Responses to Comments from the Associate Editor on Manuscript JCGS-20-212

- *The reviewers found the general contribution potentially of use to the community, but identified several areas for improvement. We request that the authors submit a revised version of this manuscript that addresses these concerns.*

Response: Thank you and two reviewers for the careful review and constructive comments. These comments are very helpful for us to improve our work. We have revised our manuscript by taking into account all the comments. Our point-by-point responses to these comments are as follows, with the original comments in italics for convenience.

- *Compare with other methods for sequence data, such as RNNs or convolutional NNs, including experimental comparisons*

Response: Thank you for your suggestion! First, to address the lack of comparison to recurrent neural networks, we would like to point out that types of problems approached by them is different than the ones that presented here. In our humble experience, we have often seen recurrent neural networks used for problems where the response of interest is the next time point in a continuum; consider our bike example – the response in this problem is the aggregate bike total in a given day where the covariate is the temperature curve for the corresponding day. We intended to use the temperature curve to predict the response in our scenarios so we used methods that made sense for such problems (namely, functional methods). If our goal was to use the preceding days of bike rental counts to predict the next day, then we would definitely have considered recurrent neural networks. Moreover, it's not clear how to structure data for when the problem is as it is in the bike example – there is a single scalar response value tied to 24 values along some continuum that is on a different domain than the scalar value. Another example can be seen with the Tecator data set where we are attempting to predict the fat contents of meat samples using spectrometer curves. There is no temporal component to the meat samples and each combination of functional covariate and response could be safely assumed to be independent from one another. Application of RNNs in this kind of example is not as clear as it is with functional neural networks where we can formulate the problem as having one functional covariate and a scalar response As far as 1D convolutional neural networks (CNN) are concerned, we thought your point was warranted and have now included the results for them in all of our examples, experiments, and simulations.

- *Elaborate on claims of interpretability*

Response: Thank you for this suggestion. Before discussing our claims of interpretability, we need to establish a definition for interpretability. We appeal to Fan et al. (2021) in which interpretability is defined as "the extent of human's ability to understand and reason a model". Furthermore, we are concerned with the relative interpretability of FNNs in comparison with conventional neural networks (which are referred to as multilayer perceptrons in Tibshirani et al. (2009)) since that is the infrastructure on which we have developed these networks. In conventional neural networks, the inordinate number of parameters make interpretability a difficult task and this large amount of parameters is responsible for the "black-box" reputation of such neural networks. We are not concerned with interpretability in any sense other than the one that would be associated with the parameters – in Zhang et al. (2020), this is referred to as hidden semantic interpretability i.e. the interpretability associated with a hidden layer(s) of a neural network. A lot of the work that has been done on hidden semantic interpretability is associated with convolutional neural networks (Zhang et al., 2018; Simonyan et al., 2013) and not conventional neural networks. What makes FNNs more interpretable versus the parameters you would find usually in conventional neural networks is their ability to be visualized. For every training iteration of an FNN, we can view the impact of the functional input by visualizing the functional weights of the network. In the case of the conventional neural network, this change would be hundreds or thousands (a value equivalent to the number of weights) of minuscule movements in parameter values. In the case of the functional neural network, this is a shift in a curve. We postulate that this visual component of the FNN functional weights has a much higher likelihood of allowing a human to understand the relationship between the functional covariate and the scalar response than the large assortment of small movements associated with a conventional neural network. For example, in the case of using temperature curves to predict daily bike rentals, we found that the (average) functional weight associated with the functional covariate of temperature was consistently highest in the middle of the day and tapered off during earlier and later hours of the day (and near zero over night); the interpretation of this functional weight here is that the relationship between the functional covariate of temperature and bike rentals is most pronounced during the early afternoon which coincides with our intuition – bike rental stores are probably not open later in the day and people probably also want to ride bikes during the day time (or, at least rent them earlier). We also see in the case of the simulation examples where the goal is to recover $\beta(t)$, we tend to perform better when we use functional neural networks as opposed to the functional linear model – this serves as empirical support for our claims with respect to the accuracy of the functional recovery in comparison to the true value in nature. We do understand that these claims are not substantiated by theory, but we feel that there is an intuitive aspect to this kind of interpretability which we hoped to relay with our examples. We recognize that some questions naturally arise from this explanation such as how to summarize the functional weights into a single statistic. We see this as an open area of research. One potential direction is to use the work done in Dalvi et al. (2019) in which neurons are ranked on their significance in a linear model – a similar approach could be used to identify which functional weight should be considered if there is not an appetite for a summary curve. These significance rankings could also constitute weights. These points are added in Section 2.3 of the manuscript.

- *Perform ablation studies to explore the roles of hyperparameters and design decisions.*

Response: Following your recommendation, we have performed ablation studies to understand the role of the different hyperparameters. The study results are summarized in Section S3 of the supplemental document as well as in Section 2.4 in the manuscript. In particular, we considered 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. The results are shown in Figure S1 of the supplementary file. 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.

Responses to Comments from the Reviewer 1 on Manuscript JCGS-20-212

- *This paper proposes a neural network architecture for functional data. The key idea is to construct a layer taking both functional inputs and other covariates to be connected to subsequent fully-connected layers.*

  Response: Thank you for your careful review and constructive comments. These comments are very helpful for us to improve our work. We have revised our manuscript by taking into account all the comments. Our point-by-point responses to these comments are as follows, with the original comments in italics for convenience.

- *My biggest concern about the proposed work is its lack of comparison (both theoretically and empirically) to the neural networks processing sequence data. As far as I understand, the actual functional data are discretized to be represented as a sequence of function values evaluated at some time-steps (if only a function expression is given as input we can just equally divide an interval and evaluate function values on those equally spaced points to discretize the function). For such data, the first option I would consider is the recurrent neural network or a 1D convolutional neural network which use a fixed number of parameters (number of parameters does not scale with the number of time-steps) and yet can effectively extract some feature representations of the functions (represented by sequences). If the number of parameters is not a big issue, we may resort to transformer-like architecture which is good at capturing long-range dependencies. Is there any reason that these baselines in neural networks context not applicable to the problem discussed in this paper? If interpretability is an issue, there are plenty of works letting the neural networks for sequences interpretable, for instance, we can construct interpretable RNNs as in [Choi et al., 2016, Huh et al., 2018].*

  To summarize, for now, I don't see a strong reason to consider the proposed model as an alternative for modeling functional data (at least among neural networks), so this part should be clarified. I'd like to see a strong reason not to compare the proposed model to RNNs, or actually empirical comparison to show the benefit of the proposed model.

Response: First, to address the lack of comparison to recurrent neural networks, we would like to point out that types of problems approached by them is different than the ones that presented here. In our humble experience, we have often seen recurrent neural networks used for problems where the response of interest is the next time point; consider our bike example – the response in this problem is the aggregate bike total in a given day where the covariate is the temperature curve for the corresponding day. We intended to use the temperature curve to predict the response in our scenarios so we used methods that made sense for such problems (namely, functional methods). If our goal was to use the preceding days of bike rental counts to predict the next day, then we would definitely have considered recurrent neural networks. Moreover, it's not clear how to structure data for when the problem is as it is in the bike example – there is a single scalar response value tied to 24 values along some continuum that is on a different domain than the scalar value. As far as 1D convolutional neural networks (CNN) are concerned, we thought your criticism was warranted and have now included the results for them in all of our examples and simulations. We note that we often obtained good results from the functional neural network (FNN) by using a functional weight curve ($\beta(t)$) that is made up much fewer terms than the raw data – this resulted in every example of ours having less parameters for the FNN than the CNN. We invite you to reconsider your opinion on these networks as here, they demonstrate lower error rates than comparable neural network methods while using less parameters. As far as interpretability is concerned, we were simply pointing out the functional nature of the FNN model weights – we can see how a random initialization of this weight curve changes as the network learns and, because this curve is weighted over the same domain as the functional observations ($x(t)$), we have observed some interpretable nature to these functional weights (see Example 3.1 in the manuscript).

Responses to Comments from the Reviewer 2 on Manuscript JCGS-20-212

- *The paper proposes a neural network architecture for functional data. Specifically, it models scalar responses with multiple functional and scalar covariates. The model is trained using stochastic gradient descent, and hyperparameters are selected by grid search. Experiments are performed on simulated data and three real-world datasets.*

Response: Thank you for your careful review, nice summary, and constructive comments. These comments are very helpful for us to improve our work. We have revised our manuscript by taking into account all the comments. Our point-by-point responses to these comments are as follows, with the original comments in italics for convenience.

- *I think the proposed model is a useful contribution to neural networks with functional data. The paper would be easy to read for a deep learning expert, but could be confusing for a "traditional statistician". In particular, since the authors propose a new architecture, they should study the role of each hyperparameter to help the reader understand the model. Finally, writing should be improved to avoid confusion, and improve clarity.*

Response: Following your recommendation, we have added an experimental study to understand the role of the different hyperparameters. The study results are summarized in Section S3 of the supplemental document. In particular, we considered 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. The results are shown in Figure S1 of the supplementary file. 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.

We have also improved our writing to avoid confusion and improve clarity.

- *Introduction, page 2, line 20: The transition from the semi-functional partial linear model to neural networks is not smooth. Multiple key terms appear in the introduction without a proper introduction (neuron, activation function, vanishing gradient, early stopping, dropout, etc). I do not think it is a good idea to assume the reader is familiar with all the details in neural network modeling. Perhaps, this should not appear in the introduction and should be properly introduced in a future section.*

Response: We have followed your suggestion and moved the expression for functional neural networks to Section 2, where we introduce key terms in details.

- *An experimental study should be performed to understand the role of the different hyperparameters: M_k, the number of basis functions, initialization, learning rate, depth, etc. The sensitivity to the numerical integration method is also important.*

Response: Thank you for the feedback! Following your recommendation, we have added an experimental study to understand the role of the different hyperparameters. The study results are summarized in Section S3 of the supplemental document.

We have summarized our work in a previous question but will repeat here for convenience. In particular, we considered the number of basis functions defining the functional weight, the learning rate, the number of neurons, the validation split rate, the decay rate, and the number of epochs. 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 results are shown in Figure S1 of the supplementary file. 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.

- *Why densely connected? Many other architectures have been considered in the literature. For example, one-dimensional convolutional networks have been successfully used with sequential/time series data. The authors should clearly motivate their choices.*

Response: Thank you for the question. We recognize that there is no reason that the layer specifics of the functional neural network be limited to those of densely-connected neural networks and have amended this point in the manuscript.

- *The authors claim that their model is "interpretable". However, they have not introduced or cited some of the related literature on interpretability/explainability in neural networks.*

Response: Thank you for this suggestion. Before discussing our claims of interpretability, we need to establish a definition for interpretability. We appeal to Fan et al. (2021) in which interpretability is defined as "the extent of human's ability to understand and reason a model". Furthermore, we are concerned with the relative interpretability of FNNs in comparison with conventional neural networks (which are referred to as multilayer perceptrons in Tibshirani et al. (2009)) since that is the infrastructure on which we have developed these networks. In conventional neural networks, the inordinate number of parameters make interpretability a difficult task and this large amount of parameters is responsible for the "black-box" reputation of such neural networks. We are not concerned with interpretability in any sense other than the one that would be associated with the parameters – in Zhang et al. (2020), this is referred to as hidden semantic interpretability i.e. the interpretability associated with a hidden layer(s) of a neural network. A lot of the work that has been done on hidden semantic interpretability is associated with convolutional neural networks (Zhang et al., 2018; Simonyan et al., 2013) and not conventional neural networks. What makes FNNs more interpretable versus the parameters you would find usually in conventional neural networks is their ability to be visualized. For every training iteration of an FNN, we can view the impact of the functional input by visualizing the functional weights of the network. In the case of the conventional neural network, this change would be hundreds or thousands (a value equivalent to the number of weights) of minuscule movements in parameter values. In the case of the functional neural network, this is a shift in a curve. We postulate that this visual component of the FNN functional weights has a much higher likelihood of allowing a human to understand the relationship between the functional covariate and the scalar response than the large assortment of small movements associated with a conventional neural network. For example, in the case of using temperature curves to predict daily bike rentals, we found that the (average) functional weight associated with the functional covariate of temperature was consistently highest in the middle of the day and tapered off during earlier and later hours of the day (and near zero over night); the interpretation of this functional weight here is that the relationship between the functional covariate of temperature and bike rentals is most pronounced during the early afternoon which coincides with our intuition – bike rental stores are probably not open later in the day and people probably also want to ride bikes during the day time (or, at least rent them earlier). We also see in the case of the simulation examples where the goal is to recover $\beta(t)$, we tend to perform better when we use functional neural networks as opposed to the functional linear model – this serves as empirical support for our claims with respect to the accuracy of the functional recovery in comparison to the true value in nature. We do understand that these claims are not substantiated by theory, but we feel that there is an intuitive aspect to this kind of interpretability which we hoped to relay with our examples. We recognize that some questions naturally arise from this explanation such as how to summarize the functional weights into a single statistic. We see this as an open area of research. One potential direction is to use the work done in Dalvi et al. (2019) in which neurons are ranked on their significance in a linear model – a similar approach could be used to identify which functional weight should be considered if there is not an appetite for a summary curve. These significance rankings could also constitute weights. These points are added in Section 2.3 of the manuscript

- *In the experiments, data and methods should be properly introduced. The authors should not have to read references to understand the data you have considered.*

Response: Thank you for this comment! We have added more detailed introduction of data and methods.

- *Regularization is often used with deep nets. This is not discussed in the paper. If so, the hyperparameter associated to regularization should be clearly identified.*

Response: Thank you for the question. We did not regularize the models presented in this paper. While your question is valid, it was not the primary concern of the paper. It is possible to regularize and this method can be incorporated into the functional neural network if needed however, this can be said about any number of customization options that a neural network can go through. If not purely for the sake of brevity, we do not consider regularization at this time.

- *Learning curves plot (training and validation error as a function of iterations/epochs) should be provided to confirm the model has been properly learned (at least in the appendix).*

Response: We have now included these for every example and simulation as figures in the supplementary document.

- *Theorem 1 is a direct application of Cybenko (1989). Is "Theorem" the right term to use?*

Response: Thank you for the observation. As this is indeed a direct application, we have amended this part of the text to now refer to it as a corollary.

- *Page 9, "the number of training iterations" is known as the number of epochs. Epoch is mentioned in Figure 2, without a proper introduction.*

Response: Thanks for pointing this out. We have added an introduction for "epoch" in the main text.

- *Page 9, you switch between M and M_k. Please be consistent.*

Response: We've made the suggested change and the notation is now consistent across the entire paper.

- *Page 9, the link between P and K was not clear to me. Please update your explanation.*

Response: We have revised our explanation as follows: "Lastly, we would like to emphasize that the number of parameters in the functional neural network presented here can often be lower than the amount required in neural networks that use raw data. Suppose we only have one functional covariate as the input in the neural network, and we have repeat measurements of the functional covariate at P points along its continuum. Passing this information into any conventional multi-layer neural network will mean that the number of parameters in the first layer will be $(P + 1) \cdot n_1$, where n_1 is the number of neurons in the first hidden layer. In our network, the number of parameters in the first layer is the number of basis functions we use to define the functional weight. The number of basis functions M_1 will be less than P because there is no need to have a functional weight that interpolates across all our observed points – we prefer a smooth effect across the continuum to avoid fitting to noise. Therefore, a good practice indicates that the number of parameters in the first layer of our network is $(M_1 + 1) \cdot n_1$ where $M_1 < P$. This example trivially generalizes to K functional covariates."

- *Algorithm 1 is not clearly written.*
  -> The inputs of the algorithm should be clearly specified. "Learning rate gamma", "Loss function R(theta)", "number of basis functions, M or M_k*", etc.*
  -> In 3.ic, I suggest assigning the results in a variable, and then use it in 3.id.
  -> Please refer to formulas in the text for clarity. For example, refer to the loss function in page 8, to the theta' in page 9, etc.
  -> Your stopping criterion is not standard. The number of iterations is hard to identify. We typically also look at the reduction in validation loss.

Response: We have followed your suggestion and re-written Algorithm 1. The stopping criterion is also revised in the Algorithm on Page 10 to follow a more standard one. References are also added.

- *Page 11, it is not clear why averaging the weights is a meaningful way of interpreting the coefficients. Are there any weaknesses to this approach?*

Response: Thank you for this comment. You are correct in pointing out that averaging isn't necessarily the best approach with respect to interpretation. Our intention with that suggestion was more to provide a summary of the estimated functional weights and be able to inspect a single plot as opposed to n_1 plots (one for each neuron). There is the potential that particular functional weights (associated with a particular neuron) may end up modelling different aspects of the relationship between the functional covariate and the response – this can be akin to principle components in PCA. The interpretation mentioned here is more context dependent and it's our apology for making it seem more grandiose than that. In some examples, when considering the context, it can be relatively easy to understand why the functional weights turn out the way they do. For example, in the bike example, the relationship found via the functional weights (after training) has an explanation that is in-line with how a bike rental store may operate when considering daily temperature – this interpretation is offered up in the paper. However, we recognize that in some examples, the merit of the method is dependent on its predictive prowess and reduction in parameters and less so in the interpretation of the functional weights. The conclusion is then that yes, averaging weights can have the weakness of muddying up a more clear interpretation that may appear from looking at individual estimations of $\beta(t)$, but the advantage is the consolidation of potentially hundreds of curves. The summary of these functional weights is definitely not limited to an average.

- *Section 2: "Another approach is to consider a zero-initialization"*
  -> This is considered as a bad initialization method since no gradient updates would be applied.
  -> You did not mention "random search", another popular hyperparameter search approach.
  -> You have already used K before for the number of covariates.
  -> "One important parameter in this particular kind of network is the number of basis functions that govern the functional weights"
  -> It is not clear to me that this is an important parameter. Did you (empirically) investigate its importance? If not, this should be added to the experimental study.

Response: Thank you for pointing this out. Our initial thought was to just point out approaches to weight initialization and did not pay much consideration to the general viewpoint of these approaches. We have now removed the comments about zero-initialization and appended on random search. While initialization is important for any algorithm of this nature, it was not the main focus here. There is no restriction on what the initialization should be for this approach so, any of the usual approaches are fair game. We have also adjusted the naming so that we refer to the cross-validation as a B-Fold Cross-Validation – running out of letters! Finally, with respect to the number of basis functions (and its importance), we have now added in an empirical study which can found in Figure S1 of the supplemental document.

- *Section 3.1: "Our functional observations of temperature (to be passed into the network) are formed using 31 Fourier basis functions." "We used 29 Fourier basis functions to estimate the functional observations."*
  -> what does that mean? Is it the procedure you use to go from discrete data to functional data? -> Also, what is K, J, M?

Response: We have now added additional information in each example to make it explicit what K, J, and M are. And yes, you are correct – that is the procedure of going from the discrete data to the functional data. We have added additional text in all of the examples to make the distinction between the raw data, $x(t)$, and $\beta(t)$ clearer.

- *"Since we are working with functional covariates, we also have access to their derivatives;"*
  -> please be more precise.

Response: Thank you for this fair comment. We have added in a couple of sentences in Section 3.2 (Page 15) on what this means in the context of functional data analysis.

- *The number of parameters is given in page 15, but not for other networks.*

Response: Thank you for the comment. We have revamped every example so that all relevant parameter count information is now contained within each section.

- *Table 2, use the same number of digits.*

Response: We have now adjusted all results so that they use the same number of significant digits.

- *Table 2, the different models should be properly introduced. "we did use water as a scalar covariate."*
  -> you did not talk about water before.

Response: Thank you for the feedback. We have now rewritten the introduction to that section so that it contains our reasoning for the use of water as a scalar covarirate. The gist of that reasoning has to do with testing functional neural networks under different conditions.

- *Section 4.1*
  -> l is not defined.
  -> lambda is not defined.

Response: We have defined ℓ and λ in Section 4.1.

- *Page 18, when talking about the g function, please refer to the paragraph where it is defined.*

Response: Thank you for the comment. To make this part more clear, we added sentences to describe the g function and linked it to the equation for each simulation.

- *A three-layer network with rectifier*
  -> not defined

Response: We actually did refer to the rectifer unit earlier in the text. However, we note that this is more hidden than ideal so we have made an additional reference to the original definition.

- *"we cross-validate over a grid to find the optimal λ parameter to smooth the resulting functional weight"*
  -> not clear. Not introduced before.

Response: We have introduced the smoothing parameter λ at the second paragraph of Page 20.

- *"We note that with a more rigorous tuning of the functional neural networks, we could further improve these results."*
  -> Do you suggest that you did not do it rigorously?

Response: Thank you for this valid question. We did not intend for the reader to think that we did not do it rigorously. This was a comment on how well the functional neural network performs without a very exhaustive tuning regimen – it was meant more to be a testament to the results (the FNN performed near the best). We also want to add that the large number of hyperparameters and the arbitrary choice of grids can make it so that tuning is a never ending process.

- *Section 4.2, why did you not apply the same models with real data?*

Response: Thank you for the question. The reasoning is that our initial goal was to compare the functional neural network with other *functional* models – not a general comparison with all methods. We were more concerned with observing how these models (including FNNs) handle functional observations. However, we have now added neural networks and convolutional neural networks into the real data examples.

- *"deep learning research has resulted in enormous breakthroughs in computer vision, classification, and scalar prediction"*
  -> what do you mean by "scalar prediction"? Deep learning has also been very successful in structured prediction problems.

Response: The scalar prediction means the prediction for scalar responses instead of the functional responses. A response is a scalar response if the value of response is just a single number, while a response is a functional response if the value of response is a continuous function defined on a continuum domain. We realize that the word "scalar" may have different meaning in different contexts. Therefore, we have remove "scalar" in the revised version.

- *Page 29, a reference is missing (see "?")*

Response: We have added the reference in the revised manuscript.

- *Table S2: remove the c().*

Response: Thank you for the comment. We have removed the c() as well as added the suggested ranges into the text and into a supplementary document.

- *"adam() optimizer"*
  -> Why are you using code here?

  Response: Thank you for this observation. This was an oversight and was fixed.

- *The link between the experiments and the previous sections is missing. Please refer to previous equations/sections in the experiments.*

  Response: Thank you. In the revised manuscript, we have made a more concerted effort to make sure that the relevant references have been made.

- *It seems that the better performance of your neural network is due to its nonlinear form. It is also not clear how the depth of your network contributes to better performance.*

  Response: You are correct that the better performance of our neural network is due to its nonlinear form. Increasing the depth is not necessarily leading to better performance, because it is much harder to train and the model may be easier to overfit. We have added these comments at the second paragraph of Page 26.

- *Table 1 and 2 do not include a conventional neural network. Table 3 does. Why? All of them should compare with a conventional neural network. You should also clearly state what kind of "conventional" neural network you used.*

  Response: Thank you. We have added neural networks as well as a convolutional neural networks to the tables. We added information on what the conventional neural network is but for additional clarity, it is a multilayer perceptron.

- *Standard errors should be added to the Tables. Tests for significant differences should be applied too.*

  Response: Thank you for the suggestion. The standard errors as well as paired t-tests have now been added. The paired t-tests are done by pairing the fold error rate of FNNs with the other models. The t-value, p-value, and 95% upper and lower bounds have been given in the supplementary document separately for each particular example.

- *Since Figure 3 shows beta(t), we suggest you define beta(t) in math mode (not inline as on page 11, line 1).*

Response: Thank you for the observation. We have made the recommended change.

References

Dalvi, F., Durrani, N., Sajjad, H., Belinkov, Y., Bau, A., and Glass, J. (2019), "What is one grain of sand in the desert? analyzing individual neurons in deep nlp models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 6309–6317.

Fan, F.-L., Xiong, J., Li, M., and Wang, G. (2021), "On interpretability of artificial neural networks: A survey," *IEEE Transactions on Radiation and Plasma Medical Sciences*.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013), "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*.

Tibshirani, R., Hastie, T., and Friedman, J. (2009), *The elements of statistical learning: data Mining, inference, and prediction*, New York: Springer.

Zhang, Q., Wu, Y. N., and Zhu, S.-C. (2018), "Interpretable convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8827–8836.

Zhang, Y., Tiňo, P., Leonardis, A., and Tang, K. (2020), "A Survey on Neural Network Interpretability," *arXiv preprint arXiv:2012.14261*.