Nonlinear Functional Modeling Using Neural Networks

Aniruddha Rajendra Rao and Matthew Reimherr
Department of Statistics, Pennsylvania State University, University Park, PA

ABSTRACT

We introduce a new class of nonlinear models for functional data based on neural networks. Deep learning has been very successful in nonlinear modeling, but there has been little work done in the functional data setting. We propose two variations of our framework: a functional neural network with continuous hidden layers, called the Functional Direct Neural Network (FDNN), and a second version that uses basis expansions and continuous hidden layers, called the Functional Basis Neural Network (FBNN). Both are designed explicitly to exploit the structure inherent in functional data. To fit these models we derive a functional gradient based optimization algorithm. The effectiveness of the proposed methods in handling complex functional models is demonstrated by comprehensive simulation studies and real data examples. Supplementary materials for this article are available online.

1. Introduction

In today’s world, repeated measurements taken over time or space are collected across a variety of different areas of industry, government, and academia. For instance, health monitoring of patients, daily weather information, and stock prices, to name only a few. In such cases, the data of interest changes continuously over time. One approach to modeling such data is functional data analysis (Ramsay and Silverman 1997; Kokoszka and Reimherr 2018).

Functional Data Analysis (FDA) is a branch of statistics that enables researchers to analyze and model relationships between variables measured over a particular domain, such as time or space (Ramsay and Silverman 1997; Ferraty and Vieu 2006; Ferraty and Romain 2011; Horváth and Kokoszka 2012; Kokoszka and Reimherr 2018). FDA has been a rich area of research in the last few decades. The advantage FDA has over multivariate data analysis is that FDA enables smoothing and interpolation procedures that can yield functional representations of a finite set of observations. Also, FDA has applicability to data that are recorded irregularly over time or some other domain.

Functional regression can mainly be classified into three major categories according to the role played by the functional data in each model: scalar response and functional predictors (scalar-on-function regression); functional response and scalar predictors (function-on-scalar regression); and functional response and functional predictors (function-on-function regression). This article concentrates on the first case, that is the scalar-on-function regression. Even though Functional Regression, especially scalar-on-function regression, is perhaps one of the most thoroughly researched topics within the broader literature on FDA, there is still a great need for superior nonlinear models. It is not only a very active area for research, but also is very challenging with functional data due to the curse of dimensionality (Geenens 2011).

Some common ways of dealing with nonlinearity in functional data are by using the generalized additive model, also known as Continuously Additive Model (CAM) (Maller, Wu, and Yao 2013; Wang and Ruppert 2013; McLean et al. 2014; Fan, James, and Radchenko 2015; Ma and Zhu 2016; Reimherr, Sriperumbudur, and Taoufik 2017), Quadratic Model (Yao and Muller 2010), Single Index Model (Stoker 1986; Eilers, Li, and Marx 2009; Ferraty 2011), and Multiple Index Model (James and Silverman 2004; Chen, Hall, and Müller 2012; Ferraty et al. 2013). The single index model has the outcome related to a linear functional of the predictors through a nonlinear transformation, whereas the multiple index model is analogous but with multiple linear projections. In these models, you can either choose to capture the complex nonlinear relations or the higher order interactions. In particular, the multiple index model allows for complex interactions but nonlinearity is weak because we are only using a few projections whereas in CAM, we can capture the nonlinearity but interactions are completely ignored, and the quadratic model assumes a linear structure with interactions without any provision for nonlinearity. In general, these models assume a very particular structure, which limits the kind of nonlinear patterns that can be learned, especially in terms of interactions within the time points for the predictors. Most of these approaches have been discussed in the review papers by Reiss et al. (2016) and Morris (2014).

Neural Network (NN) and Deep Learning (DL) have attracted a great deal of attention and research due to their application in areas like time series (Dorffner 1996; Khashei and Bijari 2010; Qin et al. 2017), image processing/computer vision, and many others.
vision (Kanellopoulos and Wilkinson 1997; Gregor et al. 2015; Shi et al. 2016; Zhang et al. 2017; Wang and Yang 2019), speech processing/natural language processing (NLP) (Seide, Li, and Yu 2012; Graves, Mohamed, and Hinton 2013; Chan et al. 2016; Morchid 2018), genetics (Asadi et al. 2014; Singh and Singh 2017), self-driving cars (Chen and Huang 2017; Maqueda et al. 2018), and video learning (Karpathy et al. 2014; Yue-Hei Ng et al. 2015). Neural network models by combining multiple linear combinations of the predictors after passing them through activation/link functions (called hidden layers) to learn complex nonlinear relationships. Even with so much success, neural network and deep learning are still relatively undeveloped for functional data. A traditional neural network includes no special provision for the temporal dependency seen in FDA. Recurrent Neural Network (RNN), especially Long Short-Term Memory (LSTM), have an edge over conventional feed-forward neural networks when it comes to time series or sequential data. This is due to their ability to selectively remember patterns for a long duration of time compared to regular RNN (Joshi, Khosravy, and Gupta 2020). The main drawback of deep learning techniques is that they can’t handle curves directly, especially if the curves are measured at different time points or with a different number of observations, that is if the data is irregular.

Rossi, Conan-Guez, and Fleuret (2002), Rossi and Conan-Guez (2006), and Conan-Guez and Rossi (2002) developed and demonstrated the efficacy of Functional Neural Network (FNN). FNN was then further investigated by Wang et al. (2019) and Wang et al. (2019), where they demonstrated the superiority of FNN to state-of-the-art alternatives (CNN, RNN, LSTM). The extension of the neural network to functional data is intriguing, as it enables efficient nonlinear learning when the predictors are curves or surfaces. Rossi, Conan-Guez, and Fleuret (2002) proposed a way to feed functional data as an input into a neural network and proved the validity by theoretical arguments and simulation experiments. But their initial work (Rossi, Conan-Guez, and Fleuret 2002) required the data to be dense and regular. This requirement was overcome with a slightly different FNN version (Conan-Guez and Rossi 2002) where they used curve projections. Wang et al. (2019) extended this work to deal with sparsity. The basic approach is that it only requires the entire time interval to be well covered by pooled observations across all the subjects. They pool measurements among all the subjects to estimate a set of basis functions and then use these basis functions to build a sparse functional neuron that extracts features for each subject. This method is developed for sparse FDA but it is equivalent to the original FNN with dense data. The main idea of FNN is that only the first layer is a functional hidden layer with functional neurons that give a scalar output, which is then connected to regular hidden layers and finally the output layer. In short, the functional layer is followed by a traditional neural network. This is a straightforward approach but it does not embrace the functional data completely. Other extensions of FNN for Spatio-Temporal modeling and Function-on-Function regression were explored in Rao et al. (2020) and Wang et al. (2020). We use the idea of FNN as an inspiration to develop our novel approach where we extend the initial functional hidden layer with functional neurons to multiple continuous hidden layers with continuous neurons with no regular hidden layers.

In this work, we study the problem of nonlinear modeling from functional data using neural networks. There are mainly two challenges: (a) How to design a continuous hidden layer consisting of continuous neurons for a neural network that can learn functional representation from the functional inputs; and (b) how to train such a model. We propose a novel approach that takes the functional input and gives a scalar output, which can be solved using either Functional Direct Neural Network (FDNN) or Functional Basis Neural Network (FBNN). The main contributions of this work are as follows:

- We introduce a new structure for nonlinear modeling of functional data with a scalar response.
- We give two strategies, FDNN and FBNN, to solve the problem.
- We derive the functional gradient based learning algorithm for both strategies to optimize the parameters.
- We conduct simulation studies to demonstrate the effectiveness of FDNN and FBNN in varying settings. We also apply the models to several real-world examples.

The rest of this article is organized as follows: In Section 2, we first present some background on FDA and FNN. Then we describe our approach and give two strategies to solve it, Functional Direct Neural Network (FDNN) and Functional Basis Neural Network (FBNN). We follow this with simulations and several benchmark applications in Section 3. We consider multiple simulation settings depending on how the nonlinear relation is defined between the response and functional predictor. We demonstrate the efficiency of our methods over current state-of-the-art approaches. In the final section, we present our concluding remarks and future research directions which pertain to deeper statistical theory and extension to a variety of functional data settings.

2. Methods

We begin by providing the necessary notation used in the article, followed by a summary of a few of the major nonlinear models for scalar-on-function regression before describing our methodology. Suppose we have \( N \) subjects observed over a compact time interval \( \mathcal{T} \), and for the \( i \)th subject, the inputs are \( L \) random functions that can be denoted as \( \{X_{i1}, \ldots, X_{iL}\} \in \mathcal{L}(\mathcal{T}) \). The output is a scalar variable \( Y_i \in \mathcal{R} \) for \( i = 1, \ldots, N \). We assume \( X_{ir} := \{X_{ir}(t) : t \in [0,1]\} \) for \( r = 1,\ldots, L \) are fully observed functions, which is also known as dense functional data analysis (Kokoszka and Reimherr 2018). The main objective is to learn the nonlinear mapping, \( F : \mathcal{L}(\mathcal{T}) \times \cdots \times \mathcal{L}(\mathcal{T}) \rightarrow \mathcal{R} \) from the functional inputs \( X_{ir} \) to the scalar output \( Y_i \)

\[
E[Y_i|X_{i1}, \ldots, X_{iL}] = f(X_{i1}, \ldots, X_{iL}).
\]

2.1. Functional Regression

There is vast literature in FDA when it comes to scalar-on-function regression and we briefly describe the most popular ones, which will be later used in simulations for comparison of different methods. In these models, the integrals without limits are taken over the entire domain.
The scalar-on-function regression model under different relation between the functional predictors and the scalar response is as follows:

- **Linear Model:**
  
  \[
  E[Y_i|X_{i,1},\ldots,X_{i,R}] = \alpha + \sum_{r=1}^{R} \int \beta_r(t)X_{i,r}(t)dt
  \tag{2}
  \]
  
  where \( \beta_r(t) \) is the parameter function (Ramsay and Silverman 1997; Kokoszka and Reimherr 2018). This is also known as Functional Linear Model (FLM).

- **Continuously Additive Model (CAM):**

  \[
  E[Y_i|X_{i,1},\ldots,X_{i,R}] = \sum_{r=1}^{R} \int f(X_{i,r}(t),t)dt
  \tag{3}
  \]

  where bivariate parameter function \((x,t) \to f(x,t)\) is smooth (Maller, Wu, and Yao 2013; Wang and Ruppert 2013; McLean et al. 2014; Fan, James, and Radchenko 2015; Ma and Zhu 2016).

- **Single Index Model:**

  \[
  E[Y_i|X_{i,1},\ldots,X_{i,R}] = g\left( \sum_{r=1}^{R} \beta_r X_{i,r} \right)
  \tag{4}
  \]

  where the function \(g()\) is any smooth function defined on the real line (Eilers, Li, and Marx 2009; Chen, Hall, and Müller 2012).

- **Multiple Index Model:**

  \[
  E[Y_i|X_{i,1},\ldots,X_{i,R}] = g\left( \sum_{r=1}^{R} \beta_{1r} X_{i,r} + \ldots + \sum_{r=1}^{R} \beta_{Pr} X_{i,r} \right)
  \tag{5}
  \]

  Multiple Index models are extensions of the single index models (Muller and Yao 2008; Chen, Hall, and Müller 2012; Ferraty et al. 2013; Zhu, Yao, and Zhang 2013).

- **Quadratic Model:**

  \[
  E[Y_i|X_{i,1},\ldots,X_{i,R}] = \sum_{r=1}^{R} \beta_r(t)X_{i,r}(t)dt + \sum_{r=1}^{R} \int \beta(s,t)X_{i,r}(t)X_{i,r}(s)dt ds,
  \tag{6}
  \]

  \[
  E[Y_i|X_{i,1},\ldots,X_{i,R}] = \sum_{r=1}^{R} \int f(X_{i,r}(t),t)dt + \sum_{r=1}^{R} \int f(X_{i,r}(t),X_{i,r}(s),s,t)ds ds,
  \tag{7}
  \]

  Equation (6) represents a Quadratic Model given by Yao and Muller (2010) whereas (7) represents a Complex Quadratic Model defined for simulation purposes.

The above models are the most common approaches used in functional regression for linear and nonlinear cases. They are commonly fit using basis expansions, Functional Principal Component Analysis (FPCA), or penalization methods. The nonlinear models are analogous to their multivariate counterparts, which help to overcome the curse of dimensionality. In practice, it is difficult to know the actual form of the relation between the response and the functional predictors. Therefore, having a method that can deal with any kind of relationship without knowing the truth will be very useful.

### 2.2. Functional Neural Network

A neural network consists of three components: an input layer, hidden layers, and an output layer. The input layer and output layer consist of neurons/nodes that are linked by multiple hidden layers consisting of neurons. In the forward propagation, that is to construct a predicted value, each neuron sums the weighted signal it receives from the input connections and produces an output signal which is a nonlinear function of these inputs. Back propagation (Rumelhart, Hinton, and Williams 1986) (that is gradient descent) is used to determine the correct weights that minimize the overall cost in the neural network.

Figure 1 shows the architecture of the FNN model developed by Rossi, Conan-Guez, and Fleuret (2002) using the idea of a generalized neuron (Stinchcombe 1999). In FNN, the first hidden layer is a functional hidden layer which consists of functional neurons that give scalar outputs. These scalar values are then fed into a regular neural network framework. A functional neuron is defined as follows:

\[
H(X, W_r) = \sigma\left(b + \sum_{r=1}^{R} W_r(t)X_r(t)dt\right),
\tag{8}
\]

where \(b \in \mathcal{R}\) is the unknown intercept, \(W_r(t)\) is the parameter function \((r = 1,\ldots,R)\) and \(\sigma()\) is a nonlinear activation function from \(\mathcal{R}\) to \(\mathcal{R}\).

In forward propagation, as seen in Figure 1, we first pass through the functional layer having functional neurons. For the value of \(H(X, W_r(t))\), we use (8), where numerical approximation is used to get the integral value. The forward propagation calculation in the subsequent hidden layers is straightforward because of the regular neural network layers. In the backpropagation step, we can go from the output layer to the second hidden layer using the partial derivatives that can be easily computed. It is critical to make sure that the partial derivatives at the first hidden layer for the parameters \(W_r(t)\) exist. Under certain assumptions (Rossi, Conan-Guez, and Fleuret 2002; Wang et al. 2019), we can compute the partial derivatives required in the first hidden layer as

\[
\frac{\partial H(X, W_r)}{\partial W_r}(t) = \sigma'(b + \sum_{r'=1}^{R} W_{r'}(t')X_{r'}(t')dt') \times X_r(t)
\tag{9}
\]

where \(\sigma'(\cdot)\) represents the first derivative of the activation function \(\sigma(\cdot)\).
2.3. Functional Direct Neural Network and Functional Basis Neural Network

In our proposed approach, we develop a continuous mapping from layer to layer. To do this, we make use of what we call *continuous neurons* and refer to a hidden layer with multiple continuous neurons as a *continuous hidden layer*. In our neural network framework, we have three types of layers: an input layer, which is the first layer that takes in the functional predictors, the continuous hidden layers containing continuous neurons, and a functional output layer that provides the final fitted value using functional neurons. Unlike neural network, which learns using scalar weights, we learn using weight functions that are continuous over time (or some other continuum). The functional output layer that provides the final fitted value using functional neurons that output a function and learn bivariate functional relationships between the functional predictors and the scalar response via a chain of functional neurons only in the last layer to get the final output value.

![Diagram of a neural network with functional and numerical neurons](image)

**Figure 1.** The architecture of a FNN with two functional neurons in the first layer and two numerical neurons in the second layer.

We implement our proposed approach using two strategies: (a) a more direct solution when the data is observed on a common grid or (b) using basis expansions to allow for more irregular sampling of the functional predictors. The former approach we call as Functional Direct Neural Network (FDNN) while the latter approach we refer to as Functional Basis Neural Network (FBNN). FDNN works well with densely observed data on a common grid. We can apply standard tools from the neural network in FDNN directly, making it straightforward to implement. We are learning the weight functions similar to the way we learn the weights in the neural network and also have the flexibility to change the grid length (s) between the continuous hidden layers. FBNN uses basis expansion to learn the relation and also allows for greater parsimony if the data is sampled on a very dense grid, as we need to only learn the weights of the basis expansion.

**Functional Direct Neural Network (FDNN):**

For simplicity, we assume that Figure 2 has $R$ inputs (functional predictors), $L$ continuous hidden layers, each with $J$ incoming and $K$ outgoing connections. FDNN has the same flexibility as a neural network, where we can specify the number of continuous hidden layers and the number of continuous neurons in each of them. Commonly used activation functions in the continuous neurons include ReLU, tanh, sigmoid/logistic and linear. The activation function used in the functional output layer depends on the response. In our simulations, for the continuous response, we use the linear activation function in the functional output layer, whereas for the binary response, we use the sigmoid activation function. In the forward propagation phase, we reach from the functional predictors to the scalar response via a chain of multiple CAMs, and in the backpropagation phase, we want to learn the weight functions and the intercept functions.

We derive a functional gradient based optimization algorithm to optimize the network parameters. The gradient of the weight function measures the change in a functional to a change in a function on which the functional depends. The needed assumptions and mathematical tools can be found in Rossi, Conan-Guez, and Fleuret (2002) and Wang et al. (2019). We extend the gradient descent optimization technique to the functional setting using calculus of variations (Olver 2019), in particular by taking Fréchet derivatives. In the backpropagation step, we pass through the network in reverse, as we do in a neural network, but we calculate the partial derivative for the weight functions, the intercept functions and the intercept. Let the loss function be $L(\theta)$, where $\theta$ is the collection of all functional and scalar parameters. The loss function is defined based on the response, quadratic loss for a continuous and a...
cross entropy loss for a binary response. The partial derivatives required for the parameters of the continuous neurons in the continuous hidden layers of our proposed architecture (as shown in Figure 2) are as follows:

\[
\frac{\partial H^{(l)}_{(k)}}{\partial b^{(l)}_{(k)}}(s) = \sigma'(b^{(l)}_{(k)}(s)) + \sum_{j=1}^{l} \int w^{(l)}_{(j,k)}(s,t) H^{(l-1)}_{(j)}(t) dt
\]

\[
\frac{\partial H^{(l)}_{(k)}}{\partial w^{(l)}_{(j,k)}}(s,t) = \sigma'(b^{(l)}_{(k)}(s)) + \sum_{j=1}^{l} \int w^{(l)}_{(j,k)}(s,t') H^{(l-1)}_{(j)}(t') dt'
\]

\[
\times H^{(l-1)}_{(j)}(t)
\]

where \( l = 1, 2, 3, \ldots, L \), \( H^{(0)}_{(i)}(t) := X_i(t) \), and \( \sigma'() \) represents the first derivative of \( \sigma() \).

Next, the partial derivatives required for the parameters of the functional neurons in the functional output layer are,

\[
\frac{\partial H^{(L+1)}}{\partial b^{(l+1)}} = \sigma'(b^{(L+1)}) + \sum_{j=1}^{l} \int w^{(L+1)}_{(j)}(t) H^{(L)}_{(j)}(t) dt
\]

\[
\frac{\partial H^{(L+1)}}{\partial w^{(L+1)}_{(j)}} = \sigma'(b^{(L+1)}) + \sum_{j=1}^{l} \int w^{(L+1)}_{(j)}(t') H^{(L)}_{(j)}(t') dt'
\]

\[
\times H^{(L)}_{(j)}(t).
\]

Section 1 of the supplementary materials contains in-depth information regarding the partial derivatives. We can consider the values of the grid/domain \((s)\), the number of continuous hidden layers, and the number of continuous neurons in each continuous hidden layer as hyperparameters. The general way of optimizing for the parameters is still the same in FDNN, using backpropagation with the help of the gradients given above.

**Functional Basis Neural Network (FBNN):** Similar to FDNN, we have continuous hidden layers with continuous neurons in FBNN, but the continuous layers are expanded with basis expansion. Unlike FDNN, we learn the weights for the basis functions rather than the weight function itself. As we are using basis functions, we can deal with irregular data more easily (Kokoszka and Reimherr 2018). We get the forward feed in FBNN by using basis expansion where we replace the weight functions \((w^{(l)}_{(j,k)}(s,t))\) in (10) by a linear combination of basis functions as seen below. We will again see the details of our approach using the architecture shown in Figure 2. Let subscripts \(j, k\) represent the continuous neurons of the neural network and superscript \(l\) denote the \(l\)th layer for the weight parameter \((W^{(l)}_{(j,k)})\). We define \(v\) to be some standard collection of basis functions, like splines, wavelets, or sine and cosine functions. Also, let us assume that we use the same basis function and the same number of basis in each of the continuous neurons in the continuous hidden layer across the network for simplicity. This gives us the following equation:

\[
H^{(l)}_{(k)}(s) = a(b^{(l)}_{(k)}(s) + \sum_{j=1}^{l} w^{(l)}_{(j,k)}(s,t) H^{(l-1)}_{(j)}(t) dt)
\]

\[
= a\left(\sum_{b=1}^{B} b^p_{(b)}(s) v^p_b(s) + \sum_{i=1}^{C} \sum_{d=1}^{D} w^{(l)}_{(j,k),cd} v_c(s) A^{(l)}_{(j,d)}\right)
\]

\[
\times \int v_d(t) H^{(l-1)}_{(j)}(t) dt
\]

\[
= a\left(\sum_{b=1}^{B} b^p_{(b)}(s) v^p_b(s) + \sum_{i=1}^{C} \sum_{d=1}^{D} w^{(l)}_{(j,k),cd} v_c(s) A^{(l)}_{(j,d)}\right)
\]

\[
\cdot \int v_d(t) H^{(l-1)}_{(j)}(t) dt
\]

\[
= a\left(\sum_{b=1}^{B} b^p_{(b)}(s) v^p_b(s) + \sum_{i=1}^{C} \sum_{d=1}^{D} w^{(l)}_{(j,k),cd} v_c(s) A^{(l)}_{(j,d)}\right)
\]

\[
\times \int v_d(t) H^{(l-1)}_{(j)}(t) dt
\]

\[
= a\left(\sum_{b=1}^{B} b^p_{(b)}(s) v^p_b(s) + \sum_{i=1}^{C} \sum_{d=1}^{D} w^{(l)}_{(j,k),cd} v_c(s) A^{(l)}_{(j,d)}\right)
\]

\[
\times \int v_d(t) H^{(l-1)}_{(j)}(t) dt
\]

\[
= a\left(\sum_{b=1}^{B} b^p_{(b)}(s) v^p_b(s) + \sum_{i=1}^{C} \sum_{d=1}^{D} w^{(l)}_{(j,k),cd} v_c(s) A^{(l)}_{(j,d)}\right)
\]

\[
\times \int v_d(t) H^{(l-1)}_{(j)}(t) dt
\]

\[
= a\left(\sum_{b=1}^{B} b^p_{(b)}(s) v^p_b(s) + \sum_{i=1}^{C} \sum_{d=1}^{D} w^{(l)}_{(j,k),cd} v_c(s) A^{(l)}_{(j,d)}\right)
\]

\[
\times \int v_d(t) H^{(l-1)}_{(j)}(t) dt
\]

\[
= a\left(\sum_{b=1}^{B} b^p_{(b)}(s) v^p_b(s) + \sum_{i=1}^{C} \sum_{d=1}^{D} w^{(l)}_{(j,k),cd} v_c(s) A^{(l)}_{(j,d)}\right)
\]

\[
\times \int v_d(t) H^{(l-1)}_{(j)}(t) dt
\]

\[
= a\left(\sum_{b=1}^{B} b^p_{(b)}(s) v^p_b(s) + \sum_{i=1}^{C} \sum_{d=1}^{D} w^{(l)}_{(j,k),cd} v_c(s) A^{(l)}_{(j,d)}\right)
\]

\[
\times \int v_d(t) H^{(l-1)}_{(j)}(t) dt
\]

\[
= a\left(\sum_{b=1}^{B} b^p_{(b)}(s) v^p_b(s) + \sum_{i=1}^{C} \sum_{d=1}^{D} w^{(l)}_{(j,k),cd} v_c(s) A^{(l)}_{(j,d)}\right)
\]

\[
\times \int v_d(t) H^{(l-1)}_{(j)}(t) dt
\]

\[
= a\left(\sum_{b=1}^{B} b^p_{(b)}(s) v^p_b(s) + \sum_{i=1}^{C} \sum_{d=1}^{D} w^{(l)}_{(j,k),cd} v_c(s) A^{(l)}_{(j,d)}\right)
\]

\[
\times \int v_d(t) H^{(l-1)}_{(j)}(t) dt
\]

\[
= a\left(\sum_{b=1}^{B} b^p_{(b)}(s) v^p_b(s) + \sum_{i=1}^{C} \sum_{d=1}^{D} w^{(l)}_{(j,k),cd} v_c(s) A^{(l)}_{(j,d)}\right)
\]

\[
\times \int v_d(t) H^{(l-1)}_{(j)}(t) dt
\]
hidden layers, the number of continuous neurons in each continuous hidden layer, the selection of the basis functions, and the number of basis as hyperparameters. We learn using the basis weights of the basis functions, to reach from the functional predictors to the scalar response via a chain of multiple CAMs in the forward propagation phase. The partial derivatives required for the parameters of the continuous neurons in the continuous hidden layers under FBNN in our proposed architecture (as seen in Figure 2) are as follows:

\[
\frac{\partial H^{(i)}(s)}{\partial B^{(i)}(k)} = \sigma'(B^{(i)}(k)v(s) + \sum_{j=1}^{l} W^{(i)}_{(j,k)}v(s)A^{(i)}_{(j)}) \times v^s(s)
\]

\[
\frac{\partial H^{(i)}(s)}{\partial W^{(i)}_{(j,k)}} = \sigma'(B^{(i)}(k)v(s) + \sum_{j'=1}^{l} W^{(i)}_{(j',k)}v(s)A^{(i)}_{(j')}) \times v(s)A^{(i)}_{(j)}
\]

where \(l = 1, 2, \ldots, L\), \(H^{(i)}(s) := X_i(t)\), and \(\sigma'(-)\) represents the first derivative of \(\sigma(-)\).

Next, the partial derivatives required for the parameters of the functional neurons in the functional output layer are,

\[
\frac{\partial H^{(L+1)}}{\partial b^{(L+1)}} = \sigma'(b^{(L+1)} + \sum_{j=1}^{l} W^{(L+1)}_{(j)}A^{(L+1)}_{(j)})
\]

\[
\frac{\partial H^{(L+1)}}{\partial W^{(L+1)}_{(j)}} = \sigma'(b^{(L+1)} + \sum_{j'=1}^{l} W^{(L+1)}_{(j')A^{(L+1)}_{(j')}}) \times A^{(L+1)}_{(j)}
\]

The general way of optimizing for the parameters is the same in FBNN, using backpropagation, but the gradients are found using the above partial derivatives.

**Regularization:** Over-fitting is a common problem with machine learning and nonparametric statistical methods. They are often over-parameterized and thus, care must be taken to ensure that the results still produce reliable out of sample predictions. Regularization in neural network to deal with over-fitting is an active area of research (Ludwig, Nunes, and Araujo 2014; Ayinde, Inanc, and Zurrada 2019; Yang et al. 2020). Common methods used to address this issue include penalization, drop out, data augmentation, and early stopping method. We concentrate on early stopping, which uses a validation set to stop the model fitting process early before it starts over-fitting. Naturally, both FDNN and FBNN tend to over-fit if the models are too deep or run for a large number of iterations. Early stopping is easy to implement and, as we will see in Section 3, it is effective at improving the performance of FDNN and FBNN.

### 3. Empirical Results

In this section, we present multiple simulations to evaluate the effectiveness of FDNN and FBNN, as well as benchmarking with multiple real world datasets. We demonstrate the following objectives through our results: (a) FDNN and FBNN are effectively trained by the derived optimization method (b) FDNN and FBNN capture the relation between the functional predictors and the scalar response (c) FBNN and FDNN outperform compared to other methods.

#### 3.1. Simulation Study

For simplicity, we consider a single predictor function (that is \(R=1\)) which is dense and regular. We simulate \(n = 1500\) iid random curves \(\{X_1(t), \ldots, X_n(t)\}\) from a Gaussian process with mean 0 and covariance

\[
C_X(t, s) = \frac{\sigma^2}{\Gamma(\nu)} \left( \frac{\sqrt{2\nu}|t - s|}{\rho} \right)^\nu K_\nu \left( \frac{\sqrt{2\nu}|t - s|}{\rho} \right),
\]

which is the Matérn covariance function and \(K_\nu\) is the modified Bessel function of the second kind. We set \(\rho = 0.5\), \(\nu = 5/2\) and \(\sigma^2 = 1\). These curves are evaluated at \(m = 200\) equally spaced time points from \([0, 1]\).

The data generating models for the continuous and binary response are given in Tables 1 and 2, respectively. We consider \(\epsilon_i\) (random noise) to our continuous response model to adjust the signal to noise ratio. For the binary response, we assume that we have an outcome \(Y_i\) related to function \(X_i\) via a link function \(g : E(Y_i|X_i) = g^{-1}(\eta_i)\) where \(\eta_i\) is defined as the response in any of the models mentioned in Section 2.2. Since we have a binary response, the link function used is the logistic function, which is also the canonical link function. Each of these settings is simulated 100 times and we divide the 1500 samples into 1000 for training and 500 for testing. For the Deep Learning methods, we use early stopping which requires a validation set. We use 500 samples from the training set as the validation set in such cases. We are primarily interested in the prediction performance and therefore, we report the Root Mean Square Error (RMSE) of the prediction for the continuous response, and the classification error as well as the log-likelihood for the binary response.

For FDNN and FBNN, we consider multiple combinations of continuous hidden layers \((l = 1, 2)\) and continuous neurons \((j = 1, 2, 4)\). The value for the grid \((s)\) in the continuous neurons is in the range \((30, 300)\). For FBNN, we use B-splines as the basis functions and for the knots, we consider equal spacing. We tune the number of B-splines which corresponds to the number of inner knots, for which consider the range \((5, 10)\). We determine the best values of the hyper parameters using grid search. We observe that mostly the number of continuous neurons and the number of continuous hidden layers change the result drastically while other hyper parameters have marginal effect.

In Table 3, we summarize the performance of each method under different simulation settings with a continuous response. We observe that when the true model is Linear, four models do equally well: FLM, CAM, and both of our methods. When the truth is CAM, then our approaches along with CAM do equally well. More substantial gains are realized as the true model becomes more complex and interactions are considered. In such situations, our methods dominate others. Interestingly, under the Complex Quadratic settings, all the methods struggle substantially compared to our approach, but there is still potential for improvement (say, through deeper networks) as the optimal RMSE is one (since that is the variance of the error). The gain from early stopping can be seen in Section 2 of the supplementary materials. Table 4 shows a similar trend with the binary response. When the truth is Logistic or CAM, then FLM, CAM, and both of our methods perform equally well. Our methods outperform others as the true model becomes more complex and interactions are considered. We see similar results...
when comparing the log-likelihood values as seen in Section 3 of the supplementary materials. In general, we see superior results to other approaches in several settings, and we are always comparable if not better to the best method in each setting. Note that all the table values for the deep learning approaches are obtained using early stopping. We have also compared the deep learning parameters under the same parameter setting in Section 4 of the supplementary materials.

### 3.2. Real Data

We analyze three different datasets as seen in Figure 3. The fat content spectrometric data (Ferraty and Vieu 2006) consists of food samples that contain finely chopped pure meat with different fat content, the growth curves from Berkeley Growth Study (Tuddenham and Snyder 1954) contains information about the heights (in cm) of the children at different age and the speech recognition data from TIMIT (available at https://hastie.su.domains/ElemStatLearn/) where we have speech signals for different phonemes. The spectrometric data is recorded on a Tecator Infracert Food and Feed Analyzer working in the wavelength range 850–1050nm by the near-infrared (NIR) transmission principle. For each meat sample, the data consists of a 100 channel spectrum of absorbances (−log10 of the transmittance measured by a spectrometer) and a fat content measure derived from an analytic chemical processing. We predict the fat content of a meat sample on the basis of its NIR absorbance spectrum. The Growth data contains the heights of 39 boys and 54 girls from age 1 to 18 and the ages at which they were collected. The goal is to build a classifier for the gender using

---

**Table 1.** Different simulation scenarios with continuous response and a single predictor function.

| Simulation Scenarios | y = α + Σ t=1 R β(t)X(t)dt + ε | where β(t) = 5 × sin(2πt) and α = 0 |
|----------------------|----------------------|----------------------|
| Logistic             | y = Σ t=1 R f(X(t), t)dt + ε | where f(X(t), t) = (X(t)^2) |
| Single Index: logit  | g(Σ t=1 R βt, X(t)) + ε | where g(a) = a^2, β(t) = 5 × sin(2πt) |
| Multiple Index: logit| g(Σ t=1 R β1, X(t) and Σ t=1 R β2, X(t)) + ε | where g(a, b) = a^2 + b^2, β1(t) = 5 × sin(3πt), β2(t) = 5 × sin(3πt) |
| Quadratic            | y = Σ t=1 R β(t)X(t)dt + Σ t=1 R β(t)X(t)dt + ε | where β(t, s) = β1(t) + β2(s), β1(t) = 5 × sin(3πt), β2(s) = 5 × sin(πt) |
| Complex Quadratic    | y = Σ t=1 R f(X(t), t)dt + Σ t=1 R f(X(t), t)dt + ε | where f(X(t), t) = (X(t)^2), f(X(t), X(t), t, s) = (X(t) × X(s))^2 |

**Table 2.** Different simulation scenario with binary response and a single predictor function.

| Simulation Scenarios | logit(ηi) = α + Σ t=1 R β(t)X(t)dt | where β(t) = 5 × sin(2πt) and α = 0 |
|----------------------|----------------------|----------------------|
| Logistic             | logit(ηi) = Σ t=1 R f(X(t), t)dt | where f(X(t), t) = sin(X(t)) |
| Single Index: logit  | g(Σ t=1 R βt, X(t)) + ε | where g(a) = sin(a), β(t) = 5 × sin(2πt) |
| Multiple Index: logit| g(Σ t=1 R β1, X(t) and Σ t=1 R β2, X(t)) + ε | where g(a, b) = sin(a) + b, β1(t) = 5 × sin(2πt), β2(t) = 5 × sin(3πt) |
| Quadratic            | logit(ηi) = Σ t=1 R β(t)X(t)dt + Σ t=1 R β(t)X(t)dt + ε | where β(t, s) = β1(t) + β2(s), β1(t) = 5 × sin(3πt), β2(s) = 5 × sin(πt) |
| Complex Quadratic    | logit(ηi) = Σ t=1 R f(X(t), t)dt + Σ t=1 R f(X(t), t)dt + ε | where f(X(t), t) = sin(X(t)), f(X(t), X(t), t, s) = sin(X(t) × X(s)) |

**Table 3.** Comparison of the RMSE for the methods under different setting between the functional predictors and the continuous response.

| Mapping              | FLM | CAM | NN  | FNN | FDNN | FBNN |
|----------------------|-----|-----|-----|-----|------|------|
| Linear               | 1.011 | 1.015 | 1.185 | 1.150 | 1.019 | 1.011 |
| CAM                  | 1.705 | 1.053 | 1.451 | 1.298 | 1.076 | 1.058 |
| Single Index         | 1.711 | 1.532 | 1.515 | 1.225 | 1.067 | 1.061 |
| Multiple Index       | 1.808 | 1.350 | 1.609 | 1.290 | 1.100 | 1.060 |
| Quadratic            | 1.646 | 1.211 | 1.480 | 1.273 | 1.092 | 1.092 |
| Complex Quadratic    | 1.807 | 1.414 | 1.905 | 1.632 | 1.232 | 1.202 |

**Table 4.** Comparison of the classification error for the methods under different setting between the functional predictors and the binary response.

| Mapping              | Logistic | CAM | Single Index | Multiple Index | Quadratic | Complex Quadratic |
|----------------------|----------|-----|--------------|---------------|-----------|-------------------|
| FLM                  | 0.276    | 0.276 | 0.291 | 0.288 | 0.278 | 0.277 |
| CAM                  | 0.390    | 0.392 | 0.402 | 0.398 | 0.393 | 0.392 |
| Single Index         | 0.365    | 0.366 | 0.373 | 0.372 | 0.357 | 0.355 |
| Multiple Index       | 0.374    | 0.376 | 0.392 | 0.410 | 0.370 | 0.368 |
| Quadratic            | 0.363    | 0.328 | 0.344 | 0.338 | 0.322 | 0.324 |
| Complex Quadratic    | 0.394    | 0.392 | 0.403 | 0.398 | 0.392 | 0.393 |
the growth curves. In the speech recognition data, the predictor is a voice signal as a function of time and the response is the name of the vowel sounds pronounced, which can be regarded as a discrete random variable. We only take a subset of the data, similar to Li and Song (2017) and only consider two phonemes (response) transcribed as follows: “aa” as the vowel in “dark” and “ao” as the first vowel in “water.” The distinction between the two groups is not straightforward as seen in Figure 3, which shows the phoneme curves for each of the two groups computed by a log-periodogram of 150 (Epifanio 2008). We use the speech signals as functional predictors to classify the different phonemes.

In the Fat Spectrum data, we have 216 samples of which 176 are for training (40 samples for validation) and 40 samples are kept aside for testing. The response is the fat content value. It should be noted that the spectra are finely sampled, leading to very smooth curves as seen in Figure 3. Next, we look into the growth curves from Berkeley Growth Study. The heights (in cm) of the children in the dataset are measured from 1 to 18 years at 31 (m=31) timepoints. We divide the data into 73 and 20 samples for the training (20 samples for validation) and testing, respectively. We take the response as Gender. In the last dataset, we divide the data into 640 and 160 samples for training (160 samples for validation) and testing, respectively, and the response is the phoneme. We can see from Table 5 that FDNN and FBNN are performing much better compared to all the traditional methods in the Spectrometric data for predicting the fat content of the meat sample. For the two classification problems, we see a similar trend where our methods have lower classification error compared to other methods whether the sample size is small or large. The true nature of the relation between the functional predictors and the response is never known in real world settings but our methods can learn any kinds of patterns and excel in such situations as seen in Table 5.

4. Discussion and Conclusions

In this article, a novel approach for complex functional regression with a scalar response has been developed. It is a natural extension of a neural network to functional data. We use continuous hidden layers with continuous neurons, allowing us to capture temporal dependency and complicated relationships. We have devised two ways to implement this approach, FDNN and FBNN. Our approach is flexible as we don’t assume a particular structure that limits the kinds of nonlinear patterns learned, especially, interactions within time points for the predictors.

Understanding both the methods further will help us determine when it is more beneficial to use FDNN or FBNN. Both, in general, seem to be working on par with the best existing methods under simple cases but are outperforming in more complicated nonlinear settings. We achieve superior results even with such shallow networks. The proposed approach is also appealing in practice as it provides a way for researchers to systematically conduct deep learning tasks when they have functional predictors. The main advantage of FDNN over FBNN is

---

**Figure 3.** Sample curves for each of the three datasets.

**Table 5.** Comparison of different methods on the real data using RMSE (Fat Spectrum) and classification error (Growth and Phonemes).

| Data       | FLM  | CAM  | NN   | FNN  | FDNN | FBNN |
|------------|------|------|------|------|------|------|
| Fat Spectrum | 2.517 | 5.103 | 1.249 | 2.777 | 0.822 | 1.197 |
| Growth    | 0.150 | 0.150 | 0.450 | 0.450 | 0.100 | 0.100 |
| Phonemes  | 0.187 | 0.231 | 0.237 | 0.232 | 0.156 | 0.160 |
that it is straightforward to apply on dense and nicely gridded data, works on the same ideas as a neural network and requires less tuning, so it is a good approach to start with. FBNN is more flexible as it can deal with irregular samples but is dependent on the imputation quality as well as the selection of the basis functions. FBNN learns basis weights allowing for greater parsimony with very dense grids. Further theoretical investigation of these approaches is needed. Hence, an avenue for future work is developing statistical theory to give these methods sound backing. Also, the performance of FBNN in a sparse setting can be of interest but needs nontrivial extensions.

Lastly, Functional regression has been studied extensively (especially the scalar-on-function models) and nonlinear modeling for functional data has received more attention recently. But there are still many questions that remain open, in particular function-on-function models. The applicability of such models is of huge significance as they can be used in many areas including time series, speech analysis, and computer vision. There is a need for a more sophisticated and flexible function-on-function model in such domains, as functional data becomes more prevalent in today's world. Hence, extending our work to the function-on-function case will enable us to model functional predictors with functional response and help capture their complex relation.

### Supplemental Materials

**Title:** Nonlinear Functional Modeling using Neural Networks-supplementary materials (pdf).

**Section 1:** We share the mathematical details for the functional gradients related to FDNN.

**Section 2:** We share results demonstrating the power of early stopping in reducing over-fitting in our approach.

**Section 3:** We share the log-likelihood values under different simulation setting for the binary response.

**Section 4:** We compare Neural Network and Functional Neural Network to our approach under the same parameter setting.

### Funding

This research was supported in part by the following grant to Pennsylvania State University: NSF SES-1853209

### References

Asadi, E., Silva, M. G., Antunes, C. H., Dias, L., and Glicksman, L. (2014), "Multi-Objective Optimization for Building Retrofit: A Model Using Genetic Algorithm and Artificial Neural Network and an Application," *Energy and Buildings*, 81, 444–456. [1249]

Ayinde, B. O., Inanc, T., and Zarada, J. M. (2019), "Regularizing Deep Neural Networks by Enhancing Diversity in Feature Extraction," *IEEE Transactions on Neural Networks and Learning Systems*, 30, 2650–2661. [1253]

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016), "Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4960–4964. [1249]

Chen, D., Hall, P., and Müller, H. (2012), "Single and Multiple Index Functional Regression Models with Nonparametric Link," *Annals of Statistics*, 39, 1720–1747. [1248,1250]

Chen, Z., and Huang, X. (2017), "End-to-End Learning for Lane Keeping of Self-Driving Cars," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1856–1860. [1249]

Conan-Guez, B., and Rossi, F. (2002), "Multi-Layer Perceptrons for Functional Data Analysis: A Projection Based Approach," in *ICANN*. [1249]

Dorffner, G. (1996), "Neural Networks for Time Series Processing," *Neural Network World*, 6, 447–468. [1248]

Eilers, P. H., Li, B., and Marx, B. D. (2009), "Multivariate Calibration with Single-Index Signal Regression," *Chemometrics and Intelligent Laboratory Systems*, 96, 196–202. Chimiometrie 2007, Lyon, France, 29–30 November 2007. [1248,1250]

Epifanio, I. (2008), "Shape Descriptors for Classification of Functional Data," *Technometrics*, 50, 284–294. [1255]

Fan, Y., James, G. M., and Radchenko, P. (2015), "Functional Additive Regression," *Annals of Statistics*, 43, 2296–2325. [1248,1250]

Ferraty, F. (2011), *Recent Advances in Functional Data Analysis and Related Topics*, Heidelberg: Springer. [1248]

Ferraty, F., Goia, A., Salinelli, E., and Vieu, P. (2013), "Functional Projection Pursuit Regression," *TEST*, 22, 293–320. [1248,1250]

Ferraty, F., and Romain, Y. (2011), *The Oxford Handbook of Functional Data Analysis*, Oxford: Oxford University Press. [1248]

Ferraty, F., and Vieu, P. (2006), *Nonparametric Functional Data Analysis: Theory and Practice* (*Springer Series in Statistics*), Berlin, Heidelberg: Springer-Verlag. [1248,1254]

Geenens, G. (2011), "Curse of Dimensionality and Related Issues in Nonparametric Functional Regression," *Statistics Surveys*, 5, 30–43. [1248]

Groes, A., Mohamed, A., and Hinton, G. (2013), "Speech Recognition with Deep Recurrent Neural Networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649. [1249]

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. (2015), "Draw: A Recurrent Neural Network for Image Generation," in *Proceedings of the 32nd International Conference on Machine Learning*, eds. F. Bach and D. Blei, PMLR, vol. 37, pp. 1462–1471. Available at [https://proceedings.mlr.press/v37/gregor15.html](https://proceedings.mlr.press/v37/gregor15.html) [1249]

Horváth, L., and Kokoszka, P. (2012), *Inference for Functional Data with Applications*, Springer Series in Statistics, New York: Springer. [1248]

James, G., and Silverman, B. (2004), "Functional Adaptive Model Estimation," *Journal of the American Statistical Association*, 100, 565–576. [1248]

Joshi, A., Khosravy, M., and Gupta, N. (2020), *Machine Learning for Predictive Analysis: Proceedings of ICTIS 2020*, Lecture Notes in Networks and Systems. Singapore: Springer. [1249]

Kanellopoulos, I., and Wilkinson, G. G. (1997), "Strategies and Best Practice for Neural Network Image Classification," *International Journal of Remote Sensing*, 18, 711–725. [1249]

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014), "Large-Scale Video Classification with Convolutional Neural Networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732. [1249]

Khashei, M., and Bijari, M. (2010), "An Artificial Neural Network (p,d,q) Model for Timeseries Forecasting," *Expert Systems with Applications*, 37, 479–489. [1248]

Kokoszka, P., and Reimherr, M. (2018), *Introduction to Functional Data Analysis*, New York: Chapman and Hall/CRC. [1248,1249,1250,1252]

Li, B., and Song, J. (2017), "Nonlinear Sufficient Dimension Reduction for Functional Data," *Annals of Statistics*, 45, 1059–1095. [1255]

Ludwig, O., Nunes, U., and Araujo, R. (2014), "Eigenvalue Decay: A New Method for Neural Network Regularization," *Neurocomputing*, 124, 33–42. [1253]

Ma, H., and Zhu, Z. (2016), "Continuously Dynamic Additive Models for Functional Data," *Journal of Multivariate Analysis*, 150, 1–13. [1248,1250]

Maller, H., Wu, Y., and Yao, F. (2013), "Continuously Additive Models for Nonlinear Functional Regression," *Biometrika*, 100, 607–622. [1248,1250]

Maqueda, A. I., Loquercio, A., Gallego, G., García, N., and Scaramuzza, D. (2018), "Event-based Vision Meets Deep Learning on Steering Prediction for Self-Driving Cars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [1249]

McLean, M. W., Hooker, G., Staicu, A., Scheipl, F., and Ruppert, D. (2014), "Functional Generalized Additive Models," *Journal of Computational and Graphical Statistics*, 23, 249–269. [1248,1250]

Morchid, M. (2018), "Parsimonious Memory Unit for Recurrent Neural Networks with Application to Natural Language Processing," *Neurocomputing*, 314, 48–64. [1249]

Morris, J. (2014), "Functional Regression," *Annual Review of Statistics and Its Application*, 2, 321–359. [1248]
Muller, H., and Yao, F. (2008), "Functional Additive Models," Journal of the American Statistical Association, 103, 1534–1544. [1250]
Oliver, P. J. (2019), Introduction to the Calculus of Variations, London: Imperial College Press. [1251]
Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., and Cottrell, G. (2017), "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction," in Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, IJCAI'17, AAAI Press, pp. 2627–2633. Available at https://dl.acm.org/doi/10.5555/3172077.3172254 [1248]
Ramsay, J., and Silverman, B. (1997), Functional Data Analysis, Springer Series in Statistics, New York: Springer. [1248,1250]
Rao, A. R., Wang, Q., Wang, H., Khorasgani, H., and Gupta, C. (2020), "Spatio-Temporal Functional Neural Networks," in 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), pp. 81–89. [1249]
Rossi, F., and Conan-Guez, B. (2006), "Theoretical Properties of Projection based Multilayer Perceptrons with Functional Inputs," Neural Processing Letters, 23, 55–70. [1249]
Rossi, F., Conan-Guez, B., and Fleuret, F. (2002), "Functional Data Analysis with Multi Layer Perceptrons," in Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN’02 (Cat. No.02CH37290) (Vol. 3), pp. 2843–2848. [1249,1250,1251]
Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986), Learning Internal Representations by Error Propagation, Cambridge, MA: MIT Press, pp. 318–362. [1250]
Seide, F., Li, G., and Yu, D. (2012), "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks," in Proceedings of the 29th International Conference on International Conference on Machine Learning, Edinburgh, Scotland, ICML’12, Madison, WI: Omnipress, pp. 437–440. Available at https://dl.acm.org/doi/abs/10.5555/3042573.3042574 [1249]
Shi, W., Caballero, J., Huszar, F., Totz, J., Atkin, A. P., Bishop, R., Rueckert, D., and Wang, Z. (2016), "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-pixel Convolutional Neural Network," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [1249]
Singh, A., and Singh, K. K. (2017), "Satellite Image Classification using Genetic Algorithm Trained Radial Basis Function Neural Network, Application to the Detection of Flooded Areas," Journal of Visual Communication and Image Representation, 42, 173–182. [1249]