Supplementary Document for the Manuscript entitled "Deep Learning with Functional Inputs"

S1 Proof of Corollary 1.

Corollary 1. *Let $g : \mathbb{R} \to \mathbb{R}$ be any continuous sigmoidal function, I_n denote the $n-$dimensional hypercube $[0,1]^n$ and $\mathcal{C}(I_n)$ denote the space of continuous functions. Then, the finite sum of the following form is dense in $\mathcal{C}(I_n)$:*

$$h(t) = \sum_{i=1}^{n_1} \Psi_i g \left(\sum_{k=1}^{K} \left(\int_{\mathcal{T}} \beta_{ik}(t) x_k(t) \, \mathrm{d}t \right) + \sum_{j=1}^{J} w_{ij} z_j + b_i \right),$$

meaning that for any $f(t) \in \mathcal{C}(I_n)$ and for $\epsilon > 0$, the function $h(t)$ obeys:

$$|h(t) - f(t)| < \epsilon.$$

Proof. The proof is to show that $h(t)$ can be rewritten into Cybenko form, Equation (2) from (Cybenko, 1989),

$$G(z) = \sum_{i=1}^{n_1} \alpha_i \sigma \left(y_i^T z + \theta_i \right) = \sum_{i=1}^{n_1} \alpha_i \sigma \left(\sum_{j=1}^{J} y_{ij} z_j + \theta_i \right).$$

To begin we consider our function g and its argument,

$$g \left(\sum_{k=1}^{K} \left(\int_{\mathcal{T}} \beta_{ik}(t) x_k(t) \, \mathrm{d}t \right) + \sum_{j=1}^{J} w_{ij} z_j + b_i \right),$$

and we note that

$$\sum_{j=1}^{J} w_{ij} z_j + b_i,$$

is already in Cybenko form. Therefore, all we need to show is that

$$\sum_{k=1}^{K} \left(\int_{\mathcal{T}} \beta_{ik}(t) x_k(t) \, \mathrm{d}t \right),$$

depends only on n, the neuron number index. If we expand our weight function as a finite linear combination of basis functions ϕ_{ikm}, we get,

$$\beta_{ik}(t) = \sum_{m=1}^{M_k} c_{ikm}\phi_{ikm}(t).$$

Therefore,

$$\begin{aligned}
\sum_{k=1}^{K}\left(\int_{\mathcal{T}} \beta_{ik}(t)x_k(t)\,\mathrm{d}t\right) &= \sum_{k=1}^{K}\left(\int_{\mathcal{T}} \sum_{m=1}^{M_k} c_{ikm}\phi_{ikm}(t)x_k(t)\,\mathrm{d}t\right) \\
&= \sum_{k=1}^{K}\sum_{m=1}^{M_k} c_{ikm} \underbrace{\left(\int_{\mathcal{T}} \phi_{ikm}(t)x_k(t)\,\mathrm{d}t\right)}_{a_{ikm}} \\
&= \sum_{k=1}^{K}\sum_{m=1}^{M_k} c_{ikm}a_{ikm} \\
&= \tilde{b}_i
\end{aligned}$$

Note, since we have a finite sum under the integral, we can integrate term-by-term and switch the integral and sum in the second step above. In total, we have:

$$g\left(\sum_{k=1}^{K}\left(\int_{\mathcal{T}} \beta_{ik}(t)x_k(t)\,\mathrm{d}t\right) + \sum_{j=1}^{J} w_{ij}z_j + b_i\right) = g\left(\sum_{j=1}^{J} w_{ij}z_j + \left(b_i + \tilde{b}_i\right)\right)$$

Therefore, if we let $\alpha_i = \Psi_i$, $y_{ij} = w_{ij}$, and $\theta_i = b_i + \tilde{b}_i$, we obtain Cybenko's form for $g(\cdot)$. ■

S2 List of All Parameters

Parameter	Symbol	Type	Details
Functional Weights	$\{\beta(t)\}$	Estimated	Coefficient function found by the FNN.
Scalar Weights	$\{w\}$	Estimated	The scalar covariate weights.
Biases	$\{b\}$	Estimated	The intercept in each neuron.
Number of Layers	–	Hyperparameter	The depth of the FNN.
Neurons per Layer	$\{n_u\}$ for $u = 1, 2, ..$	Hyperparameter	Number of neurons in each layer of the FNN.
Learn Rate	γ	Hyperparameter	The learning rate of the FNN.
Decay Rate	–	Hyperparameter	A weight on the learning process across training iterations for the FNN.
Validation Split	–	Hyperparameter	The split of train/test set.
Functional Weight Basis Expansion Sizes	$\{M_k\}$	Hyperparameter	The number of terms, M_k (for each k), for the estimation of $\beta_k(t)$.
Functional Weight Basis Functions	–	Hyperparameter	The type of basis expansion (e.g. Fourier) for the estimation of $\beta(t)$.
Functional Weight Domains	–	Hyperparameter	The domain for the integration of $\beta(t)$ and $x(t)$.
Epochs	–	Hyperparameter	The number of learning iterations.
Batch Size	–	Hyperparameter	Subset of data per pass of the FNN.
Activation Functions	–	Hyperparameter	The choice of $g(\cdot)$ for each layer.
Early Stop	–	Hyperparameter	Stops the model building process if no improvement in error.
Threshold for Min. Improvement	–	Hyperparameter	For the early stop parameter – the threshold for what constitutes “no improvement”.
Loss Choice	$R(\theta)$	Hyperparameter	Loss function used in the learning process for FNNs.
Dropout	–	Hyperparameter	Optional parameter that randomly drops observations between layers.

Table S1: A list of the parameters in the network.

S3 Additional Simulation Studies

We have added an experimental study to understand the role of the different hyperparameters. The study results are visualized in Figure S1 below. In particular, we consider the number of basis functions defining the functional weight (range: [3, 31]), the learning rate (range: [0.0001, 1]), the number of neurons (range: [2, 256]), the validation split rate (range: [0.05, 0.5]), the decay rate (range: [0, 1]), and the number of epochs (range: [5, 500]). These were measured using mean squared prediction errors (MSPEs) as defined in the main manuscript. The goal was to determine how changing values for a particular hyperparameter affects the measure of MSPE. In the study for each particular hyperparameter, we fix all the others and pick from a grid the value chosen for the hyperparameter under scrutiny. The cross-validated error is then stored and the process is repeated for the next value in the grid. The data used in these studies is the bike data set. As expected, the number of basis functions seems to have an impact on the error rate. The error rate decreases until a minimum is hit and then slowly, the error begins to climb. This is expected because the functional weights are supposed to model the relationship between the functional covariates and the response. In order to capture the relationship, enough basis terms are required so that all significant points along the domain of the functional covariate are free to be weighted. The learn rate seems to be steady until a particular point from where it begins to increase. The behaviour is similar with the number of neurons however, we recognize that some of these behaviours are highly context dependent; perhaps on a larger data set, this behaviour wouldn't be preserved. The number of epochs have an expected behaviour as well in that the decrease in error plateaus after about a 100 iterations; this is expected as we can attribute this to the network as being appropriately trained via some local extrema being found. The validation split parameter also had a clear minimum. Lastly, the decay rate parameter is minimized at 0, the boundary; we observe that adding a decay rate here only worsens model predictions and hence, even in our application in the manuscript, do not use this parameter.

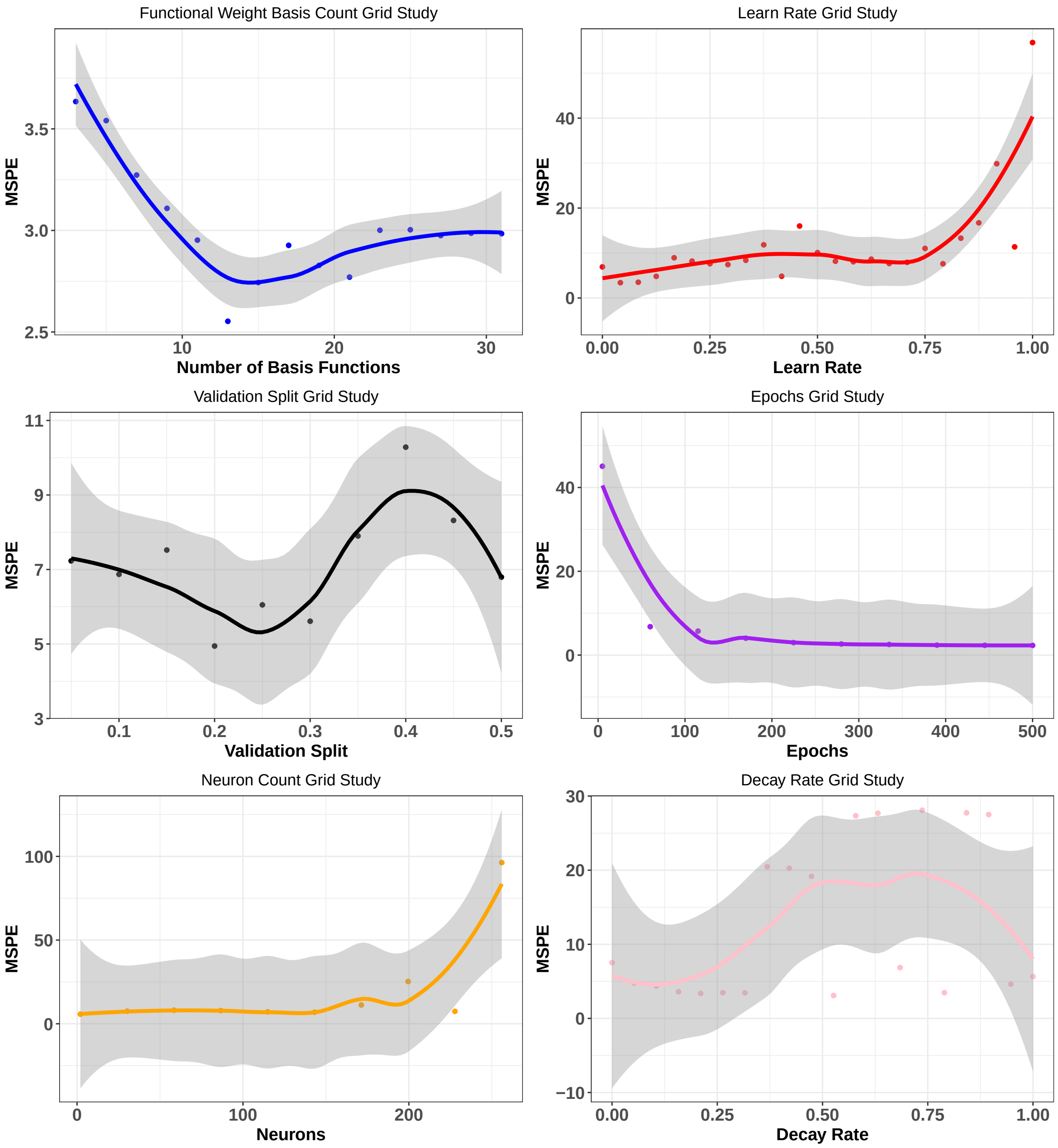

Figure S1: Hyperparameter studies plot featuring an analysis of the functional weight basis count, learn rate, validation split, epochs, neuron count, and decay rate. Superimposed is a local regression approximation (curve) along with the associated standard errors (shaded).

S4 Applications

S4.1 Bike

S4.1.1 Model Configurations

Model	Layers	Num. Neurons	Activations	Functional Weight Function	Functional Weight Terms	Learn Rate
Bike	4	32, 32, 32, 32	sigmoid, sigmoid, relu, linear	Fourier	9	0.002

Table S2: Configurations for the functional neural network in the bike example.

S4.1.2 Paired T-Tests

Model Compared	P-Value	T-Value	95% Lower Bound	95% Upper Bound
Functional Linear Model (Basis)	0.132	1.19	-0.228	0.939
Functional PC Regression	0.0286	2.18	0.0707	1.34
Functional PC Regression (2nd Deriv Penalization)	0.00127	4.14	1.53	4.29
Functional PC Regression (Ridge Regression)	0.00218	3.78	0.376	1.19
Functional Partial Least Squares	0.167	1.02	-0.322	1.02
Functional Partial Least Squares (2nd Deriv Penalization)	0.0662	1.65	-0.0641	0.759
Convolutional Neural Networks	0.307	0.524	-0.719	1.24
Neural Networks	0.126	1.23	-0.364	1.58

Table S3: Paired T-Tests comparing FNN results with other models for the bike example.

S4.1.3 Training Plots

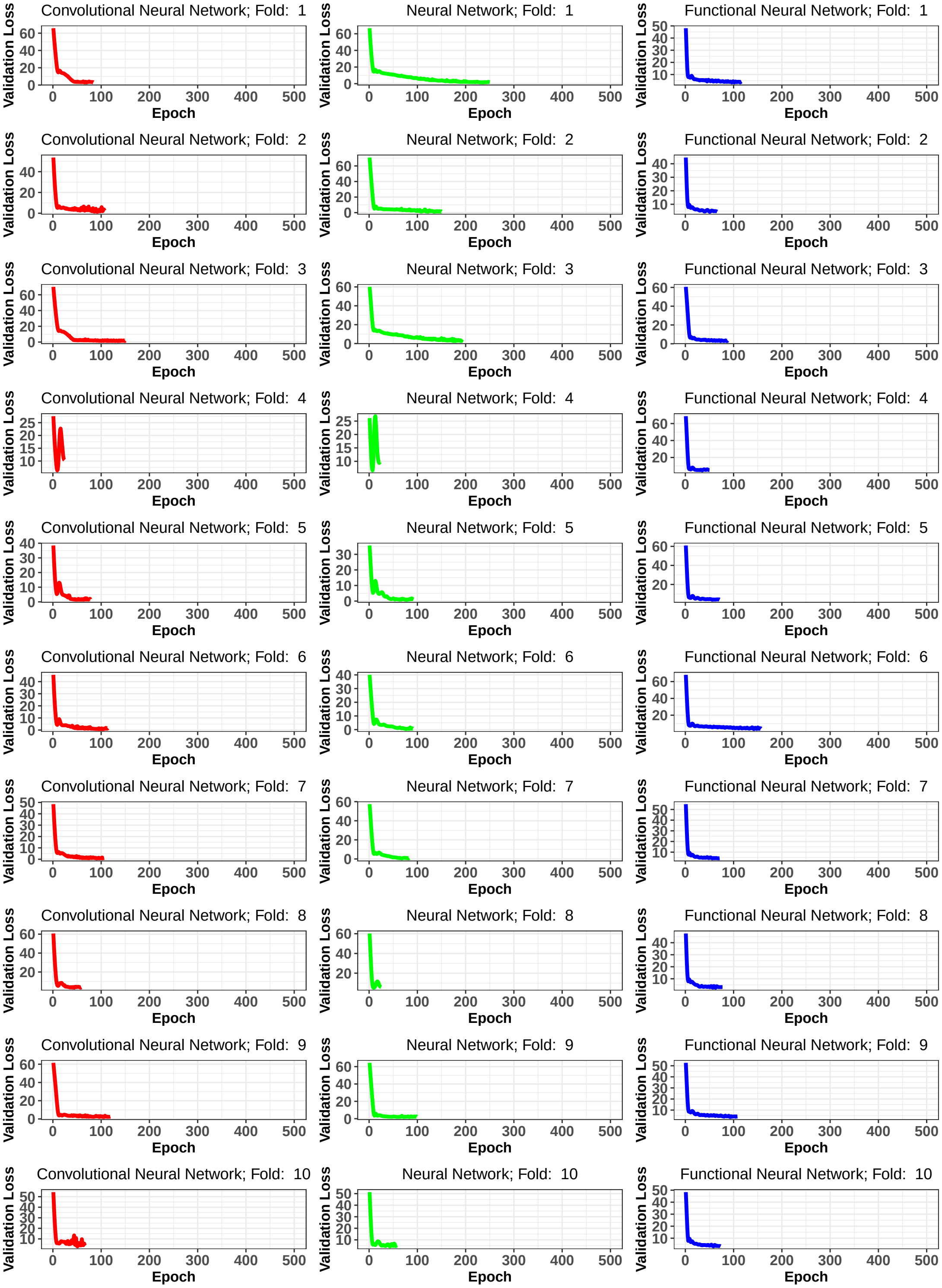

Figure S2: Training plots for the neural network models in the bike example.

S4.1.4 Functional Neural Network Weights

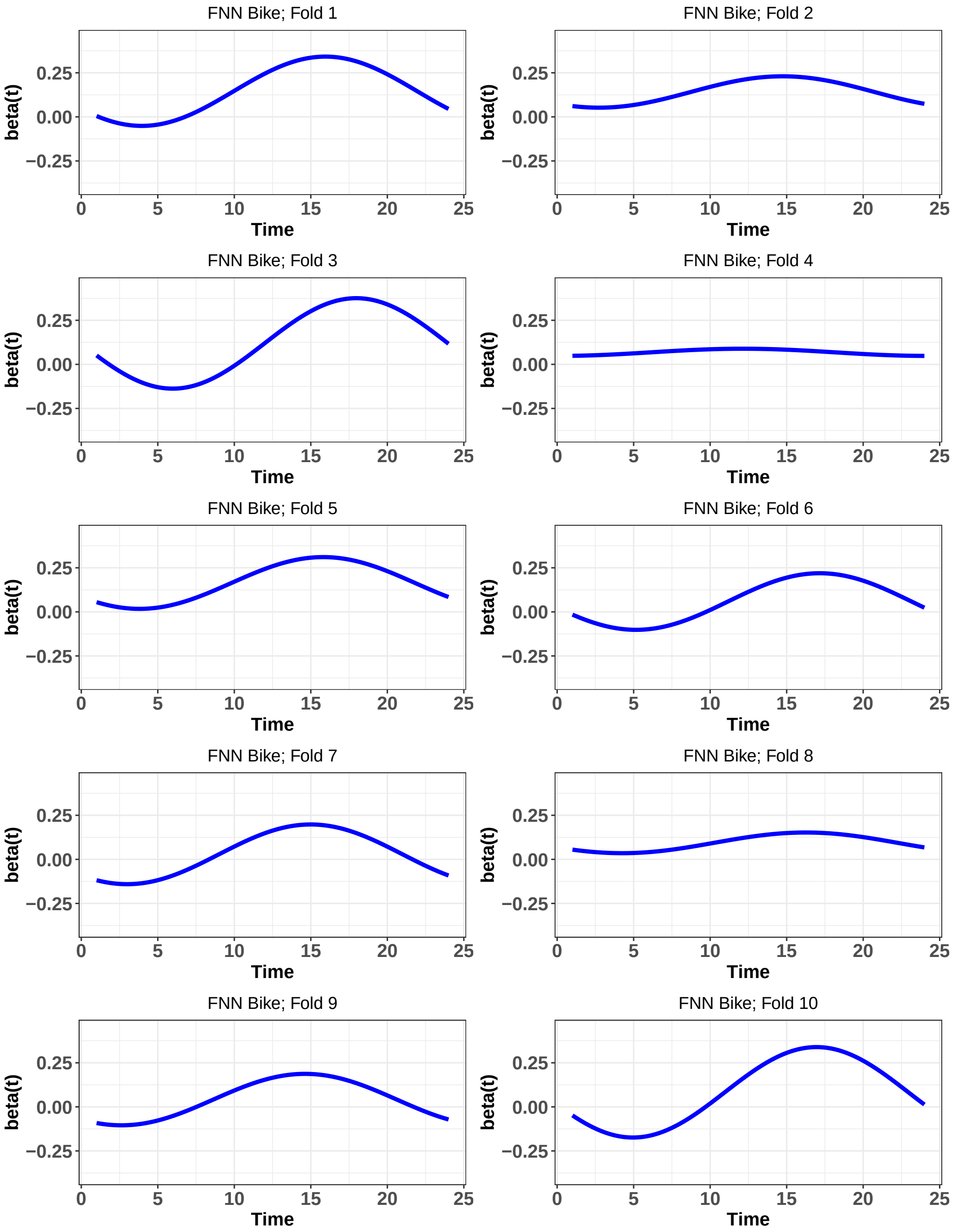

Figure S3: Estimate of $\beta(t)$ for each fold in the bike example (FNN).

S4.1.5 Functional Linear Model Weights

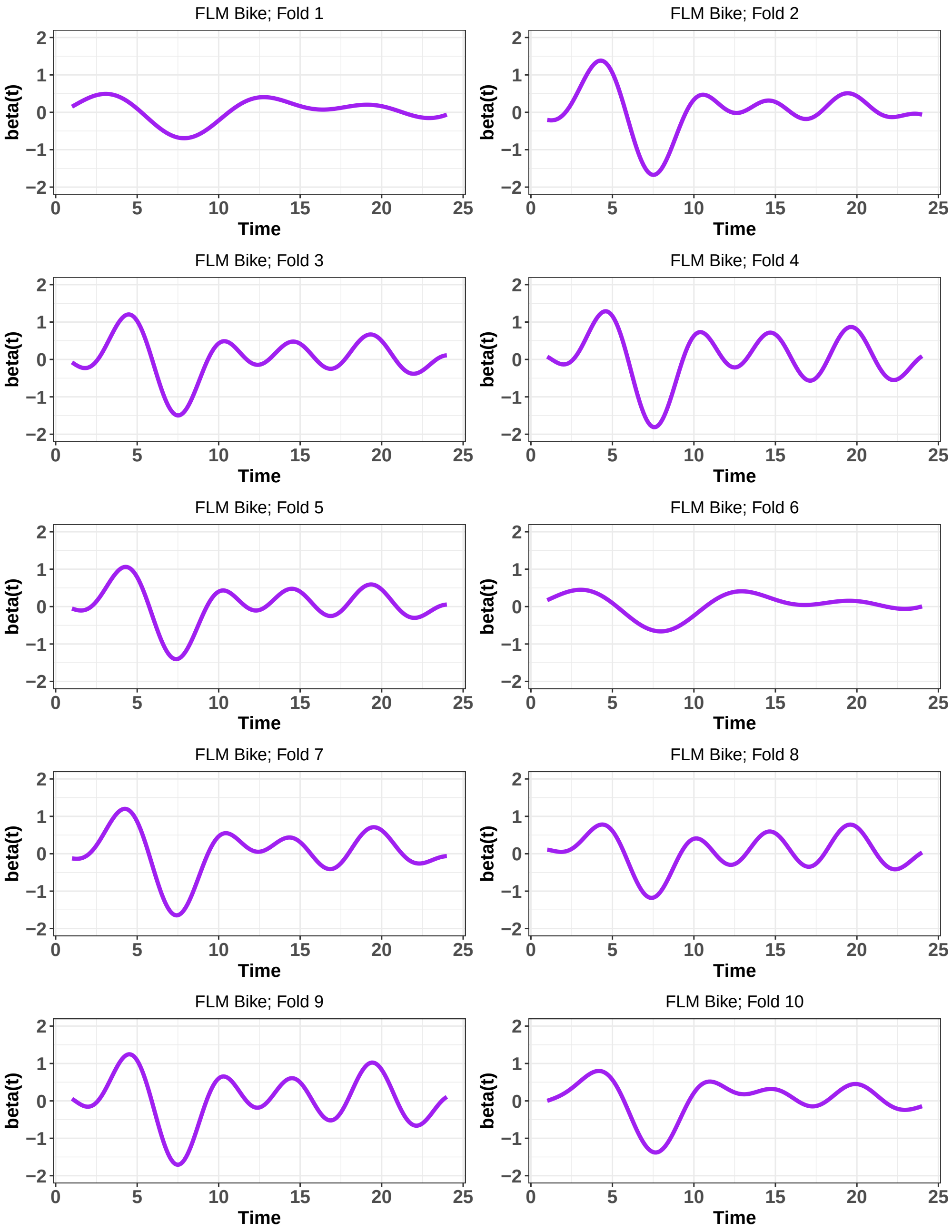

Figure S4: Estimate of $\beta(t)$ for each fold in the bike example (FLM).

S4.2 Tecator

S4.2.1 Training Plots

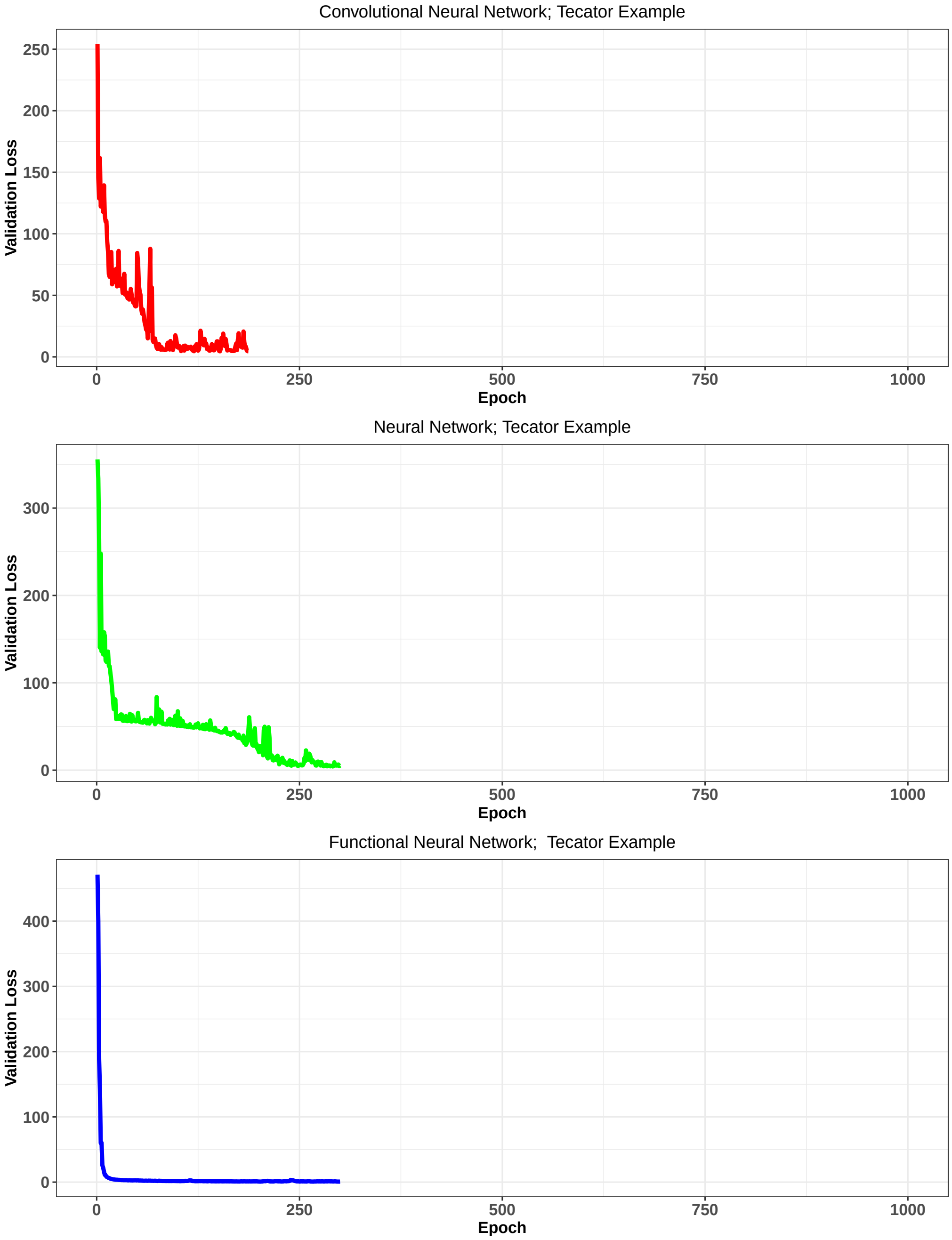

Figure S5: Training plots for the neural network models in the tecator example.

S4.2.2 Model Configurations

Model	Layers	Num. Neurons	Activations	Functional Weight Function	Functional Weight Terms	Learn Rate
Tecator	6	24, 24, 24, 24, 24, 58	relu, relu, relu, relu, relu, relu	Fourier	3	0.005

Table S4: Configurations for the functional neural network in the tecator example.

S4.3 Weather

S4.3.1 Model Configurations

Model	Layers	Num. Neurons	Activations	Functional Weight Function	Functional Weight Terms	Learn Rate
Weather	2	16, 8	relu, sigmoid	Fourier	5	0.05

Table S5: Configurations for the functional neural network in the weather example.

S4.3.2 Paired T-Tests

Model Compared	P-Value	T-Value	95% Lower Bound	95% Upper Bound
Functional Linear Model (Basis)	0.136	1.17	-0.387	1.53
Functional Non-Parametric Regression	0.112	1.30	-0.0211	0.106
Functional PC Regression	0.331	0.451	-0.019	0.0296
Functional PC Regression (2nd Deriv Penalization)	0.202	0.875	-0.0308	0.0804
Functional PC Regression (Ridge Regression)	0.360	0.370	-0.0194	0.0284
Functional Partial Least Squares	0.126	1.23	-0.0141	0.0610
Functional Partial Least Squares (2nd Deriv Penalization)	0.0982	1.40	-0.0110	0.0652
Convolutional Neural Networks	0.0510	1.82	-0.00139	0.0380
Neural Networks	0.0805	1.53	-0.00360	0.0291

Table S6: Paired T-Tests comparing FNN results with other models for the weather example.

S4.3.3 Training Plots

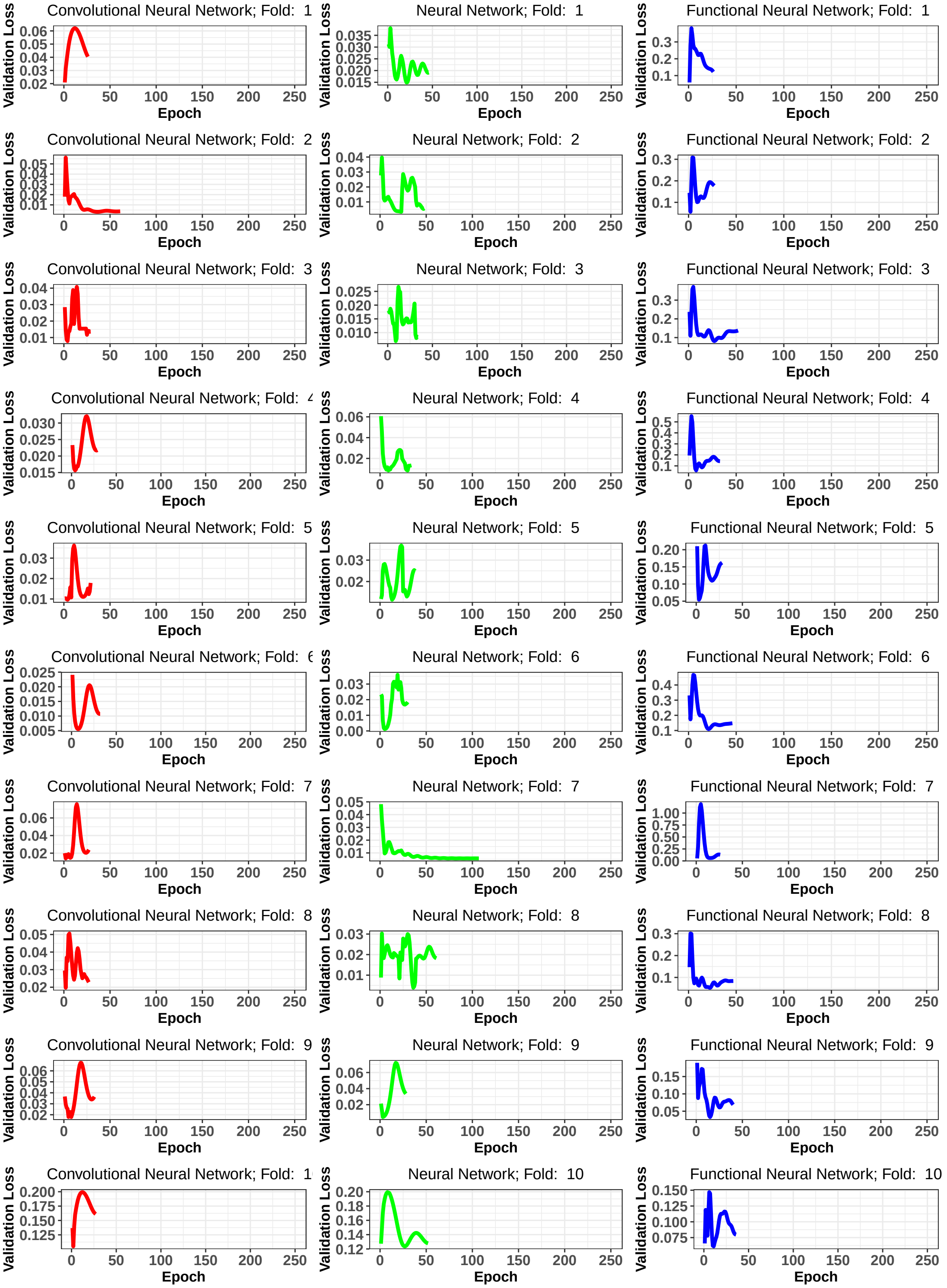

Figure S6: Training plots for the neural network models in the weather example (FNN).

S4.3.4 Functional Neural Network Weights

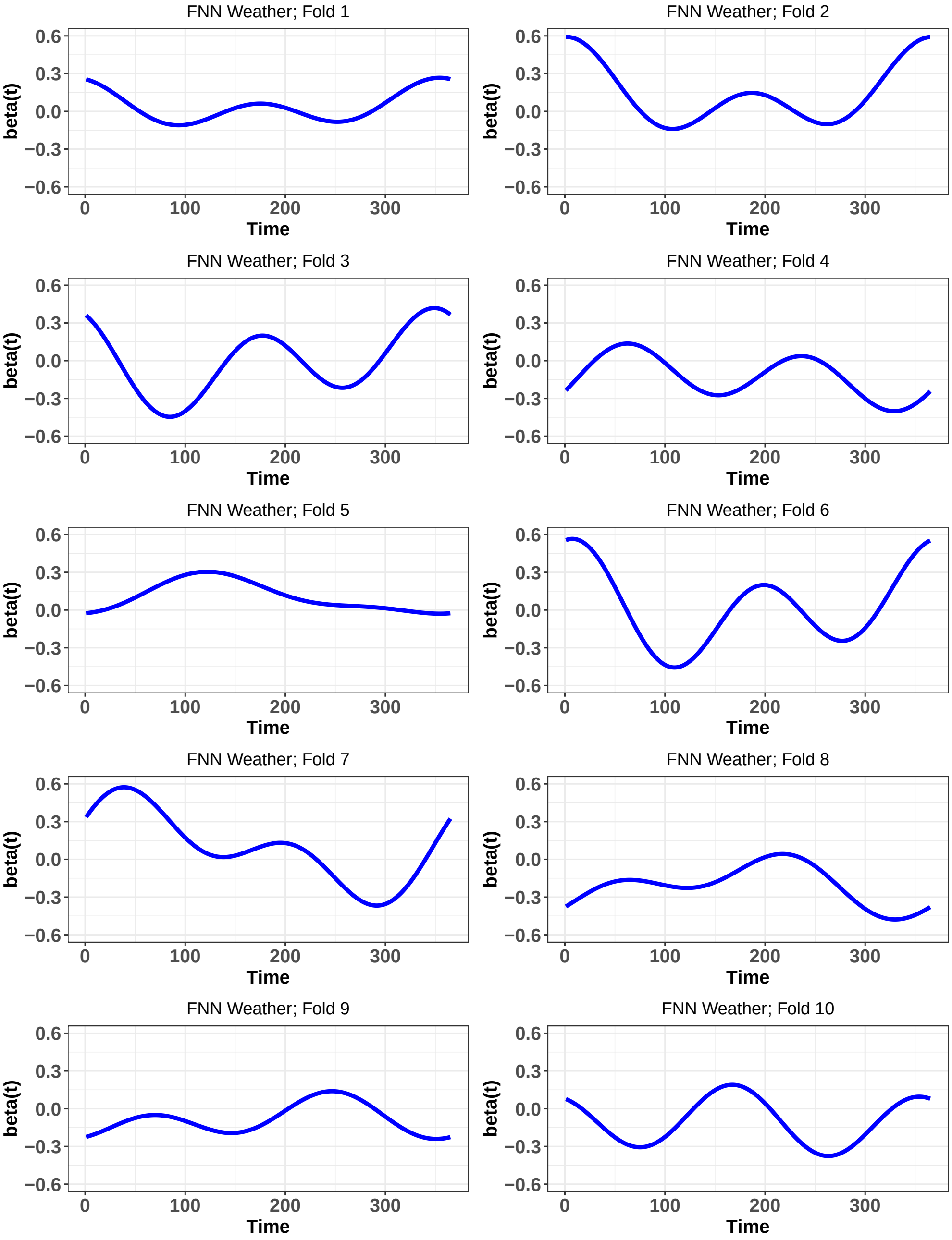

Figure S7: Estimate of $\beta(t)$ for each fold in the weather example.

S4.3.5 Functional Linear Model Weights

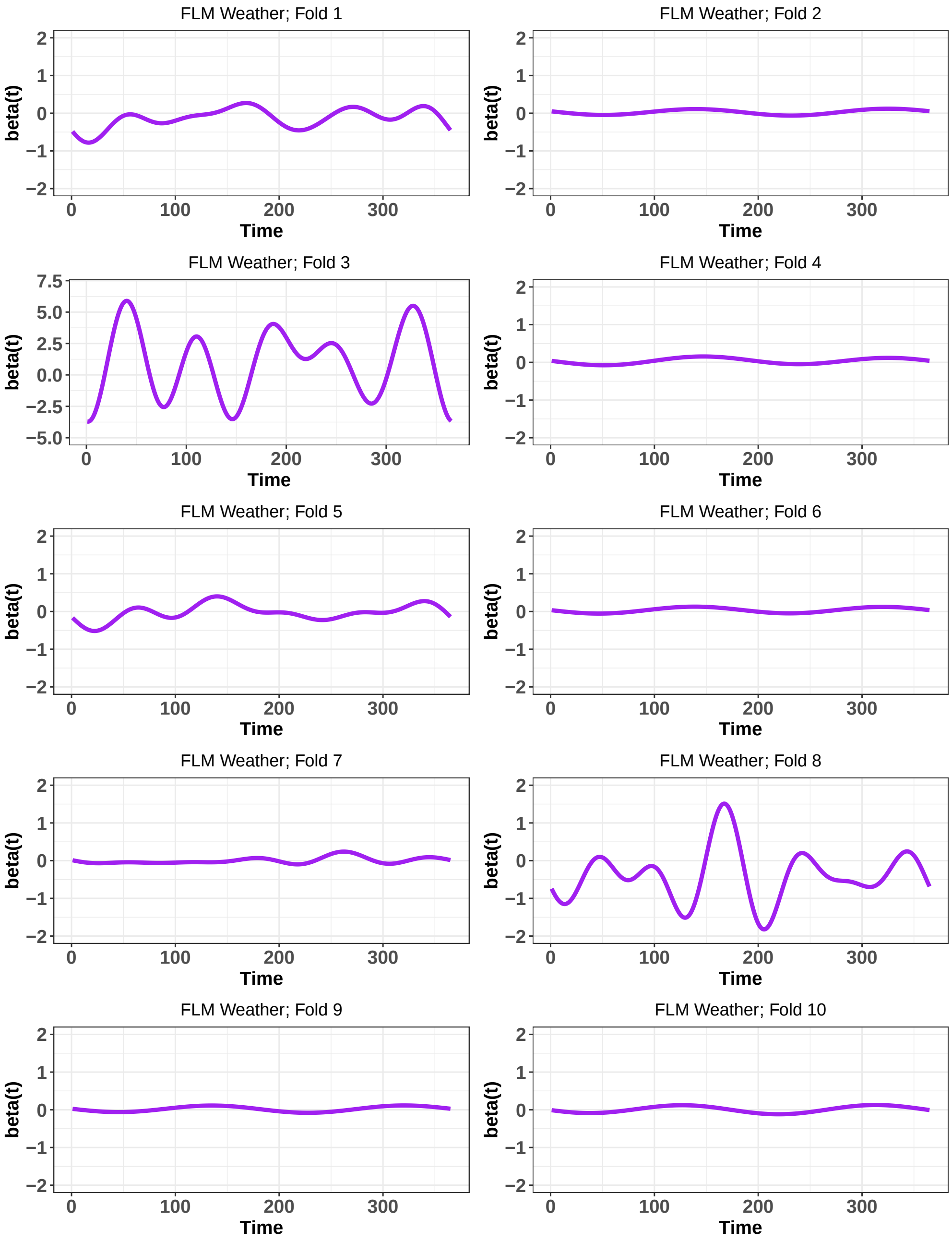

Figure S8: Estimate of $\beta(t)$ for each fold in the weather example (FLM).

S4.4 Simulation Recovery

S4.4.1 Model Configurations

Model	Layers	Num. Neurons	Activations	Functional Weight Function	Functional Weight Terms	Learn Rate
Sim 1 (Linear)	3	16, 16, 16	relu, linear, linear	Fourier	5	0.001
Sim 2 (Exponential)	3	16, 16, 16	relu, linear, linear	Fourier	5	0.001
Sim 3 (Sigmoidal)	1	16	sigmoid	Fourier	5	0.001
Sim 4 (Logarithmic)	3	16, 16, 16	relu, linear, linear	Fourier	5	0.001

Table S7: Configurations for the functional neural network in the inference simulation example.

S4.4.2 Paired T-Tests

Model Compared	P-Value	T-Value	95% Lower Bound	95% Upper Bound
Sim 1 (Linear)	< 0.0001	33.0	0.111	0.126
Sim 2 (Exponential)	< 0.0001	-54.9	-0.167	-0.156
Sim 3 (Sigmoidal)	< 0.0001	-38.7	-0.281	-0.254
Sim 4 (Logarithmic)	< 0.0001	179	0.954	0.975

Table S8: Paired T-Tests comparing FNN with the functional linear model IMSE results for four simulation scenarios.

S4.5 Simulation Prediction

S4.5.1 Model Configurations

Model	Layers	Num. Neurons	Activations	Functional Weight Function	Functional Weight Terms	Learn Rate
Sim 1 (Linear)	3	16, 16, 16	relu, linear, linear	Fourier	5	0.001
Sim 2 (Exponential)	3	16, 16, 16	relu, linear, linear	Fourier	5	0.001
Sim 3 (Sigmoidal)	3	16, 16, 16	relu, linear, linear	Fourier	5	0.001
Sim 4 (Logarithmic)	3	16, 16, 16	relu, linear, linear	Fourier	5	0.001

Table S9: Configurations for the functional neural network in the prediction simulation example.

S4.5.2 Error Rates

Sim/Model	Simulation 1	Simulation 2	Simulation 3	Simulation 4
Functional Linear Model (Basis)	0.233	0.731	0.0194	0.365
Functional PC Regression	0.232	0.743	0.0200	0.367
Functional PC Regression (2nd Deriv Penalization)	0.123	0.733	0.0169	0.367
Functional PC Regression (Ridge Regression)	0.232	0.747	0.0201	0.376
Functional Partial Least Squares	0.160	0.812	0.0205	0.610
Functional Partial Least Squares (2nd Deriv Penalization)	0.144	0.818	0.0214	0.582
Convolutional Neural Networks	0.256	0.684	0.0374	0.357
Neural Networks	0.727	1.14	0.0815	0.781
Functional Neural Networks	0.140	0.646	0.0205	0.261
Multiple Linear Regression	0.458	1.34	0.0637	1.95
LASSO - Min λ	0.170	0.809	0.0191	0.376
LASSO - 1SE λ	0.196	0.846	0.0207	0.383
Random Forest	0.266	0.713	0.0193	0.247
Gradient Boosting	0.280	0.723	0.0195	0.251
Projection Pursuit Regression	0.363	1.04	0.0392	1.07
Extreme Gradient Boosting	0.256	0.919	0.0217	0.255

Table S10: MSPE values for simulated data predictions. The FNN approach outperforms the others in 3 of the 4 simulations. In the one case that it does not, it performs comparable tot he best.

S4.5.3 Paired T-Tests

Model Compared	Sim Number	P-Value	T-Value	95% Lower Bound	95% Upper Bound
Functional Linear Model (Basis)	1	< 0.0001	8.87	0.0723	0.113
Functional PC Regression	1	< 0.0001	6.96	0.00663	0.118
Functional PC Regression (2nd Deriv Penalization) (Ridge Regression)	1	0.0597	-1.58	-0.0379	0.00413
Functional PC Regression (Ridge Regression)	1	< 0.0001	6.60	0.0648	0.119
Functional Partial Least Squares	1	0.0613	1.56	-0.00522	0.0456
Functional Partial Least Squares (2nd Deriv Penalization)	1	0.367	0.339	-0.0206	0.0292
Convolutional Neural Network	1	< 0.0001	8.26	0.0883	0.143
Neural Network	1	< 0.0001	5.65	0.384	0.791
Multiple Linear Regression	1	< 0.0001	7.39	0.234	0.403
LASSO - Min λ	1	0.00724	2.49	0.00652	0.0548
LASSO - 1SE λ	1	0.000231	3.62	0.0259	0.0869
Random Forest	1	< 0.0001	5.12	0.0781	0.175
Gradient Boosting	1	< 0.0001	5.07	0.0858	0.194
Projection Pursuit Regression	1	< 0.0001	12.2	0.187	0.258
Extreme Gradient Boosting	1	< 0.0001	4.68	0.0680	0.166
Functional Linear Model (Basis)	2	0.00197	2.95	0.0283	0.140
Functional PC Regression	2	0.000992	3.18	0.0371	0.157
Functional PC Regression (2nd Deriv Penalization)	2	0.00141	3.06	0.0311	0.142
Functional PC Regression (Ridge Regression)	2	0.000859	3.22	0.0395	0.162
Functional Partial Least Squares	2	< 0.0001	4.92	0.0997	0.232
Functional Partial Least Squares (2nd Deriv Penalization)	2	< 0.0001	5.69	0.113	0.231
Convolutional Neural Network	2	0.127	1.14	-0.0268	0.102
Neural Network	2	< 0.0001	6.17	0.340	0.656
Multiple Linear Regression	2	< 0.0001	6.58	0.489	0.905
LASSO - Min λ	2	< 0.0001	4.02	0.0836	0.242
LASSO - 1SE λ	2	< 0.0001	4.38	0.110	0.289
Random Forest	2	0.0149	2.20	0.00732	0.125
Gradient Boosting	2	0.0119	2.29	0.0111	0.142
Projection Pursuit Regression	2	< 0.0001	6.33	0.274	0.519
Extreme Gradient Boosting	2	< 0.0001	7.94	0.206	0.340
Functional Linear Model (Basis)	3	0.238	-0.715	-0.00428	0.00199
Functional PC Regression	3	0.370	-0.332	-0.00348	0.00248
Functional PC Regression (2nd Deriv Penalization)	3	0.000236	-3.616	-0.00562	-0.00167
Functional PC Regression (Ridge Regression)	3	0.392	-0.276	-0.00325	0.00245
Functional Partial Least Squares	3	0.481	0.0480	-0.00340	0.00357
Functional Partial Least Squares (2nd Deriv Penalization)	3	0.114	1.22	-0.000566	0.00241
Convolutional Neural Network	3	0.000606	3.33	0.00696	0.0268
Neural Network	3	< 0.0001	8.52	0.0470	0.0751
Multiple Linear Regression	3	< 0.0001	10.5	0.0351	0.0512
LASSO - Min λ	3	0.168	-0.965	-0.00422	0.00143
LASSO - 1SE λ	3	0.440	0.152	-0.00314	0.00366
Random Forest	3	0.230	-0.741	-0.00449	0.00202
Gradient Boosting	3	0.280	-0.585	-0.00424	0.00229
Projection Pursuit Regression	3	< 0.0001	10.4	0.0152	0.0222
Extreme Gradient Boosting	3	0.235	0.724	-0.00207	0.00449
Functional Linear Model (Basis)	4	< 0.0001	9.19	0.0819	0.126
Functional PC Regression	4	< 0.0001	9.40	0.0843	0.129
Functional PC Regression (2nd Deriv Penalization)	4	< 0.0001	9.49	0.0846	0.129
Functional PC Regression (Ridge Regression)	4	< 0.0001	9.74	0.0918	0.138
Functional Partial Least Squares	4	< 0.0001	12.7	0.2954	0.403
Functional Partial Least Squares (2nd Deriv Penalization)	4	< 0.0001	16.5	0.283	0.359

Model Compared	Sim Number	P-Value	T-Value	95% Lower Bound	95% Upper Bound
Convolutional Neural Network	4	< 0.0001	4.06	0.0499	0.143
Neural Network	4	< 0.0001	8.93	0.407	0.635
Multiple Linear Regression	4	< 0.0001	6.99	1.21	2.16
LASSO - Min λ	4	< 0.0001	11.9	0.0966	0.134
LASSO - 1SE λ	4	< 0.0001	11.6	0.102	0.143
Random Forest	4	0.0774	-1.43	-0.0315	0.00488
Gradient Boosting	4	0.156	-1.01	-0.0279	0.00888
Projection Pursuit Regression	4	< 0.0001	16.5	0.718	0.919
Extreme Gradient Boosting	4	0.273	-0.606	-0.0246	-0.0130

Table S11: Paired T-Tests comparing FNN results with other models for the simulation prediction example.

S4.5.4 Relative MSPE Boxplots

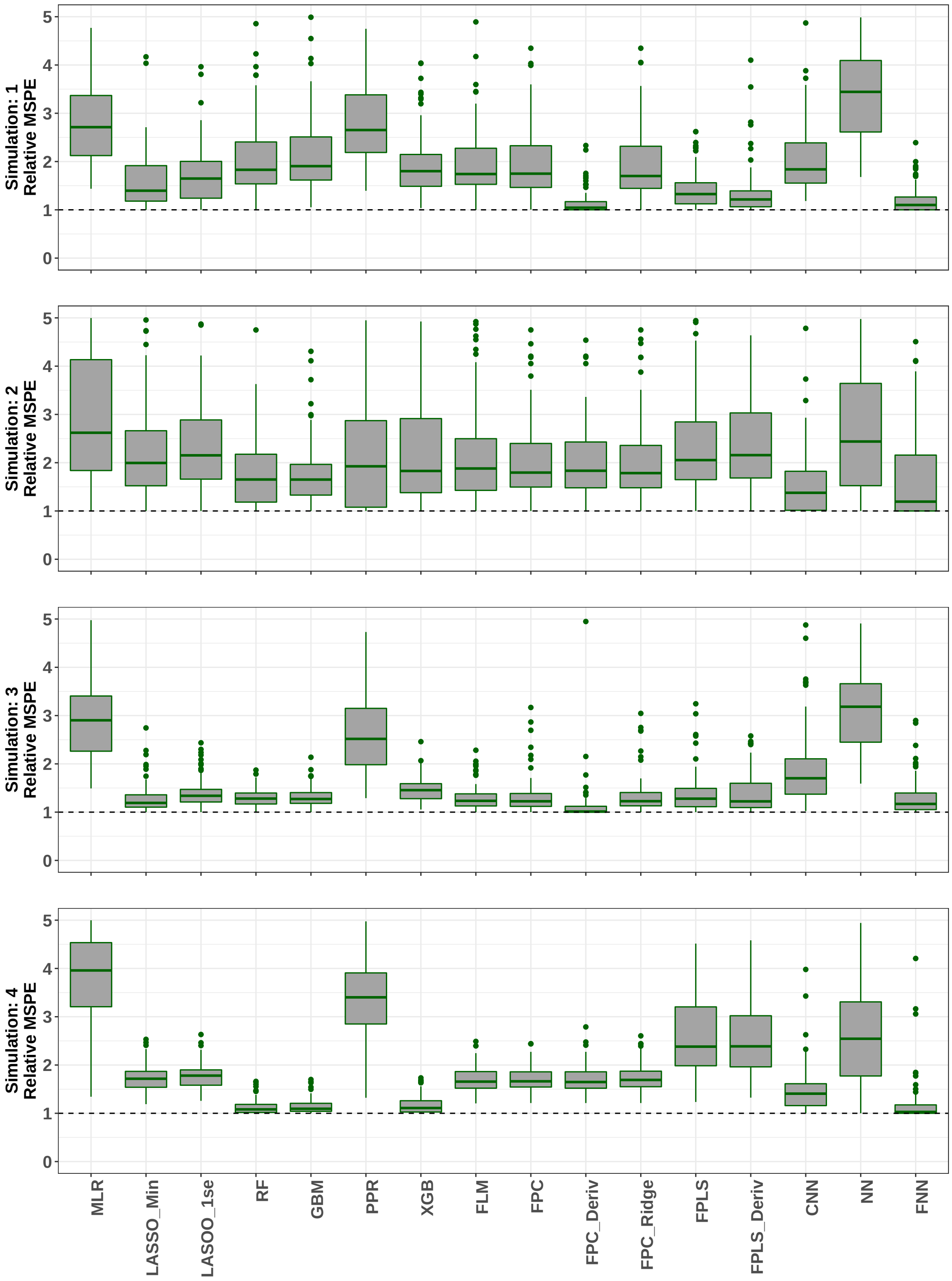

Figure S9: Boxplot of the relative mean squared predicted errors (rMSPEs) for sixteen methods in four simulation scenarios.

References

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, **2**(4), 303–314.