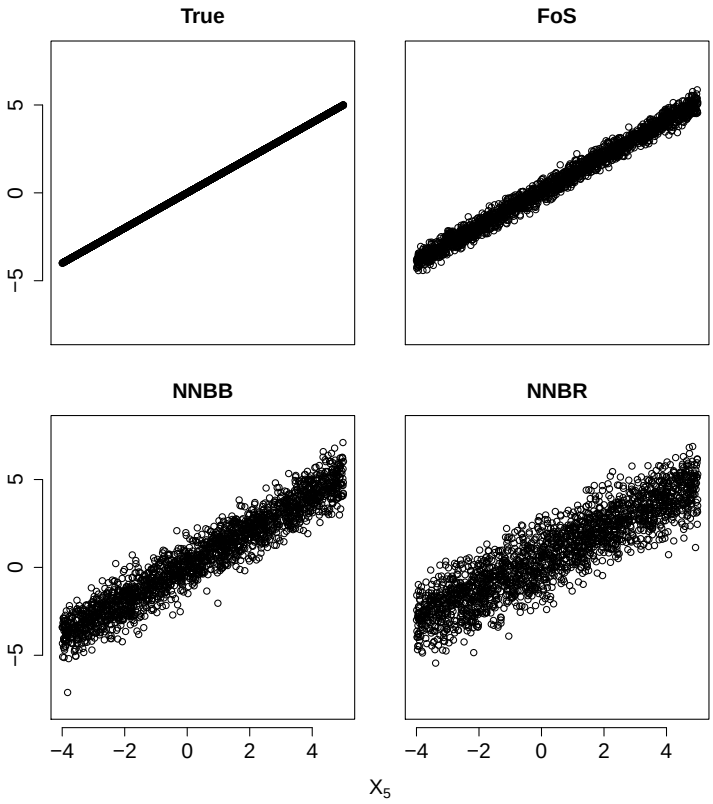

Fig. 6 Scatter plots of the true c_5 (ζ_5 as per the generator), FoS-predicted $\hat{c}_{5,\text{FoS}}$, NNBB-predicted $\hat{c}_{5,\text{NNBB}}$, and NNBR-predicted $\hat{c}_{5,\text{NNBR}}$ against X_5, from left to right respectively.

Design 2 (nonlinear scenario): 20 random predictors ($K = 20$) are generated, with X_k being binary variables for $k = 1, 3, 5, 7$, eight-level categorical variables for $k = 2, 4, 6$, and i.i.d. uniform random variables from $[a, b]$ with $a \in \{-4, -3, -2, -1, 0\}, b \in \{3, 4, 5, 6, 7\}$ for $k \geq 8$. Likewise, approximate 50% of the continuous variables are later transformed by a polynomial function with different degrees, where $\zeta_k(X) = \text{polynomical}(X_k)$ with the second and the third degree for $k = 14, 16, 17, 18$ and $k = 8, 10, 19, 20$, separately, and $\zeta_k(X) = X_k$ for the remainder. Then we construct the response curve as $Y(t) = \sum_{k=1}^{20} \zeta_k(\mathbf{X})\psi_k(t)$, coupled with $\psi_k(t) = \sum_{l=1}^{13} \beta_{k,l} B_l(t)$, where $\{B_l(t)\}_{l=1}^{13}$ are the B-spline basis functions with order 4 and $\beta_{k,l}$ are i.i.d. random variable following normal distribution $\mathcal{N}(0, 4)$.

The mean and SD of the MSE_p's on the testing observations for all methods in comparison can be found in Table 3. We observe that all our proposed methods remain superior to the FoS method in this setting. The NN-based models with basis-coefficient output continue to be the top performer, especially the NNBR model trained using the response variable directly. The performance of

Methods	FoS	NNBB	NNSS	NNBR	NNSR
Mean	4559.20	38.33	286.23	36.38	265.50
Std. Dev.	159.01	9.18	79.68	18.76	16.05
p-value	-	<2.2e-16	<2.2e-16	<2.2e-16	<2.2e-16

Table 3 Table of MSE for various models when the data is generated using a combination of B-spline-based random curves and polynomial-transformed covariates to construct the response variable.

the FoS is still significantly lower than the NN models and still has higher variance. For both of these designs, some coefficient functions ζ_k have significant non-linear curvature being polynomial function of degree 2 or 3. Consequently, the dominance of the NN model was expected, even now with a more realistic scenario where a predictor affects the response on its entire domain $\mathcal{T}$.

Design 3 (nonlinear scenario): We continue to set $K = 20$, and X_k is generated as binary variables, eight-level categorical variables and i.i.d. uniform random variables from $[a, b]$ with $a \in \{-4, -3, -2, -1, 0\}, b \in \{3, 4, 5, 6, 7\}$ for different sets of k (same as Design 2). We trigger the nonlinear configuration by passing the predictors $\mathbf{X}$ through the 3-hidden-layer NN introduced in the section 5.1, resulting in $\zeta(\mathbf{X}) = \text{NN}(\mathbf{X})$. Same as Design 2, we choose to use the random curves $\psi_k(t) = \sum_{l=1}^{13} \beta_{k,l} B_l(t)$, where $\{B_l(t)\}_{l=1}^{13}$ are the B-spline basis functions with order 4 and $\beta_{k,l} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 4)$. In consequence, the functional response is generated following $Y(t) = \sum_{k=1}^{20} \text{NN}(\mathbf{X})\psi_k(t)$.

Methods	FoS	NNBB	NNSS	NNBR	NNSR
Mean	4.17	4.18	4.16	4.13	4.17
Std. Dev.	0.02	0.03	0.02	0.03	0.02
p-value	-	1.1e-02	6.2e-02	2.7e-08	1.8e-01

Table 4 Table of MSE for various models when the data is generated using a combination of B-spline-based random curves and NN-transformed covariates to construct the response variable.

Table 4 reports the mean and SD of MSE_p on the testing sets for all models with data generated by Design 3. It is shown that most models perform similarly on predicting the functional curve. The resulting ζ_K produced by the NN generator were barely nonlinear, having only small curvature with respect to the predictors. This leads to a reduction in the performance gap between our NN models and the FoS model. However, the NNBR model comes ahead once again, being significantly superior to the FoS model.

Design 4 (linear scenario): In the last design, we consider setting up a scenario where the functional response $Y(t)$ is linearly associated the scalar predictors. 20 predictors are generated in the manner of $\{X_k\}_{k=1}^{20}$ being binary variables, eight-level categorical variables and i.i.d. uniform random variables

from $[a, b]$ with $a \in \{-4, -3, -2, -1, 0\}, b \in \{3, 4, 5, 6, 7\}$ for different subsets of $k = 1, 2, ..., 20$ (same as Design 2 & 3). The linear setting is simply achieved as $\zeta_k(\mathbf{X}) = X_k$. We continue with the random curves $\psi_k(t) = \sum_{l=1}^{13} \beta_{k,l} B_l(t)$, where $\{B_l(t)\}_{l=1}^{13}$ are 4th-order B-spline basis functions, together with $\beta_{k,l} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 4)$. The functional response is generated as a linear combination of the random curves and predictors, following $Y(t) = \sum_{k=1}^{20} X_k \psi_k(t)$.

Methods	FoS	NNBB	NNSS	NNBR	NNSR
Mean	4.01	4.19	6.66	4.07	5.94
Std. Dev.	0.05	0.06	0.15	0.05	0.05
p-value	-	3.2e-08	4.3e-13	7.8e-07	2.2e-09

Table 5 Table of MSE for various models when the data is generated using a linear combination of B-spline-based random curves and covariates to construct the response variable.

Table 5 displays the results of all models in the linear setting described. In the linear setting, we expect FoS to be the top performer and indeed, it has the lowest prediction error than all the proposed methods. This is reasonable as the $Y(t)$ is constructed linearly with respect to the predictors in this design. However, despite the proposed approaches being significantly worse, NNBB and NNBR remain competitive with strong performances closely following the one of FoS. From an absolute perspective, the gap between NNBR and FoS is much smaller in the scenario favouring FoS (**Design 4**), with FoS being marginally ahead, than it is in the scenario favouring NNBR (**Design 1** and **Design 2**), with NNBR being dramatically ahead.

6 Conclusion

In the article, we introduce a new solution for the regression of functional responses on scalar predictors, which consistently outperforms the current models when the relation between the functional response and the scale predictors is nonlinear. We designed models producing two types of output, either basis coefficients or FPC scores, both of them being the traditional techniques to analyse functional data. A modification to the objective function allows us to train the model directly with the functional response variable thus bypassing the necessity to firstly estimate those coefficients using conventional FDA approaches. The modified objective function decompress the NN output to the functional curves in a linear manner, allowing us to use back-propagation. We implemented all of our proposed models in a way they can be trained on both regularly and irregularly spaced domain. Additionally, we provided the necessary tools to control the smoothness of the predicted response curves by implementing two different roughness penalties. Furthermore, our family of models scales better with the number of predictors.

Based on a real data application, we demonstrated a case where the models we propose are superior to already established techniques such as FoS and FAMM. Moreoever, through several simulation studies, we not only showed the superior predictive power of our NN-based approaches, but also their strong ability to recover nonlinear relation between predictors and coefficients representing the response curves.

On the other hand, the developed methods rely on a large number of hyper-parameters, including the number of hidden layers, the number of neurons in each of the hidden layers, the number of basis functions (NN output size), the number of training epochs, etc. This is the main weakness of our approach since conducting a grid search on that space is particularly annoying and time-consuming. For both B-spline expansion models, the performance varied dramatically from one hyper-parameters configuration to another and this is certainly we need to bring up. Comparatively, the FPCA-based model varied only slightly while changing the number of principal components, given once we have the first few leading principal components, we capture the vast majority of the variability between curves. The performance was also much more stable across various parameterizations, though those models were rarely top performers. In contrast, the FoS model is the easiest to fit among all the models applied, howbeit it performed poorly in non-linear situations.

Multiple directions of further research are considered at the moment. Our proposed models can be extended to predict multidimensional (mainly two-dimensional) functional response where $t \in \mathbb{R}^q, q > 1$. In this scenario, we can borrow the basis expansion or FPCA technique to compress the multidimensional functional data to a vector of finite scalar basis coefficients or FPC scores [4, 10, 16, 18, 41]. Additionally, our current models rely on the existing NN architecture with a scalar output layer, which requires us to firstly project the functional response to some finite-dimensional coefficients and then feed the NN with the scalar representatives obtained. Extending the NN to a more dynamic architecture allowing a functional output layer could be a more appropriate tool for such type of regression problem. This could be achieved by defining functional nodes and functional layers, a concept we are currently exploring. Furthermore, a combination of our proposed NN models with existing literature that tackles the SoF problem can be explored to create a general and complete framework for using NN to analyse and solve regression problems for various forms of functional data.

Declarations

The authors would like to acknowledge the financial support of the Canadian Statistical Sciences Institute (CANSSI) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[2] Rina Foygel Barber, Matthew Reimherr, and Thomas Schill. The function-on-scalar lasso with applications to longitudinal gwas. *Electronic Journal of Statistics*, 11(1):1351–1389, 2017.

[3] Cédric Beaulac, Jeffrey S Rosenthal, and David Hodgson. A deep latent-variable model application to select treatment intensity in survival analysis. *Proceedings of the Machine Learning for Health (ML4H) Workshop at NeurIPS 2018*, 2018.

[4] Lu-Hung Chen and Ci-Ren Jiang. Multi-dimensional functional principal component analysis. *Statistics and Computing*, 27(5):1181–1192, 2016.

[5] Jeng-Min Chiou, Hans georg Müller, and Jane ling Wang. Functional response models. *Statistica Sinica*, pages 675–693, 2004.

[6] Jeng-Min Chiou, Hans-Georg Müller, and Jane-Ling Wang. Functional quasi-likelihood regression models with smooth random effects. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(2):405–423, 2003.

[7] François Chollet et al. Keras. https://keras.io, 2015.

[8] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of Computing*, pages 1–6, 1987.

[9] Carl d. Boor. *A Practical Guide to Splines*. Springer Verlag, New York, 1978.

[10] Paul Dierckx. Computation of least-squares spline approximations to data over incomplete grids. *Computers & Mathematics with Applications*, 10(3):283–289, 1984.

[11] Paul H. C. Eilers and Brian D. Marx. Flexible smoothing with B-splines and penalties. *Statistical Science*, 11(2):89–121, 1996.

[12] Ana-Maria Staicu Fabian Scheipl and Sonja Greven. Functional additive mixed models. *Journal of Computational and Graphical Statistics*, 24(2):477–501, apr 2015.

[13] Jan Gertheiss Fabian Scheipl and Sonja Greven. Generalized functional additive mixed models. *Electronic Journal of Statistics*, 10(1):1455–1492, 2016.

[14] Frédéric Ferraty and Philippe Vieu. *Nonparametric Functional Data Analysis: Theory and Practice*. Springer-Verlag, New York, 2006.

[15] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2. Springer, 2009.

[16] J. G. Hayes and J. Halliday. The least-squares fitting of cubic spline surfaces to general data sets. *IMA Journal of Applied Mathematics*, 14(1):89–103, 1974.

[17] Hemant Ishwaran, Udaya B Kogalur, Eugene H Blackstone, and Michael S Lauer. Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860, 2008.

[18] Andrada Ivanescu. A note on bivariate smoothing for two-dimensional functional data. *International Journal of Statistics and Probability*, 2(2), 2013.

[19] Jared L Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(1):1–12, 2018.

[20] Paola Lecca. Machine learning for causal inference in biological networks: perspectives of this challenge. *Frontiers in Bioinformatics*, page 45, 2021.

[21] Jeffrey S. Morris. Functional regression. *Annual Review of Statistics and Its Application*, 2(1):321–359, 2015.

[22] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[23] J. O. Ramsay, Spencer Graves, and Giles Hooker. *fda: Functional Data Analysis*, 2021. R package version 5.5.1.

[24] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis (Second Edition)*. Springer, 2005.

[25] Aniruddha Rajendra Rao, Qiyao Wang, Haiyan Wang, Hamed Khorasgani, and Chetan Gupta. Spatio-temporal functional neural networks, 2020.

[26] F. Rossi, B. Conan-Guez, and F. Fleuret. Functional data analysis with multi layer perceptrons. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, volume 3, pages 2843–2848, 2002.

[27] Fabrice Rossi and Brieuc Conan-Guez. Functional multi-layer perceptron: a non-linear tool for functional data analysis. *Neural Networks*, 18(1):45–60, 2005.

[28] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[29] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[30] Volker Strassen et al. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.

[31] Arnab Maity Suchit Mehrotra. Simultaneous variable selection, clustering, and smoothing in function-on-scalar regression. *Canadian Journal of Statistics*, 50(1):180–199, 2022.

[32] Barinder Thind, Kevin Multani, and Jiguo Cao. Neural networks as functional classifiers. *arXiv preprint arXiv:2010.04305*, 2020.

[33] Barinder Thind, Kevin Multani, and Jiguo Cao. Deep learning with functional inputs. *Journal of Computational and Graphical Statistics*, 0:1–27, 2022.

[34] Jane-Ling Wang, Jeng-Min Chiou, and Hans-Georg Müller. Functional data analysis. *Annual Review of Statistics and Its Application*, 3(1):257–295, 2016.

[35] Lifeng Wang, Guang Chen, and Hongzhe Li. Group scad regression analysis for microarray time course gene expression data. *Bioinformatics*, 23(12):1486–1494, 2007.

[36] Q. Wang, H. Wang, C. Gupta, A. Rao, and H. Khorasgani. A non-linear function-on-function model for regression with time series data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 232–239, Los Alamitos, CA, USA, dec 2020. IEEE Computer Society.

[37] Q. Wang, H. Wang, C. Gupta, A. Rao, and H. Khorasgani. A non-linear function-on-function model for regression with time series data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 232–239,

Los Alamitos, CA, USA, dec 2020. IEEE Computer Society.

[38] Fang Yao, Hans-Georg Müller, and Jane-Ling Wang. Functional data analysis for sparse longitudinal data. *Journal of the American statistical association*, 100(470):577–590, 2005.

[39] Junwen Yao, Jonas Mueller, and Jane-Ling Wang. Deep learning for functional data analysis with adaptive basis layers, 2021.

[40] Xiaoke Zhang and Jane-Ling Wang. Varying-coefficient additive models for functional data. *Biometrika*, 102(1):15–32, 12 2014.

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