A New PHO-rmula for Improved Performance of Semi-Structured Networks

David Rügamer [1 2]

Abstract

Recent advances to combine structured regression models and deep neural networks for better interpretability, more expressiveness, and statistically valid uncertainty quantification demonstrate the versatility of semi-structured neural networks (SSNs). We show that techniques to properly identify the contributions of the different model components in SSNs, however, lead to suboptimal network estimation, slower convergence, and degenerated or erroneous predictions. In order to solve these problems while preserving favorable model properties, we propose a non-invasive post-hoc orthogonalization (PHO) that guarantees identifiability of model components and provides better estimation and prediction quality. Our theoretical findings are supported by numerical experiments, a benchmark comparison as well as a real-world application to COVID-19 infections.

1. Introduction

A linear model in its original form is inherently interpretable due to its structural model space assumption: Given features $\boldsymbol{X}$, the expected outcome $\mathbb{E}(\boldsymbol{y}|\boldsymbol{X})$ of a variable of interest $\boldsymbol{y}$ is the linear combination $\boldsymbol{X}\boldsymbol{\beta}$ of the features $\boldsymbol{X}$ and weights $\boldsymbol{\beta}$. When fixing all but one of the features $\boldsymbol{x}_j$, the change in $\mathbb{E}(\boldsymbol{y}|\boldsymbol{X})$ can be easily quantified and interpreted by the change in $\boldsymbol{x}_j$ multiplied by its weight β_j (ceteris paribus). A deep neural network (DNN), on the other hand, can also be seen as a linear combination of features and weights, say $\boldsymbol{U}\boldsymbol{\gamma}$, where $\boldsymbol{U}$ are latent features learned in the penultimate layer of the neural network and $\boldsymbol{\gamma}$ are the weights from the connection between the last hidden and the output layer (in this case with linear activation function). As $\boldsymbol{U}$ itself is often a non-linear and complex transformation of the actual DNN inputs $\boldsymbol{Z}$, the linear combination $\boldsymbol{U}\boldsymbol{\gamma}$ is not "structured" in a way that users can directly relate the effects $\boldsymbol{\gamma}$ to the original inputs $\boldsymbol{Z}$. While there is a plethora of literature characterizing the learned DNN effects in terms of $\boldsymbol{Z}$, we in this work focus on models that combine a structured linear model-type predictor $\boldsymbol{\eta}^{str} := \boldsymbol{X}\boldsymbol{\beta}$ and an unstructured DNN predictor $\boldsymbol{\eta}^{unstr} := \boldsymbol{U}\boldsymbol{\gamma}$ to combine interpretability and predictive performance. The latent features $\boldsymbol{U}$ are learned from a set of features $\boldsymbol{Z}$ with arbitrary shape (e.g., image tensors) and potentially overlapping with $\boldsymbol{X}$. An example of such a model is the semi-structured neural network (SSN). SSNs, exemplary depicted in Figure 1, assume a late fusion of the structured and unstructured model by combining both model parts additively, i.e.,

$$\boldsymbol{\eta}^{str} + \boldsymbol{\eta}^{unstr} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{U}\boldsymbol{\gamma}. \tag{1}$$

This simple fusion has some attractive properties. Dorigatti et al. (2023), e.g., promote SSNs in the medical application context where researchers require some parts of the network to be a white box and derive statistically valid uncertainty quantification for $\boldsymbol{\eta}^{str}$. We will motivate our proposed approach in this paper using SSNs, but later results also generalize to more complex network structures.

Why make structured model parts explicit? As DNNs can be designed very flexibly and deep architectures often come with some form of universal approximation guarantee, DNNs can also capture linear effects. This raises the question as to why the structured part needs to be made explicit in models such as the SSN. The reason is the same as for the usage and success of residual connections (He et al., 2016): while DNNs can represent increasingly more complex and non-linear functions with greater depth, it also gets increasingly more difficult to learn the identity function of some input for these networks. By concatenating the outputs of (parts of) a DNN with the inputs of this DNN, the identity map can be made explicit and thereby allows the combined network to better learn linear effects of the inputs.

Identifiability In late semi-structured fusion models such as the SSN, the interpretability of the structured part can not be preserved without additional constraints (see, e.g., Baumann et al., 2021; Fritz et al., 2022; Rügamer et al., 2023). This is due to an identifiability problem. The DNN can also capture linear effects – to some extent – but it is unclear

[1]Department of Statistics, LMU Munich, Munich, Germany [2]Munich Center for Machine Learning (MCML), Munich, Germany. Correspondence to: David Rügamer <david@stat.uni-muenchen.de>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

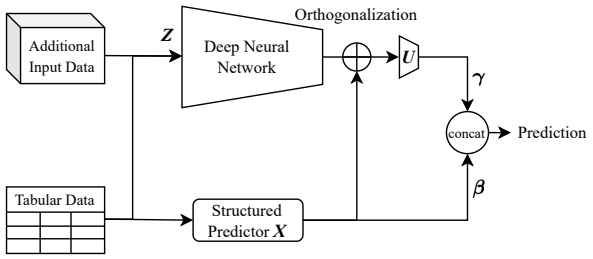

Figure 1. Exemplary SSN learning latent features $\boldsymbol{U}\boldsymbol{\gamma}$ from tabular and additional input data sources through a DNN and combining these with structured effects $\boldsymbol{X}\boldsymbol{\beta}$ based on tabular data $\boldsymbol{X}$. As $\boldsymbol{X}$ is contained in both network parts, an orthogonalization (top right) is required to identify the true linear effect $\boldsymbol{\beta}$.

to the analyst, how much additional linearity in addition to $\boldsymbol{\eta}^{str}$ has been learned by the DNN in $\boldsymbol{\eta}^{unstr}$. An obvious solution is to restrict the DNN in such a way that only the structured model part can learn structured effects and the unstructured deep network part captures the remaining variation while preserving the predictive performance of the DNN. While current literature takes this route and suggests that it is necessary to constrain the unstructured model part to ensure the interpretability of $\boldsymbol{\eta}^{str}$, restricting the network might potentially deteriorate model performance.

Our Contributions In this work, we provide new insights into approaches that combine structured and unstructured predictors and answer the following open questions:

- Why and when do constrained SSNs work?
- What are the limitations of semi-structured approaches with enforced identifiability?
- How can existing limitations be overcome?
- How can SSN approaches be generalized?

Our investigations allow advancing research in the direction of SSNs and interpretability in neural networks in general. Our proposed solutions to existing limitations are generic and applicable wherever a structured linear assumption is made, thereby also addressing use cases beyond SSNs. More specifically, we propose a "non-invasive" method to obtain valid interpretability post-model fitting and show both theoretically and in numerical experiments that this approach is superior to existing methods.

2. Related Literature

Methods that represent or learn structured model parts in a neural network are discussed in the literature in many different aspects. De Waal & Du Toit (2011), e.g., discuss how generalized additive models (GAMs) can be framed as a neural network, laying the foundation to combine GAMs with other network parts within one large neural network. In particular, this allows scaling GAMs to large amounts of observations or features (Rügamer et al., 2023, see, e.g.,). In contrast, neural additive models (NAMs; Agarwal et al., 2021) combine neural networks and additive models by learning the basis representations of additive effects using a neural network. Generalized linear models and neural network predictors have been combined by Tran et al. (2020) as well as Hubin et al. (2018), and used under the name *wide & deep learning* for product recommendation by Cheng et al. (2016). Pölsterl et al. (2020) and Kopper et al. (2021; 2022) use the idea of combining structured and unstructured predictors to learn a survival model. A combination of state space models and neural networks has, e.g., been proposed by Amoura et al. (2011). Another strain of literature combines transformation models and DNNs (Baumann et al., 2021; Sick et al., 2021) in a semi-structured manner to, e.g., facilitate the combination of structured predictors and complex DNNs for ordinal regression (Kook et al., 2022). Next to different model combinations, properties of semi-structured approaches such as the previously mentioned identifiability (Rügamer et al., 2023) and also uncertainty quantification of SSNs (Dorigatti et al., 2023) have been discussed.

We will focus on the aspect of identifiability in this paper and point out limitations in existing approaches. Our suggested solutions guarantee correct interpretability, are applicable to most of the methods and models discussed in the literature, and are efficient to compute without modifying the SSN.

3. Identifiability and Limitations

Rügamer et al. (2023) discuss the identifiability problem in the setup of a probabilistic density regression network that learns $K \geq 1$ distribution parameters $\theta_k, k = 1, \ldots, K$ using the Maximum Likelihood approach. In their framework, each of the distribution parameters θ_k is linked to an additive predictor

$$\boldsymbol{\eta}_k = \boldsymbol{X}_k\boldsymbol{\beta}_k + \boldsymbol{U}_k\boldsymbol{\gamma}_k.$$

For the purpose of this paper, we simplify this approach and focus on the special case of mean regression.

Notation We assume n observations $\boldsymbol{y} \in \mathbb{R}^n$ drawn (conditionally) independent from a distribution $\mathcal{D}_{Y|X,Z}$ based on a fixed set of features $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ and a set of features $\boldsymbol{Z}$ with arbitrary shape (e.g., image tensors) that are potentially overlapping with $\boldsymbol{X}$. As model for the data, we consider an SSN. In the SSN, features $\boldsymbol{X}$ are modeled as a structured linear effect with weights $\boldsymbol{\beta} \in \mathbb{R}^p$. For the unstructured model part, a DNN with $\boldsymbol{Z}$ as input learns latent features $\boldsymbol{U} \in \mathbb{R}^{n\times q}$. These latent features are weighted in a final layer with weights $\boldsymbol{\gamma} \in \mathbb{R}^q$ and then added to the structured

model part (cf. Fig. 1). Put together, the SSN learns

$$\mathbb{E}(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{Z}) = \boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{U}\boldsymbol{\gamma} \tag{2}$$

and we optimize the network using the mean squared error $\ell(\boldsymbol{\eta}) = \frac{1}{2n}||\boldsymbol{y} - \boldsymbol{\eta}||_2^2$. For better readability, we focus on mean regression and not a general distributional regression as in Rügamer et al. (2023), but note that results presented in the following also hold for the distributional case.

Identifiability Based on this (simplified) regression approach defined in (2), we adapt the definition of *semi-structured identifiability* as follows:

Definition 3.1. Semi-structured identifiability We say that a semi-structured regression model is identified in its structured model parts if there exists no $\boldsymbol{\xi} \in \mathbb{R}^n \backslash \{\mathbf{0}\}$ such that

$$\begin{aligned}\boldsymbol{\eta} &= \boldsymbol{\eta}^{str} + \boldsymbol{\eta}^{unstr} = (\boldsymbol{\eta}^{str} - \boldsymbol{\xi}) + (\boldsymbol{\eta}^{unstr} + \boldsymbol{\xi}) \\ &= \breve{\boldsymbol{\eta}}^{str} + \breve{\boldsymbol{\eta}}^{unstr} = \breve{\boldsymbol{\eta}},\end{aligned} \tag{3}$$

where $\ell(\breve{\boldsymbol{\eta}}) = \ell(\boldsymbol{\eta})$, i.e., both parameterizations perform equally well and yield the same loss.

Simply put, the above definition says that the structured predictor is identified, if we cannot find another truly different structured predictor that yields the same loss as the original predictor. In particular, the definition only relates to the structured part as $\boldsymbol{\eta}^{unstr}$, a DNN, is in most cases never uniquely identified. The solution to guarantee identifiability in Rügamer et al. (2023) proposes an orthogonalization cell that projects the unstructured (DNN) predictor $\boldsymbol{U}$ into the orthogonal complement space $\mathcal{X}^\perp$ spanned by the columns space of $\boldsymbol{X}$, thereby ensuring identifiability and thus, in turn, meaningful interpretability of the structured effects $\boldsymbol{\beta}$. More specifically, if the model's predictor is as in (2), the adapted SSN with orthogonalization computes the mean via

$$\mathbb{E}(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{Z}) = \boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta} + \mathcal{P}_X^\perp \boldsymbol{U}\boldsymbol{\gamma}, \tag{4}$$

where $\mathcal{P}_X^\perp = (\boldsymbol{I}_{n\times n} - \mathcal{P}_X) \in \mathbb{R}^{n\times n}$ is the orthogonal projection matrix defined as the difference between the identity matrix $\boldsymbol{I}_{n\times n}$ and the projection matrix $\mathcal{P}_X$ onto the space $\mathcal{X}$ spanned by $\boldsymbol{X}$. We will call this procedure "online orthogonalization" (ONO) as the respective constraints and orthogonalization operations are applied during training.

In the following, we will formalize some of the limitations of the ONO approach and then propose solutions addressing these restrictions.

3.1. Limitations

Rügamer et al. (2023) proved that ONO yields identifiability of the structured model part and thereby fulfilling the purpose of preserving its interpretability. Before addressing some downsides induced by this approach, we first show another desirable property of their orthogonalization approach not mentioned in their work.

Lemma 3.2. ***ONO Hypothesis space*** *Let $n > p$. Then the hypothesis space $\mathcal{H}_{ONO}$ of an SSN with orthogonalization is equivalent to the hypothesis space $\mathcal{H}$ of the same SSN without the orthogonalization constraint.*

The proof of Lemma 3.2 can be found in the Appendix. Contrary to the intuition that the orthogonalization directly entails a constraint in the model space, the previous result shows that an SSN with ONO is *a priori* as expressive as the same model without the constraint. ONO, however, induces a different optimization problem, resulting in slower convergence in practice and a restricted hypothesis space if the $n > p$-assumption of Lemma 3.2 is not met. We formalize this in the following:

Limitation 3.3 (Rügamer et al. (2023), Remark 2.5)**.** *In the case where $p > n$, the orthogonalized unstructured effect is equal to* $\mathbf{0}$.

This is due to the fact that $\mathcal{P}_X^\perp = \mathbf{0}_{n\times n}$ for $p > n$. Using the same argument, we have the following limitation for mini-batch training:

Limitation 3.4. *In the case where $p < n$ but the batch size $b < p$, the theoretical unstructured effect is different from* $\mathbf{0}$ *but is practically estimated as a null-effect due to the mini-batch training.*

The proof is a simple result of the fact that $\mathcal{P}_X^\perp$ is a $b \times b$ matrix formed for every batch in mini-batch training and by construction, $\mathcal{P}_X^\perp = \mathbf{0}_{b\times b}$ if $b < p$. This finding also relates to the network's prediction as stated in the following:

Limitation 3.5. *Due to the orthogonalization, the prediction of an orthogonalized network depends on the batch size used for prediction.*

This is due to the fact that the network's prediction is formed through the matrix $\mathcal{P}_X^\perp \in \mathbb{R}^{b\times b}$, implying that every of the b observations during prediction influences the $b-1$ other observations (whereas in many other networks, the predictions are independent of each other for fixed network weights). As a consequence, an SSN network with ONO will have an additional source of prediction error σ_E^2, irrespective of its optimization quality when the batch size b at test time is small. Let $\boldsymbol{X}^*$ and $\boldsymbol{Z}^*$ be the test data sets fed into the structured and unstructured model part, respectively, and let $\hat{\boldsymbol{U}}^* = \hat{\boldsymbol{U}}(\boldsymbol{Z}^*)$ be the penultimate layer's features on this test set. We can quantify the behavior of this error w.r.t. b exemplarily using $p = q = 1$ as follows:

Lemma 3.6 (**Error rate ONO prediction**)**.** *Assume that the latent feature $\hat{\boldsymbol{U}}^* = (\hat{u}_1^*, \ldots, \hat{u}_b^*)^\top$ reside in $\mathcal{X}^\perp$, and let features u_i^* be independent realizations of some distribution with mean zero and variance σ_U^2. Further assume*

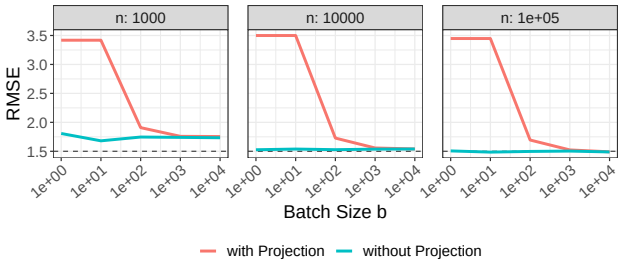

Figure 2. Root mean squared error (RMSE) of predictions for different training set sizes (facets) and different batch sizes (x-axis) when activating and deactivating the orthogonal projection. Results show that deactivation always leads to an optimal error which is also achieved in the limit as $b \to \infty$. For increasing n both approaches further converge to the Bayes error (dashed black line).

that features $\boldsymbol{X}^* = (x_1^*, \dots, x_b^*)^\top$ are realizations of some other distribution with mean zero and variance σ_X^2. Then σ_E^2 is $\mathcal{O}(1/b)$.

A corresponding proof can be found in the Appendix. This result is also confirmed in our simulations, showing that the additional prediction error induced by the orthogonalization decreases at a linear rate. While this is usually considered a good rate of convergence, note that this rate is w.r.t. the batch size b, which does not grow in practice but is usually fixed to a moderately small number (e.g., $b = 32$). In other words, the linear decrease in the additional prediction error has only limited benefits and might even be irrelevant in practice if the test set is small and hence with no option to increase b. Fortunately, this problem can be easily solved. Define the estimated network weights for the structured and unstructured model part as $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\gamma}}$. Then the mean prediction on the test data is

$$\boldsymbol{\eta}^* = \boldsymbol{X}^*\hat{\boldsymbol{\beta}} + \mathcal{P}_{X^*}^{\perp}\hat{\boldsymbol{U}}^*\hat{\boldsymbol{\gamma}} = \boldsymbol{X}^*\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{U}}^*\hat{\boldsymbol{\gamma}} - \mathcal{P}_{X^*}\hat{\boldsymbol{U}}^*\hat{\boldsymbol{\gamma}},$$

where the last term $\mathcal{P}_{X^*}\hat{\boldsymbol{U}}^*\hat{\boldsymbol{\gamma}} \to \mathbf{0}$ for $b \to \infty$ as the SSN has been trained to learn features $\hat{\boldsymbol{U}}$ with entries (rows) residing in $\mathcal{X}^\perp$. Thus, predictions of an SSN with ONO in expectation (and in the limit) are given by $\boldsymbol{X}^*\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{U}}^*\hat{\boldsymbol{\gamma}}$. In other words, if the ONO has successfully learned to generate features $\hat{\boldsymbol{U}}$ orthogonal to the space $\mathcal{X}$, the orthogonalization cell can be deactivated at test time (also confirmed in practice as shown in Figure 2).

While the previous result implies that inferior prediction performance due to test set size dependency can be solved straightforwardly, it does not solve the other issues of ONO.

4. Post-Hoc Orthogonalization (PHO)

To overcome the remaining previously mentioned problems, we propose the procedure described in Algorithm 1 and dubbed PHO (Post-Hoc Orthogonalization).

Algorithm 1 PHO

Input: Data set $\mathcal{D}$; unconstrained SSN
1. Train SSN on $\mathcal{D}$
2. Replace $\hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}}$ with $\mathcal{P}_X^{\perp}\hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}}$ in the last layer
3. Set $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}} + \boldsymbol{X}^\dagger\hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}}$ with $\boldsymbol{X}^\dagger$ being the Moore-Penrose pseudoinverse of $\boldsymbol{X}$

Output: Return adapted SSN

The underlying idea of PHO is that all constraints proposed in Rügamer et al. (2023) only relate to the identifiability of $\boldsymbol{\beta}$, but not to the identifiability of the additive predictors $\boldsymbol{\eta}$. In other words, the unconstrained model is able to find the same $\boldsymbol{\eta}$ as the constrained model, implying that the optimal amount of structured effects needs to be contained in $\boldsymbol{\eta}$ in some form. Otherwise, the optimization routine could just change $\hat{\boldsymbol{\beta}}$ and thereby improve the predictions $\hat{\boldsymbol{\eta}}$. Moreover, the unconstrained model is not only theoretically able to learn the same effects as the one with ONO but will have more degrees of freedom in the optimization in practice. It is therefore not unlikely that it will find a similar or even better solution compared to the constrained model. The adaptions of fitted effects of the unconstrained SSN in steps 2 and 3 of Algorithm 1 can then be justified as follows. Given the final (not meaningfully interpretable) model output $\hat{\boldsymbol{\eta}}$ we have

$$\begin{aligned} \hat{\boldsymbol{\eta}} &= \boldsymbol{X}\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}} = \boldsymbol{X}\widetilde{\boldsymbol{\beta}} + \widetilde{\boldsymbol{U}}\hat{\boldsymbol{\gamma}}, \\ \text{with } \widetilde{\boldsymbol{\beta}} &= \hat{\boldsymbol{\beta}} + \boldsymbol{X}^\dagger\hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}} \;\text{ and }\; \widetilde{\boldsymbol{U}} = \mathcal{P}_X^{\perp}\hat{\boldsymbol{U}}, \end{aligned} \tag{5}$$

where $\boldsymbol{X}^\dagger$ is the Moore-Penrose pseudoinverse of $\boldsymbol{X}$. The reformulation stems from the fact that we can split any predictor into

$$\begin{aligned} \hat{\boldsymbol{\eta}}_k &= \boldsymbol{X}\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}} \\ &= \boldsymbol{X}\hat{\boldsymbol{\beta}} + (\mathcal{P}_X\hat{\boldsymbol{U}} + \mathcal{P}_X^{\perp}\hat{\boldsymbol{U}})\hat{\boldsymbol{\gamma}} \\ &= \boldsymbol{X}\hat{\boldsymbol{\beta}} + \boldsymbol{X}\boldsymbol{X}^\dagger\hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}} + \mathcal{P}_X^{\perp}\hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}} \\ &= \boldsymbol{X}(\hat{\boldsymbol{\beta}} + \boldsymbol{X}^\dagger\hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}}) + \mathcal{P}_X^{\perp}\hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}} \\ &=: \boldsymbol{X}\widetilde{\boldsymbol{\beta}} + \widetilde{\boldsymbol{U}}\hat{\boldsymbol{\gamma}}, \end{aligned} \tag{6}$$

where $\mathcal{P}_X$ is the projection matrix into the column space spanned by $\boldsymbol{X}$ and $\mathcal{P}_X^{\perp}$ its orthogonal complement. In other words, $\widetilde{\boldsymbol{\beta}}$ is defined such that all linear effects of $\boldsymbol{X}$ found in $\hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}}$ are moved to the structured model part. From (5) it follows that the adjusted structured effects are $\widetilde{\boldsymbol{\beta}}$ and the unstructured predictions are $\widetilde{\boldsymbol{U}}\hat{\boldsymbol{\gamma}}$. Notably, as also apparent from (5), the prediction of the model is not affected by this reformulation. Similarly, other constraints required to identify a model can be applied post-hoc (e.g., mean-centering to identify the global bias term of the model).

4.1. Properties, Efficient Computation and Prediction

Why does using PHO solve the aforementioned problems? First, training the SSN unconstrained removes the

constrained-induced difficulties in optimizing the SSN. Second, while Limitation 3.3 still exists also for PHO, we now have the guarantee that this does not deteriorate the prediction performance (as (5) states that predictions are unchanged by PHO). Moreover, this finding also solves Limitation 3.5 as no orthogonalization is required for predictions (and, in contrast to ONO, not part of the network). As PHO is applied on the final model output and does not require any form of mini-batch updates, computations in (5) can be applied on the whole data set and thereby solves Limitation 3.4.

A possible downside of PHO is that $\mathcal{P}_X^{\perp} \in \mathbb{R}^{n \times n}$ is potentially very large and its computation or storage is not possible. An alternative and computationally more efficient algorithm to compute PHO is the following Algorithm 2 using the partitioning of the data set $\mathcal{D}$ into batches $\mathcal{B}_1, \ldots, \mathcal{B}_M$ with sizes $b_1, \ldots, b_M$. This algorithm is an **exact** procedure and is inspired by distributed computations of linear models (e.g., Karr et al., 2005), i.e., no information is lost by switching to a mini-batch routine in this case, allowing us to scale our method arbitrarily. A similar algorithm can be used to calculate the different contributions of structured and unstructured network parts on a new data set $\mathcal{D}^*$ (see Appendix B.1) when computing predictions. The total storage required for Algorithm 2 is $\mathcal{O}(p^2 + n)$. As p is typically relatively small in semi-structured model applications and therefore negligible, the relevant memory requirement is to store a vector of size n (i.e., the same size as the outcome vector $\boldsymbol{y}$). The time complexity of Algorithm 2 is $\mathcal{O}(c + n \cdot p^2 + p^3)$ where c is the total cost of a single forward pass of the SSN using all M mini-batches. Again, if p is small and since c is usually a multiple of n, the time complexity is practically not different from the one of a single forward pass (in batches) over the whole data set.

Algorithm 2 Mini-Batch PHO

Input: Data set $\mathcal{D}$; unconstrained SSN; batches $\mathcal{B}_1, \ldots, \mathcal{B}_M$
1. Train SSN on $\mathcal{D}$
2. Initialize $\boldsymbol{H} = \boldsymbol{0}_{p \times p}$; $\boldsymbol{s} = \boldsymbol{0}_{p \times 1}$ and compute summary statistics as follows:
for $m = 1, \ldots, M$ **do**
  a) Compute $\hat{\boldsymbol{U}}_{\mathcal{B}_m}$ ($\hat{\boldsymbol{U}}$ for batch $\mathcal{B}_m$)
  b) Compute and store $\hat{\boldsymbol{\zeta}}_m := \hat{\boldsymbol{U}}_{\mathcal{B}_m} \hat{\boldsymbol{\gamma}} \in \mathbb{R}^{b_m \times 1}$
  c) Compute $\boldsymbol{H} = \boldsymbol{H} + (\boldsymbol{X}_{\mathcal{B}_m}^\top \boldsymbol{X}_{\mathcal{B}_m})$
  d) Compute $\boldsymbol{s} = \boldsymbol{s} + \boldsymbol{X}_{\mathcal{B}_m}^\top \hat{\boldsymbol{\zeta}}_m$
end for
3. Compute $\boldsymbol{\alpha} := \boldsymbol{H}^{-1} \boldsymbol{s}$
4. Set $\widetilde{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}} + \boldsymbol{\alpha}$
5. Compute $\boldsymbol{\eta}^{unstr} = \texttt{stack}((\hat{\boldsymbol{\zeta}}_m - \boldsymbol{X}_{\mathcal{B}_m} \boldsymbol{\alpha})_{m=1,\ldots,M})$
Output: $\widetilde{\boldsymbol{\beta}}$, $\boldsymbol{\eta}^{unstr}$

Note that it is straightforward to extend the presented algorithm to K additive predictors $\boldsymbol{\eta}_k, k = 1, \ldots, K$ as discussed in Rügamer et al. (2023) by simply iterating over k and applying Algorithm 2 to all K predictors.

4.2. Semi-Structured Importance Measures

As the last line of (5) creates two functional parts which are orthogonal to each other, the model decomposition follows the axioms proposed in the functional ANOVA approach (Hooker, 2004). We can thus derive importance measures post-hoc, relating to the variance explained by the different model parts (structured/unstructured) in the SSN. Various definitions are given in Appendix C.

4.3. Relationship to other Methods

The proposed approach opens up several links to other methods in the literature on model interpretability, which we want to briefly discuss in the following.

Post-hoc Explanation Methods As mentioned in the previous subsection, the PHO approach for SSNs is a functional ANOVA-type decomposition as introduced in Hooker (2004). Similar techniques have been applied to other machine learning models and have also been recently discussed for interaction effects and tree-based models (Lengerich et al., 2020). The SSN approach already specifies the hypothesis space using structural assumptions (for reasons discussed in the introduction) while other existing techniques assume no structure a priori and only impose the structure post-model fitting.

Feature Adjustment (Trend Adjustment) In statistical literature, feature (or trend) adjustment in linear models (see, e.g., Robinson & Jewell, 1991) follows a similar principle as our approach. By regressing the outcome on a feature or trend variable $\boldsymbol{\chi}$ that needs to be adjusted for, one can remove the influence of this variable before regressing the resulting linear model residuals (the remaining information in the outcome left after adjusting for χ) on the actual set of features $\boldsymbol{X}$. In the special case of a linear model where the mean is parametrized by a semi-structured predictor, i.e., $\boldsymbol{\eta} = \boldsymbol{\eta}^{str} + \boldsymbol{\eta}^{unstr}$, our approach coincides with the feature adjustment approach by setting $\boldsymbol{\chi} = \hat{\boldsymbol{\eta}}^{unstr}$. For general parametric regression models, our approach can be seen as repeatedly applying the first step of feature detrending to (every of the K) additive predictor(s) separately.

4.4. PHO with Splines

Various approaches listed in Section 2 use an SSN but define the structured model part $\boldsymbol{X}$ using a spline basis representation. This allows modeling structured non-linearities in $\boldsymbol{\eta}^{str}$ and thereby defining a combination of a DNN and a generalized additive model. In many of these applications,

the structured part defined by the spline basis representation is, however, estimated using a penalized smoothing approach (i.e., using a quadratic difference penalty matrix for spline coefficients in $\boldsymbol{\beta}$). This leads to another limitation of ONO and, in this case, also of our PHO approach.

Limitation 4.1 (Orthogonalization in penalized SSNs). *If the structured model part $\boldsymbol{\eta}^{str}$ in an SSN is estimated using a quadratic penalty matrix, the orthogonalization via $\mathcal{P}_X^{\perp}$ results in an "over-orthogonalization" by removing more contribution than necessary from $\boldsymbol{\eta}^{unstr}$.*

We can overcome this restriction by accounting for the penalization when projecting the unstructured model part onto the space $\mathcal{X}$ as done in (5). Limitation 4.1 can also be seen as some form of overfitting. When applying $\boldsymbol{X}^{\dagger}\hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}}$ in (5) without regularization, we are more likely to learn spline coefficients $\boldsymbol{\beta}$ that interpolate the data. Instead, given $\boldsymbol{X}$, the quadratic spline penalty $\boldsymbol{K}$ as well as the amount of penalization $\lambda \geq 0$, we can perform the corrected update

$$\widetilde{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}} + (\boldsymbol{X}^{\top}\boldsymbol{X} + \lambda\boldsymbol{K})^{-1}\boldsymbol{X}^{\top}\hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}}. \tag{7}$$

While some of the related literature use a pre-defined smoothness λ and the previous equation then directly yields the updated structured parameters, the projection in (7) can also be implemented by running a generalized additive model (GAM) routine with inputs $\boldsymbol{X}$, penalty matrix $\boldsymbol{K}$ and response $\hat{\boldsymbol{\eta}}^{unstr}$ to estimate λ in a data-driven fashion. We call this approach PHOGAM in our experiments.

4.5. Generalization Beyond Semi-structured Networks

Lastly, we address one final limitation of ONO and present a generalization of the orthogonalization approach to a much larger class of networks beyond SSNs.

Limitation 4.2 (Model-class restriction of ONO). *As ONO iteratively performs a projection into the space $\mathcal{X}^{\perp}$, $\mathcal{P}_X^{\perp}$ must be a non-random matrix and thus known prior to model training.*

The reason for $\mathcal{P}_X^{\perp}$ to be known a priori is that it would otherwise not be possible to calculate gradients for this projection. While this is not a limitation for the class of SSNs as the structured model part is always defined a priori by the modeler, Limitation 4.2 implies that the separation into structured and unstructured components can only be done if $\boldsymbol{X}$ is fixed. This, e.g., excludes cases such as neural additive models (NAMs; Agarwal et al., 2021), where additive components are learned features themselves and thus not known a priori. In contrast to ONO, our PHO approach only applies the orthogonalization post-hoc and therefore allows its application to additive combinations with learnable inputs which only need to be fixed after training. This effectively generalizes the approach presented in Rügamer et al. (2023) to a much larger model class and allows the investigation of (non-)linearities in DNNs with learnable structured parts.

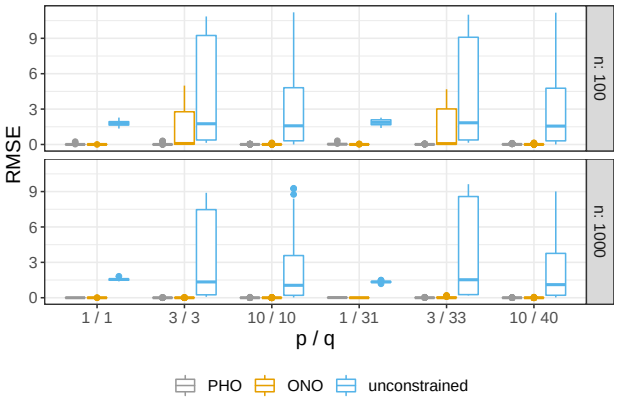

Figure 3. Linear coefficient estimation performance of different methods (colors) for different numbers of structured and unstructured features (p / q, x-axis) and data sizes n (rows).

5. Numerical Experiments

In the following, we investigate all derived properties of ONO as well as the performance of PHO using simulated and real-world data. Further experiments and details such as optimization, computing environment, and hyperparameters can be found in Appendices D and E. All code is made available on Github.

5.1. Convergence of ONO

We first empirically investigate the convergence speed of ONO compared to PHO. As the convergence speed is influenced by the projection $\mathcal{P}_X$, we investigate different settings of structured features $p \in \{1, 3, 10\}$ as well as different numbers of observations $n \in \{100, 1000\}$. We use a DNN with 2 hidden layers with 100 and 50 units and ReLU activation each followed by a dropout layer with a dropout probability of 20%. A final layer learning the effects $\boldsymbol{\gamma}$ is defined by a linearly activated fully-connected layer. Results (see Figure 8 in the Appendix) confirm our hypothesis, showing that additional iterations required by ONO compared to PHO increase linearly with the number of features p, implying that the projection can slow down the training of an SSN notably. While this issue seems to be mitigated with more observations, ONO still requires a multiple of iterations of PHO for $p = 10$.

5.2. Semi-Structured Models with Linear Effects

In our next numerical experiment, we investigate the estimation quality of structured effects $\boldsymbol{\beta}$ using ONO, PHO, and an unconstrained model (Figure 3). We adopt the setup of the previous analysis to investigate different numbers of features p and q as well as the number of observations n. The deep neural network and optimization routine is defined as

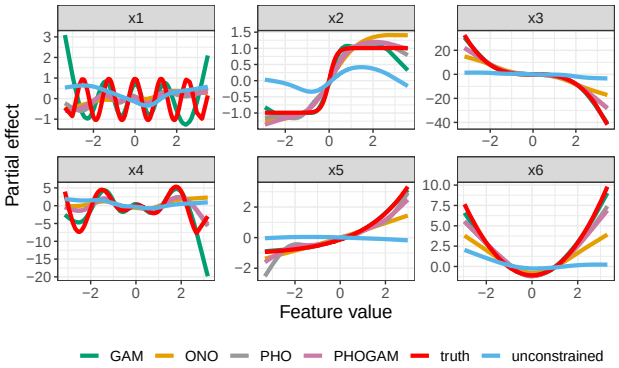

Figure 4. Visual comparison of different methods (colors) for six of the simulated non-linear functions (different facets; truth in red) visualized by the average curve over 20 repetitions (lines) for $n = 1000$ and exact overlap between $\boldsymbol{X}$ and $\boldsymbol{Z}$.

in Subsection 5.1. The estimation quality of the structured part is measured using the root mean squared error (RMSE) between the estimated and true model weights ($\boldsymbol{\beta}$) which are defined as equispaced values between -2.5 and 2.5. The results are depicted in Figure 3 showing that the unconstrained model is not able to find the right coefficients (the RMSE in most cases is larger than 1.5, which is the average absolute value of $\boldsymbol{\beta}$). In contrast, both ONO and PHO perform well in general, but the optimization of ONO sometimes does not work perfectly, showing larger RMSEs in some cases.

5.3. Semi-Structured Models with Non-Linear Effects

We now investigate SSNs with non-linear structured effects based on a spline representation. We adopt the setup of the two previous subsections but define a non-linear structured relationship in the true data-generating process (i.e., the data-generating process is an additive model). The $p \in \{1, 3, 10\}$ non-linear functions along with further details are defined in Appendix D. Six of the non-linear functions are visualized in Figure 4 together with estimated splines using ONO, PHO, PHOGAM, an unconstrained model and the gold-standard method for GAMs (Wood, 2017). As Figure 4 suggests, the unconstrained model has difficulties properly estimating the non-linear effects while all other methods work similarly well. Quantitative results (Fig. 9 in Appendix E.2) show that PHOGAM indeed improves over PHO in some cases, suggesting that accounting for the model's penalization is beneficial. PHO, in turn, performs again better than ONO in most cases. All neural approaches exhibit a shrinkage effect towards zero, most likely caused by the implicit regularization of the first-order optimization routine.

5.4. Prediction Error of ONO

In a final simulation study, we investigate the additional prediction error of ONO. For this, we create an SSN with ONO and simulate random features $\boldsymbol{X} \in \mathbb{R}^{n \times 10}$ and $\boldsymbol{Z} \in \mathbb{R}^{n \times 20}$ from a standard random normal distribution, set $\boldsymbol{\beta} = \mathbf{0}$ and define the true non-linear data generating model as $\boldsymbol{y} = \sin(\boldsymbol{Z}_{[:,1]}) + \boldsymbol{Z}^2_{[:,2]} + \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. The latent features $\boldsymbol{U}$ are generated using a single hidden layer with 10 units, no bias, and ReLU activation. A final dense layer with one unit learns the last layer's weights $\boldsymbol{\gamma}$ using a linear activation. We vary $n \in \{1e4, 1e5, 1e6\}$ and train the network with an 80/20-split for training and validation data with early stopping the validation data with a patience of 50 iterations. We then evaluate the network on a separate test data set of size n which we predict with a batch size $b \in \{1e0, 1e1, 1e2, 1e3, 1e4\}$. Finally, we evaluate the prediction performance using the RMSE both with activated and deactivated projection. We repeat this simulation 4 times and average the prediction errors across runs. The visual result from this simulation study was given in Section 3 in Figure 2, confirming our theoretical understanding. We use the same data to also investigate the behavior of the additional prediction error σ^2_E in terms of the batch size b and find that numerically the rate is very close to $1/b$.

5.5. Benchmark

Table 1. Prediction performances (average MSE and its standard deviation in brackets) for different methods (rows) and data sets (columns) based on 10 train-test splits. The best method per data set is highlighted in bold.

	Airfoil	Concrete	Diabetes	Energy	ForestF	Yacht
GAM	119 (1.9)	7.1 (0.53)	140 (9.6)	3.4 (0.36)	1.5 (0.16)	3.0 (0.43)
DNN (large)	22 (1.6)	5.0 (0.66)	63 (7.7)	**3.2** (0.39)	**1.4** (0.10)	2.0 (0.44)
DNN (small)	23 (1.4)	7.0 (1.40)	58 (7.1)	3.3 (0.31)	1.4 (0.11)	2.8 (0.54)
ONO (large)	113 (1.6)	7.2 (0.55)	120 (9.4)	3.4 (0.36)	1.5 (0.12)	2.9 (0.43)
ONO (small)	115 (1.6)	7.2 (0.56)	130 (9.5)	3.4 (0.36)	1.5 (0.15)	2.9 (0.43)
PHO (large)	14 (1.5)	**4.9** (0.80)	57 (7.8)	3.3 (0.36)	1.5 (0.16)	**1.9** (0.49)
PHO (small)	**6** (0.4)	6.3 (0.63)	**57** (7.2)	3.3 (0.37)	1.5 (0.13)	2.3 (0.56)

The previous sections showed that PHO and ONO yield the same structured effects while the first does not need any network adaptions before or during model training. We now compare both approaches in terms of prediction performance. As our approach is less restrictive during training and implies an easier optimization problem, we hypothesize that this will also result in better prediction performance. We, therefore, compare the two approaches (constrained vs. unconstrained) on different benchmark data sets from the UCI repository (Dua & Graff, 2017) using 10 different train-test splits. We report average MSE values and their standard deviation. In addition to the previously analyzed SSN models, we run the two obvious baselines, a model without an unstructured (neural network) predictor (i.e., a GAM) and a model without a structured (additive model) predictor (i.e., a DNN). For the models that include a DNN

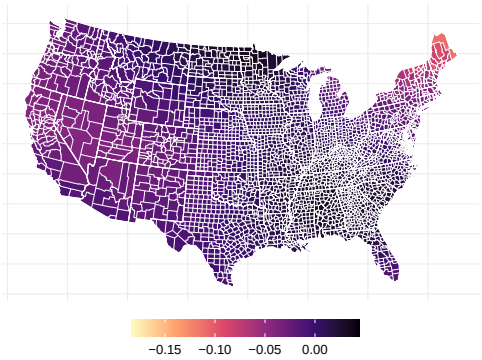

Figure 5. Spatial effect on the logarithmic prevalence of infection in the US estimated by PHOGAM (lower means infections are less likely).

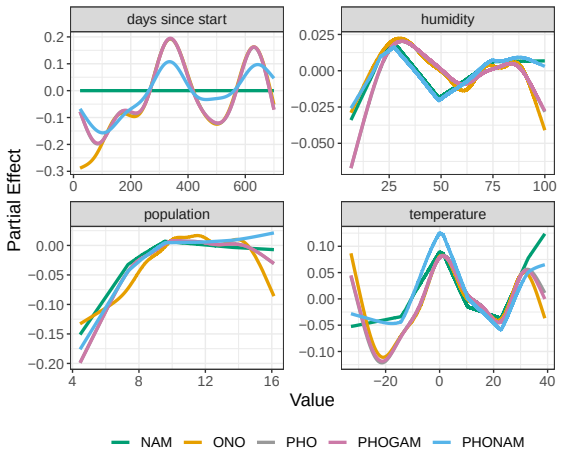

Figure 6. Estimated non-linear effects (mean-centered) for different features (facets) using different methods (colors).

component, we choose between four pre-defined multi-layer perceptron architectures (see Appendix D for more details).

Table 1 summarizes the results when choosing the best-performing hyperparameter set per method and data set. Further details on DNN architectures, hyperparameters, the full list of results, and details on benchmark data sets can be found in Section D in the Appendix. Results indicate that PHO works notably better in prediction compared to ONO and is almost on par with the DNN in cases where the deep unstructured model yields the best performance. In particular, for Airfoil, ONO (due to its limited flexibility) is effectively estimated as a GAM with almost zero DNN part, hence the similar performance to the GAM. Both DNN and PHO perform similarly well but the additional inductive bias through the structural assumption of the SSN (PHO) allows further improvement over the DNN.

6. Real-World Applications

Finally, we demonstrate our approach by applying it to two large real-world data sets.

6.1. Covid-19 Analysis

We first analyze a recently published data set on global-scale spatially granular meta-data for coronavirus disease modeling (Wahltinez et al., 2022). We subset the data to complete observations from the United States (US) and calculate the prevalence of infection per 1,000 people in the provided administration regions. We then model the logarithmic prevalence using an SSN including the average temperature, the relative humidity, the time (days since the start of the first outbreak), longitude, and latitude as well as the logarithmic population size as predictors. All features except the longitude and latitude are modeled as univariate splines whereas the geo-coordinates are modeled using a bivariate tensor-product spline to account for spatial interaction. We fit SSNs with ONO, PHO, and PHOGAM and compare them with a semi-structured neural additive model (NAM; Agarwal et al., 2021). In contrast to the SSN approaches, the NAM learns each of the univariate non-linear effects as well as the bivariate non-linear effect using a feature-specific DNN. Details on network architectures and optimization routines can be found in Appendix F. As PHO(GAM) can also work with learnable features, we apply PHO also to the NAM (which we call PHONAM in the following) to demonstrate the generality of our approach discussed in Section 4.5.

Results All models provide well-interpretable structured effects (see, e.g., Figure 5 for a spatial effect visualization of PHOGAM). When comparing the different approaches, we find similar patterns across models for all SSN approaches (Figure 6, and Figure 10 in Appendix F). In contrast to the pre-defined basis representation approaches (ONO, PHO, PHOGAM), the learned non-linear effects of NAM are less smooth as clearly visible in Figure 6 or show no trend (likely in cases when the optimization of the respective feature-specific network gets stuck in a local minimum). PHONAM is able to correct for missing smoothness, but can only learn as much additional feature information as present in NAM's deep predictor. Further effect comparisons can be found in Appendix F.

6.2. Flight Delay Data

To demonstrate the scalability of our approach, we further analyze the flight delay data set (Harvard Dataverse, 2008). The data set contains detailed information about commercial flights within the US from October 1987 to April 2008, comprising approximately 120 million records. Data includes flight arrival and departure specifics, such as times, carrier codes, flight numbers, delays, cancellation details, and more.

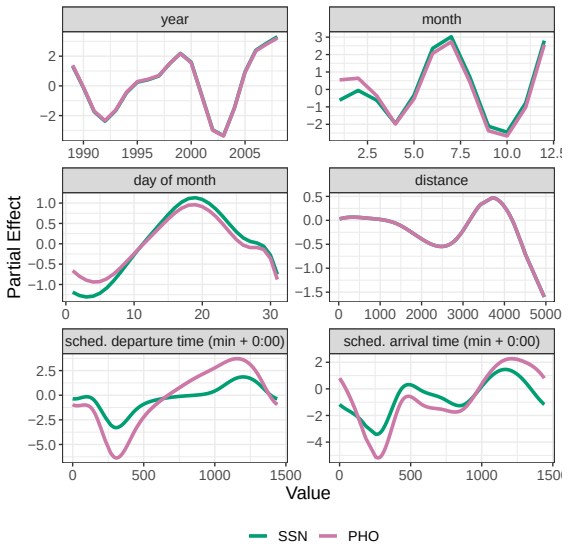

Figure 7. Partial effect (average arrival delay in minutes) of splines for all numerical features from both SSN (without adjusting for identifiability) and PHO.

After filtering the data for non-canceled and non-diverted flights as well as routes and carriers that are present for all 22 years, we analyze the data with an SSN by modeling the year, the month, the day of the month, and the distance using splines, the scheduled departure and arrival time as cyclic splines (to enforce continuity for the transition between 11:59 pm and 0:00 am), as well as the origin, destination, their interaction, the day of the week and the carrier as factor effects (i.e., dummy-encoded binary features). The deep neural network also processing this information is defined by a 3-hidden layer ReLU network with 100 neurons each followed by 1 output neuron.

Results Upon inspecting the fitted model, it is apparent that the interaction effects modeled by the DNN do not substantially influence the prediction. This finding is supported by Figure 7, which only displays minor adjustments after running PHO for the majority of variables. The exceptions are the scheduled arrival and departure times, which show more significant variation and larger effects when we extract their unidentified univariate information from the DNN component of the SSN. Overall, the estimated effects display a plausible pattern. For instance, flights scheduled early in the morning tend to be less delayed on average (indicating a negative partial effect). Conversely, flights during the summer and winter months typically experience more delays on average.

7. Conclusion

Semi-structured networks can be used to explicitly learn structured (non-)linear feature effects while jointly modeling other relationships using a deep network. The combination of structured and unstructured models can lead to identifiability issues, impeding the interpretability of the structured part. While this problem has been successfully addressed in previous work, we show that there exists an easy-to-implement, scalable, better-performing, and more general algorithm that theoretically circumvents the downsides of the previous solution and also excels in practice.

Limitations While solving most of the limitations of previous work, our approach is not able to solve Limitation 3.3. A possible solution could be to use PHOGAM together with a sparsity-inducing regularization. Another restriction of the approach is given when applied to learned features as presented in the previous section. If structured features learned by the network (e.g., by the feature-specific networks in NAMs) exhibit large correlation, an additional regularization is required to allow PHO to extract additional linear information from the unstructured model part.

Future Research Our approach opens up several future research directions. As mentioned in Section 4.5, PHO – in contrast to ONO – can be applied to learned bases. Similar to the functional ANOVA approach (Hooker, 2004), this, e.g., also allows for a hierarchical decomposition post-model fitting, can be used to solve identifiability issues in NAMs, or to speed up networks that include structured assumptions.

Caveat We further want to emphasize that the given application in this paper only addresses a small subset of processes involved in the spread of COVID-19 and should not be (the sole) basis for decision-making in the future. A more elaborate analysis could, e.g., impute missing information on the stringency index to explain some of the temporal dynamics now comprised in the time effect in our analysis.

Acknowledgements

We would like to thank the three anonymous reviewers for their comments that helped to further improve the paper.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. pp. 265–283, 2016.

Agarwal, R., Melnick, L., Frosst, N., Zhang, X., Lengerich, B., Caruana, R., and Hinton, G. E. Neural additive mod-

els: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems*, 34: 4699–4711, 2021.

Amoura, K., Wira, P., and Djennoune, S. A state-space neural network for modeling dynamical nonlinear systems. In *IJCCI (NCTA)*, pp. 369–376, 2011.

Baumann, P. F. M., Hothorn, T., and Rügamer, D. Deep conditional transformation models. In *Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pp. 3–18. Springer International Publishing, 2021.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10. ACM, 2016.

Cortez, P. and Morais, A. d. J. R. A data mining approach to predict forest fires using meteorological data. 2007.

De Waal, D. A. and Du Toit, J. V. Automation of generalized additive neural networks for predictive data mining. *Applied Artificial Intelligence*, 25(5):380–425, 2011.

Dorigatti, E., Schubert, B., Bischl, B., and Ruegamer, D. Frequentist uncertainty quantification in semi-structured neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 1924–1941. PMLR, 2023.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

Fritz, C., Dorigatti, E., and Rügamer, D. Combining graph neural networks and spatio-temporal disease models to predict covid-19 cases in germany. *Scientific Reports*, 12: 2045–2322, 2022.

Harvard Dataverse. Data Expo 2009: Airline on time data, 2008.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hooker, G. Discovering additive structure in black box functions. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pp. 575–580. Association for Computing Machinery, 2004.

Hubin, A., Storvik, G., and Frommlet, F. Deep Bayesian regression models. 2018.

Karr, A. F., Lin, X., Sanil, A. P., and Reiter, J. P. Secure regression on distributed databases. *Journal of Computational and Graphical Statistics*, 14(2):263–279, 2005.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kook, L., Herzog, L., Hothorn, T., Dürr, O., and Sick, B. Deep and interpretable regression models for ordinal outcomes. *Pattern Recognition*, 122:108263, 2022.

Kopper, P., Pölsterl, S., Wachinger, C., Bischl, B., Bender, A., and Rügamer, D. Semi-structured deep piecewise exponential models. In *Survival Prediction-Algorithms, Challenges and Applications*, pp. 40–53. PMLR, 2021.

Kopper, P., Wiegrebe, S., Bischl, B., Bender, A., and Rügamer, D. DeepPAMM: Deep Piecewise Exponential Additive Mixed Models for Complex Hazard Structures in Survival Analysis. In *Advances in Knowledge Discovery and Data Mining (PAKDD)*, pp. 249–261. Springer International Publishing, 2022.

Lengerich, B., Tan, S., Chang, C.-H., Hooker, G., and Caruana, R. Purifying interaction effects with the functional anova: An efficient algorithm for recovering identifiable additive models. In *International Conference on Artificial Intelligence and Statistics*, pp. 2402–2412. PMLR, 2020.

McFadden, D. et al. Conditional logit analysis of qualitative choice behavior. 1973.

Ortigosa, I., Lopez, R., and Garcia, J. A neural networks approach to residuary resistance of sailing yachts prediction. In *Proceedings of the international conference on marine engineering MARINE*, volume 2007, pp. 250, 2007.

Pölsterl, S., Sarasua, I., Gutiérrez-Becker, B., and Wachinger, C. A wide and deep neural network for survival analysis from anatomical shape and tabular clinical data. *Communications in Computer and Information Science*, pp. 453–464, 2020.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2022.

Robinson, L. D. and Jewell, N. P. Some surprising results about covariate adjustment in logistic regression models. *International Statistical Review/Revue Internationale de Statistique*, pp. 227–240, 1991.

Rügamer, D., Kolb, C., Fritz, C., Pfisterer, F., Kopper, P., Bischl, B., Shen, R., Bukas, C., de Andrade e Sousa, L. B., Thalmeier, D., Baumann, P., Kook, L., Klein, N., and Müller, C. L. deepregression: a flexible neural network framework for semi-structured deep distributional regression. *Journal of Statistical Software*, 2022.

Rügamer, D., Bender, A., Wiegrebe, S., Racek, D., Bischl, B., Müller, C. L., and Stachl, C. Factorized structured regression for large-scale varying coefficient models. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part V*, pp. 20–35. Springer, 2023.

Rügamer, D., Kolb, C., and Klein, N. Semi-structured distributional regression. *The American Statistician*, 0 (ja):1–25, 2023.

Sick, B., Hothorn, T., and Dürr, O. Deep transformation models: Tackling complex regression problems with neural network based transformation models. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2476–2481. IEEE, 2021.

Tran, M.-N., Nguyen, N., Nott, D., and Kohn, R. Bayesian deep net GLM and GLMM. *Journal of Computational and Graphical Statistics*, 29(1):97–113, 2020.

Tsanas, A. and Xifara, A. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49: 560–567, 2012.

Wahltinez, O., Cheung, A., Alcantara, R., Cheung, D., Daswani, M., Erlinger, A., Lee, M., Yawalkar, P., Lê, P., Navarro, O. P., et al. Covid-19 open-data a global-scale spatially granular meta-dataset for coronavirus disease. *Scientific data*, 9(1):1–11, 2022.

Wood, S. N. Thin plate regression splines. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(1):95–114, 2003.

Wood, S. N. *Generalized additive models: an introduction with R*. Chapman and Hall/CRC, 2017.

Yeh, I.-C. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.

A. Proofs

A.1. Proof of Lemma 3.2

First, note that

$$\mathcal{H}_{PHO} \equiv \mathcal{H}, \tag{8}$$

where $\mathcal{H}$ is the hypothesis space of the unconstrained model. Eq. 8 holds as PHO first uses an unconstrained model to find the model parameters and given the model parameters $\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}}$, PHO just shifts explained variance from one to the other predictor by adjusting the weights, yielding the same model output, i.e.,

$$\hat{\boldsymbol{\eta}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}} = \boldsymbol{X}\widetilde{\boldsymbol{\beta}} + \widetilde{\boldsymbol{U}}\hat{\boldsymbol{\gamma}} = \hat{\boldsymbol{\eta}}. \tag{9}$$

For ONO, it holds:

$$\mathcal{H}_{ONO} = \left\{\hat{\boldsymbol{\eta}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{U}}\hat{\boldsymbol{\gamma}} : \hat{\boldsymbol{\beta}} \in \mathbb{R}^p, \hat{\boldsymbol{\gamma}} \in \mathbb{R}^q, \hat{\boldsymbol{U}} \in \mathcal{X}^{\perp^{n\times q}}\right\}, \tag{10}$$

where $\mathcal{X}^{\perp}$ is the orthogonal complement of the column space $\mathcal{X}$ spanned by $\boldsymbol{X}$. This might already look like a hypothesis space restriction as for $\mathcal{H}$, we allow $\hat{\boldsymbol{U}} \in \mathbb{R}^{n\times q} \supset \mathcal{X}^{\perp^{n\times q}}$ while this is not the case for ONO. However, the following Corollary shows that this is, in fact, not restrictive.

Corollary A.1. *For every $\hat{\boldsymbol{y}} \in \mathcal{H}$ where $\hat{\boldsymbol{U}} \in \mathcal{X}^{n\times q}, n > p$, there is always a parameter set $\breve{\boldsymbol{\beta}}, \breve{\boldsymbol{\gamma}}$ with $\breve{\boldsymbol{U}} \in \mathcal{X}^{\perp^{n\times q}}$ that yields $\mathcal{H}_{ONO} \ni \breve{\boldsymbol{\eta}} = \boldsymbol{X}\breve{\boldsymbol{\beta}} + \breve{\boldsymbol{U}}\breve{\boldsymbol{\gamma}} = \hat{\boldsymbol{\eta}}$.*

The proof is, in turn, given by (6) when using $\hat{\boldsymbol{y}} = \hat{\boldsymbol{\eta}}_k$ yielding $\breve{\boldsymbol{\beta}} = \widetilde{\boldsymbol{\beta}}$, $\breve{\boldsymbol{U}} = \widetilde{\boldsymbol{U}}$, and $\breve{\boldsymbol{\gamma}} = \hat{\boldsymbol{\gamma}}$. As a consequence, the previous corollary also proves Lemma 3.2 since SSNs are a function of their additive predictor $\boldsymbol{\eta}$ only (and the same also holds for distributional regression models with K predictors $\boldsymbol{\eta}_k, k = 1, \ldots, K$).

A.2. Proof of Lemma 3.6

In order to prove Lemma 3.6, we make the following assumptions:

Assumption A.2. Let $\hat{u}_1^*, \ldots, \hat{u}_b^*$ be independent realizations from a random variable $\mathfrak{U}$ with $\mathbb{E}(\mathfrak{U}) = 0$ and $\text{Var}(\mathfrak{U}) = \sigma_U^2$. Let $x_1^*, \ldots, x_b^*$ be independent realizations from a random variable $\mathfrak{X}$ with $\mathbb{E}(\mathfrak{X}) = 0$ and $\text{Var}(\mathfrak{X}) = \sigma_X^2$.

Note that these assumptions are not very restrictive as 1) no distribution assumption is made, 2) the latent features $\hat{\boldsymbol{U}}$ are usually mean zero as the bias term is modeled in the structured model part in SSNs, 3) features $\boldsymbol{X}$ can always be mean-centered to fulfill the zero mean assumption, and 4) independence for data points are given per definition of the analysis setup (Section 3). When the projection of the SSN works properly and $\mathfrak{U}$ and $\mathfrak{X}$ are orthogonal, i.e., $\mathbb{E}(\mathfrak{X}\mathfrak{U}) = 0$, then the inner product $\boldsymbol{X}^{*\top}\hat{\boldsymbol{U}}^* = \sum_{i=1}^b x_i^*\hat{u}_i^*$ is the sum of b independent variables with expectation zero. Therefore, by the central limit theorem, this inner product converges in distribution to a normal distribution with mean 0 and variance $\sigma_E^2 = \sigma_U^2\sigma_X^2 b^{-1}$. Thus the variance of the inner product converges to 0 with rate $\mathcal{O}(1/b)$ for $b \to \infty$.

While this proves the limiting behavior of the variance induced by the projection, it does not discuss additional biases induced by it. However, as the mean of $\boldsymbol{X}^{*\top}\hat{\boldsymbol{U}}^*$ is zero (also theoretically), the additional term $\mathcal{P}_{X^*}\hat{\boldsymbol{U}}^*\hat{\boldsymbol{\gamma}}$ in the projection also has mean zero, implying that the projection does not induce any additional bias but only additional variance yielding to the increase in RMSE. This variance is proportional to σ_E^2.

B. Further Algorithms

B.1. Computing Out-of-sample Contributions

Given a new data set $\mathcal{D}^*$ with structured features $\boldsymbol{X}^*$ and unstructured features $\boldsymbol{Z}^*$ and trained network on features $(\boldsymbol{X}, \boldsymbol{Z})$, the contributions of both network parts are given by

$$\boldsymbol{\eta}_k^{str} = \boldsymbol{X}^*\widetilde{\boldsymbol{\beta}}, \tag{11}$$

$$\boldsymbol{\eta}_k^{unstr} = (\hat{\boldsymbol{U}}(\boldsymbol{Z}^*) - \boldsymbol{X}^*\boldsymbol{X}^{\dagger}\hat{\boldsymbol{U}}(\boldsymbol{Z}))\hat{\boldsymbol{\gamma}}, \tag{12}$$

where $\hat{\boldsymbol{U}}(\cdot)$ indicates the penultimate layer's features as a function of the unstructured network inputs. The contribution in (12) follows from the fact that

$$\boldsymbol{\eta}_k^{unstr} = \boldsymbol{X}^*\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{U}}(\boldsymbol{Z}^*)\hat{\boldsymbol{\gamma}} - \boldsymbol{X}^*\widetilde{\boldsymbol{\beta}}$$

after orthogonalization.

C. Semi-Structured Importance Measures

Using the orthogonality property of predictors post-processed by PHO, we can define importance measures in the spirit of a functional ANOVA.

Definition C.1. Structured predictor importance The importance of a structured predictor in a post-hoc orthogonalized SSN can be characterized by $\text{EV}^{str} = \text{Var}(\boldsymbol{X}\widetilde{\boldsymbol{\beta}})/\text{Var}(\hat{\boldsymbol{\eta}})$.

The explained variance EV^{str} is a proportion $\in [0, 1]$ and describes how much of the model's explanation can be related to the structured model part. Furthermore, a feature-based importance measure (also for SSNs beyond simple Gaussian mean regression) can be constructed as follows:

Definition C.2. Structured feature importance Let ℓ be the model's likelihood. Then a measure of importance for the jth structured predictor in $\boldsymbol{\eta}^{str}$ is $R_j^2 :=$

$1 - (\ell(\hat{\boldsymbol{\eta}})/\ell(\hat{\boldsymbol{\eta}}_{-j}))$, where $\hat{\boldsymbol{\eta}}_{-j}$ is the predictor excluding the jth term in the structured predictor $\boldsymbol{\eta}^{str}$.

The importance measure defined in Definition C.2 is also known as McFadden's pseudo-R^2 (McFadden et al., 1973), yielding values between 0 and 1 (the higher, the more explained "variance"). Both importance measures can also be further generalized to modeling setups with K predictors in various ways. The structured predictor importance can, e.g., be calculated by considering the variance of only one of the K structured predictors $\boldsymbol{\eta}_k^{str}$, yielding the importance of the structured model part for the kth parameter. For the structured feature importance, on the other hand, a meaningful definition would remove the jth feature from all predictors $\boldsymbol{\eta}_k, k = 1, \ldots, K$, that include this feature.

D. Simulation and Benchmark Details

D.1. Implementation

To allow for a fair comparison between the methods, we use the package `deepregression` (Rügamer et al., 2022) that implements both GAMs, DNNs, and SSNs in TensorFlow (Abadi et al., 2016) and R (R Core Team, 2022). This mitigates differences in performance due to different software implementations.

D.2. Simulation Details

For all simulations, we generate features using a standard normal distribution. The data-generating process of the simulation introduced in Subsection 5.3 defines the 10 non-linear functions as follows: $f_0(x_0) = \cos(5x_0)$, $f_1(x_1) = \tanh(3x_1)$, $f_2(x_2) = -(x_2^3)$, $f_3(x_3) = -3x_3\cos(3x_3-2)$, $f_4(x_4) = \exp(0.5x_4)-1$, $f_5(x_5) = x_5^2$, $f_6(x_6) = \sin(x_6)\cos(x_6)$, $f_7(x_7) = \sqrt{|x_7|}$, $f_8(x_8) = \phi(x_8) - 1/8$ with standard normal density ϕ, and $f_9(x_9) = -x_9\tanh(3x_9)\sin(4x_9)$.

D.3. Benchmark Details

Benchmark Data Sets Table 2 gives an overview of the different data sets used in our benchmark.

Network Architecture and Optimization For all DNNs in our benchmark, we define architectures consisting of fully-connected layers, each with a 0.1 dropout rate after every hidden layer. Hidden layers are defined using a ReLU activation, and the output layer uses no activation. The numbers of units of the four options are a) 200/1, b) 200/200/1, c) 20/1, and d) 20/20/1. All models are optimized using Adam (Kingma & Ba, 2014) with a learning rate of 0.001, and early stopping based on 10% validation data with a patience of 50 iterations.

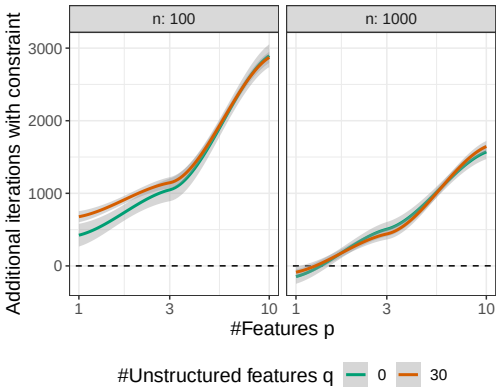

Figure 8. Difference in iterations when training the SSN with and without constraint (positive values indicate longer convergence with constraint) for different numbers of p (x-axis), q (colors), and data sizes n (facets).

Table with all Results

E. Additional Experiments and Results

E.1. Convergence of ONO

The following Figure 8 shows the additional iteration required by ONO compared to PHO until the validation error is not improving anymore and early-stopping is triggered.

Results suggest a linear increase with p whereas larger n seems to mitigate the issue. However, as in general with more data fewer iterations are required, it is not directly clear whether this trend is indeed related to the projection. The size of q, i.e., the number of unstructured features, does not seem to play a role in the convergence speed. We, however, note that in general, many different factors can potentially influence the convergence and this simulation only shows that there is a downside in convergence speed which will get worse with increasing p.

E.2. Semi-Structured Models with Non-Linear Effects

The results of Subsection 5.3 are visualized in the following Figure 9.

As for linear effects, we see that the unconstrained model is not able to correctly estimate the splines, yielding large RMSE values in most settings. As a gold standard, the GAM model defines a lower bound on the performance, which is often reached by the SSNs. While there is no clear trend or ranking between ONO, PHO, and PHOGAM, larger RMSE values are more often obtained using ONO. In various cases, PHOGAM further shows better performance than PHO, suggesting an improvement when accounting for the model's penalization.

Table 2. Data set characteristics and references.

Data set	# Obs.	# Feat.	Pre-processing	Reference
Airfoil	1503	5	-	Dua & Graff (2017)
Concrete	1030	8	-	Yeh (1998)
Diabetes	442	10	-	Efron et al. (2004)
Energy	768	8	-	Tsanas & Xifara (2012)
ForestF	517	12	logp1 transformation for `area`; numerical representation for `month` and `day`	Cortez & Morais (2007)
Yacht	308	6	-	Ortigosa et al. (2007); Dua & Graff (2017)

	GAM	ONO (a)	ONO (b)	ONO (c/d)	PHO (a)	PHO (b)	PHO (c)	PHO (d)	MLP (a)	MLP (b)	MLP (c)	MLP (d)
Airfoil	120 (1.9)	110 (1.6)	110 (1.8)	110 (1.6)	14 (1.5)	3.4 (0.22)	9.5 (1.2)	5.9 (0.41)	22 (1.6)	3.3 (0.24)	23 (1.4)	5.8 (0.28)
Concrete	7.1 (0.53)	7.2 (0.56)	7.2 (0.55)	7.2 (0.56)	6 (0.58)	4.9 (0.8)	6.3 (0.63)	7.6 (0.63)	7.5 (1.5)	5 (0.66)	7.4 (0.87)	7 (1.4)
Diabetes	140 (9.6)	130 (9.5)	120 (9.4)	130 (9.5)	57 (7.8)	67 (9.5)	57 (7.9)	57 (7.2)	63 (7.7)	68 (8.5)	62 (8.3)	58 (7.1)
Energy	3.4 (0.36)	3.4 (0.36)	3.4 (0.36)	3.4 (0.36)	3.3 (0.36)	3.4 (0.42)	3.3 (0.37)	3.3 (0.38)	4.2 (0.44)	3.2 (0.39)	4.9 (0.56)	3.3 (0.31)
ForestF	1.5 (0.16)	1.5 (0.15)	1.5 (0.12)	1.5 (0.15)	1.5 (0.16)	1.7 (0.18)	1.5 (0.13)	1.5 (0.14)	1.4 (0.1)	1.6 (0.14)	1.4 (0.11)	1.4 (0.11)
Yacht	3 (0.43)	2.9 (0.43)	2.9 (0.43)	2.9 (0.43)	2.6 (0.42)	1.9 (0.49)	2.8 (0.46)	2.3 (0.56)	5 (0.54)	2 (0.44)	5 (0.61)	2.8 (0.54)

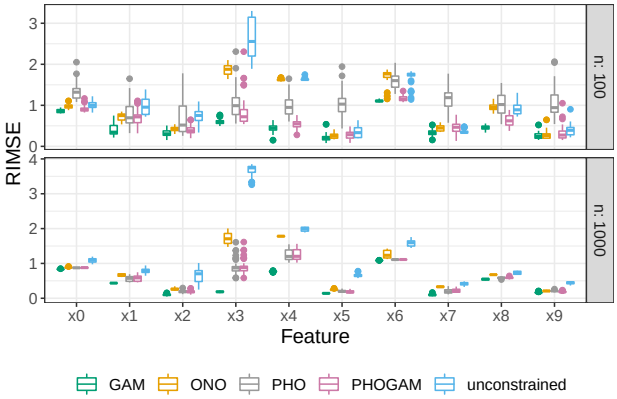

Figure 9. Spline estimation performance of different methods (colors) for different features (x-axis) and data sizes n (rows).

F. Further Details on the Application

We now give further details on the application presented in Section 6

Structured Model We model the time, temperature, humidity, and logarithmic population using a penalized thin plate regression spline with 9 basis functions and difference penalty following Wood (2003; 2017). The bivariate effect of longitude and latitude is modeled using a tensor-product spline using thin plate bases with a total of 25 basis functions. Penalization is again enforced through a difference penalty.

Deep Neural Network For the DNNs for ONOand PHO we use 2 hidden layers each with 100 units, ReLU activation followed by a dropout layer with 20% dropout rate. A final linear fully-connected layer with one unit is used to learn γ. For the DNN of NAM we use a different DNN as we found that the otherwise either the feature-specific DNNs or the DNN for the unstructured part learns a relationship with the response but not both simultaneously. We, therefore, specify the DNN with two hidden layers each with 20 hidden units and tanh activation as well as a subsequent dropout layer with 20% dropout rate. The feature networks are defined as in the software accompanying Agarwal et al. (2021) using an activation layer (here with ReLU activation) and 64 units, followed by two hidden layers with 64 and 32 units (also activated using ReLU). The final layer is again specified using a linearly activated layer with 1 unit. For NAM's feature-specific DNN for the bivariate effect, we use the same architecture as for the other feature-specific DNNs for both latitude and longitude, but add another dense layer with 5 units and linear activation for both variables. We then calculate the row-wise tensor product of those two times 5 features, yielding the new DNN-learned feature basis of dimension 25 (analogous to the 25 basis function in the SSNs' structured predictors), which is finally combined with one dense layer with 1 hidden unit and linear activation.

Optimization and Evaluation All networks are optimized using Adam (Kingma & Ba, 2014) with a learning rate of 0.001, a batch size of 1024 (chosen relatively large due to the size of the data set), 100 epochs, a validation split of 0.1, and a patience of 3 for early stopping. For NAM, we chose 150 epochs and a patience of 15 to account for the more erratic training curves.

Results Figure 10 shows the comparison between the estimated spatial effects of ONO, PHOGAM (which is practically identical with PHO), NAM, and PHONAM. ONO and PHOGAM show very similar spatial effects whereas the NAM estimates a much more concentrated positive effect in the northeast of the US. The PHONAM variant corrects this effect and shows positive effects mainly in the south of the US.

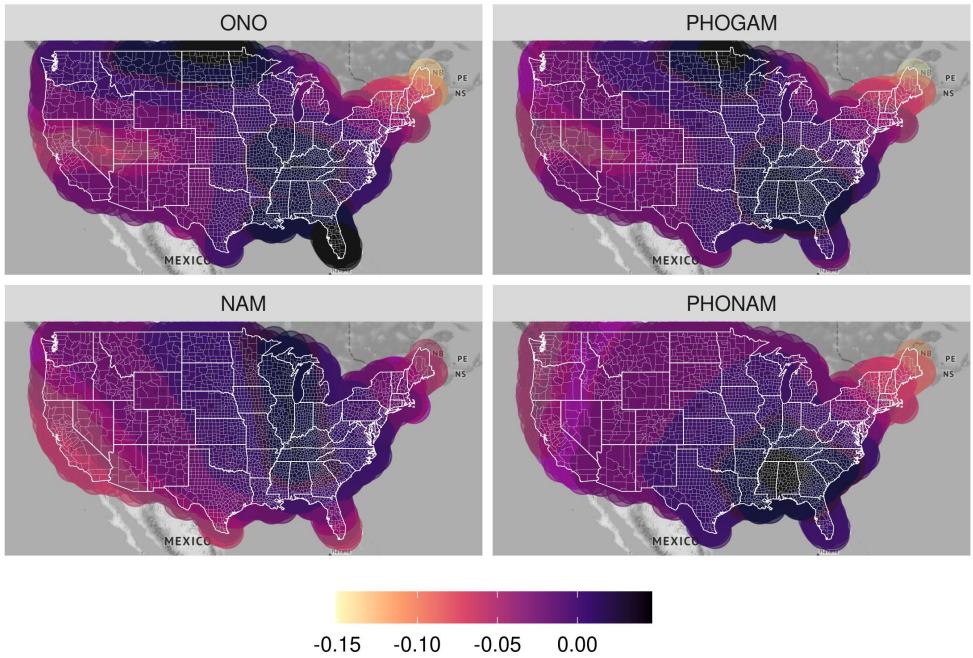

Figure 10. Estimated spatial effect of four of the methods (different facets).