Supplementary Document for the Manuscript entitled "Functional Nonlinear Learning"

S1 An Example of Definition 1

Here is an example of Definition 1 in the main article. Suppose we have a single data pair $(\mathbf{x}_1 \equiv x_1(t), y_1)$ where $\mathbf{x}_1 \equiv x_1(t)$ represents the functional input and y_1 indicates the class of input $x_1(t)$. Furthermore, we assume only two classes of y, i.e., $\{0, 1\}$. For the single data pair, we assume $y_1 = 1$. If we specify that the margin $\gamma = 0.1$, we can obtain the loss as

$$\begin{aligned} l_\gamma(f_{\mathbf{w}}) &= \mathbb{P}_{(x,y)\sim\mathcal{D}}\Bigg[f_{\mathbf{w}}(\boldsymbol{x})[y] \leq \gamma + \max_{j\neq y}\{f_{\mathbf{w}}(\boldsymbol{x})[j]\}\Bigg] \\ &= \mathbb{P}_{(x,y)\sim\mathcal{D}}\Bigg[f_{\mathbf{w}}(\boldsymbol{x}_1)[1] \leq 0.1 + f_{\mathbf{w}}(\boldsymbol{x}_1)[0]\Bigg]. \end{aligned}$$

S2 Proof of the generalization bound

A generalization bound for any classifier can be specified from Lemma 1 in Neyshabur et al. (2018).

Lemma 1. *Let $f_{\boldsymbol{w}}(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ be any predictor (not necessarily a neural network) with parameters $\boldsymbol{w}$, and P be any distribution on the parameters that is independent of the training data. Then, for any γ, $\delta > 0$, with probability $\geq 1-\delta$ over the training set of size m, for any $\boldsymbol{w}$, and any random perturbations $\boldsymbol{u}$ s.t. $\mathbb{P}_{\boldsymbol{u}}[max_{\boldsymbol{x}\in\mathcal{X}}|f_{\boldsymbol{w}+\boldsymbol{u}}(\boldsymbol{x}) - f_{\boldsymbol{w}}(\boldsymbol{x})|_\infty < \frac{\gamma}{4}] \geq \frac{1}{2}$, we have*

$$l_0(f_{\boldsymbol{w}}) \leq \hat{l}_\gamma(f_{\boldsymbol{w}}) + 4\sqrt{\frac{KL(\boldsymbol{w}+\boldsymbol{u}||P) + ln(6m/\delta)}{m-1}}. \tag{S1}$$

Let $f_{\mathbf{w}}(\mathbf{x}) : \mathcal{X} \to \mathbb{R}^Q$ be the classifier based on some covariates $\mathbf{x} \in \mathcal{X}$ and Q is the number of classes in $\mathbf{y}$. The output of $f_{\mathbf{w}}(\mathbf{x})$ is a Q-dimensional vector. Each element in the vector is a normalized value by which the maximum will be used to produce the label of the input $\mathbf{x}$. Then, $f_{\mathbf{w}}(\mathbf{x})[y]$ is the prediction (a normalized value) of class y in $\mathbf{y}$. In order to establish a generalization bound on the proposed method, we need the following assumptions.

Assumption 1. For each dimension of the multivariate function $\mathbf{X}(t) = (X_1(t), ..., X_D(t))$, $||X_d(t)||_\infty < B_{X_d}$, and $||\mathbf{x}_j||_2 < B_x = \sum_{d=1}^{D} B_{X_d}$ where $\mathbf{x}_j$ is $(X_1(t_j), ..., X_D(t_j))$.

Assumption 2. Activation functions $g(\cdot)$'s are ρ-Lipschitz, e.g., $g_h(||\boldsymbol{a}-\boldsymbol{a}'||) \leq \rho_h||\boldsymbol{a}-\boldsymbol{a}'||$.

Assumption 3. Coefficient matrices are bounded by $B_W, B_U, B_V, B_M, B_{\tilde{W}}, B_{\tilde{U}}, B_G$.

In preparation for proving the main theorem, we also need the following lemmas based on the order of propagation in model (2.1). First, a bound on the hidden states $\boldsymbol{h}_j$ is defined as follows:

Lemma 2.

$$\beta_{h_j} = \rho_h B_W B_x \frac{1-(\rho_h B_U)^j}{1-\rho_h B_U}$$

Proof.

$$\begin{aligned}
||\boldsymbol{h}_j|| &= ||\sigma_h(\boldsymbol{W}\boldsymbol{x}_j + \boldsymbol{U}\boldsymbol{h}_{j-1})|| \\
&\leq \rho_h(||\boldsymbol{W}\boldsymbol{x}_j + \boldsymbol{U}\boldsymbol{h}_{j-1}||) \\
&\leq \rho_h(||\boldsymbol{W}||\boldsymbol{x}_j|| + ||\boldsymbol{U}||||\boldsymbol{h}_{j-1}||) \\
&= \rho_h(B_W B_x + B_U||\boldsymbol{h}_{j-1}) \\
&\leq \rho_h(B_W B_x + B_U(\rho_h(||\boldsymbol{W}\boldsymbol{x}_{j-1} + \boldsymbol{U}\boldsymbol{h}_{j-2}||))) \\
&= \rho_h B_W B_x + \rho_h B_W B_x \rho_h B_U + \rho_h B_W B_x(\rho_h B_U)^2 + \cdots \\
&= \rho_h B_W B_x \sum_{l=0}^{j-1} (\rho_h B_U)^l \\
&= \rho_h B_W B_x \frac{1-(\rho_h B_U)^j}{1-\rho_h B_U}
\end{aligned}$$

□

Subsequently, we can specify the lemma for a bound on $||\boldsymbol{h}_{j1} - \boldsymbol{h}'_j||$

Lemma 3. $||\boldsymbol{h}_j - \boldsymbol{h}'_j|| \leq \rho_h B_x \frac{1-(\rho_h B_U)^j}{1-(\rho_h B_U)} ||\boldsymbol{W} - \boldsymbol{W}'|| + j\rho_h \beta_{h_j} ||\boldsymbol{U} - \boldsymbol{U}'||$ *with initializations* $\boldsymbol{h}_0, \boldsymbol{h}'_0 = \boldsymbol{0}$.

Proof.

$$
\begin{aligned}
||\boldsymbol{h}_j - \boldsymbol{h}'_j|| &= ||\sigma_h(\boldsymbol{W}\boldsymbol{x}_j + \boldsymbol{U}\boldsymbol{h}_{t-1}) - \sigma_h(\boldsymbol{W}'\boldsymbol{x}_j + \boldsymbol{U}'\boldsymbol{h}'_{j-1})|| \\
&\leq \rho_h(||\boldsymbol{W}\boldsymbol{x}_j - \boldsymbol{W}'\boldsymbol{x}_j + \boldsymbol{U}\boldsymbol{h}_{j-1} - \boldsymbol{U}'\boldsymbol{h}'_{j-1}||) \\
&\leq \rho_h(||\boldsymbol{W}\boldsymbol{x}_j - \boldsymbol{W}'\boldsymbol{x}_j + \boldsymbol{U}\boldsymbol{h}_{j-1} - \boldsymbol{U}'\boldsymbol{h}_{j-1} + \boldsymbol{U}'\boldsymbol{h}_{j-1} - \boldsymbol{U}'\boldsymbol{h}'_{j-1}||) \\
&\leq \rho_h(||\boldsymbol{W} - \boldsymbol{W}'||B_x + ||\boldsymbol{U} - \boldsymbol{U}'||||\boldsymbol{h}_{j-1}|| + B_U||\boldsymbol{h}_{j-1} - \boldsymbol{h}'_{j-1}||) \\
&\leq \rho_h(||\boldsymbol{W} - \boldsymbol{W}'||B_x + ||\boldsymbol{U} - \boldsymbol{U}'||\beta_{h_{j-1}} \\
&\qquad + B_U(\rho_h(||\boldsymbol{W} - \boldsymbol{W}'||B_x + ||\boldsymbol{U} - \boldsymbol{U}'||\beta_{h_{j-2}} + B_U||\boldsymbol{h}_{j-2} - \boldsymbol{h}'_{j-2}||))) \\
&= \rho_h B_x||\boldsymbol{W} - \boldsymbol{W}'|| + \rho_h\beta_{h_{j-1}}||\boldsymbol{U} - \boldsymbol{U}'|| + \rho_h^2 B_x B_U||\boldsymbol{W} - \boldsymbol{W}'|| + \rho_h^2\beta_{h_{j-2}}B_U||\boldsymbol{U} - \boldsymbol{U}'|| \\
&\qquad + (\rho_h B_U)^2||\boldsymbol{h}_{j-2} - \boldsymbol{h}^2_{j-2}|| \\
&\leq \rho_h B_x||\boldsymbol{W} - \boldsymbol{W}'||\sum_{l=0}^{j-1}(\rho_h B_U)^l + \rho_h||\boldsymbol{U} - \boldsymbol{U}'||\sum_{l=0}^{j-1}(\rho_h B_U)^{j-1-l}\beta_{h_l} \\
&= \rho_h B_x||\boldsymbol{W} - \boldsymbol{W}'||\sum_{l=0}^{j-1}(\rho_h B_U)^l + j\rho_h\beta_{h_j}||\boldsymbol{U} - \boldsymbol{U}'|| \\
&= \rho_h B_x\frac{1 - (\rho_h B_U)^j}{1 - (\rho_h B_U)}||\boldsymbol{W} - \boldsymbol{W}'|| + j\rho_h\beta_{h_j}||\boldsymbol{U} - \boldsymbol{U}'||
\end{aligned}
$$

where

$$
\begin{aligned}
\sum_{l=0}^{j-1}(\rho_h B_U)^{j-1-l}\beta_{h_l} &= \sum_{l=0}^{j-1}(\rho_h B_U)^{j-1-l}\rho_h B_x B_W(\sum_{k=0}^{l-1}(\rho_h B_U)^k) \\
&= \rho_h B_x B_W((\rho_h B_U)^0\sum_{j=0}^{j-2}(\rho_h B_U)^j + (\rho_h B_U)^1\sum_{j=0}^{j-3}(\rho_h B_U)^j + ...) \\
&\leq j\rho_h B_x B_W\sum_{l=0}^{j-1}(\rho_h B_U)^l \\
&= j\rho_h B_x B_W\frac{1 - (\rho_h B_U)^j}{1 - (\rho_h B_U)} = j\rho_h\beta_{h_j}
\end{aligned}
$$

□

Lemma 3 institutes the bound on latent features which will be used for the classification.

Lemma 4. *The difference in latent features $||\boldsymbol{z} - \boldsymbol{z}'||$ with respect to perturbation in param-*

eters is bounded by $\rho_z||\boldsymbol{V}-\boldsymbol{V}'||\beta_{h_J}+\rho_z B_V\rho_h B_x\frac{1-(\rho_h B_U)^j}{1-(\rho_h B_U)}||\boldsymbol{W}-\boldsymbol{W}'||+j\rho_h\beta_{h_j}||\boldsymbol{U}-\boldsymbol{U}'||$.

Proof.

$$\begin{aligned}
||\boldsymbol{z}-\boldsymbol{z}'|| &= ||\sigma_z(\boldsymbol{V}\boldsymbol{h}_J)-\sigma_z(\boldsymbol{V}'\boldsymbol{h}_J')|| \\
&\le \rho_z||\boldsymbol{V}\boldsymbol{h}_J-\boldsymbol{V}'\boldsymbol{h}_J+\boldsymbol{V}'\boldsymbol{h}_J-\boldsymbol{V}'\boldsymbol{h}_J'|| \\
&\le \rho_z(||\boldsymbol{V}-\boldsymbol{V}'||\beta_{h_J}+B_V||\boldsymbol{h}_J-\boldsymbol{h}_J'||) \\
&\le \rho_z||\boldsymbol{V}-\boldsymbol{V}'||\beta_{h_J}+\rho_z B_V\rho_h B_x\frac{1-(\rho_h B_U)^J}{1-(\rho_h B_U)}||\boldsymbol{W}-\boldsymbol{W}'||+J\rho_h\beta_{h_J}||\boldsymbol{U}-\boldsymbol{U}'||
\end{aligned}$$

□

With Lemma 4, we can work on the prediction difference from the model trained by different data sets.

Lemma 5. $||\boldsymbol{y}-\boldsymbol{y}'||\le H_{y,M}||\boldsymbol{M}-\boldsymbol{M}'||+H_{y,V}||\boldsymbol{V}-\boldsymbol{V}'||+H_{y,W}||\boldsymbol{W}-\boldsymbol{W}'||+H_{y,U}||\boldsymbol{U}-\boldsymbol{U}'||$ *where* $H_{y,M}=\rho_y\rho_z B_V\beta_{h_J}$, $H_{y,V}=\rho_y\rho_z B_M\beta_{h_J}$,$H_{y,W}=\rho_z B_V\rho_h B_x\frac{1-(\rho_h B_U)^J}{1-(\rho_h B_U)}$ *and* $H_{y,U}=J\rho_h\beta_{h_J}$.

Proof.

$$\begin{aligned}
||\boldsymbol{y}-\boldsymbol{y}'|| &= ||\rho_y(\boldsymbol{M}\boldsymbol{z})-\rho_y(\boldsymbol{M}'\boldsymbol{z}')|| \\
&\le \rho_y||\boldsymbol{M}\boldsymbol{z}-\boldsymbol{M}'\boldsymbol{z}'|| \\
&\le \rho_y(||\boldsymbol{M}\boldsymbol{z}-\boldsymbol{M}'\boldsymbol{z}||+||\boldsymbol{M}'\boldsymbol{z}-\boldsymbol{M}'\boldsymbol{z}'||) \\
&\le \rho_y B_M||\boldsymbol{z}-\boldsymbol{z}'||+\rho_y\rho_z B_V\beta_{h_J}||\boldsymbol{M}-\boldsymbol{M}'|| \\
&\le \rho_y B_M\rho_z||\boldsymbol{V}-\boldsymbol{V}'||\beta_{h_J}+\rho_z B_V\rho_h B_x\frac{1-(\rho_h B_U)^J}{1-(\rho_h B_U)}||\boldsymbol{W}-\boldsymbol{W}'||+J\rho_h\beta_{h_J}||\boldsymbol{U}-\boldsymbol{U}'|| \\
&\quad +\rho_y\rho_z B_V\beta_{h_J}||\boldsymbol{M}-\boldsymbol{M}'||
\end{aligned}$$

□

The proof of the main theorem can be provided by using the previous lemmas.

Proof. Let $P = \mathrm{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$ be the prior distribution for coefficient matricies $\boldsymbol{w} = (\boldsymbol{M}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{U})$ and pertubations $\boldsymbol{u} = (\delta\boldsymbol{M}, \delta\boldsymbol{V}, \delta\boldsymbol{W}, \delta\boldsymbol{U})$. The bound of $\boldsymbol{u}$ is equivalent to the bound of the union of all coefficients, hence $||\delta\boldsymbol{M}||, ||\delta\boldsymbol{V}||, ||\delta\boldsymbol{W}||, ||\delta\boldsymbol{U}|| < \sqrt{2k\sigma^2 \ln 16k}$ with probability at least $\frac{1}{2}$ given the following bound

$$\mathbb{P}_{\delta\boldsymbol{M} \sim \mathrm{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_{k*k})}(||\delta\boldsymbol{M}|| < \epsilon) \geq 2k \exp -\frac{\epsilon^2}{2k\sigma^2}$$

as well as on other matrices. Using the spectral norm bounds on the perturbations and substituting into Lemma 5, we get

$$\begin{aligned}
||\boldsymbol{y} - \boldsymbol{y}'|| &\leq H_{y,M}||\boldsymbol{M} - \boldsymbol{M}'|| + H_{y,V}||\boldsymbol{V} - \boldsymbol{V}'|| + H_{y,W}||\boldsymbol{W} - \boldsymbol{W}'|| + H_{y,U}||\boldsymbol{U} - \boldsymbol{U}'|| \\
&= H_{y,M}||\delta\boldsymbol{M}|| + H_{y,V}||\delta\boldsymbol{V}|| + H_{y,W}||\delta\boldsymbol{W}|| + H_{y,U}||\delta\boldsymbol{U}|| \\
&= (H_{y,M} + H_{y,V} + H_{y,W} + H_{y,U})\sqrt{2k\sigma^2 \ln 16k} \leq \frac{\gamma}{4}
\end{aligned}$$

with probability at least $\frac{1}{2}$ where $\sigma = \frac{\gamma}{4(H_{y,M}+H_{y,V}+H_{y,W}+H_{y,U})\sqrt{2k\ln 16k}}$. The KL-divergence term in Lemma (1) can be obtained as

$$\begin{aligned}
&D(\boldsymbol{w} + \boldsymbol{u}||P) \leq \frac{||\boldsymbol{w}||^2}{2\sigma^2} \\
&\leq \mathcal{O}(\frac{(H_{y,M} + H_{y,V} + H_{y,W} + H_{y,U})k\ln k}{\gamma^2}(\frac{||\boldsymbol{M}||^2}{||\boldsymbol{M}||_F^2} + \frac{||\boldsymbol{V}||^2}{||\boldsymbol{V}||_F^2} + \frac{||\boldsymbol{W}||^2}{||\boldsymbol{W}||_F^2} + \frac{||\boldsymbol{U}||^2}{||\boldsymbol{U}||_F^2}))
\end{aligned}$$

□

S3 Modify network complexity

S3.1 Stacking of neural networks

The specification (2.1) only uses 1-layer networks to estimate nonlinear functions. However, it may not suffice for more complex functional data. For that matter, we can add more layers into networks for better approximations of target functions. The stacking of recurrent neural networks takes the following form:

$$\boldsymbol{h}_j^l = \begin{cases} g_h(\boldsymbol{W}^1 \boldsymbol{x}_j + \boldsymbol{U}^1 \boldsymbol{h}_{j-1}), & \text{for} \quad l = 1, \\ g_h(\boldsymbol{W}^l \boldsymbol{h}_j^{l-1} + \boldsymbol{U}^l \boldsymbol{h}_{j-1}^l), & \text{for} \quad l = 2, ..., L. \end{cases}$$

where l corresponds to the layer index of recurrent networks. The stacked recurrent neural networks can be applied to the estimation of both ϕ and ψ. Adding layers to the network approximating f is straightforward as:

$$\mathbf{y}^l = \begin{cases} g_y(\mathbf{M}^1 \mathbf{z}), & \text{for} \quad q = 1, \\ g_y(\mathbf{M}^q \mathbf{y}^{q-1}), & \text{for} \quad q = 2, ..., Q. \end{cases}$$

where $q = 1, ..., Q$ indicates the number of hidden layers. For the remainder of this article, we will use 1-layer neural networks to describe the estimation process and illustrate the generalization property.

S3.2 Bidirectional network for handling functional data

The vanilla RNN only considers the forward continuity of functional data. We can add a reverse RNN layer in parallel to the forward layer such that the proposed method can use both past and future information for achieving continuity. Let $\mathbf{z} = (\vec{\mathbf{z}}, \overleftarrow{\mathbf{z}})$ be the latent vector representation as the concatenation of $\vec{\mathbf{z}}$ (using past information) and $\overleftarrow{\mathbf{z}}$ (using future information). For illustration, we demonstrate the implementation of RNN for approximating ϕ. In addition to $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{V}$, we will have addition parameter matrices $\overleftarrow{\mathbf{W}}$, $\overleftarrow{\mathbf{U}}$, and $\overleftarrow{\mathbf{V}}$. Hence, the bidirectional RNN for using both past and future information to achieve continuity has the specification as follows:

$$
\begin{aligned}
\mathbf{h}_j &= g_h(\mathbf{W}\mathbf{x}_j + \mathbf{U}\mathbf{h}_{j-1}) \\
\vec{\mathbf{z}} &= g_z(\mathbf{V}\mathbf{h}_J), \\
\overleftarrow{\mathbf{h}}_j &= g_h(\mathbf{W}\overleftarrow{\mathbf{x}}_j + \mathbf{U}\overleftarrow{\mathbf{h}}_{j-1}) \\
\overleftarrow{\mathbf{z}} &= g_z(\mathbf{V}\overleftarrow{\mathbf{h}}_J),
\end{aligned}
$$

where $\overleftarrow{\mathbf{x}}_j = \mathbf{x}_{J-j+1}$. As a result, the final $\mathbf{z} = (\vec{\mathbf{z}}, \overleftarrow{\mathbf{z}})$ is more comprehensive by using two RNNs to process information along both directions of the support. Equivalently, it can achieve continuity in both directions by using past and future information of trajectories. In applications on the `Earthquakes` and `Swedishleaf` data sets, the prediction accuracy was improved around 4% (over random data splittings) by using this bidirectional RNN to model functional data.

S4 Modified training process and regularization method

Distorting observations with noises can be viewed as a regularization method to avoid overfitting the data (Bishop (1995)). For simpler notations, we will assume that the corruption process is only adding random noises. Let a single function $u(x(t)) = \psi\phi(x(t))$ represent the reconstructed trajectory. It is corrupted by $\epsilon(t)$ with realizations $\epsilon_{1:J}$ at $t_1, ..., t_J$ such that $\mathbb{E}\epsilon_j = 0$ and $\mathbb{E}\epsilon_j\epsilon_{j'} = \sigma^2\delta(j = j')$ where $\delta(j = j') = 1$ if $t_j = t_{j'}$ and 0 otherwise. Then, the reconstruction loss (using squared error loss) of a single observation $x_{1:J}$ is

$$\tilde{\mathcal{L}}_r = ||x_{1:J} - u(x_{1:J} + \epsilon_{1:J})||^2. \tag{S2}$$

The quadratic approximation to $u(x_{1:J} + \epsilon_{1:J})$ at $x_{1:J}$ is

$$u(x_{1:J} + \epsilon_{1:J}) = u(x_{1:J}) + \nabla u|_{x_{1:J}}\epsilon_{1:J} + \frac{1}{2}(\epsilon_{1:J}\nabla)^2 \cdot u|_{x_{1:J}}.$$

Substituting the approximation of $u(x_{1:J} + \epsilon_{1:J})$ into S2 and taking the expectation with respect to $\epsilon_{1:J}$ gives

$$\begin{aligned}\mathbb{E}\tilde{\mathcal{L}}_r &= \frac{1}{N}\sum_{i=1}^{N} ||x^i_{1:J} - u(x^i_{1:J})||^2 \\ &+ \frac{\sigma^2}{N}\sum_{i=1}^{N}\{\sum_{s=1}^{T}[(\nabla u_s|_{x^i_{1:J}})^2 - (x^i_{1:J} - u(x^i_{1:J}))\nabla^2 u_s|_{x^i_{1:J}}]\} \\ &= \mathcal{L}_r + \sigma^2 P(u(\cdot)).\end{aligned} \tag{S3}$$

As a result, adding uncorrelated Gaussian errors to trajectories is equivalent to introducing a penalty term $P(u(\cdot))$ which controls the roughness of the reconstruction $u(\mathbf{x}(t))$. It resembles the roughness penalty in the smoothing splines method but does not have the analytic form of a proper penalty. Furthermore, $P(u(\cdot))$ is not bounded from below in comparison to a penalty taking the form of $\int[D^p(u(\cdot))]^2 d\cdot$. As suggested in Bishop (1995), using a neural network to estimate ϕ and ψ with a noisy layer is equivalent to minimizing with a standard Tikhonov penalty given that the σ^2, the variance of corruption noise, is small enough. By penalizing the roughness of the reconstructed trajectory, we are able to indirectly enforce the smoothness of both ϕ and ψ. As a result, the representation model is robust to noisy and corrupted observations.

S5 The gradients of parameters in the FunNoL model

$$
\begin{aligned}
\frac{d\mathcal{L}}{d\mathbf{M}} &= \frac{d\mathcal{L}_c}{d\mathbf{M}}, \\
\frac{d\mathcal{L}}{d\mathbf{G}} &= \frac{d\mathcal{L}_r}{d\mathbf{G}} = \sum_{j=1}^{J} \frac{dl_r(t_j)}{d\mathbf{G}}, \\
\frac{d\mathcal{L}}{d\tilde{\mathbf{W}}} &= \frac{d\mathcal{L}_r}{d\tilde{\mathbf{W}}} = \sum_{j=1}^{J} \frac{dl_r(t_j)}{d\tilde{\mathbf{W}}} \\
&\qquad \text{where } \frac{dl_r(t_j)}{d\tilde{\mathbf{W}}} = \sum_{s=0}^{j} \frac{dl_r(t_j)}{dd\tilde{\mathbf{h}}_j} \frac{d\tilde{\mathbf{h}}_j}{d\tilde{\mathbf{h}}_s} \frac{d\tilde{\mathbf{h}}_s}{d\tilde{\mathbf{W}}} \text{ and } \frac{d\tilde{\mathbf{h}}_j}{d\tilde{\mathbf{h}}_s} = \prod_{k=s+1}^{j} \frac{d\tilde{\mathbf{h}}_k}{d\tilde{\mathbf{h}}_{k-1}}, \\
\frac{d\mathcal{L}}{d\tilde{\mathbf{U}}} &= \frac{d\mathcal{L}_r}{d\tilde{\mathbf{U}}} = \sum_{j=1}^{J} \frac{dl_r(t_j)}{d\tilde{\mathbf{U}}} \\
&\qquad \text{where } \frac{dl_r(t_j)}{d\tilde{\mathbf{U}}} = \sum_{s=0}^{j} \frac{dl_r(t_j)}{dd\tilde{\mathbf{h}}_j} \frac{d\tilde{\mathbf{h}}_j}{d\tilde{\mathbf{h}}_s} \frac{d\tilde{\mathbf{h}}_s}{d\tilde{\mathbf{U}}} \text{ and } \frac{d\tilde{\mathbf{h}}_j}{d\tilde{\mathbf{h}}_s} = \prod_{k=s+1}^{j} \frac{d\tilde{\mathbf{h}}_k}{d\tilde{\mathbf{h}}_{k-1}}, \\
\frac{d\mathcal{L}}{d\mathbf{V}} &= \frac{d\mathcal{L}_c}{d\mathbf{V}} + \frac{d\mathcal{L}_r}{d\mathbf{V}} \\
&\qquad \text{where } \frac{d\mathcal{L}_r}{d\mathbf{V}} = \sum_{j=1}^{J} \frac{dl_r(t_j)}{d\mathbf{V}} \text{ and } \frac{dl_r(t_j)}{d\mathbf{V}} = \sum_{s=0}^{j} \frac{dl_r(t_j)}{d\tilde{\mathbf{h}}_j} \frac{d\tilde{\mathbf{h}}_j}{d\tilde{\mathbf{h}}_s} \frac{d\tilde{\mathbf{h}}_s}{d\mathbf{V}}, \\
\frac{d\mathcal{L}}{d\mathbf{W}} &= \frac{d\mathcal{L}_c}{d\mathbf{W}} + \frac{d\mathcal{L}_r}{d\mathbf{W}} \\
&\qquad \text{where } \frac{d\mathcal{L}_c}{d\mathbf{W}} = \frac{d\mathcal{L}_c}{d\mathbf{z}} \frac{d\mathbf{z}}{d\mathbf{W}} = \frac{d\mathcal{L}_c}{d\mathbf{z}} \frac{d\mathbf{z}}{d\mathbf{h}_J} \frac{d\mathbf{h}_J}{\mathbf{W}} = \frac{d\mathcal{L}_c}{d\mathbf{z}} \frac{d\mathbf{z}}{d\mathbf{h}_J} \Big(\sum_{s=1}^{J} \frac{d\mathbf{h}_J}{d\mathbf{h}_s} \frac{d\mathbf{h}_s}{d\mathbf{W}}\Big) \\
&\qquad\qquad \frac{d\mathcal{L}_r}{d\mathbf{W}} = \sum_{j=1}^{J} \frac{dl_r(t_j)}{d\mathbf{W}} = \sum_{s=0}^{j} \frac{dl_r(t_j)}{d\tilde{\mathbf{h}}_j} \frac{d\tilde{\mathbf{h}}_j}{d\tilde{\mathbf{h}}_s} \frac{d\tilde{\mathbf{h}}(s)}{d\mathbf{z}} \frac{d\mathbf{z}}{d\mathbf{h}_J} \Big(\sum_{l=1}^{J} \frac{d\mathbf{h}_J}{d\mathbf{h}_l} \frac{d\mathbf{h}_l}{d\mathbf{W}}\Big), \\
\frac{d\mathcal{L}}{d\mathbf{U}} &= \frac{d\mathcal{L}_c}{d\mathbf{z}} \frac{d\mathbf{z}}{d\mathbf{h}_J} \Big(\sum_{s=1}^{J} \frac{d\mathbf{h}_J}{d\mathbf{h}_s} \frac{d\mathbf{h}_s}{d\mathbf{U}}\Big) + \sum_{j=1}^{J} \sum_{s=0}^{j} \frac{dl_r(t_j)}{d\tilde{\mathbf{h}}_t} \frac{d\tilde{\mathbf{h}}_t}{d\tilde{\mathbf{h}}_s} \frac{d\tilde{\mathbf{h}}_s}{d\mathbf{z}} \frac{d\mathbf{z}}{d\mathbf{h}_J} \Big(\sum_{l=1}^{J} \frac{d\mathbf{h}_J}{d\mathbf{h}_l} \frac{d\mathbf{h}_l}{d\mathbf{U}}\Big).
\end{aligned}
$$

S6 Gradients of the long short-term memory model

$$\frac{dm_j}{dm_{j-1}} = \frac{\partial m_j}{\partial m_{j-1}} + \frac{\partial m_j}{\partial f_j}\frac{\partial f_j}{\partial h_{j-1}}\frac{\partial h_{j-1}}{\partial m_{j-1}} + \frac{\partial m_j}{\partial i_j}\frac{\partial i_j}{\partial h_{j-1}}\frac{\partial h_{j-1}}{\partial m_{j-1}} + \frac{\partial m_j}{\partial \tilde{m}_j}\frac{\partial \tilde{m}_j}{\partial h_{j-1}}\frac{\partial h_{j-1}}{\partial m_{j-1}},$$

where

$$\begin{aligned}
\frac{\partial m_j}{\partial f_j}\frac{\partial f_j}{\partial h_{j-1}}\frac{\partial h_{j-1}}{\partial m_{j-1}} &= m_{j-1}\boldsymbol{U}_{f,dd}g'(\boldsymbol{W}_f\boldsymbol{x}_j + \boldsymbol{U}_f\boldsymbol{h}_{j-1})_{[d]}o_j\mathrm{tanh}'(m_{j-1}),\\
\frac{\partial m_j}{\partial i_j}\frac{\partial i_j}{\partial h_{j-1}}\frac{\partial h_{j-1}}{\partial m_{j-1}} &= \tilde{m}_j\boldsymbol{U}_{i,dd}g'(\boldsymbol{W}_i\boldsymbol{x}_j + \boldsymbol{U}_i\boldsymbol{h}_{j-1})_{[d]}o_j\mathrm{tanh}'(m_{j-1}),\\
\frac{\partial m_j}{\partial \tilde{m}_j}\frac{\partial \tilde{m}_j}{\partial h_{j-1}}\frac{\partial h_{j-1}}{\partial m_{j-1}} &= i_j\boldsymbol{U}_{m,dd}\mathrm{tanh}'(\boldsymbol{W}_m\boldsymbol{x}_j + \boldsymbol{U}_m\boldsymbol{h}_{j-1})_{[d]}o_j\mathrm{tanh}'(m_{j-1}).
\end{aligned}$$

Hence

$$\begin{aligned}
\frac{dm_j}{dm_{j-1}} = f_j + o_j\mathrm{tanh}'(m_{j-1})[&m_{j-1}\boldsymbol{U}_{f,dd}g'(\boldsymbol{W}_f\boldsymbol{x}_j + \boldsymbol{U}_f\boldsymbol{h}_{j-1})_{[d]}\\
&+ \tilde{m}_j\boldsymbol{U}_{i,dd}g'(\boldsymbol{W}_i\boldsymbol{x}_j + \boldsymbol{U}_i\boldsymbol{h}_{j-1})_{[d]}\\
&+ i_j\boldsymbol{U}_{m,dd}\mathrm{tanh}'(\boldsymbol{W}_m\boldsymbol{x}_j + \boldsymbol{U}_m\boldsymbol{h}_{j-1})_{[d]}].
\end{aligned}$$

The recursive term dm_j/dm_{j-1} is varying at any t_j and will not be bounded to a value that eventually leads to vanishing/exploding gradients. Instead, it relies on the gating functions to adjust the propagation of training errors and the long-term history dependency of hidden states.

S7 Benchmark models

This section explains the benchmark models used for comparing with the proposed FunNoL method.

S7.1 Univariate FPCA

Let $X(t)$ be a functional variable which is square integratable over the interval $[a, b]$. For simplicity, we assume the mean function $\mathbb{E}X(t) = 0$ for all t. Let $\{x_i(t)\}_{i=1}^N$ be the sample functions of X(t), and each sample function has discrete observations y_{ij} for $j = 1, ..., J$ such that

$$y_{ij} = x_i(t_j) + \epsilon_{ij}$$

where we assume $t_1 = a$ and $t_J = b$ and all $x_i(t)$'s are observed on the same set of points $t_1, ..., t_J$. ϵ_{ij}'s are mutually independent random errors with mean 0 and variance σ^2. Let $C(s, t)$ be the covariance function (a semi-positive kernel) of $X(t)$, and the Mercer's theorem states that

$$C(s,t) = \sum_{k=1}^{\infty} \lambda_k \phi_k(s)\phi_k(t)$$

where λ_k and $\phi_k(t)$ are the k-th eigenvalue and eigenfunction. This also admits the Karhunen-Loeve representation of random functions, i.e.,

$$X(t) = \sum_{k=1}^{\infty} \xi_k \phi_k(t)$$

where ξ_k's are the FPC scores. By limiting the upper limit of the summation to a fixed number K, we can use a K-dimensional vector $(\xi_{i1}, ..., \xi_{iK})$ to represent individual trajectory through the following

$$x_i(t) = \sum_{k=1}^{K} \xi_{ik} \phi_k(t)$$

Both the covariance function $C(s, t)$ and its eigenfunctions $\phi_k(t)$'s are smooth functions, and we will use local linear regression to estimate $C(s, t)$. Let $K : \mathbb{R} \to \mathbb{R}$ be a density kernel function over $[-1, 1]$ and b be the bandwidth of the kernel. The covariance function

$\hat{C}(s,t) = \hat{\beta}_0$ is the estimated intercept of a local linear plane, where $(\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2)$ are obtained by minimizing

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{\substack{1\le j,j'\le J\\ j\neq j'}}[y_{ij}y_{ij'} - \beta_0 - \beta_1(s-t_j) - \beta_2(t-t_{j'})]^2 K(\frac{t_j - s}{b})K(\frac{t_{j'}-t}{b})$$

The eigenvalues and eigenfunctions are obtained by solving the following equation

$$\langle C(s,t), \phi_k(t)\rangle_{\mathbb{H}} = \lambda_k \phi_k(t)$$

where $\langle \phi_k(t), \phi_{k'}(t)\rangle_{\mathbb{H}} = \delta_{kk'}$. The indicator $\delta_{kk'} = 1$ if $k = k'$ and 0 otherwise. We obtain the estimates of $\hat{\lambda}_k$ and $\hat{\phi}_k(t)$ by discretization of the smooth covariance function. Let the matrix $\hat{\mathbf{C}}$ be the evaluation of $\hat{C}(s,t)$ over some arbitrary points, then it is straightforward to solve the following eigenequation:

$$\hat{\mathbf{C}}\hat{\phi}_k = \hat{\lambda}_k\hat{\phi}_k$$

where $\hat{\phi}_k$ is the estimated eigenfunction over evaluation points.

There are two scenarios for obtaining the FPC scores for individual trajectory: when data are fully observed or sparsely observed. First, the k-th FPC score for i-th function can be obtained by numerical integration of the following:

$$\hat{\xi}_{ik} = \langle x_i(t), \hat{\phi}_k(t)\rangle_{\mathbb{H}} = \int_a^b x_i(t)\hat{\phi}_k(t)dt$$

When functions are sparsely observed, the above integration are not satisfactory to estimate $\hat{\xi}_{ik}$'s. Instead, we can use the conditional expectation approach in Yao et al. (2005). That is, we assume the FPC score $\{\xi_{ik}\}_{k=1}^{K}$ and random errors $\{\epsilon_{ij}\}_{j=1}^{J}$ follow a joint Gaussian distribution, then

$$\hat{\xi}_{ik} = \mathbb{E}[\xi_{ik}|\mathbf{y}_i] = \hat{\lambda}_k\hat{\phi}_{ik}^T\mathbf{\Omega}_i^{-1}\mathbf{y}_i$$

where $\mathbf{y}_i = (y_{i1}, ..., y_{iJ})^T$, $\mathbf{\Omega}_i = \mathrm{cov}(\mathbf{y}_i, \mathbf{y}_i)$, and $\hat{\phi}_{ik}$ is a vector contains evaluation of $\hat{\phi}_k(t)$ over observation points of i-th trajectory. The selection of K will be based on the selected dimension of latent representation from the proposed FunNoL method, so that we can compare two vector spaces of the same dimension.

S7.2 Multivariate FPCA

Let $\mathbf{X}(t) = (X_1(t), ..., X_D(t))^T$ be the multivariate functional variable where each $X_d(t)$ is square-integratable over the interval $[a, b]$. Then, $\mathbf{X}(t)$ is defined in a Hilbert space $\mathbb{H}$ of D-dimensional vector of functions. We assume the functions are centered, i.e., $\mathbb{E}\mathbf{X}(t) = 0$. The covariance function $\mathbf{C}(s, t) = \{C_{dd'}(s, t)\}$ for $\mathbf{X}(t)$ is defined as follows

$$C_{dd'}(s, t) = \text{cov}(X_d(t), X_{d'}(t))$$

for $d, d' = 1, ..., d$. For any two vectors $\mathbf{f}$ and $\mathbf{g}$ is $\mathbb{H}$, the inner product is

$$\langle \mathbf{f}, \mathbf{g} \rangle_{\mathbb{H}} = \sum_{d=1}^{D} \langle f_d(t), g_d(t) \rangle = \sum_{d=1}^{D} \int f_d(t) g_d(t) dt$$

The multivariate Karhunen-Loeve representation of $\mathbf{X}(t)$ becomes

$$\mathbf{X}(t) = \sum_{k=1}^{\infty} \xi_k \phi_k(t)$$

where $\phi_k(t) = (\phi_{k1}(t), ..., \phi_{kD}(t))^T$'s are the multivariate FPCs and ξ_k's are the FPC scores. The FPCs satisfy the following orthogonality constraint

$$\langle \phi_k(t), \phi_{k'}(t) \rangle_{\mathbb{H}} = \delta_{kk'} = 1 \quad \text{if} \quad k = k' \quad \text{and 0 otherwise.}$$

The FPC scores are calculated by

$$\xi_k = \langle \mathbf{x}_t, \phi_k(t) \rangle_{\mathbb{H}} = \sum_{d=1}^{D} \langle x_d(t), \phi_{kd}(t) \rangle = \sum_{d=1}^{D} \int x_d(t) \phi_{kd}(t) dt$$

The estimation of multivariate FPC and FPC scores are not very different compared to the univariate case as introduced in Ramsay and Silverman (2005). The key difference is to string multivariate trajectories into univariate ones. Let $\mathbf{y}_i = (\mathbf{y}_{1i}^T, ..., \mathbf{y}_{Di}^T)$ be the observations of $\mathbf{x}_i(t)$, where $\mathbf{y}_{di} = (\mathbf{y}_{di1}, ..., \mathbf{y}_{diJ}$ and

$$y_{dij} = x_{di}(t_{ij}) + \epsilon_{dij}$$

for some mutually independent random errors ϵ_{dij}. Lets consider a vector $\mathbf{z}_i$ such that

$$\mathbf{z}_i = (y_{1i1}, ..., y_{1iJ}, ..., y_{Di1}, ..., y_{DiJ})$$

which is a $D * J$-dimensional vector. After concatenation into a univariate trajectory, we can apply the previous estimation method for FPCA based on the observations $\{\mathbf{z}_i\}_{i=1}^N$. FPC scores are directly obtained, and we can truncate the obtained eigenfunctions into each d dimension for $d = 1, ..., D$.

S7.3 mFPCA

The estimation of mFPCA by Happ and Greven (2018) is based on univariate FPCA to each d-dimension of $\mathbf{X}(t)$. For each d, we consider the univariate representation

$$X_d(t) = \sum_{k=1}^{\infty} \xi_k^{(d)} \phi_k^{(d)}(t)$$

We apply the aforementioned univariate FPCA on $\{x_{id}(t)\}_{i=1}^N$ for $d = 1, ..., D$ to obtain the univariate FPC scores $\{\xi_{ik}^{(d)}\}_{i=1,k=1}^{i=N,k=K_d}$ for a selected number of components K_d. Consider a matrix $\mathbf{\Xi}$, N by $\sum K_d$, where the i-th row is

$$\mathbf{\Xi}_i = (\xi_{i1}^{(1)}, ..., \xi_{i1}^{(d)}, ..., \xi_{iK_d}^{(d)}, ..., \xi_{iK_D}^{(D)})$$

Let the $\mathbf{\Gamma} = 1/(N-1)\mathbf{\Xi}^T\mathbf{\Xi}$, we can obtain the eigenpairs $\mathbf{c}_l$'s and ρ_l's for $l = 1, ..., L$ given a selected number of components L. At the end, the multivariate FPC score of i-th trajectory is obtained as

$$\omega_{il} = \mathbf{\Xi}_i \cdot \rho_l$$

for $l = 1, ..., L$. We can see that mFPCA is a weighted version of the previous MFPCA method. To apply mFPCA on simulated and real data sets, we used the author's R package `mFPCA`.

S8 Additional application results

S8.1 Data set descriptions

`Starlight` - This data set contains the brightness of stars as a function of time. The interest in studying star light curves is to identify and distinguish the sources as celestial objects. The 8236 curves are observed for three stars where the brightness of them has been recorded over 1024 equally-spaced time points.

`Wafer` - The `Wafer` data set is to study microelectronics fabrication. The inline process control measurements of one wafer are recorded by one sensor over 152 points as a function of time. There are two classes of measurements: normal and abnormal.

`SwedishLeaf` - This data set studies the leaf outlines. The original 2D outline of a leaf has been preprocessed by a visual descriptor to a one-dimensional curve over 128 arbitrary location points. The data set contains samples for 15 different leaves for which we are trying to classify.

`Earthquakes` - The data set focuses on predicting whether a major event is about to happen by using the most recent Richter scale readings. For both the positive case (an earthquake occurs) and the negative case (no earthquake occurs), we use the recent 512 hourly readings to predict whether a major event (over 5 on the Richter scale) will occur in the following hour.

`Gesture` - This data set contains 8 simple gestures: a check-mark wave; up, down, left, and right movement; clockwise and counter-clockwise circular movement; a square movement. Each sample gesture is recorded by accelerometers in three-dimension over 945 time points. The interest is to classify the gesture based on a three-dimensional trajectory as a function of time.

`Cricket` - This data set contains 12 movements that would be observed by a cricket player. The movements are recorded by accelerometers, and the observed three-dimensional trajectory is treated as a function of time. Each trajectory is observed over 300 time points. The study's interest is to classify the movement based on a given multivariate function.

Figure S1 presents some sample trajectories from each data set.

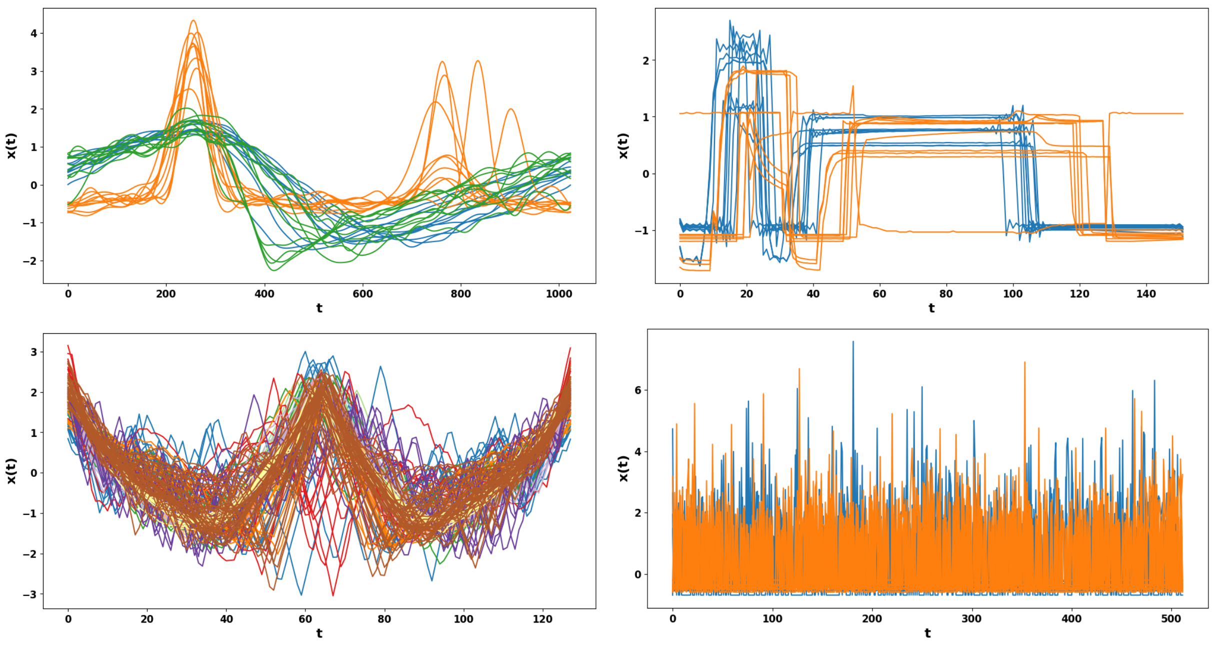

Figure S1: The data sets are `Starlight`, `Wafer`, `SwedishLeaf`, `Earthquakes` ordered from left to right and top to bottom. From each data set, we select a few sample trajectories for illustration. Different colors indicate different classes of trajectories in each plot.

S8.2 Using neural networks for predicting labels

In addition to using logistic regression for predicting curve labels, we also build a classifier based on neural networks. This section presents the prediction performances when a neural networks is used for classification.

Table S1: Prediction accuracy based on features from FunNoL (with or without the corruption step, denoted by FunNoL_c and FunNoL_nc respectively) and FPCA on four data sets, `Starlight`, `Wafer`, `SwedishLeaf`, and `Earthquakes`. The top table presents the prediction accuracy by using the logistic regression models. The bottom table presents the prediction accuracy by using the neural networks.

(a) Prediction accuracy by using logistic regression models.

	`Starlight`	`Wafer`	`SwedishLeaf`	`Earthquakes`
# Training samples	8236	6164	625	322
FunNoL_c	.975(.003)	.995(.002)	.398(.052)	.839(.035)
FunNoL_nc	.951(.003)	.995(.002)	.381(.042)	.814(.027)
FPCA	.847(.006)	.927(.005)	.471(.057)	.810(.032)

(b) Prediction accuracy by using neural networks.

	`Starlight`	`Wafer`	`SwedishLeaf`	`Earthquakes`
# Training samples	8236	6164	625	322
FunNoL_nnet_c	.978(.003)	.996(.002)	.593(.031)	.889(.031)
FunNoL_nnet_nc	.952(.004)	.996(.002)	.600(.036)	.857(.036)
FPCA_nnet	.872(.010)	.929(.005)	.567(.047)	.829(.024)

The neural networks for making predictions consist of 6 densely-connected layers (with 256 hidden nodes), and the networks for different methods have the same structure and are trained separately. The prediction accuracy of test samples over 100 data splits are summarized in Table S1b for each data set. For data sets `Starlight`, `Wafer`, and `Earthquakes`, the prediction accuracy are above 80% for all methods when we use logistic regression for prediction. In such case, we can consider that predictions are generally good, and the improvement by using neural networks for classification is not significant. The most improvement is on the `Earthquakes` data set where the accuracy increases by 6% from 0.814 to 0.857.

On the other hand, the biggest improvement by using neural networks is on the `SwedishLeaf` data set. This data set has 15 labels and relatively small sample sizes for each label. It is difficulty for all methods to establish a good feature space. This conclusion is reflected by the low prediction accuracy of logistic regression models. In this case, using neural networks improves the prediction performances, as shown in Table S1b.

On the comparison between Table S1a and Table S1b, a good representation model of functional data should provide information which can be used by simple logistic regression models. Hence, it can minimize the subsequent modeling efforts while providing performances on par with more complex models. Even though we could enhance the following statistical model to a more complex one, that should not affect the priority and standard on the latent representations.

Table S2: Prediction accuracy based on features from FunNoL (with or without the corruption step, denoted by FunNoL_c and FunNoL_nc respectively), FPCA on multivariate functional data, and the mFPCA method by Happ and Greven (2018). The first row indicates the training sample size for data sets, `Gesture` and `Cricket`. Each cell in the last four rows provides the classification accuracy with standard errors in parentheses on curve labels in the test sets.

(a) Prediction accuracy by using logistic regression models.

	`Gesture`	`Cricket`
# Training samples	3582	390
FunNoL_c	.963(.0044)	.316(.031)
FunNoL_nc	.920(.0055)	.368(.033)
FPCA	.880(.0080)	.0791(.023)
mFPCA	.887(.0076)	.0842(.021)

(b) Prediction accuracy by using neural networks.

	`Gesture`	`Cricket`
# Training samples	3582	390
FunNoL_nnet_c	.975(.0043)	.379(.031)
FunNoL_nnet_nc	.964(.0064)	.386(.033)
FPCA_nnet	.949(.012)	.0974(.018)
mFPCA_nnet	.952(.0098)	.0996(.014)

For FunNoL_c and FunNoL_nc on the `Gesture` data set, the latent features are already good enough for using a logistic regression model for prediction. Hence, the improvement by using neural networks for prediction is not significant. On the other hand, linear representations from FPCA and mFPCA can be boosted by using neural networks for prediction. The performances are slightly worse than our propose method, but they are more competitive than using a logistic regression model for prediction. For the `Cricket` data set, the performances of the propose method is superior over FPCA and mFPCA. For this data set, using a complex classifier is more beneficial for FunNoL_c and FunNoL_nc.

S8.3 Curve reconstruction

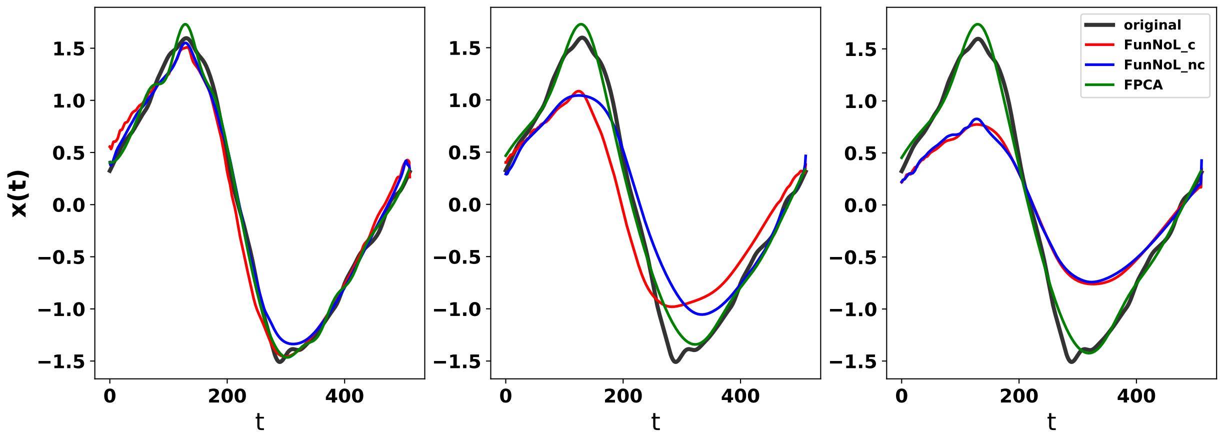

Figure S2: Reconstruction of an individual curve from the *Starlight* data set. The leftmost plot corresponds to the case where we use complete data set for estimation. The middle plot comes from the case when the model is estimated based on curves with 30% missing observations. The rightmost plot is from the model when each curve has 60% of all observations missing. The black line is the observed trajectory. The red line is the reconstruction of the FunNoL method with corrupted input, the blue line is from the FunNoL method with observed input, and the green line is the reconstruction by using FPCA.

Figure S2 demonstrates the reconstructions of a selected curve from the *Starlight* data set. When the proposed FunNoL method and FPCA are estimated with the complete data set, we can see that the reconstructions between these two methods are not different. When the data set becomes more sparse, as going from left to right in Figure S2, FPCA outperforms in curve reconstruction over the FunNoL method. Even though the FunNoL method does not reconstruct the curve to the true one, it is still able to capture the structure of the curve.

When data are sparse, FPCA uses local linear regression to pull observations from all curves around a single observation point to bypass missing observations and estimate the basis functions. This is advantageous for smoothing discrete observations into a trajectory. On the other hand, the proposed FunNoL method and its framework of recurrent neural networks will skip points with missing observations for updating parameters. It is more suitable for estimating the mapping from functional data to representations but less advantageous for reconstructing individual trajectories.

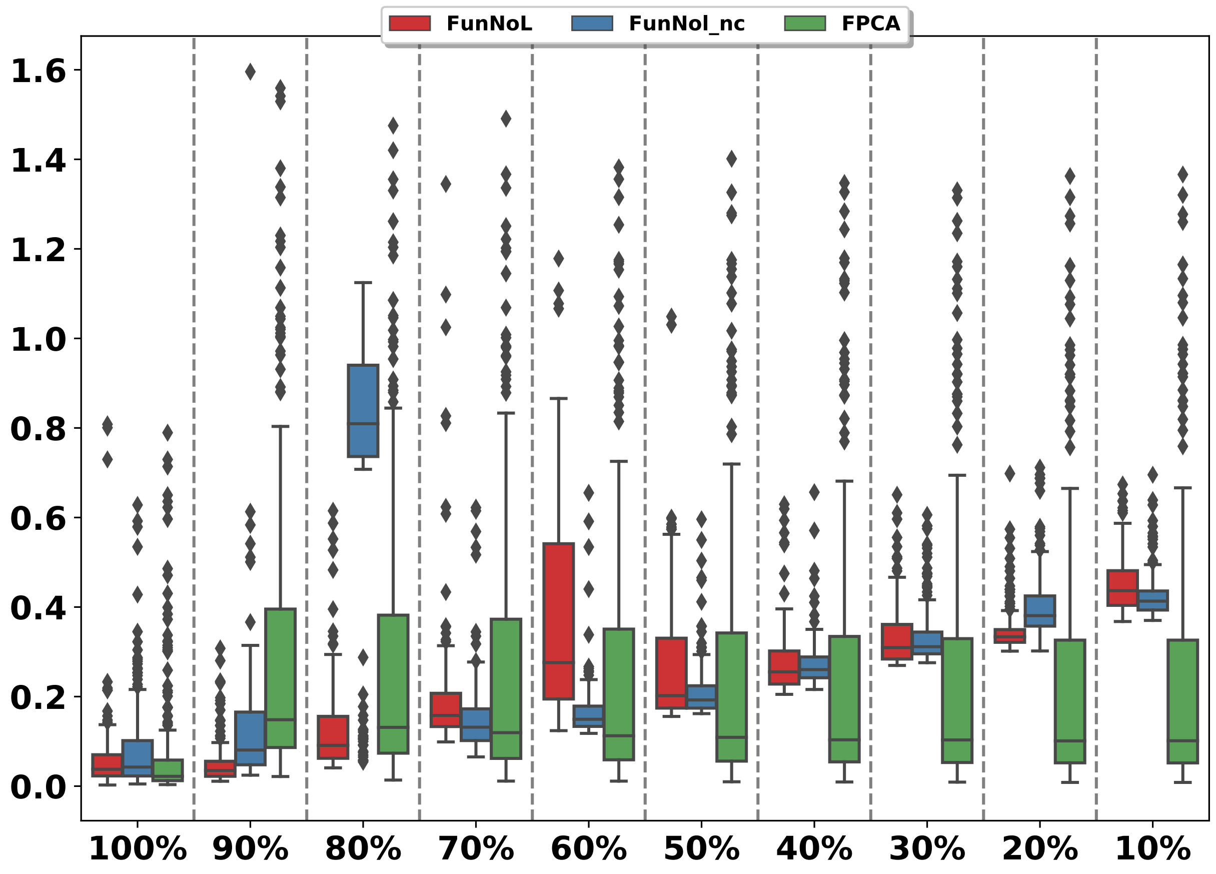

Figure S3: Mean squared error of reconstructions of curves from the *Starlight* data set based on the FunNoL method with corrupted input (red), the FunNoL method with observed data (blue), and the FPCA (green).

Let the mean squared error of reconstructions of curves to be

$$\frac{1}{T}\sum_{j=1}^{T}(x_i(t_j) - \hat{x}_i(t_j))^2$$

where $x_i(t_j)$ is the value of individual functional data at point t_j and $\hat{x}_i(t_j)$ is that of the reconstructed curve. Figure S3 demonstrates that the FunNoL method produces well reconstructions when the data sparsity is low. As the sparsity increases, FPCA provides better reconstructions of individual functional data.

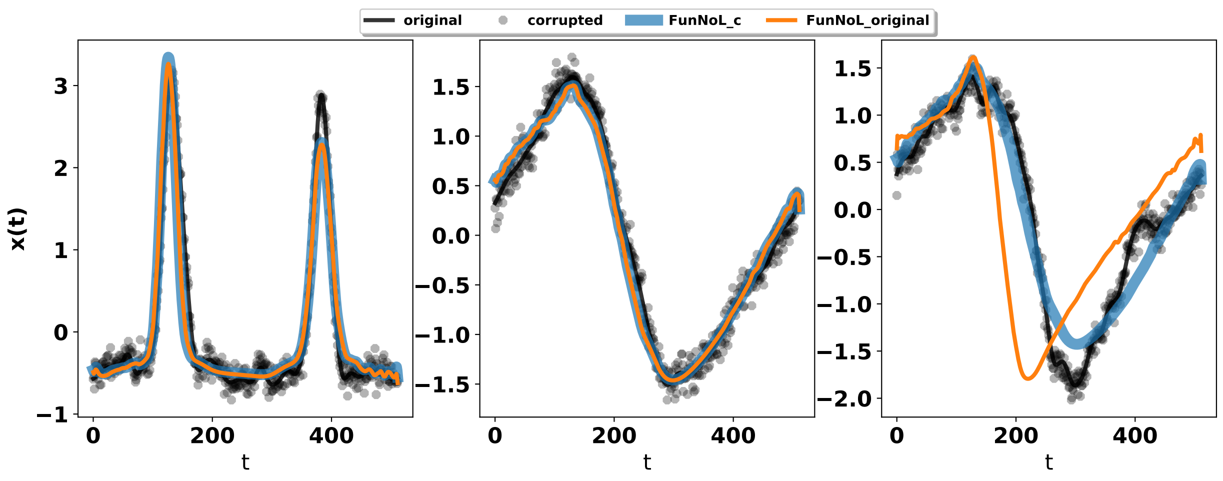

Figure S4: Checking the robustness of the FunNoL method. The black line is the originally observed trajectory. The grey points are the corrupted input to the FunNoL model. The bold blue line is the curve reconstruction with corrupted inputs, and the orange line is that with original inputs.

In terms of checking the robustness of the proposed FunNoL method, we can also see whether the FunNoL method can provide the same or similar curve reconstructions when using either the corrupted inputs or the original ones. Figure S4 demonstrates that the FunNoL method is able to produce similar curve reconstructions by using either observed or corrupted data. This highlights the robustness of the FunNoL methods, and it is a desirable property when functional data is usually influenced by uncontrollable random noises.

S9 Additional simulation results

In the main article, we have included the simulation studies on multivariate functional data focusing on the classification performance. In Section S8.1, we will present simulation results on univariate functional data. In Section S8.2, we will show the simulation results on multivariate functional data on curve classification and reconstruction.

S9.1 Simulation on univariate functional data

First, we conduct simulations on univariate functional data sets. The second task for the representations is to reconstruct trajectories given noisy observations. The reconstruction mapping ψ has to be estimated where FPCA has a natural inverse transformation. In addition to classification, it is important to verify that representations can be passed on to ψ for producing stable reconstructions. In real applications, curves might not be fully observed, and individual curves have missing observations. We select two curves from the `Starlight` data set for illustrating the simulation results, and the curve has only 30% data observed. The reconstructions of these two curves are presented in Figure S5. There are three classes of curves in the `Starlight` data set, one distinct class and two overlapping classes. Figure S5 shows that FunNoL exceeds in curve reconstruction for curves from the distinct class, whereas FPCA is able to provide better reconstruction for the two overlapping classes. The overlapping curves are more likely to have a common covariance function that helps FPCA on the reconstruction of such curves. On the other hand, the FunNoL method equally attends to different classes of curves. Similar to the simulations on classification, the usefulness of additional corruption step on curve reconstruction depends on the portion of missing observations in each curve. If the portion of missing observations is large, then FunNoL produces better reconstructions without a corruption step. In the top left panel of Figure S5, FunNoL_c misses the second bump in the observed trajectory where FunNoL_nc captures both bumps. Next, we can investigate the robustness of FunNoL to random noises in the observations. First, we estimate the FunNoL model with a corruption step to get FunNoL_nc. Then, we use FunNoL_nc to predict trajectories with original observations where the reconstructions are labeled by FunNoL_original. As shown in the bottom panel of Figure S5, the reconstruction from FunNoL_c and predicted one from FunNoL_original are nearly identical. As a result, the FunNoL model, trained with a corruption step, is able to find the underlying trajectory and be robust to corruption in functional data.

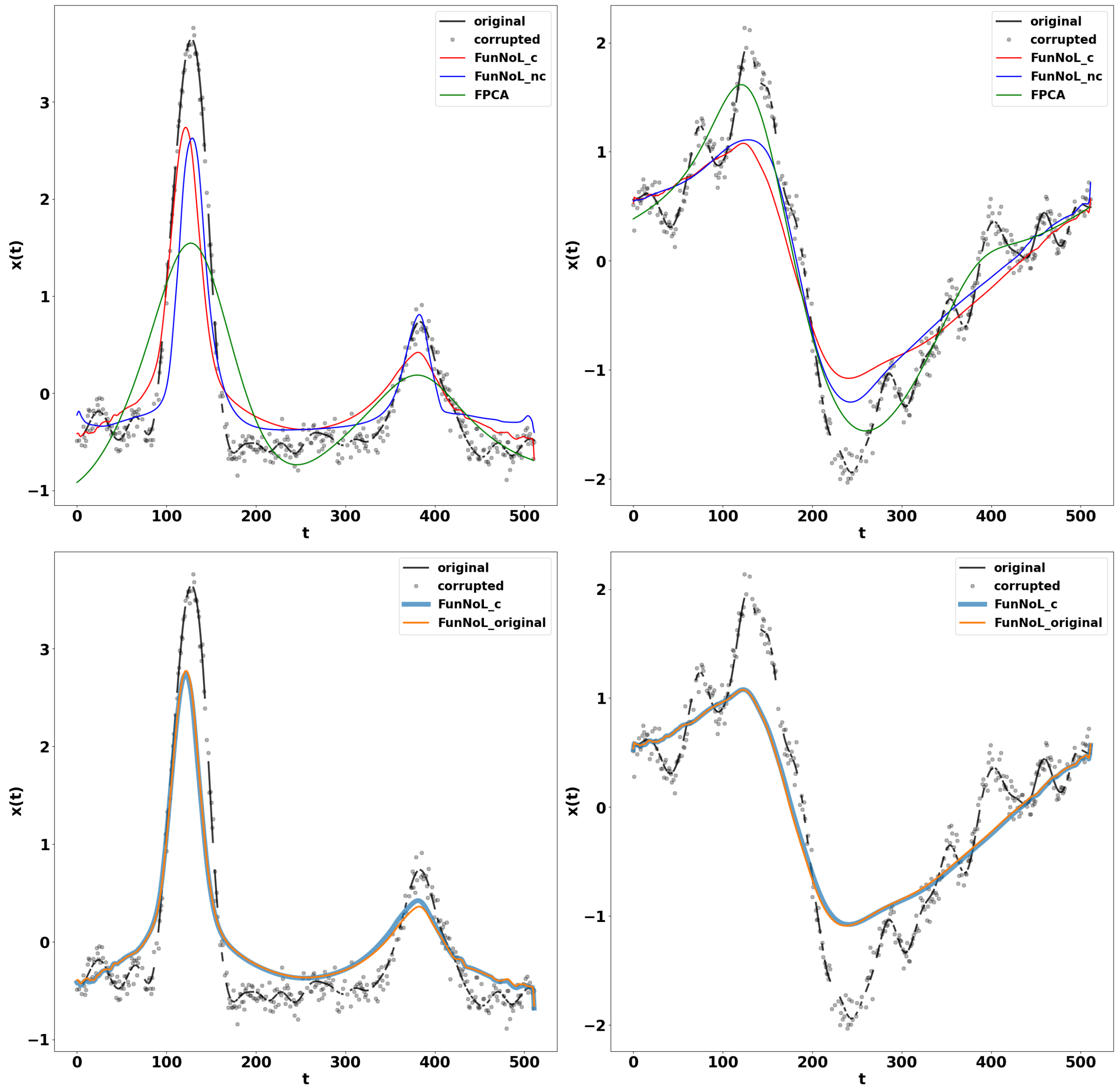

Figure S5: Curve reconstructions of two selected curves (ordered from left to right) from the FunNoL model and FPCA. The top panel presents the case when we sample 30% of available observations. The bottom panel shows the reconstruction by inputting original observations to a fitted model with corrupted ones. FunNoL_c is the the FunNoL model trained with corrupted observations. The thin line is to use FunNoL_c with corrupted observations whereas the thick line is that with original observations.

S9.2 Simulation on multivariate functional data

We will present simulation studies on multivariate functional data. Additional simulation results on the curve reconstruction are provided in this section. We show the simulation results on multivariate functional data where the focus is on the reconstruction of multivariate trajectories. One assessment is based on the root mean square error of reconstructions over time, RMSE(t), which is defined as:

$$\text{RMSE}_d(t_{ij}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_d(t_{ij}) - \tilde{x}_d(t_{ij}))^2},$$

where $d = 1, ..., D$, corresponding to each dimension in multivariate data, $x_d(t_{ij})$'s are the observations from dimensions, and $\tilde{x}_d(t_{ij})$'s are the reconstructions from FPCA or FunNoL. The left panel of Figure S6 shows that RMSE(t) from FPCA is constantly lower than that from FunNoL when the estimations are based on complete data. As the portion of missing observations increases, FunNoL produces better curve reconstructions in terms of RMSE(t). The right panel in Figure S6 presents the reconstructions when 30% of observations from each trajectory are sampled to estimate the model. We can observe that FPCA produces flatter trajectories compared to either FunNoL or the case of using complete data. This may be caused by concatenating multiple dimensions into a univariate one. When we sample observations from multivariate trajectories, we are not restricting the sampled time points to be the same across dimensions. This may cause the kernel window to be wider in estimating the covariance function. By contrast, FunNoL is not affected by the missing observations since it does not require the calculation of a covariance function. Based on the simulations on classification and reconstruction, we conclude that FunNoL is superior in handling to multivariate functional data.

References

Bishop, C. M. (1995). Training with noise is equivalent to tikhonov regularization. *Neural Computation 7*(1), 108–116.

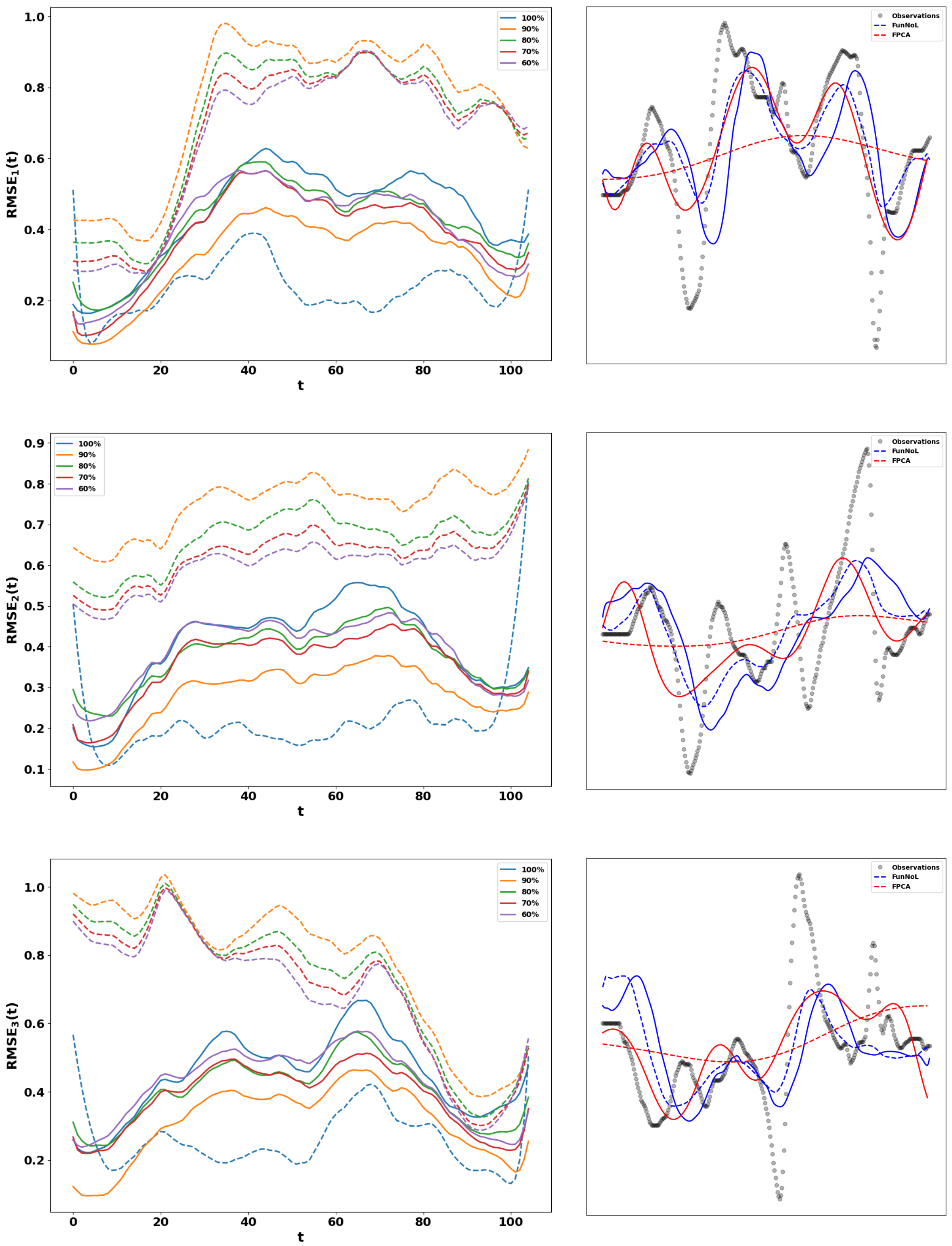

Figure S6: The left panel presents the RMSE(t) with respect to the three dimensions in the `Gesture` data set. The dimensions are ordered from top to bottom. The solid lines are RMSE(t) of FunNoL_c whereas the dashed lines are from FPCA. The right panel showcases a selected trajectory with reconstructions from FunNoL and FPCA. Blue lines are from FunNoL, and red lines are from FPCA. The solid lines correspond to the case where models estimated with complete data. The dashed lines correspond to the case where 30% of observations are sampled for doing estimations.

Happ, C. and S. Greven (2018). Multivariate functional principal component analysis for data observed on different (dimensional) domains. *Journal of the American Statistical Association 113*, 649–659.

Neyshabur, B., S. Bhojanapalli, and N. Srebro (2018). A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*.

Ramsay, J. and B. Silverman (2005). *Functional Data Analysis*. New York: Springer.

Yao, F., H.-G. Müller, and J.-L. Wang (2005). Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association 100*, 577–590.