A Functional Extension of Semi-Structured Networks

David Rügamer[1,2], Bernard Liew[3], Zainab Altai[3], Almond Stöcker[4]

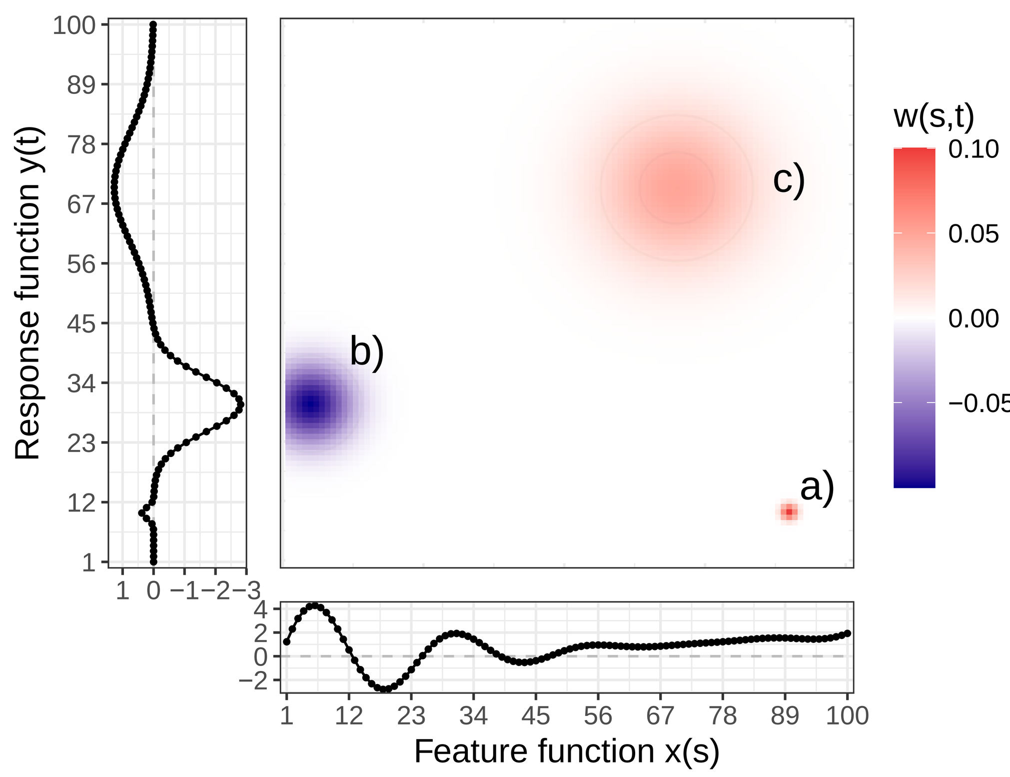
[1]LMU Munich [2]Munich Center for Machine Learning [3]University of Essex [4]École Polytechnic Fédéral de Lausanne

Function-on-Function Regression (FFR)

- Outcome: Stoch. process $Y(t)$ over $\mathcal{T} \subset \mathbb{R}$, realizations $y \in L^2(\mathcal{T})$
- Features: Stoch. processes $X_j, j \in [J]$, realizations $x_j \in L^2(\mathcal{S}_j)$
- Model:

$$\mu(t) := \mathbb{E}(Y(t)|X = x) = \tau\left(b(t) + \sum_{j=1}^{J} \int_{\mathcal{S}_j} w_j(s,t)x_j(s)ds\right) \quad (1)$$

with functional bias $b(t)$, weight surfaces $w_j(s,t)$, activ. function τ.

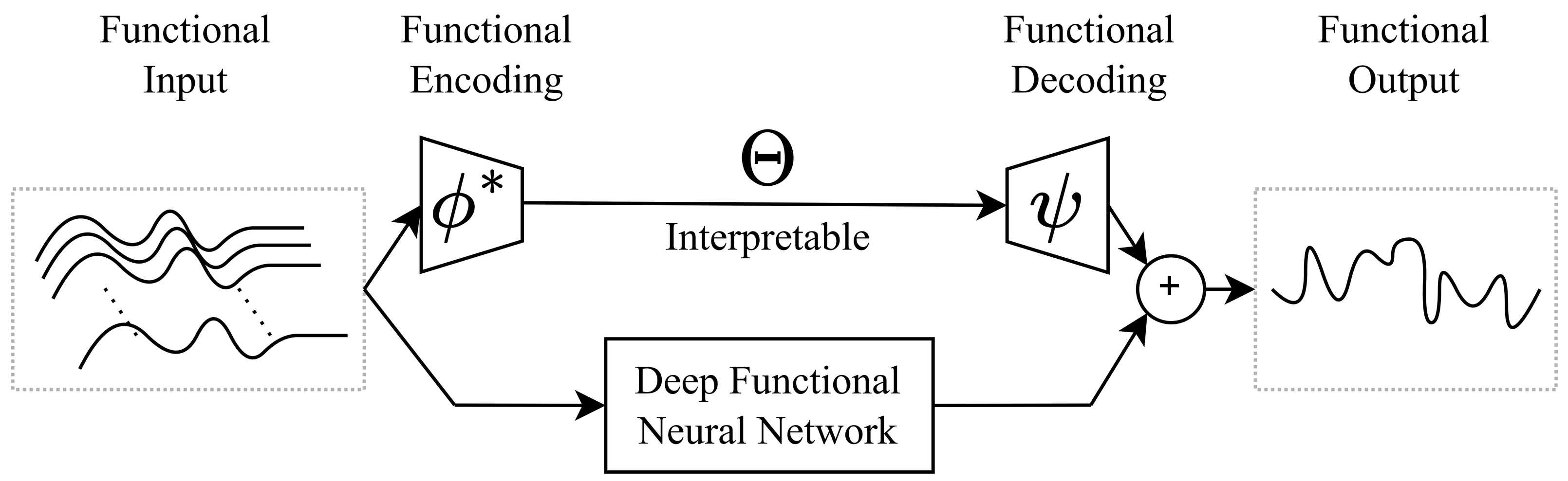


Semi-Structured Networks (SSNs)

- Idea: Combine structured model λ^+ and deep neural network λ^-
- Advantage: Interpretability of λ^+, flexibility through λ^-
- Challenge: Ensure interpretability of λ^+ in the presence of λ^-

This Paper

- Embed FFR in neural networks — in a scalable manner
- Extend FFR to SSNs — without losing their interpretability



Scalable Implementation

▶ **Representation**: Given time points $s_r, r \in [R]$, approximate the structured part as

$$\lambda_j^+(t) = \int_{\mathcal{S}_j} x_j(s) w_j(s,t) ds \approx \boldsymbol{\Phi}_j^* \boldsymbol{\Theta}_j \boldsymbol{\psi}(t) \quad (2)$$

with tensor-product spline basis for $w_j(s,t) = \boldsymbol{\psi}(t)^\top \boldsymbol{\Theta}_j \boldsymbol{\phi}_j(s)$,

$$\boldsymbol{\Phi}_j^* = (\boldsymbol{\Delta}_j \circ \boldsymbol{x}_j)\boldsymbol{\Phi}_j^\top,$$

$\boldsymbol{\Phi}_j = [\boldsymbol{\phi}_j(s_1), \ldots, \boldsymbol{\phi}_j(s_R)]$, $\boldsymbol{x}_j = (x_j(s_r))_{r \in [R]}$, Hadamard product $\circ$, $\boldsymbol{\Delta}_j = [\Delta_j(s_1), \ldots, \Delta_j(s_R)]^\top$ defines integration weights.

▶ **Loss**: For observed time points $t_q, q \in [Q]$, the ith loss contrib. is

$$\ell(y^{(i)}, \hat{\mu}^{(i)}) = \int_{\mathcal{T}} l(y^{(i)}(t), \hat{\mu}^{(i)}(t))dt \approx \sum_{q=1}^{Q} \Xi(t_q) l(y^{(i)}(t_q), \hat{\mu}^{(i)}(t_q))$$

where $\Xi(\cdot)$ are integration weights and l a point-wise loss.

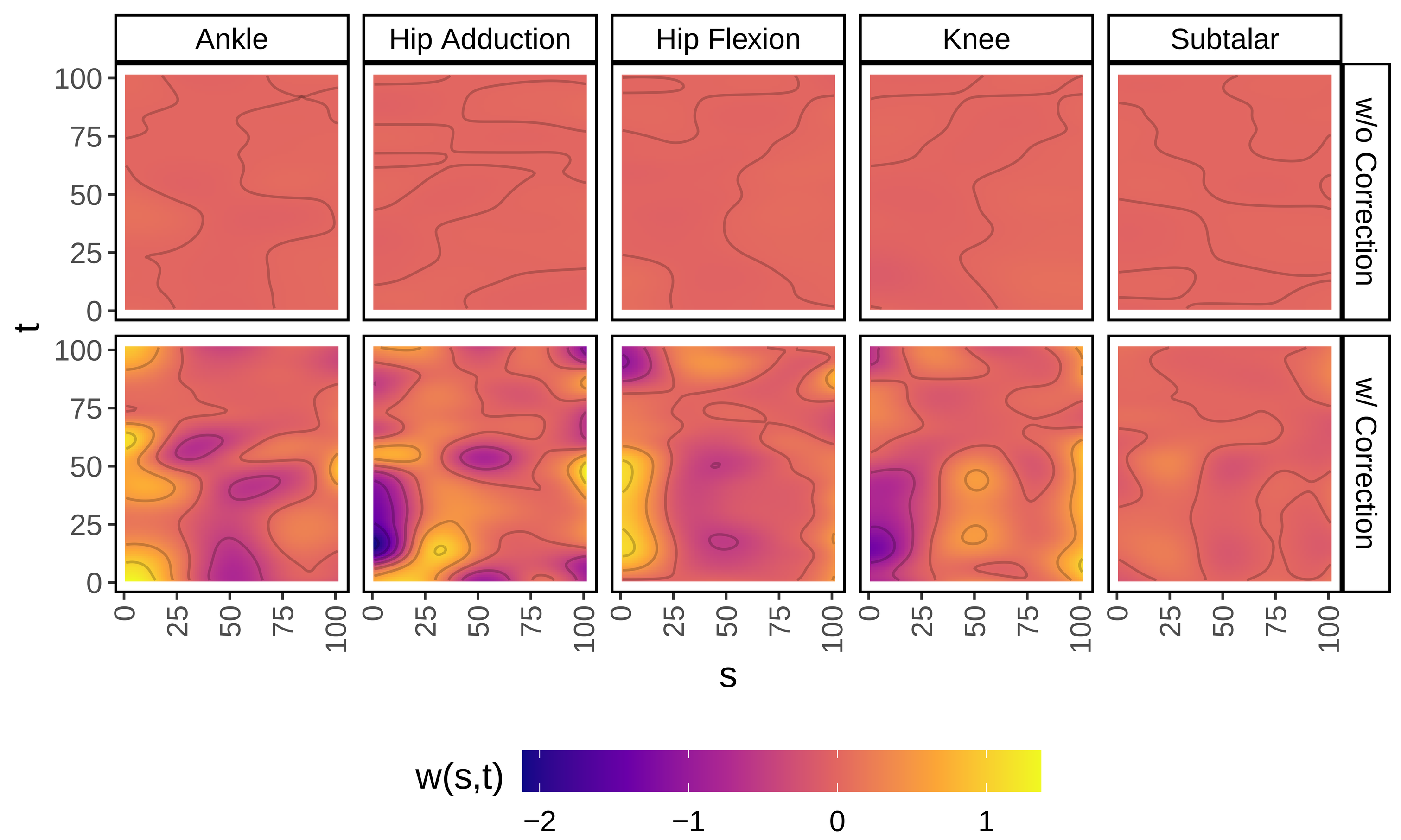
▶ **Scalability**: 1. Mini-batch training, 2. Array computation (Eq. 2), 3. Basis recycling of $\boldsymbol{\Phi}_j$ for all $j \in [J]$.

Orthogonalization

Step 1: Reformulate $\lambda^+(t)$ as

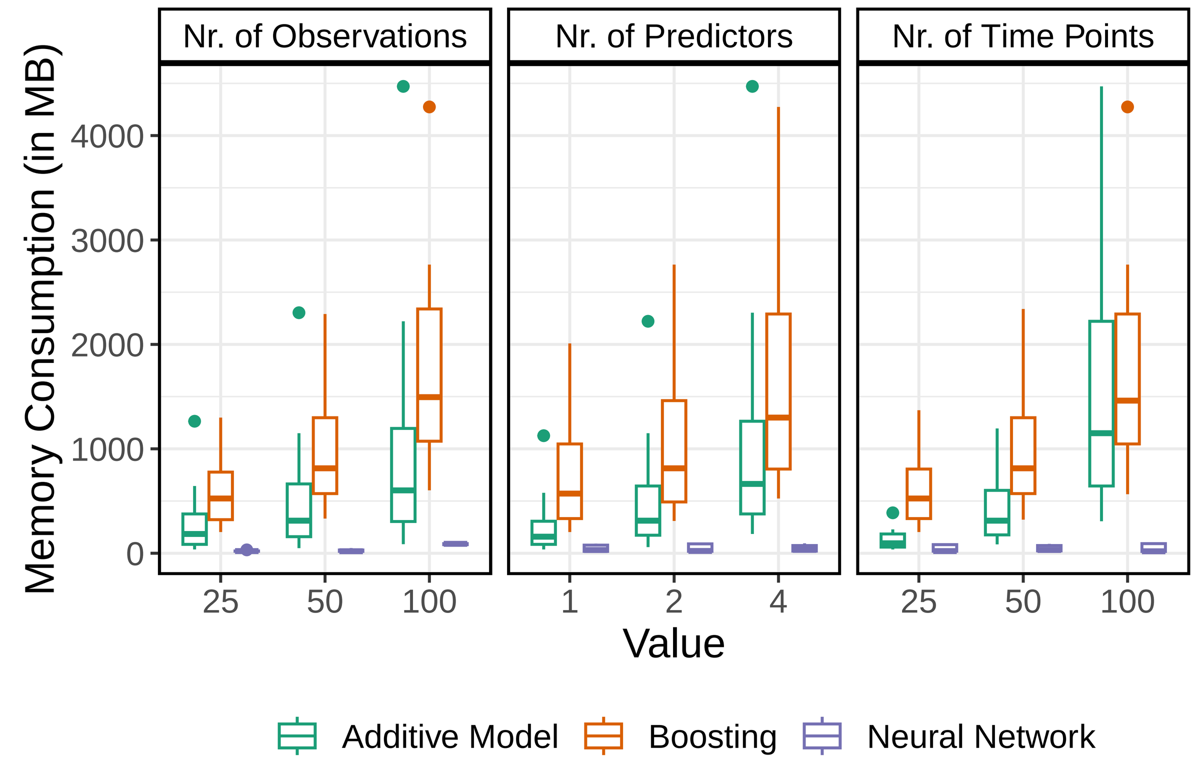
$$\text{vec}(\lambda^+(t)) = \text{vec}\left(\sum_{j=1}^{J} \boldsymbol{\Phi}_j^* \boldsymbol{\Theta}_j \boldsymbol{\psi}(t)\right) = \sum_{j=1}^{J} \underbrace{(\boldsymbol{\psi}(t)^\top \otimes \boldsymbol{\Phi}_j^*)}_{\boldsymbol{\Omega}_j(t)} \underbrace{\text{vec}(\boldsymbol{\Theta}_j)}_{\boldsymbol{\theta}_j}$$

Step 2: Project λ^- into the orthogonal complement of the column space of $[\boldsymbol{\Omega}_j(t)]_{i \in [n \cdot Q], j \in [J]}$

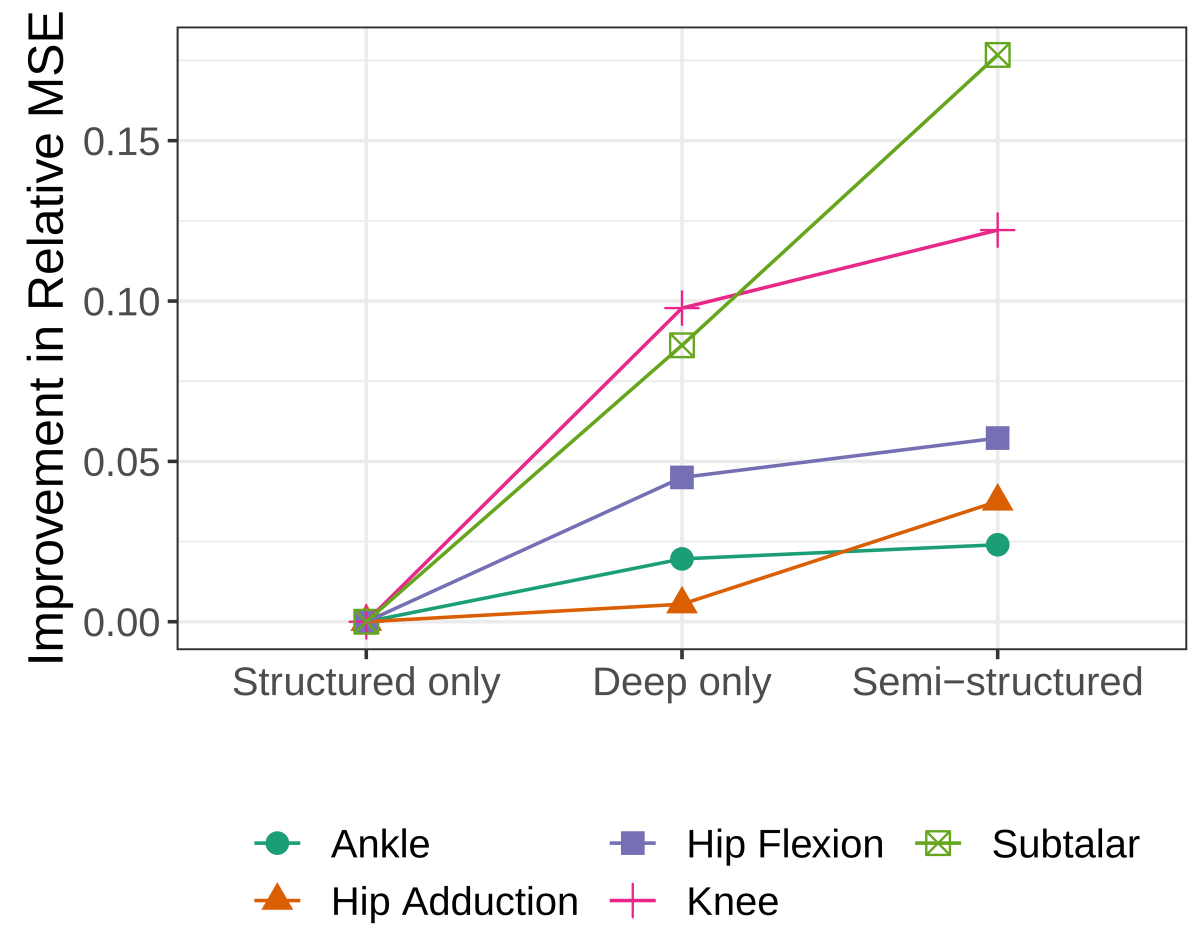


Experimental Results

Computational Complexity



Prediction Performance



Contact

✉ **david.ruegamer@lmu.de**