Analyzing Complex Models by Orthogonal Input-Output Decompositions

Pavel Loskot
ZJU-UIUC Institute
Haining, China
pavelloskot@intl.zju.edu.cn

***Abstract*—Devising mathematical models with high accuracy, but otherwise low interpretability as well as explainability is no longer sufficient. This paper proposes a universal, model-agnostic method for achieving a certain degree of explainability of complex mathematical models including the models that are used, for example, in computer simulations. Specifically, the proposed method prescribes how to effectively perform Sobol's outer decomposition of a complex model by exploiting masking of the model inputs by a default value. The masking can not only divide the inputs into multiple orthogonal subspaces, but it also determines the granularity, at which the model explainability is studied. The outputs corresponding to every masked input can be orthogonalized by some existing methods including, for example, the Gram-Schmidt process. It enables readily finding the optimum linear combining of these orthogonal outputs. It should be noted that such a model decomposition is not intended to improve the accuracy by ensemble modeling, but the goal is to uncover an inherent structure and properties of complex models.**

***Keywords*—*Explainability; model-agnostic; multivariate function; orthogonal decomposition; Sobol's decomposition*.**

I. Introduction

Mathematical modeling has become indispensable in many scientific and engineering disciplines. In engineering, for example, mathematical models are necessary for offering credible explanations about how the systems should be designed in any particular way, or why the given system has a particular performance. The widespread adoption of mathematical models is steadily driven by the modeling economics. Thus, designing ever more complex engineering systems and elucidating understanding of many physical and biological systems have become much cheaper and faster when they are studied as mathematical models rather than performing often costly and tedious laboratory or field experiments. Mathematical modeling allows guided searches and systematically exploring very large spaces of domain knowledge by leveraging computing technology and algorithms. The expanding ecosystem of mathematical models leads to much higher information gains, and it also facilitates automated discovery of new knowledge.

Mathematical models can appear in several basic forms. The first obvious form are mathematical expressions, which can be manipulated using algebraic rules and calculus. It is the only form, which guarantees reproducability. The second form are algorithms and computer simulations represented implicitly or explicitly by hierarchical mathematical structures. They are mainly used for computing the outputs for given input values. The third form are the sets of input-output data values. The datasets can be used to infer other forms of mathematical models with a varying degree of accuracy. Deep learning models are particularly popular nowadays due to their universality to fit different kinds of datasets within the same model structure. Such a fundamental property can be attributed to compositional sparsity of computable functions [1].

The model development and analysis often requires understanding how the model outputs are derived from its inputs. It includes understanding how the input-output transformation is affected by the model structure as well as by different sets of model parameters. Such a task has been traditionally referred to as sensitivity analysis [2], and it is key in providing the model interpretability. It is generally accepted that there is a trade-off between the model interpretability and the model accuracy [3], although this assumption is currently being debated. The model interpretability enables a variety of model-related tasks such as calibrating, optimizing, selecting, validating and simplifying the models, and making the models more robust by reducing the model uncertainties.

The basic strategy for performing a local sensitivity analysis of the model exploits derivatives in multiple input dimensions [2]. The global sensitivity analysis can be obtained by expanding the model outputs directly, or by expanding the model output means or variances in terms of the input statistics [2], [4]. The surrogate or meta models can be particularly effective in reducing the computational costs, and providing the faster convergence. The challenge is how to preserve the key properties of the original model [4].

Furthermore, since the models are usually used to provide a certain functionality within high-level applications, it may be easier as well as sufficient to examine the model explainability [5]. Unlike interpretability, explainability does not require understanding a complete model structure. Instead, explainability focuses on a more narrow objective of identifying, which model inputs are more important in determining the model outputs, This problem is also referred to as input factors screening, or attribution problem. The most popular methods for achieving the model explainability are model-agnostic. They include permutation importance of features, various dependency plots (e.g., individual conditional expectation plots, and partial dependency plots), local interpretable model-agnostic explanations (LIME) [6], and Shapley additive explanations (SHAP) [7].

In general, orthogonalizations can be used to improve the convergence, and even the performance of models [8]. The orthogonalization injects separability into the model, which then leads to a reduced complexity and increased robustness, since the model components have less effect on each other. The orthogonal model components can be added or removed without requiring to change the existing model structure. In the literature, various orthogonalization methods of linear and non-linear models were considered. For example, the Gram-Schmidt process together with variable projections was adopted in [9] to fit the data with a linear combination of non-linear models. The Gram-Schmidt process was also used in [10] to improve learning of deep neural networks. The mixing of orthogonalized linear models was examined in [11] to improve the scale and the convergence of data fitting. The properties of orthogonalized linear regression were studied in [12], [13]. A faster convergence of statistical parameter estimation was obtained by exploring orthogonal statistical moments in [14]. The methods for achieving explainability in deep neural networks were reviewed in [15], although without assuming any orthogonalization strategies.

The main problem with the above works is they implicitly assume that the model under consideration is a linear combination of multiple sub-models. This excludes a large number of other models, for example, the models that are used in computer simulations, where explicit mathematical description is often very limited. In this paper, we overcome such a limitation by defining the model components using multiple input projections with the model being considered, so it does not matter how the given model is specified.

More specifically, our objective is to explore an inherent structure of complex models by orthogonal projections of their inputs and outputs. The proposed method obviates the need to mathematically manipulate the model, so it is completely model agnostic. The only requirement is that the model being investigated is already "good enough". The proposed method is primarily motivated by Sobol's decomposition of multivariate functions [16]. The proposed method is similar to SHAP method except that it does not require obtaining multiple models, for example, retraining multiple machine learning models, for different input subspaces. Moreover, both input and output subspaces can be made orthogonal. It enables exploiting other useful properties, and providing insights into the dependency and importance of model inputs and outputs.

The rest of the paper is organized as follows. Section II reviews common decomposition strategies of multivariate functions. Section III introduces the proposed decomposition method for explainability of complex models. Numerical example is investigated in Section IV. Conclusion and future work are summarized in Section V.

II. Decompositions of Multivariate Functions

Function decompositions allow uncovering their latent structure, reducing the computational complexity by enabling divide & conquer strategies, obtaining approximations, which are amenable to optimizations and analysis, and most importantly, they can also provide explainability. In the literature, function decomposition is also sometimes referred to as function factorization, and function expansion, respectively, depending on the specific objectives assumed.

Consider a multivariate vector function,

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}(x_1,\ldots,x_I) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_O(\boldsymbol{x}) \end{bmatrix} \in \mathcal{R}^O,\ \boldsymbol{x} \in \mathcal{R}^I \quad (1)$$

representing a mapping between the Euclidean vector spaces, $\mathcal{R}^I \mapsto \mathcal{R}^O$. There are two basic function decomposition strategies. In particular, the decomposition into n product-factors can be written as,

$$\boldsymbol{f}(\boldsymbol{x}) = \prod_{i=1}^{n} \boldsymbol{f}_i(\boldsymbol{s}_i),\ \boldsymbol{s}_i \subseteq \{x_1,\ldots,x_I\} \quad (2)$$

whereas the decomposition into n sum-factors is defined as,

$$\boldsymbol{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{f}_i(\boldsymbol{s}_i),\ \boldsymbol{s}_i \subseteq \{x_1,\ldots,x_I\}. \quad (3)$$

These decompositions are very useful for effectively performing, for example, the function marginalization and maximization over all except a small number of independent input variables. Moreover, it is usually easier to express a complicated support region, $\mathcal{A}$, of the function, $\boldsymbol{f}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathcal{A}$, using a scalar decision function, $A(\boldsymbol{x})$, i.e.,

$$\boldsymbol{f}(\boldsymbol{x})A(\boldsymbol{x}) = \begin{cases} \boldsymbol{f}(\boldsymbol{x}), & \boldsymbol{x} \in \mathcal{A} \\ 0, & \boldsymbol{x} \notin \mathcal{A} \end{cases} \quad (4)$$

which enforces zero function values, when the inputs are outside the support region. Since $A(\boldsymbol{x})$ is itself a multivariate function, it can be decomposed using the same factorizations.

The multivariate Taylor expansion [17] was assumed in [4] to obtain a polynomial expansion of stochastic functions in multiple dimensions, i.e.,

$$\boldsymbol{y} \approx \sum_{i=1}^{n} a_i\, |\boldsymbol{x} - \mathrm{E}[\boldsymbol{x}]|_1^i \quad (5)$$

where $\mathrm{E}[\cdot]$ denotes the expectation, and $|\cdot|_1$ is the absolute value of a sum of the vector or matrix elements. The main issue with the Taylor-based function expansion is that it is very localized, so one has to decide about which point in the input space the function is to be approximated.

In the literature, there are different versions of the universal approximation theorem [18]. Specifically, this theorem claims that certain broad classes of multivariate functions can be approximated to an arbitrary accuracy by compounding a sufficient number of linear transformations followed by a dimension-wise non-linearity (activation function), i.e.

$$\boldsymbol{y} \approx = \cdots\ \boldsymbol{\sigma} \circ (\boldsymbol{A}_i, \boldsymbol{b}_i) \circ \cdots \boldsymbol{\sigma}(\boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{b}_1). \quad (6)$$

The corresponding computing structure is known as a multilayer perceptron (MLP).

Alternatively, the Kolmogorov-Arnold theorem claims that multivariate scalar functions defined on a unit hypercube can

be approximated to an arbitrary accuracy by assuming the decomposition [19],

$$f(\boldsymbol{x}) = \sum_{i=0}^{2n} \Phi \left(\sum_{j=1}^{n} \phi_{i,j}(x_j) \right). \tag{7}$$

The caveat is that, in practice, the non-linear functions, Φ, and, $\phi_{i,j}$, can be very peaky, or otherwise ill-shaped. The corresponding computing structure is known as the Kolmogorov-Arnold network (KAN) [20].

Another function decomposition, which will be assumed in the following section to enable explainability of complex models, is due to Sobol [2], [16]. It is referred to as Sobol's decomposition, and it is an example of the sum-factors decomposition (3). In particular, a multivariate vector function, $\boldsymbol{f}$, can be systematically expanded as,

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}_0 + \sum_{i=1}^{I} \boldsymbol{f}_i(\boldsymbol{x}_i) + \sum_{\substack{i,j=1 \\ i \neq j}}^{I} \boldsymbol{f}_{i,j}(\boldsymbol{x}_i, \boldsymbol{x}_j) \cdots + \sum_{i=1}^{I} \boldsymbol{f}_{\{1:I\}\setminus i}(\boldsymbol{x}) \tag{8}$$

where $\boldsymbol{f}_0$ is a constant vector, $\boldsymbol{x}_i$ denotes the i-th variable of the I-dimensional input vector, $\boldsymbol{x}$, and, $\{1:I\} \setminus i$, removes the index i from the index set, $\{1,2,\ldots,I\}$. Expansion (8) also allows for symmetric component functions, i.e., the functions that are invariant to permutations of their arguments. More importantly, Sobol's decomposition is not unique, however, without any further constraints, it is exact.

There is yet another model-agnostic method for approximating multivariate functions, which turned out to be very effective in many practical scenarios involving complex models that are expensive to evaluate. The basic idea of this method is to approximate the model by the realization of a multi-dimensional Gaussian process [21] assuming only a few points where the function values are known. However, this method is not considered in this paper.

III. Orthogonal Input-Output Decompositions

Kolmogorov-Arnold decomposition (7) assumes only univariate component functions, whereas Sobol's decomposition (8) combines component functions having different number of variables. Assuming the latter, it can be argued that the summands in (8) that are dependent on larger number of input variables are more accurate approximations of the given function, $\boldsymbol{f}(\boldsymbol{x})$, then other component functions having smaller number of variables. However, the component functions with more variables are not only more difficult to obtain, but they are also less interpretable.

In practice, a good trade-off between the decomposition interpretability, complexity, and accuracy can be achieved by assuming non-empty variable subsets, $\boldsymbol{s}^{(i)} \subset \{x_1,\ldots,x_I\}$, $i = 1,\ldots,N$, such that the subsets, $\boldsymbol{s}^{(i)}$, fully cover all input variables, $\boldsymbol{x}$. Here, it is proposed to only consider the following terms in decomposition (8) in order to obtain an interpretable approximate representation of the original function, i.e., let,

$$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{f}_0 + \sum_{i=1}^{N} \boldsymbol{f}_i(\boldsymbol{s}^{(i)}) + \sum_{\substack{i,j=1 \\ i \neq j}}^{N} \boldsymbol{f}_{i,j}(\boldsymbol{s}^{(i)}, \boldsymbol{s}^{(j)}). \tag{9}$$

The structural interpretation of decomposition (9) is obvious; it is a fully connected graph consisting of N vertices as depicted in Figure 1. The vertices are assigned the component functions, $\boldsymbol{f}_i(\boldsymbol{s}^{(i)})$, while the edges between the vertices are assigned the pairwise component functions, $\boldsymbol{f}_{i,j}(\boldsymbol{s}^{(i)}, \boldsymbol{s}^{(j)})$.

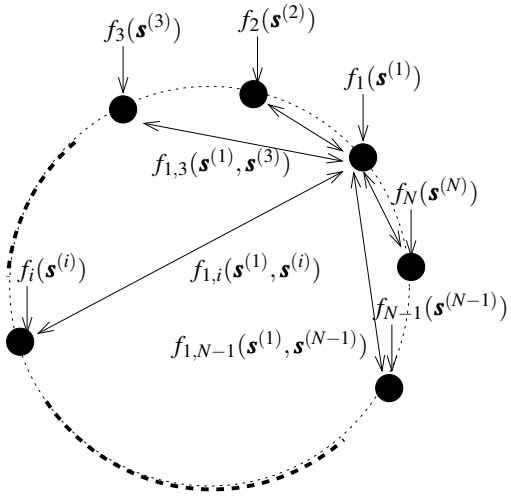

Figure 1. Truncated Sobol's decomposition of a multivariate function as an interpretable and universal representation with good approximation accuracy.

In the sequel, we assume that the multivariate vector function, $\boldsymbol{f}$, represents a complex model, $M(\boldsymbol{x};\boldsymbol{\Omega})$, which is parameterized by a set of parameters, $\boldsymbol{\Omega}$, i.e.,

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\Omega}) \equiv M(\boldsymbol{x};\boldsymbol{\Omega}) \equiv M_{\boldsymbol{\Omega}}(\boldsymbol{x}). \tag{10}$$

The parameters, $\boldsymbol{\Omega}$, represent additional input dimensions of the model. In practice, conditioning on parameters defines a whole class of models, $M_{\boldsymbol{\Omega}}(\boldsymbol{x})$, so the input dimensions are only represented by $\boldsymbol{x}$. For example, the model, M, can be a deep learning model with learnable parameters, $\boldsymbol{\Omega}$. The overall process of extracting the proposed model structure as decomposition (9) is shown in Figure 2.

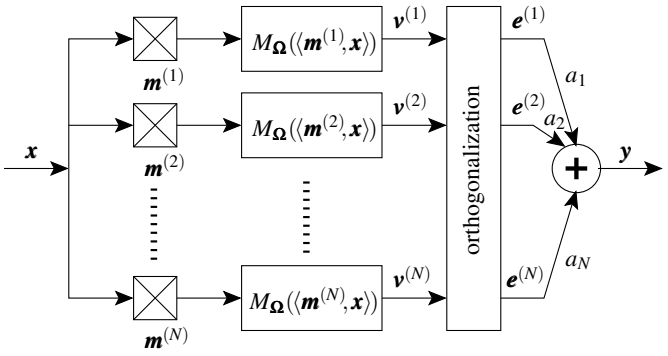

Figure 2. Model-agnostic decomposition (9) for extracting the input-output structure of complex models, $M_{\boldsymbol{\Omega}}(\boldsymbol{x})$.

A. Orthogonal Inputs

It is desirable to consider the projections of model inputs into multiple independent subspaces, since the information

contents of a sum of independent components is then equal to the sum of the information contents of these components. The vector subspaces are independent, provided that they are mutually orthogonal. Thus, assume the subsets of input variables, $\boldsymbol{s}^{(i)}$, satisfying,

$$\begin{cases} \boldsymbol{s}^{(i)} \cap \boldsymbol{s}^{(j\neq i)} = \emptyset & \text{(disjoint)} \\ \cup_i \boldsymbol{s}^{(i)} = \boldsymbol{x} & \text{(full coverage)} \end{cases}. \tag{11}$$

If the model, M, is complex, and expensive to evaluate (a typical case for deep learning models), the question arises how to effectively find the component functions, $\boldsymbol{f}_i$, and, $\boldsymbol{f}_{i,j}$, in decomposition (9). A possible solution is to combine the given multi-dimensional complex model with the orthogonal subspace projections of its inputs as shown next.

Let the input variables be arranged in a column vector, $\boldsymbol{x}$. Define the binary mask vectors, $\boldsymbol{m}^{(i)}$, $i = 1,\ldots,I$, of the same length as $\boldsymbol{x}$, i.e., $\boldsymbol{m}^{(i)} \in \{0,1\}^I$. The zeros in $\boldsymbol{m}^{(i)}$ indicate, which components in $\boldsymbol{x}$ should be masked by setting them to some default value, for example, to zero. Then, by slightly abusing the notation when using the same symbol for a set as well as for its vector representation, the orthogonal (i.e., mutually exclusive) variable subsets, $\boldsymbol{s}^{(i)}$, can be represented as vectors,

$$\boldsymbol{s}^{(i)} = \langle \boldsymbol{m}^{(i)}, \boldsymbol{x} \rangle = (\boldsymbol{m}^{(i)})^T \cdot \boldsymbol{x} = \sum_{j=1}^{I} \boldsymbol{m}_j^{(i)} \boldsymbol{x}_j \tag{12}$$

where $\langle \cdot,\cdot \rangle$ denotes the inner product of two vectors. The corresponding output vectors for every masked input are,

$$\boldsymbol{v}_i = M_{\boldsymbol{\Omega}}(\langle \boldsymbol{m}^{(i)}, \boldsymbol{x} \rangle), \quad i = 1,2,\ldots,N. \tag{13}$$

B. Orthogonal Outputs

After computing the N output vectors for all N masked inputs, it is useful to explore their structure too. In particular, the outputs can be assumed to be deterministic vectors in an O-dimensional vector space. In such a case, they can be orthogonalized by the Gram-Schmidt procedure, provided that, $O \geq N$. In particular, the orthogonal vectors, $\boldsymbol{e}_i$, are obtained as linear projections of vectors, $\boldsymbol{v}^{(i)}$, using the recurrence,

$$\boldsymbol{e}^{(i)} = \boldsymbol{v}^{(i)} - \sum_{j=1}^{i-1} \langle \boldsymbol{v}^{(i)}, \boldsymbol{e}^{(j)} \rangle \, \boldsymbol{e}^{(j)}, \; i = 2,3,\ldots,N \tag{14}$$

with the initial vector, $\boldsymbol{e}^{(1)} = \boldsymbol{v}^{(1)}$. Note that orthogonalization (14) can be expressed as a linear transformation of vectors, $\boldsymbol{v}^{(i)}$. The caveat is that the linear transformation must be recomputed for every new set of vectors, $\boldsymbol{x}$.

In practice, often, $N \gg O$, which rules out the Gram-Schmidt procedure (more precisely, the vectors with O components can span the subspaces in at most O dimensions). In such a case, other orthogonalization strategies are possible that exploit various matrix factorizations. In this paper, the vectors, $\boldsymbol{v}^{(i)}$, are decorrelated by first finding their empirical correlation matrix, $\boldsymbol{C}^{\boldsymbol{v}}$. The elements of $\boldsymbol{C}^{\boldsymbol{v}}$ are the average inner products. They can be computed recursively as soon as the k-th set of vectors, $\boldsymbol{v}^{(i)}(k)$, has been obtained, i.e.,

$$\boldsymbol{C}_{i,j}^{(\boldsymbol{v})}(k) = \frac{k}{k+1} \boldsymbol{C}_{i,j}^{(\boldsymbol{v})}(k-1) + \frac{1}{k+1} \langle \boldsymbol{v}^{(i)}(k), \boldsymbol{v}^{(j)}(k) \rangle. \tag{15}$$

Since the matrix, $\boldsymbol{C}^{(\boldsymbol{v})}$, is guaranteed to be positive-definite, it can be decomposed into a product of the square matrix, $\boldsymbol{D} \in \mathcal{R}^{N\times N}$, i.e., , $\boldsymbol{C}^{(\boldsymbol{v})} = \boldsymbol{D}^T \boldsymbol{D}^T$, for example, using a singular value decomposition (SVD). The matrix, $\boldsymbol{D}$, can be then used to decorrelate, i.e., orthogonalize the vectors, $\boldsymbol{v}^{(i)}(k)$, as,

$$\boldsymbol{E}(k) = [\boldsymbol{e}^{(1)}(k) \; \cdots \; \boldsymbol{e}^{(N)}(k)] = \boldsymbol{V}(k) \boldsymbol{D}^{-T}(k) \in \mathcal{R}^{O\times N} \tag{16}$$

where the vectors at instant, k, are gathered into matrices, $\boldsymbol{E}(k)$, and, $\boldsymbol{V}(k)$, respectively.

The resulting vectors, $\boldsymbol{e}^{(i)}$, are orthogonal, so that,

$$\langle \boldsymbol{e}^{(i)}, \boldsymbol{e}^{(j)} \rangle = \begin{cases} E_i, & i = j \\ 0, & i \neq j \end{cases} \tag{17}$$

where the squared (Euclidean) length of these vectors is,

$$E_i = \langle \boldsymbol{e}^{(i)}, \boldsymbol{e}^{(i)} \rangle = \left\| \boldsymbol{e}^{(i)} \right\|^2. \tag{18}$$

The values, E_i, can be again computed recursively as,

$$E_i(k) = \frac{k}{k+1} E_i(k-1) + \frac{1}{k+1} \left\| \boldsymbol{e}^{(i)}(k) \right\|^2. \tag{19}$$

Recall also that, when the vectors, $\boldsymbol{v}^{(i)}$, are assumed to be random (the $N \gg O$ case), the inner products are computed as empirical means (cf. eq. (15)).

Finally, the orthogonal vectors, $\boldsymbol{e}^{(i)}$, are linearly combined to create the output vector, $\boldsymbol{y}$, as,

$$\boldsymbol{y}(k) = \sum_{i=1}^{N} a_i \boldsymbol{e}^{(i)}(k). \tag{20}$$

It is immediately obvious, why to make the vectors, $\boldsymbol{v}^{(i)}$, orthogonal. Thus, by multiplying both sides of (20) by the vector, $\boldsymbol{e}^{(j)}$, and averaging, the combining coefficients can be computed one-by-one as,

$$\sum_k \langle \boldsymbol{e}^{(j)}(k), \boldsymbol{y}(k) \rangle = \sum_k \sum_{i=1}^{N} a_i \langle \boldsymbol{e}^{(j)}(k), \boldsymbol{e}^{(i)}(k) \rangle = a_j \bar{E}_j \tag{21}$$

$$\Rightarrow \quad a_j = \langle \boldsymbol{e}^{(j)}, \boldsymbol{y} \rangle / E_j = \langle \boldsymbol{e}^{(j)}, M_{\boldsymbol{\Omega}}(\boldsymbol{x}) \rangle / E_j \tag{22}$$

where the desired output vector, $\boldsymbol{y} = M_{\boldsymbol{\Omega}}(\boldsymbol{x})$.

C. Obtaining Decomposition (9)

The procedure described so far can be readily used to obtain the first two summands in decomposition (9), i.e., $(N+1)$ functions, $\{\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$. Even though it is possible to also obtain, at the same time, the second-order functions, $\boldsymbol{f}_{i,j}$, the joint orthogonalization of all $(N(N+1)/2)$ vectors, $\boldsymbol{v}^{(i)} = M_{\boldsymbol{\Omega}}(\langle \boldsymbol{m}^{(i)}, \boldsymbol{x} \rangle)$, and, $\boldsymbol{v}^{(i,j)} = M_{\boldsymbol{\Omega}}(\langle \boldsymbol{m}^{(i,j)}, \boldsymbol{x} \rangle)$, is rather cumbersome. In order to overcome this difficulty, decomposition (9) can be performed in two steps. In particular, after obtaining the zero and the first-order functions, $\{\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$, the $N(N-1)/2$ second order functions, $\boldsymbol{f}_{i,j}$, are obtained in the second step in order to approximate the left-hand side of the expression,

$$\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}_0 - \sum_{i=1}^{N} \boldsymbol{f}_i(\boldsymbol{s}^{(i)}) \approx \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \boldsymbol{f}_{i,j}(\boldsymbol{s}^{(i)}, \boldsymbol{s}^{(j)}). \tag{23}$$

This process can be continued to obtain the $N(N-1)(N-2)/6$ third-order functions, $\boldsymbol{f}_{i,j,k}$, and so on. However, and importantly, the input masks for higher-order functions are no longer assumed to be disjoint (i.e., orthogonal), as they are created by combining the first-order masks, $\boldsymbol{m}^{(i)}$; for example,

$$\begin{aligned} \boldsymbol{m}^{(i,j)} &= \boldsymbol{m}^{(i)} + \boldsymbol{m}^{(j)} \\ \boldsymbol{m}^{(i,j,k)} &= \boldsymbol{m}^{(i)} + \boldsymbol{m}^{(j)} + \boldsymbol{m}^{(k)}. \end{aligned} \tag{24}$$

IV. Numerical Example

Machine learning is assumed as an example of a complex model to illustrate the proposed explainability method. The well-known MNIST dataset [22] of hand-written digits is used with a basic MLP classifier. The training samples are gray-scale images of (28×28) pixels with pixel values between 0 and 1. The MLP has two hidden layers with 30 and 20 neurons, respectively, and 10 softmax outputs. The MLP was trained on all samples over only 10 epochs. The trained model reached the training accuracy of 97.56%, and the testing accuracy of 94.31%. The model was implemented using Python class, "MLPClassifier", from the Python module, "sklearn.neural_network".

The goal is to decompose the trained MLP model into the explainable structure in Figure 2. There are two key aspects to investigate. First, we can compare the training samples from the MNIST dataset, on which the MLP model was trained, against the randomly generated inputs. Second, the orthogonal masks can be compared with the masks generated at random. In both cases, we compute the mean-square error (MSE) between the trained MLP output, and the combined output of the expanded model in Figure 2.

In the experiment concerning different distributions of inputs, in addition to training and testing samples from the MNIST dataset, the same number of (28×28) independent random inputs were generated from a uniform distribution. The resulting MSE values are summarized in Table I, for $N = 7, 14, 28, 56$ and 112 subspace projections, respectively. The MSE values were averaged over K input samples and N model components as follows:

$$\begin{aligned} \overline{\mathrm{MSE}}_0 &= \frac{1}{K}\sum_{k=1}^{K} \|\boldsymbol{y}(k)\|^2 \\ \overline{\mathrm{MSE}}_1 &= \frac{1}{NK}\sum_{i=1}^{N}\sum_{k=1}^{K} \left\|\boldsymbol{v}^{(i)}(k) - \boldsymbol{y}(k)\right\|^2 \\ \min \mathrm{MSE}_1 &= \min_{1\le i\le N} \frac{1}{K}\sum_{k=1}^{K} \left\|\boldsymbol{y}^{(i)}(k) - \boldsymbol{y}(k)\right\|^2 \\ \overline{\mathrm{MSE}}_2 &= \frac{1}{K}\sum_{k=1}^{K} \|\tilde{\boldsymbol{y}}(k) - \boldsymbol{y}(k)\|^2 \end{aligned} \tag{25}$$

where $\boldsymbol{y} = M_{\boldsymbol{\Omega}}(\boldsymbol{x})$ is the output of the trained model, $\boldsymbol{v}^{(i)}$ is the output of the model using the i-th mask as its inputs, and $\tilde{\boldsymbol{y}}$ denotes the overall combined output of the decomposed model. Note also that $\overline{\mathrm{MSE}}_0$ values are independent of N.

Examining the MSE values in Table I, it is obvious that MSE_1 values are comparable to $\overline{\mathrm{MSE}}_0$ values, i.e., masking the inputs can substantially reduce the classifier accuracy in exchange for better interpretability. More importantly, combining all N outputs of the N model replicas with masked inputs can greatly improve the accuracy. Thus, assuming not only the first-order functions in decomposition (9) is expected to further improve the approximation accuracy. Surprisingly, the number of masks N considered seems to have a little effect on the MSE. For training samples from the MNIST dataset, using random or systematic masks make a little difference. However, for randomly generated inputs, the original MLP model and the expanded model have very similar MSE values.

TABLE I. COMPARISON of MSE VALUES

N	inputs	masks	$\overline{\mathrm{MSE}}_0$	$\overline{\mathrm{MSE}}_1$	min MSE_1	$\overline{\mathrm{MSE}}_2$
7	train.	rand.	0.953	0.821	0.717	0.791
		systm.	0.953	1.2085	0.912	0.767
	rand.	rand.	0.416	0.432	0.336	0.275
		systm.	0.418	0.892	0.466	0.253
14	train.	rand.	0.953	0.921	0.805	0.739
		systm.	0.953	1.1028	0.835	0.739
	rand.	rand.	0.417	0.430	0.330	0.270
		systm.	0.417	0.727	0.362	0.252
28	train.	rand.	0.953	0.936	0.866	0.741
		systm.	0.953	1.0348	0.844	0.751
	rand.	rand.	0.417	0.442	0.322	0.258
		systm.	0.416	0.584	0.279	0.233
56	train.	rand.	0.953	0.947	0.850	0.753
		systm.	0.953	0.986	0.877	0.749
	rand.	rand.	0.417	0.437	0.330	0.269
		systm.	0.418	0.511	0.328	0.233
112	train.	rand.	0.954	0.950	0.890	0.731
		systm.	0.954	0.972	0.878	0.742
	rand.	rand.	0.418	0.465	0.351	0.280
		systm.	0.417	0.474	0.315	0.232

Next, we display the correlation matrix, $\boldsymbol{C}^{\boldsymbol{v}}$, and, the probabilities, $\boldsymbol{P}_{i,j}$, that the MLP output predictions for the masked inputs agree with the predictions obtained when the masks are combined. We again consider four cases as in Table I. Specifically, the probabilities, $\boldsymbol{P}_{i,j}$, are defined for the pairs of masks, $\boldsymbol{m}^{(i)}$, and, $\boldsymbol{m}^{(j)}$, $1 \le i \ne j \le N$, as the ratios of the number of instances when the decisions for the two masked inputs are the same as the decision when the two masks are combined, i.e., for the mask, $\boldsymbol{m}^{(i)} + \boldsymbol{m}^{(j)}$. The calculated matrices, $\boldsymbol{C}^{\boldsymbol{v})}$, and, $\boldsymbol{P}$, are shown in Figure 3.

In Figure 3, there is an apparent line marking the main diagonal. All sub-figures are symmetric about the main diagonal. Increasing N not only increases the resolution, but also the variety of calculated values. There are clearly observable patterns (the squares of various sizes and color shades) indicating that orthogonal input subspaces might lead to similar output decisions due to inherent cross-correlations. The probabilities (right-column sub-figures) related to the final output decisions appear to have visually richer patterns than the pairwise cor-

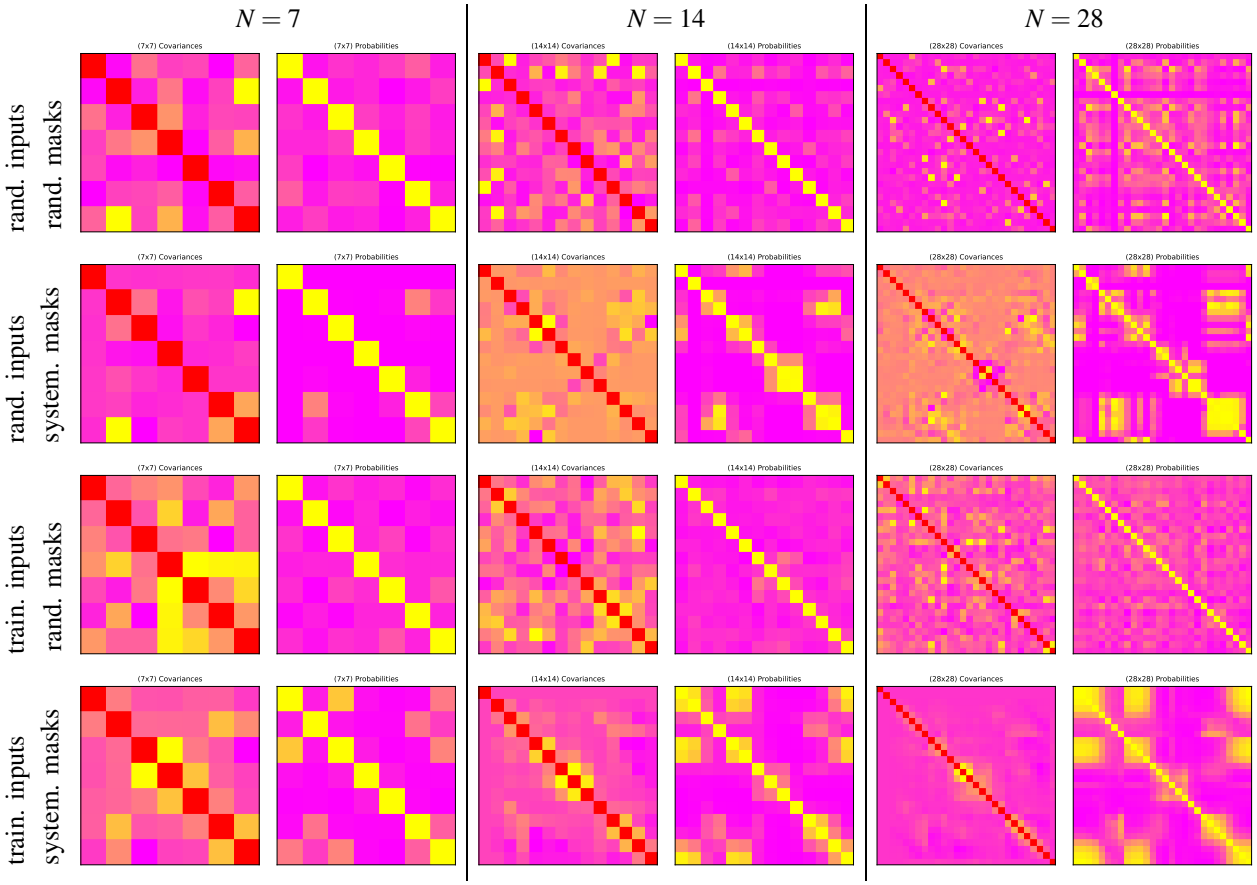

Figure 3. The calculated correlations $\boldsymbol{C}^{\boldsymbol{v}}$ (left columns), and the decision probabilites (defined in the text; right columns). For comparison purposes, the units are arbitrary, and the darker colors represent smaller values.

relations (left-column sub-figures). Interestingly, the patterns for $N = 28$ start emerging as being in 3D. There is a clear difference between the models processing random inputs, and processing the inputs, on which they were trained. This can be exploited for detecting the out-of-sample distributions. Similar claims can be made about systematic (orthogonal) masks vs. random masks. In both cases, the differences become more recognizable when N is increased.

V. Conclusion and Future Work

Sobol's decomposition of multivariate functions was adopted to expand complex models, and to support their input-output explainability. This was achieved by masking the inputs, which is equivalent to projections into orthogonal subspaces. The component model outputs can be orthogonalized in order to facilitate their linear combination. The number of components in model expansion is a trade-off between computational complexity and explainability. For MLP classifier trained on the MNIST dataset, a good value of the number of components for (28×28) inputs seems to be $N = 28$.

The proposed method opens up many opportunities for future research. For instance, providing explainability for complex models requires that explainability is sufficiently simple, or at least much simpler than the model to be explained. For large number of inputs and outputs, the number of possible orthogonal projections, and thus, the number of possible explanations, is exponentially large. It requires to analyze how to choose these projections for a particular explainability objective, which can be defined, for example, as an optimization problem. Moreover, the inputs can be averaged out from the model instead of being masked. There are several other methods for orthogonalizing vectors that can be considered involving, e.g., matrix factorizations (QR, Cholesky, PCA) and lattice reductions. There may be other strategies how to define the components of complex models. The models with orthogonal components can be used as complex basis functions for generating samples with the desired properties instead of focusing on explainability. The dependencies between the model components that are not orthogonal can be studied as structural causal models using statistical methods of causal inferences. Also, the vector space of model parameters can be orthogonalized similarly as the vector space of model inputs in order to perform the sensitivity analysis. Considering Sobol's expansion itself, it is useful to investigate how additional higher-order terms can be used to create higher-order graphs having beyond pairwise interactions. There is also a need to obtain the approximation bounds for Sobol's decomposition, which does seem to have been provided in the literature.

REFERENCES

[1] T. Poggio and M. Fraser, "Compositional sparsity of learnable functions," *Bulletin AMS*, vol. 61, pp. 438–456, May 2024.

[2] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global Sensitivity Analysis: The Primer*. John Wiley & Sons, Ltd, Chichester, England, 2008.

[3] U. Johansson, C. Sönströd, U. Norinder, and H. Boström, "Trade-off between accuracy and interpretability for predictive in silico modeling," *Future Medicinal Chemistry*, vol. 3, no. 6, pp. 647–663, May 2011.

[4] P. Loskot, "Polynomial representations of high-dimensional observations of random processes," *Mathematics*, vol. 9, no. 123, pp. 1–24, Jan. 2021.

[5] Y. Liang, S. Li, C. Yan, M. Li, and C. Jiang, "Explaining the black-box model: A survey of local interpretation methods for deep neural networks," *Neurocomputing*, vol. 419, pp. 168–182, Sep. 2021.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?" explaining the predictions of classifier," in *KDD*, 2016, pp. 1135–1144.

[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *NIPS'17*, 2017, pp. 4768–4777.

[8] J. A. Mumford, J.-B. Poline, and R. A. Poldrack, "Orthogonalization regressors in fMRI models," *PLOS One*, vol. 10, no. 4:e0126255, 2015.

[9] G.-Y. Chen, M. Gan, F. Ding, and C. L. P. Chen, "Modified Gram–Schmidt method-based variable projection algorithm for separable nonlinear models," *IEEE Trans. Neural Networks and Learning Systems*, vol. 30, no. 8, pp. 2410–2418, Aug. 2019.

[10] Y.-P. Zhao, Z.-Q. Li, P.-P. Xi, D. Liang, L. Sun, and T.-H. Chen, "Gram–Schmidt process based incremental extreme learning machine," *Neurocomputing*, vol. 241, pp. 1–17, Jun. 2017.

[11] M. Clyde, H. Desimone, and G. Parmigiani, "Prediction via orthogonalized model mixing," *J. American Statistical Association*, vol. 91, no. 435, pp. 1197–1208, 2012.

[12] Z. Mikulášek, "The benefits of the orthogonal LSM models," Nov. 2007, arXiv:0711.4510v1 [astro-ph].

[13] S. J. Pallavi, S. Dilbag, K. Manjit, and H. Lee, "Comprehensive review of orthogonal regression and its applications in different domains," *Archives Comput. Methods Engineer.*, vol. 29, pp. 4027–4047, 2022.

[14] L. Mackey, V. Syrgkanis, and I. Zadik, "Orthogonal machine learning: Power and limitations," in *PMLR*, vol. 80, 2018, pp. 3375–3383.

[15] G. Ras, N. Xie, M. Gerven, and D. Doran, "Explainable deep learning: A field guide for uninitiated," Apr. 2021, arXiv:2004.14545v2 [cs.LG].

[16] A. Saltelli, M. Ratto, S. Tarantola, and F. Campolongo, "Sensitivity analysis for chemical models," *Chemical Reviews*, vol. 105, pp. 2811–2827, May 2005.

[17] G. B. Folland, "Higher-order derivatives and Taylor's formula in several variables," 2010, Math 425A Course notes.

[18] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366c, Jan. 1989.

[19] G. G. Lorentz, "Metric entropy, widths, and superpositions of functions," *American Mathematical Monthly*, vol. 69, no. 6, pp. 469–485, 1962.

[20] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljaččić, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold Networks," Jun. 2024, arXiv:2404.19756 [cs.LG].

[21] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *PMLR*, vol. 28(1), 2013, pp. 115–123.

[22] Y. LeCun, C. Cortes, and C. J. Burges, "MNIST dataset," 1998, https://yann.lecun.com/exdb/mnist/.