Sparse Interaction Additive Networks via Feature Interaction Detection and Sparse Selection

James Enouen
Department of Computer Science
University of Southern California
Los Angeles, CA
enouen@usc.edu

Yan Liu
Department of Computer Science
University of Southern California
Los Angeles, CA
yanliu.cs@usc.edu

Abstract

There is currently a large gap in performance between the statistically rigorous methods like linear regression or additive splines and the powerful deep methods using neural networks. Previous works attempting to close this gap have failed to fully investigate the exponentially growing number of feature combinations which deep networks consider automatically during training. In this work, we develop a tractable selection algorithm to efficiently identify the necessary feature combinations by leveraging techniques in feature interaction detection. Our proposed Sparse Interaction Additive Networks (SIAN) construct a bridge from these simple and interpretable models to fully connected neural networks. SIAN achieves competitive performance against state-of-the-art methods across multiple large-scale tabular datasets and consistently finds an optimal tradeoff between the modeling capacity of neural networks and the generalizability of simpler methods.

1 Introduction

Over the past decade, deep learning has achieved significant success in providing solutions to challenging AI problems like computer vision, language processing, and game playing [17, 44, 37]. As deep-learning-based AI models increasingly serve as important solutions to applications in our daily lives, we are confronted with some of their major disadvantages. First, current limitations in our theoretical understanding of deep learning makes fitting robust neural networks a balancing act between accurate training fit and low generalization gap. Questions which ask how the test performance will vary given the choice of architecture, dataset, and training algorithm remain poorly understood and difficult to answer. Second, infamously known as blackbox models, deep neural networks greatly lack in interpretability. This continues to lead to a variety of downstream consequences in applications: unexplainable decisions for stakeholders, the inability to distinguish causation from correlation, and a misunderstood sensitivity to adversarial examples.

In contrast, simpler machine learning models such as linear regression, splines, and the generalized additive model (GAM) [16] naturally win in interpretability and robustness. Their smaller number of parameters often have clear and meaningful interpretations and these methods rarely succumb to overfitting the training data. The main shortcoming of these simpler methods is their inability to accurately fit more complex data distributions.

There is a growing strand of literature attempting to merge the interpretability of additive models with the streamlined differentiable power of deep neural networks (DNNs). Key works in this direction such as NAM and NODE-GAM have been able to successfully model one-dimensional main effects and two-dimensional interaction effects using differentiable training procedures [2, 8]. Although many other works have found similar success in modeling one- and two-dimensional interactions, few have made practical attempts towards feature interactions of size three or greater, which we will

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

refer to throughout as higher-order feature interactions. In this way, no existing works have been able to use the hallmark ability of neural networks to model higher-order interactions: amassing the influence of hundreds of pixels in computer vision and combining specific words from multiple paragraphs in language processing. In this work, we bring interpretable models one step closer towards the impressive differentiable power of neural networks by developing a simple but effective selection algorithm and an efficient implementation to be able to train neural additive models which fit higher-order interactions of degrees three and greater. The proposed Sparse Interaction Additive Networks (SIANs) consistently achieve results that are competitive with state-of-the-art methods across multiple datasets. We summarize our contributions as follows:

- We develop a feature interaction selection algorithm which efficiently selects from the exponential number of higher-order feature interactions by leveraging heredity and interaction detection. This allows us to construct higher-order neural additive models for medium-scale datasets unlike previous works in neural additive modeling which only consider univariate and bivariate functions.
- We provide further insights into the tradeoffs faced by neural networks between better generalization and better functional capacity. By tuning the hyperparameters of our SIAN model, we can gradually interpolate from one-dimensional additive models to full-complexity neural networks. We observe fine-grained details about how the generalization gap increases as we add capacity to our neural network model.
- We design a block sparse implementation of neural additive models which is able to greatly improve the training speed and memory efficiency of neural-based additive models. These improvements over previous NAM implementations allow shape functions to be computed in parallel, allowing us to train larger and more complex additive models. We provide code for our implementation and hope the tools provided can be useful throughout the additive modeling community for faster training of neural additive models. [1]

2 Related Work

The generalized additive model (GAM) [16] has existed for decades as a more expressive alternative to linear regression, replacing each linear coefficient with a nonparametric function. Two-dimensional extensions appeared in the literature shortly after its introduction [46]. Over the years, an abundance of works have used the GAM model as an interpretable method for making predictions, with the choice of functional model typically reflecting the most popular method during the time period: regression splines, random forests, boosting machines, kernel methods, and most recently neural networks [18, 7, 21, 49]. Two of the most prominent neural network based approaches are NAM and NODE-GA^{2}M [2, 8]. The former stacks multilayer perceptrons to build a one-dimensional GAM; the latter connects differentiable decision trees to build a two-dimensional GAM. Both have demonstrated competitive performance and interpretable trends learned over multiple tabular datasets. Other neural network extensions [47, 33] increase the modeling capacity to higher-order interactions by first training a two-dimensional model and then training an additional blackbox neural network to fit the residual error. While this approach does have higher modeling capacity, it foregoes interpretable insights on the higher-order feature interactions and suffers the same inclination to overfit held naturally by deep neural networks.

Other works in additive modeling instead focus on extending the univariate GAM to sparse additive models in the high-dimensional inference regime [24, 36, 29, 48]. Further extensions of these methods to sparse bivariate models for high-dimensional inference also exist [43, 25]. These works extend classical high-dimensional inference techniques like LASSO and LARS from linear parameters to additive functions by shrinking the effect of minimally important variables and emphasizing sparse solutions. We note that, unlike these works, we do not use sparsity in the soft-constrained sense to shrink features from a fixed selection, but instead adaptively use feature interaction measurements to hierarchically build a feasible set of interactions. The only existing work in additive modeling which uses hierarchy to induce sparsity in the same sense as this work is the GAMI-Net which uses a three stage procedure to select candidate pairs under a hierarchical constraint [49]. Extending their procedure to three or higher dimensions would require four or more stages of training and is left unexplored in their work.

[1] Available at github.com/EnouenJ/sparse-interaction-additive-networks

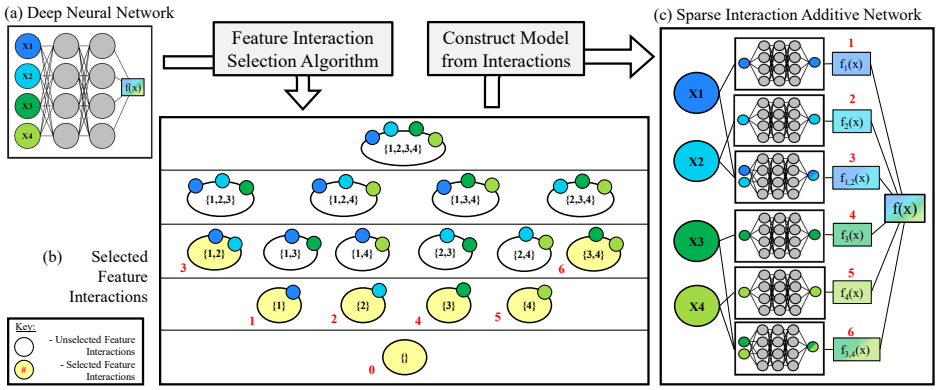

Figure 1: SIAN Pipeline Diagram. (a) We train a base DNN to be able to learn the feature interactions from the dataset given the $d = 4$ input features of X_1, X_2, X_3, X_4. (b) We feed our DNN into the FIS algorithm to be able to select from the $2^d = 16$ possible subsets of $\{X_1, X_2, X_3, X_4\}$. (c) We use our selected feature subsets as the GAM array (of length 6 in the image) for our full SIAN network. The empty set function $f_{\{\}}$ is a constant we absorb into the last additive layer of $f(x)$. Finally, we train our SIAN neural network using the specified architecture.

Although the theoretical extension to three-dimensional additive models is clear, there is currently a lack of discussion surrounding the practical challenges faced when trying to model three-dimensional shape functions. One of the few works to pursue practical implementation of higher-order GAMs on real-world datasets is the work of SALSA [21]. This work uses a specialized kernel function to fit additive models of order three and higher. Their work also corroborates our finding that optimal performance is achieved by different orders for different datasets. However, the kernel-based approaches used in this work make it unsuitable beyond small-scale datasets with few samples. This makes our work one of the first to train additive models of higher-order interactions which leverage the automatic differentiation and GPU optimization toolkits which have become commonplace in modern workstations.

3 Methods

Notation We denote a d-dimensional input as $x \in \mathbb{R}^d$ with its i-th component as x_i; its corresponding output is denoted $y \in \mathbb{R}$. We consider one-dimensional y as in regression and binary classification. We will use $f(x)$ to denote the function or model used to approximate y, implicitly considering the additive noise model $y = f(x) + \varepsilon$ for some noise term ε. We denote a subset of the set of features by $\mathcal{I} = \{i_1, \ldots, i_{|\mathcal{I}|}\} \subseteq [d] := \{1, \ldots, d\}$. Its cardinality is denoted $|\mathcal{I}|$, its complement $\setminus\mathcal{I}$, and its power set $\mathcal{P}(\mathcal{I})$. For $x \in \mathbb{R}^d$, we define $x_\mathcal{I} \in \mathbb{R}^d$ such that:

$$(x_\mathcal{I})_i = \begin{cases} x_i & \text{if} \quad i \in \mathcal{I} \\ 0 & \text{otherwise} \end{cases}$$

3.1 Generalized Additive Models

We first consider the generalized additive model (GAM) [16], which extends linear regression by allowing each input feature to have a nonlinear relationship with the output.

$$g(y) = f_\emptyset + f_1(x_1) + \cdots + f_d(x_d) \tag{1}$$

Each of the f_i 'reshapes' their respective feature x_i and then adds the reshaped feature to the total prediction. These f_i are hence called shape functions and were traditionally fit using regression splines. The function g is the link function which will be the identity function for regression and the inverse-sigmoid $g(y) = \log(\frac{y}{1-y})$ for classification. $f_\emptyset$ is a normalizing constant. This original formulation where each shape function considers only one feature we will further refer to as GAM-1.

Feature Interactions In order to extend this definition we must consider the interplay which occurs between multiple input features. A 'non-additive feature interaction' between features $\mathcal{I} \subseteq [d]$ for the function f is said to exist when the function f cannot be decomposed into a sum of $|\mathcal{I}|$ arbitrary subfunctions such that each subunction f_i excludes one of the interacting variables x_i: $f(x) \neq \sum_{i \in \mathcal{I}} f_i(x_{\{1,2,\ldots,d\} \setminus i})$ [15, 38, 41]. In other words, the entire feature set $\{x_i : i \in \mathcal{I}\}$ must be simultaneously known to be able to correctly predict the output $f(x)$.

The goal of *feature interaction detection* is to uncover these groups of features which depend on one another. For smooth functions, this can be quantitatively done by finding the sets $\mathcal{I} \subseteq [d]$ such that the *interaction strength*, $\omega(\mathcal{I})$, is positive and large.

$$\omega(\mathcal{I}) := \mathbb{E}_x \left[\frac{\partial^{|\mathcal{I}|} f(x)}{\partial x_{i_1} \partial x_{i_2} \ldots \partial x_{i_{|\mathcal{I}|}}} \right]^2 > 0. \tag{2}$$

We may now adjust the GAM definition to capture feature interactions by considering a set of T specified interactions, $\{\mathcal{I}_t\}_{t=1}^T$, where each $\mathcal{I}_t \subseteq [d]$ is an interaction of size $|\mathcal{I}_t|$:

$$g(y) = f_\emptyset + \sum_i f_i(x_i) + \sum_t f_{\mathcal{I}_t}(x_{\mathcal{I}_t}) \tag{3}$$

The third term extends GAMs to full capacity models which can represent nonlinear dependencies of arbitrary feature sets. For instance, if our set of interactions $\{\mathcal{I}_t\}_{t=1}^T$ includes the complete feature set $[d]$, then our model has exactly the same capacity as the underlying functional model we choose for the shape functions (splines, random forests, deep neural networks, etc.) An abundance of previous works have used this extended version of the GAM model, often called the Smoothing Spline ANOVA model or the functional ANOVA model [7, 36, 29, 18, 46].

Throughout this work, we will use GAM-K to refer to a GAM whose highest order interaction in $\{\mathcal{I}_t\}_{t=1}^T$ is of cardinality K. (i.e. $|\mathcal{I}_t| \leq K \; \forall t \in [T]$.) For instance, we will call the NODE-GA^{2}M model [8] a GAM-2 model since the interaction sets are all possible feature pairs: $\{\{i,j\} : i < j \text{ for } i,j \in [d]\}$. We will refer to our SIAN networks as SIAN-K using the same convention.

3.2 Feature Interaction Selection

A key concern of using neural networks to fit the shape functions is keeping the number of networks low enough that our training time computation is kept reasonable. While this is typically not a problem for the GAM-1, this can quickly become an issue for the GAM-2. For instance, if we have an input variable x with 30 features, then including all pairwise functions would need to cover the $\binom{30}{2} = 435$ possible pairs. Although learning 465 linear coefficients is reasonable, training hundreds of neural networks becomes less so. Moreover, this quantity only grows exponentially as we increase K to higher-order interactions (3, 4, 5, etc.) In an effort to combat this growing complexity, we introduce a Feature Interaction Selection (FIS) algorithm which depends on two key ingredients: an interaction detection procedure and a heredity condition.

Feature Interaction Detection In recent years, there has been a growing body of work focused on detecting and measuring the strength of feature interactions from large-scale data. Three of the most popular and generally applicable of these methods are the Shapley Additive Explanations (SHAP) [28, 26, 13], Integrated Hessians [40, 19], and Archipelago [42]. For our experiments, we primarily use an adaptation of Archipelago for higher-order interactions because of its compatibility with blackbox models. In contrast, Integrated Hessians is only applicable to sufficiently smooth networks (ReLU networks are not compatible) and SHAP only has fast implementations available for tree-based approaches. Moreover, although both SHAP and Integrated Hessians have clear ideological extensions to higher-order interactions, there are no currently available implementations. A detailed discussion surrounding Archipelago and other detection techniques can be found in Appendix C.

Heredity The practice of only modeling the pairwise interaction effect $\{i,j\} \subseteq [d]$ for some features $i,j \in [d]$ when both of the main effects $\{i\}$ and $\{j\}$ are already being modeled has a long history throughout statistics [34, 11, 5]. There are two main versions of this hierarchical principle explored in the literature: strong heredity and weak heredity. If we are given that $\omega(\{i,j\}) > 0$, strong heredity implies that both $\omega(\{i\}) > 0$ and $\omega(\{j\}) > 0$ whereas weak heredity implies that

$\omega(\{i\}) > 0$ or $\omega(\{j\}) > 0$. For our definition of a feature interaction, we have that strong heredity holds; however, our algorithm will instead focus on a computational version of heredity which asks that τ percent of subsets are above threshold θ in order for a pair (triple, tuple, etc.) to be considered as a possible interaction.

Algorithm In Algorithm 1, we detail how we use Archipelago to build our FIS algorithm. The visual overview of the SIAN pipeline is also depicted in Figure 1 above. We start by training a reference DNN which is required for the inductive insights generated by Archipelago. We then pass the trained function approximator to the FIS algorithm along with hyperparameters K (interpretability index), τ (heredity strength threshold), and θ (interaction strength threshold). This procedure efficiently searches and ranks the 2^d possible feature subsets to produce a final set of interactions which we use to specify the SIAN model architecture. Finally, we train our SIAN neural network using typical gradient descent methods. In Appendix A, we provide a further theoretical discussion in which we prove exact recovery of the true feature interactions and show how this sparse selection leads to provably lower generalization gaps in a toy additive noise model.

Algorithm 1 Feature Interaction Selection (FIS)

Inputs: Trained prediction model $f(x)$, and validation dataset $X^* = \{x^{(1)}, \ldots, x^{(V)}\}$
Parameters: Cutoff index K, cutoff threshold τ, strength threshold θ
Output: $\mathcal{I}$, a family of feature interactions with index at most K and strength above θ

Set $\mathcal{I} \leftarrow \emptyset$
Set $\mathcal{J} \leftarrow \{\{i\} : i \in [d]\}$
$k \leftarrow 1$
while $k \leq K$ **do**
  for J in $\mathcal{J}$ **do**
    $\omega_J(X^*) \leftarrow 0$
    for $x^{(v)} \in X^*$ **do**
      $\omega_J(x^{(v)}) \leftarrow$ Archipelago$(f, x^{(v)}, J)$
      $\omega_J(X^*) \leftarrow \omega_J(X^*) + \omega_J(x^{(v)})$
    end for
    $\omega_J(X^*) \leftarrow \omega_J(X^*) \cdot \frac{1}{|X^*|}$
    if $\omega_J(X^*) > \theta$ **then**
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{J\}$
    end if
  end for
  $\mathcal{J}' \leftarrow \{J : J \in \mathcal{P}([d]); |J| = k+1\}$
  for J in $\mathcal{J}'$ **do**
    $\nu(J) \leftarrow \frac{1}{|J|} \sum_{I \subseteq J} [1_{I \in \mathcal{I}} \cdot 1_{\{|J|-|I|=1\}}]$
  end for
  $\mathcal{J} \leftarrow \{J : J \in \mathcal{J}'; \nu(J) > \tau\}$
  $k \leftarrow k+1$
end while
return $\mathcal{I}$

3.3 Block Sparse Implementation

In order to improve the time and memory efficiency of the SIAN model, we implement a block sparse construction for neural additive models. The default scheme of SIAN is to use a network module for each of the shape functions and additively combine the output features, following the implementation strategies of NAM and other popular neural additive models. However, since each shape function network is computed sequentially, this greatly bottlenecks the computation speed of the model. We instead construct a large, block sparse network which computes the hidden representations of all shape function networks simultaneously, leveraging the fact that each shape subnetwork has the same depth. Using this block sparse network allows for shape features to be computed in parallel, leading to a significant improvement in training speed. The main consequence of this design is a higher footprint in memory; therefore, we also develop a compressed sparse matrix implementation of the network which has a greatly reduced memory footprint for saving network parameters. SIAN is able to interchange between these different modes with minimal overhead, allowing for faster training in the block sparse module, lower memory footprint in the compressed module, and convenient visualization of shape functions in the default module. We provide further numerical details of our gains in Section 5.1.

4 Datasets

Our experiments focus on seven machine learning datasets. Two are in the classification setting, the MIMIC-III Healthcare and the Higgs datasets [20, 3]. The other five are in the regression setting, namely the Appliances Energy, Bike Sharing, California Housing Prices, Wine Quality, and Song Year datasets [6, 14, 22, 12, 4]. More details about each dataset are provided in Table 1. We evaluate the regression datasets using mean-squared error (MSE). We measure the performance on the classification datasets using both the area under the receiver operating characteristic (AUROC) and

the area under the precision-recall curve (AUPRC) metrics. We report both metrics for the MIMIC dataset since the positive class is only 9% of examples and report only AUROC for the Higgs dataset since it is relatively well-balanced.

4.1 Experiment Details

For the baseline DNNs we are using hidden layer sizes [256,128,64] with ReLU activations. For the GAM subnetworks we are using hidden layer sizes [16,12,8] with ReLU activations. We use L1 regularization of size 5e–5. In the main results section, we report the results for each $K \in \{1, 2, 3, 5\}$ using only a single value of τ and θ. The hyperparameter τ was taken to be 0.5 throughout and θ was selected from a handful of potential values using a validation set. We train all networks using Adagrad with a learning rate of 5e–3. All models are evaluated on a held-out test dataset over five folds of training-validation split unless three folds are specified. Three folds are used for NODE-GAM on all datasets as well as Song Year and Higgs for all models. We respect previous testing splits when applicable, otherwise we subdivide the data using an 80-20 split to generate a testing set. In addition to NODE-GA2M, we compare against the interpretable models LASSO and GA^{2}M EBM as well as the popular machine learning models of support vector machines (SVM), random forests (RF), and extreme gradient boosting (XGB) [10, 30].

Table 1: Real-world datasets. n is the number of samples, d is the number of features, and p is the percentage of data samples with the positive class label.

Dataset	n	d	p
Higgs Boson	$11,000,000$	28	53.0%
MIMIC-III	$32,254$	30	9.2%
Energy Appliances	$19,735$	30	–
Bike Sharing	$17,379$	13	–
California Housing	$20,640$	8	–
Wine Quality	$6,497$	12	–
Song Year	$515,345$	90	–

Table 2: MIMIC-III Performance.

Model	AUROC (⇑)	AUPRC (⇑)
SAPS II	0.792	0.281
SOFA	0.703	0.225
LASSO	0.568	0.396
GA^{2}M EBM	0.840	0.375
NODE-GA^{2}M	0.826	0.345
SIAN-1	0.848	0.409
SIAN-2	**0.855**	**0.423**
SIAN-3	**0.856**	**0.425**
SIAN-5	**0.856**	**0.425**
RF	0.821	0.369
SVM	0.831	0.404
XGB	0.843	0.382
DNN	0.844	0.382

5 Results

Across seven different datasets, SIAN achieves an average rank of 3.00 out of the 8 models we consider. The next best performing model, NODE-GA2M, has an average rank of 3.71 out of 8. The third best performing model, DNN, has an average rank of 3.86 out of 8. We find that SIAN achieves consistent performance by being able to adapt to both the low-dimensional datasets and the high-dimensional datasets, finding a balance between good training fit and good generalization.

Table 3: Test metrics for six of seven datasets. (⇑)/(⇓) indicates higher/lower is better, respectively.

Model	Appliances Energy (⇓)	Bike Sharing (⇓)	California Housing (⇓)	Wine Quality (⇓)	Song Year (⇓)	Higgs Boson (⇑)
LASSO	0.740±0.002	1.053±0.001	0.478±0.000	0.575±0.002	1.000±0.008	0.635±0.000
GA^{2}M EBM	1.053±0.138	0.124±0.004	0.265±0.002	0.498±0.004	0.894±0.001	0.698±0.001
NODE-GA^{2}M	1.064±0.056	**0.111±0.006**	**0.222±0.005**	0.521±0.009	0.806±0.001	0.811±0.000
SIAN-1	**0.718±0.007**	0.387±0.035	0.378±0.007	0.551±0.004	0.860±0.001	0.771±0.001
SIAN-2	0.763±0.009	0.127±0.008	0.302±0.002	0.497±0.003	0.842±0.002	0.795±0.001
SIAN-3	0.808±0.026	0.125±0.013	0.278±0.001	0.497±0.003	0.831±0.001	0.798±0.001
SIAN-5	0.801±0.031	0.149±0.011	0.272±0.003	0.484±0.006	0.821±0.001	0.802±0.001
RF	1.114±0.095	0.206±0.009	0.271±0.001	**0.439±0.005**	0.994±0.005	0.654±0.002
SVM	0.740±0.008	0.168±0.001	0.262±0.001	0.457±0.008	0.940±0.012	0.698±0.001
XGB	1.188±0.119	0.157±0.003	0.229±0.002	0.465±0.014	0.881±0.002	0.740±0.000
DNN	0.945±0.054	0.374±0.017	0.283±0.005	0.495±0.007	**0.791±0.002**	**0.823±0.000**

For the MIMIC dataset, we can see the models' performances in terms of both AUROC and AUPRC in Table 2. In addition to the models we previously described, we also compare against the interpretable medical baselines of SOFA and SAPS II which are simple logic-based scoring methods [23, 45]. We see that the machine learning methods improve over the baseline performance achieved by the SAPS and SOFA methods.

In Table 3, we can see the combined results for our six other datasets. First, in the Appliances Energy dataset, we see that the SIAN-1 performs the best, with LASSO and SVM trailing slightly behind. The success of one-dimensional methods on this dataset could be indicative that many of the dataset's trends are one-dimensional. As we increase the dimension of the SIAN, the test error slowly increases as the generalization gap grows; the full-complexity DNN has even worse error than all SIAN models.

Second, in the Bike Sharing dataset, we see that the best performing model is the NODE-GA2M, with the EBM, SIAN-2, and SIAN-3 trailing only slightly behind. All of these methods are two or three dimensional GAM models, again hinting that a significant portion of the important trends in this dataset could be bivariate. Indeed, the most important bivariate trend accounts for more than 50% of the variance in the dataset, as we explore in Figure 2 below.

For the remaining datasets, we see that the performance of the SIAN model improves as we add higher and higher-order feature interactions. For the California Housing dataset, we find the best performance using the differentiable tree method of NODE-GA2M. For the Wine Quality dataset, we find the best performance using the random forest algorithm. For the larger-scale datasets of Song Year and Higgs Boson, we find that the best performance is still obtained by a full-complexity deep neural network. These two are the only datasets where SIANs of order five or less are not sufficient to outperform vanilla deep neural networks, implying there are important feature interactions of degree greater than five.

5.1 Training Speed and Storage

In Table 4 below we see how our SIAN network compares against other popular differentiable additive models in both training time and size on disk. For fair comparison we do not utilize our interaction selection algorithm for SIAN in this section, instead training SIAN-2 with all possible pairs. We see that our implementation of different modes for the SIAN architecture allows us to outperform both NAM and NODE-GAM, with 2-8x faster training on GAM-1 models and 10-80x faster training on GAM-2 models. We reiterate that because the overhead for switching between modes is negligible, SIAN enjoys the benefits of all modes: training quickly in the block sparse mode and saving succinctly in the compressed mode.

Table 4: Training Time and Memory Size. Experiments are run with a machine using a GTX 1080 GPU and 16GB of RAM. The average wall-clock time over three runs is reported.

		SIAN-2 (default)	(block-sparse)	(compressed)	**NODE-GA2M**	**SIAN-1** (default)	(block-sparse)	(compressed)	**NAM**
Training Time (minutes)	Wine	17.57	**0.69**	–	55.19	3.03	**0.66**	–	5.45
	Bike	159.65	**5.38**	–	60.75	25.65	**4.58**	–	12.09
	House	88.67	**6.09**	–	58.03	22.72	**5.88**	–	14.74
Size on Disk (KB)	Wine	7,113	7,113	**537**	1,040	182	182	**86**	1,203
	Bike	9,727	9,666	**626**	1,040	218	212	**92**	308
	House	1,526	1,526	**251**	1,040	83	83	**59**	1,921

5.2 Beyond ReLU Networks

The FIS algorithm we describe can be applied to other combinations of FID algorithm + functional model besides Archipelago + ReLU Neural Networks. To demonstrate the general applicability of our scheme, we replace the continuous, piecewise-linear functions of ReLU neural networks with the piecewise-constant, differentiable decision trees using NODE-GAM. We extend the original implementation of NODE-GAM to handle feature triplets, extending the method to a trivariate function or GAM-3 model. We run our FIS algorithm using $K = 3$ on both the Housing and Wine datasets to fit GAM-3 models using the inductive biases of the NODE architecture.

Extending from NODE-GA2M to NODE-GA3M is able to improve performance from 0.222 to 0.175 on the Housing dataset, a 21.2% further improvement over the state of the art method. The same extension is only able to deliver a 4.8% improvement from 0.521 to 0.496 on the Wine dataset; however, both the SIAN-5 and NODE-GA3M are able to improve performance over all previously available additive models. These two examples demonstrate the ability of our FIS pipeline to be applied to more general machine learning techniques to model higher-order interactions.

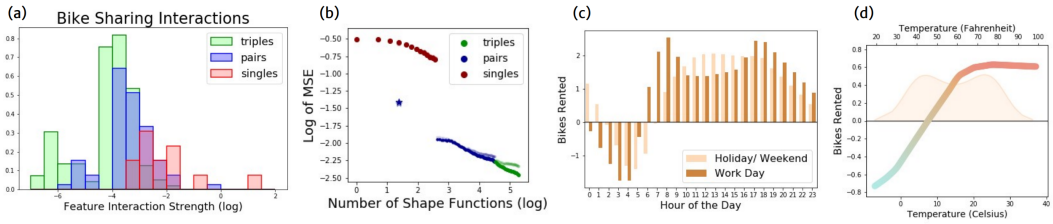

Figure 2: Analysis of Bike Sharing. (a) We plot the histogram of feature interaction strength, adjusting the vertical scaling by degree to display in a single plot. (b) We plot the training and validation performance (darker and lighter, respectively) as we add feature interactions to SIAN. We use an exponentially weighted moving average to reduce variance instead of training multiple networks of each size. The blue star is the performance of the shape functions in 2c and 2d. (c) The most important shape function showing the interaction between hour of the day and work day. (d) The next shape function showing temperature's effect on bikes rented.

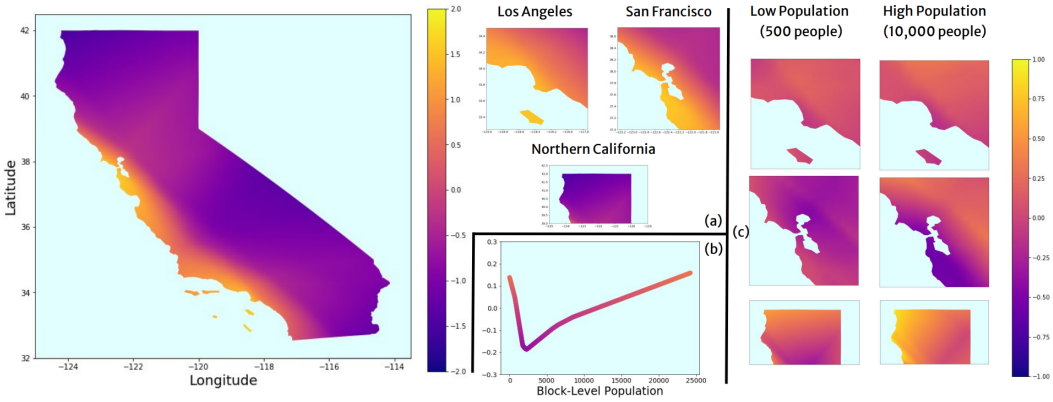

Figure 3: 3D feature interaction for California Housing. (a) The effect of the location only for the entire state of California and zoomed in to three selected locations. (b) The effect of only the block-level population. (c) The three-dimensional interaction effect of block-level population with location. We plot the effect for both a low and high population to highlight the differences in effect for each location. Note the different scales per panel.

6 Discussion

In this section, we further explore the feature interactions learned by SIAN and FIS across multiple datasets. We provide multiple visualizations of the shape functions learned by SIAN to get a sense of the diverse analysis and interpretations which are made possible by additive models. Further discussion and graphics are provided in Appendix C and D.

In Figure 2a, we visualize the interaction strength measured for each of the singles, pairs, and triples of features from the Bike Sharing dataset. The four top rated interactions (three of which are visually separated from the main body of the histogram) are, from right to left, ["hour", "hour X workday", "workday", "temperature"]. In 2b we see the effect of gradually adding feature interactions to our model. We see there is a steep jump in performance when we are able to model the first pair, depicted in 2c. Together with 2d, these two visualizations alone explain 84% of the variance in the Bike Sharing dataset. Importantly, these two trends are interpretable and agree with our intuition about when people are more likely to bike: on work days, we see peak spikes at 8 a.m. and 5 p.m., corresponding to the beginning and end of work hours; on weekends and holidays, there is a steady demand of bikes throughout the afternoon. There are also more bikers during warmer temperatures.

In Figure 3, we set out to visualize one of the three dimensional interactions which occurs between the three features of latitude, longitude, and population. In 3a, we see the rise in housing price along the coast of California, especially around the metropolitan areas of Los Angeles and San Francisco. In 3b, we see that house price does not monotonically increase with population as we might expect for higher population densities. A detailed look into the dataset reveals that the 'population' feature being used is accumulated at the census block level, creating an inverse relationship with population density as areas like LA and SF are subdivided more than their suburban and rural counterparts. Although it is

possible location and population density might have independent effects on housing price, the dataset nevertheless induces an interaction between location and population. In 3c, we attempt to visualize this 3D interaction by viewing subsamples across two population sizes and over three regions. We see the network has learned to differentiate the urban regions of LA and SF from the rural coast of northern California, where high population becomes indicative of higher population densities and higher housing prices. We note that the trends learned by SIAN tends to be very continuous, which is a potential shortcoming in learning fine-grained block-level information in cities like Los Angeles and San Francisco. In light of these concerns, we run experiments on a three-dimensional extension of the piecewise constant, discontinuous NODE-GAM model in section 5.2.

In Figure 4, we see two example trends from the SIAN trained to predict mortality given hospital data. On the left, we see that health risk gradually increases with age. On the right, we see the trend with respect to the Glasgow Coma Scale indicator which is a measure of alertness and consciousness. The lowest and highest scores correspond to severe head injury and full alertness, respectively. We see that very low GCS corresponds to high risk and that the risk decreases as GCS increases. At the score of 13, however, we see that an increase to 14 or 15 actually increases mortality risk, defying the intuition that risk should be monotone in the GCS score. It is highly likely that this dip in mortality risk comes from the special care given to patients with GCS scores below 15, compared to their counterparts with perfect scores.

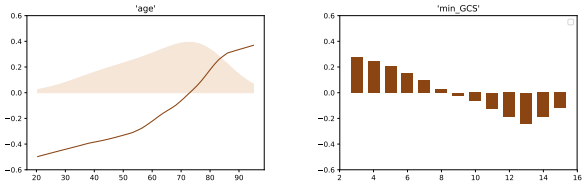

Figure 4: MIMIC-III. Age Risk and GCS Risk

This trend illustrates a phenomenon occurring throughout machine learning applications in healthcare where correlation and causation are conflated with one another. Similar results have been found linking asthma to a decreased risk of death from pneumonia [7]. These issues might go unnoticed in black-box models whereas interpretable models can uncover and resolve such discrepancies before deployment. While discovering and correcting interpretable trends brings clear advantages, it is also possible for these trends to be misinterpreted by non-experts as a causal relationship. Such false causal discoveries can not only lead to physical harm in the domain of medicine, but also to larger social harm in broader AI systems and applications.

7 Limitations and Future Work

A primary limitation of the current work is its focus on multilayer perceptrons whereas modern state-of-the-art results are dominated by geometric deep learning and transformer architectures. Extending this procedure to more general architectures is a key direction for bringing interpretability to domains like computer vision and natural language. Such domains can further customize the FIS algorithm to respect specific structures like spatial locality and knowledge graph semantics.

Another important direction of research includes a better theoretical understanding how the benefits of SIAN scale with the dimensionality of the dataset and the number of samples provided, providing yet another lens to study the implicit biases of deep neural networks. Further, developing a theory which accurately models the empirically observed distributions of feature interactions in real-world data, especially in the presence of heteroscedastic noise and correlated features, would be of great interest for this direction.

Multiple experiments confirm that SIAN can produce powerful and interpretable machine learning models which match the performance of state-of-the-art benchmarks like deep neural networks and NODE-GA2M. Further experiments show that FIS can be applied to more general machine learning algorithms. Hopefully, future work will be able to further clarify this sparse interaction perspective and help deepen our understanding of the generalization performance of neural networks.

References

[1] A. Abdul, C. von der Weth, M. Kankanhalli, and B. Y. Lim. Cogam: Measuring and moderating cognitive load in machine learning model explanations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery.

[2] R. Agarwal, N. Frosst, X. Zhang, R. Caruana, and G. E. Hinton. Neural additive models: Interpretable machine learning with neural nets. *CoRR*, abs/2004.13912, 2020.

[3] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 07 2014.

[4] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[5] J. Bien, J. Taylor, and R. Tibshirani. A lasso for hierarchical interactions. *Annals of statistics*, 41(3):1111, 2013.

[6] L. M. Candanedo, V. Feldheim, and D. Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings*, 140:81–97, 2017.

[7] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730. ACM, 2015.

[8] C.-H. Chang, R. Caruana, and A. Goldenberg. NODE-GAM: Neural generalized additive model for interpretable deep learning. In *International Conference on Learning Representations*, 2022.

[9] C.-H. Chang, S. Tan, B. Lengerich, A. Goldenberg, and R. Caruana. How interpretable and trustworthy are gams? In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 95–105, New York, NY, USA, 2021. Association for Computing Machinery.

[10] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[11] H. A. Chipman. Bayesian variable selection with related predictors. *Canadian Journal of Statistics-revue Canadienne De Statistique*, 24:17–36, 1995.

[12] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47:547–553, 11 2009.

[13] K. Dhamdhere, A. Agarwal, and M. Sundararajan. The shapley taylor interaction index. *arXiv preprint arXiv:1902.05622*, 2019.

[14] H. Fanaee-T and J. Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pages 1–15, 2013.

[15] J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, pages 916–954, 2008.

[16] T. J. Hastie and R. J. Tibshirani. Generalized additive models, 1990.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[18] G. Hooker. Generalized functional anova diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics*, 16(3):709–732, 2007.

[19] J. D. Janizek, P. Sturmfels, and S.-I. Lee. Explaining explanations: Axiomatic feature interactions for deep networks. *arXiv preprint arXiv:2002.04138*, 2020.

[20] A. E. Johnson, T. J. Pollard, L. Shen, L. wei H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3(160035), 2016.

[21] K. Kandasamy and Y. Yu. Additive approximations in high dimensional nonparametric regression via the salsa. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 69–78, New York, New York, USA, 20–22 Jun 2016. PMLR.

[22] R. Kelley Pace and R. Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

[23] J.-R. Le Gall, S. Lemeshow, and F. Saulnier. A new simplified acute physiology score (saps ii) based on a european/north american multicenter study. *JAMA*, 270(24), 1993.

[24] Y. Lin and H. H. Zhang. Component selection and smoothing in multivariate nonparametric regression. *The Annals of Statistics*, 34(5):2272–2297, 2006.

[25] G. Liu, H. Chen, and H. Huang. Sparse shrunk additive models. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6194–6204. PMLR, 13–18 Jul 2020.

[26] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):2522–5839, 2020.

[27] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From local explanations to global understanding with explainable ai for trees, 2020.

[28] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[29] L. Meier, S. van de Geer, and P. Bühlmann. High-dimensional additive modeling. *The Annals of Statistics*, 37(6B):3779 – 3821, 2009.

[30] H. Nori, S. Jenkins, P. Koch, and R. Caruana. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

[31] R. O'Donnell. Some topics in analysis of boolean functions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 569–578, New York, NY, USA, 2008. Association for Computing Machinery.

[32] R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[33] L. O'Neill, S. Angus, S. Borgohain, N. Chmait, and D. L. Dowe. Creating powerful and interpretable models with regression networks, 2021.

[34] J. L. Peixoto. Hierarchical variable selection in polynomial regression models. *The American Statistician*, 41(4):311–313, 1987.

[35] S. Raskhodnikova and G. Yaroslavtsev. Learning pseudo-boolean k -dnf and submodular functions. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 08 2012.

[36] P. Ravikumar, J. Lafferty, H. Liu, and L. Wasserman. Sparse additive models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 71(5):1009–1030, 2009.

[37] D. Silver and et al. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[38] D. Sorokina, R. Caruana, M. Riedewald, and D. Fink. Detecting statistical interactions with additive groves of trees. In *Proceedings of the 25th international conference on Machine learning*, pages 1000–1007. ACM, 2008.

[39] P. Stobbe and A. Krause. Learning fourier sparse set functions. In N. D. Lawrence and M. Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 1125–1133, La Palma, Canary Islands, 21–23 Apr 2012. PMLR.

[40] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

[41] M. Tsang, H. Liu, S. Purushotham, P. Murali, and Y. Liu. Neural interaction transparency (nit): Disentangling learned interactions for improved interpretability. In *Advances in Neural Information Processing Systems*, pages 5804–5813, 2018.

[42] M. Tsang, S. Rambhatla, and Y. Liu. How does this interaction affect me? interpretable attribution for feature interactions. *arXiv preprint arXiv:2006.10965*, 2020.

[43] H. Tyagi, A. Kyrillidis, B. Gärtner, and A. Krause. Learning sparse additive models with interactions in high dimensions. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 111–120, Cadiz, Spain, 09–11 May 2016. PMLR.

[44] A. Vaswani and et al. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[45] J. L. Vincent, R. Moreno, J. Takala, S. Willatts, A. D. Mendonça, H. Bruining, C. K. Reinhart, P. M. Suter, and L. G. Thijs. The sofa (sepsis-related organ failure assessment) score to describe organ dysfunction/failure. *Intensive Care Med.*, 22(7):707–710, 1997.

[46] G. Wahba, Y. Wang, C. Gu, R. Kleins, and B. Kle. Smoothing spline anova for exponential families. In *The Annals of Statistics*, 1994.

[47] T. Wang, J. Yang, Y. Li, and B. Wang. Partially interpretable estimators (pie): Black-box-refined interpretable machine learning, 2021.

[48] S. Xu, Z. Bu, P. Chaudhari, and I. J. Barnett. Sparse neural additive model: Interpretable deep learning with feature selection via group sparsity, 2022.

[49] Z. Yang, A. Zhang, and A. Sudjianto. Gami-net: An explainable neural network based on generalized additive models with structured interactions, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Theoretical Discussion

A.1 Introduction with pseudo-boolean function

Let us first discuss a pseudo-boolean function of the form $f : \{-1, 1\}^d \to \mathbb{R}$. Every such function has a unique decomposition as a multilinear polynomial:

$$f(x) = \sum_{I \subseteq [d]} c_I x_I$$

where x_I is the monomial $x_I = \prod_{i \in I} x_i$. This multilinear decomposition is usually referred to as the Fourier expansion of the function f and the c_I are called the Fourier coefficients. Instead of estimating the 2^d functional values, we can instead estimate the 2^d Fourier coefficients. Although there is no immediate benefit to this shift in perspective, an abundance of practical assumptions like submodularity, disjunctive normal forms, low-order approximates, and decaying Fourier spectrum can be made to imply that only a sparse subset of the full 2^d coefficients are worth estimating. A further discussion of these assumptions in the context of boolean functions can be found in [31, 39, 35, 32].

For our context, it suffices to say that only $s << 2^d$ of the full c_I coefficients are nonzero. Our problem hence becomes a sparse selection problem in the space of Fourier coefficients, which can be equivalently thought of as a Feature Interaction Selection (FIS) problem in the functional space of pseudo-boolean functions.

Lemma 1. *Given exact functional values, Archipelago recovers the upper set of Fourier coefficients. Specifically, given a subset $A \subseteq [d]$ and defining:*

$$\mathcal{A}_A(f) := \mathbb{E}_{x,y}\left[\left(\sum_{C \subseteq A} (-1)^{|C|-|A|} f(x_C + y_{C'})\right)^2\right] \cdot \frac{1}{2^{|A|}}$$

We have that the upper cone set of A is recovered by the Archipelago measure:

$$\mathcal{A}_A(f) = \sum_{I \supseteq A} |c_I|^2$$

Proof.

$$\begin{aligned}
&\mathbb{E}_{x,y}\left[\left(\sum_{C \subseteq A} (-1)^{|C|-|A|} f(x_C + y_{C'})\right)^2\right] \\
&= \mathbb{E}_{x,y}\left[\sum_{C_1 \subseteq A} \sum_{C_2 \subseteq A} (-1)^{|C_1|-|A|} (-1)^{|C_2|-|A|} f(x_{C_1} + y_{C_1'}) f(x_{C_2} + y_{C_2'})\right] \\
&= \mathbb{E}_{x,y}\left[\sum_{C_1 \subseteq A} \sum_{C_2 \subseteq A} (-1)^{|C_1|+|C_2|} \sum_{I_1 \subseteq [d]} \sum_{I_2 \subseteq [d]} c_{I_1} c_{I_2} \cdot (x_{C_1} + y_{C_1'})_{I_1} \cdot (x_{C_2} + y_{C_2'})_{I_2}\right] \\
&= \mathbb{E}_{x,y}\left[\sum_{C_1 \subseteq A} \sum_{C_2 \subseteq A} (-1)^{|C_1|+|C_2|} \sum_{J_1 \subseteq A} \sum_{J_2 \subseteq A} \sum_{K_1 \subseteq A'} \sum_{K_2 \subseteq A'} c_{I_1} c_{I_2} \cdot (x_{C_1} + y_{C_1'})_{I_1} \cdot (x_{C_2} + y_{C_2'})_{I_2}\right]
\end{aligned}$$

where $I_1 = J_1 \cup K_1$ and $I_2 = J_2 \cup K_2$. We also write $B_1 = A - C_1$ and $B_2 = A - C_2$. Continuing:

$$\begin{aligned}
= &\sum_{K_1 \subseteq A'} \sum_{K_2 \subseteq A'} \mathbb{E}_{x,y}\left[\sum_{C_1 \subseteq A} \sum_{C_2 \subseteq A} \sum_{J_1 \subseteq A} \sum_{J_2 \subseteq A} (-1)^{|C_1|+|C_2|} c_{I_1} c_{I_2} \cdot (x_{C_1} + y_{B_1} + y_{A'})_{I_1} \cdot (x_{C_2} + y_{B_2} + y_{A'})_{I_2}\right] \\
= &\sum_{K \subseteq A'} \mathbb{E}_{x,y}\left[\sum_{C_1 \subseteq A} \sum_{C_2 \subseteq A} \sum_{J_1 \subseteq A} \sum_{J_2 \subseteq A} (-1)^{|C_1|+|C_2|} c_{I_1} c_{I_2} \cdot (x_{C_1} + y_{B_1} + y_{A'})_{I_1} \cdot (x_{C_2} + y_{B_2} + y_{A'})_{I_2}\right] \\
+ &\sum_{K_1 \neq K_2 \subseteq A'} \mathbb{E}_{x,y}\left[\sum_{C_1 \subseteq A} \sum_{C_2 \subseteq A} \sum_{J_1 \subseteq A} \sum_{J_2 \subseteq A} (-1)^{|C_1|+|C_2|} c_{I_1} c_{I_2} \cdot (x_{C_1} + y_{B_1} + y_{A'})_{I_1} \cdot (x_{C_2} + y_{B_2} + y_{A'})_{I_2}\right]
\end{aligned}$$

We know that the second summand will have every term equal to zero. Without loss of generality, let k_1 belong to K_1 but not K_2 which we can always find such an element since the two sets are not equal. We can then factor out an independent term of y_{k_1} and take the expectation over these independent quantities:

$$\sum_{K_1 \neq K_2 \subseteq A'} \mathbb{E}_{x,y}\left[\sum_{C_1 \subseteq A} \sum_{C_2 \subseteq A} \sum_{J_1 \subseteq A} \sum_{J_2 \subseteq A} (-1)^{|C_1|+|C_2|} c_{I_1} c_{I_2} \cdot (x_{C_1} + y_{B_1} + y_{A'})_{I_1} \cdot (x_{C_2} + y_{B_2} + y_{A'})_{I_2}\right] =$$

$$\sum_{K_1 \neq K_2 \subseteq A'} \mathbb{E}_{y_{k_1}}[y_{k_1}] \cdot \mathbb{E}_{x,y-y_{k_1}}\Big[\dots\Big] = \sum_{K_1 \neq K_2 \subseteq A'} 0 \cdot \mathbb{E}_{x,y-y_{k_1}}\Big[\dots\Big] = 0.$$

Hence, we can now focus on a given $K \subseteq A'$ and consider what happens to:

$$\mathbb{E}_{x,y}\Big[\sum_{C_1 \subseteq A}\sum_{C_2 \subseteq A}\sum_{J_1 \subseteq A}\sum_{J_2 \subseteq A}(-1)^{|C_1|+|C_2|} c_{(J_1+K)} c_{(J_2+K)} \cdot (x_{C_1} + y_{B_1})_{J_1} \cdot (x_{C_2} + y_{B_2})_{J_2}\Big]$$

Following a similar argument, we can partition into two summands consisting of $\{J_1 = J_2\}$ and $\{J_1 \neq J_2\}$. Without loss of generality, pick one such j_1 in J_1 but not in J_2. For each choice of C_1, C_2, there is a choice of either x or y which can be factored out from one term and not the other, by assumption there is no $j_1 \in J_2$. Without loss of generality, let it be x.

$$\sum_{J_1 \neq J_2 \subseteq A} \mathbb{E}_{x,y}\Big[\sum_{C_1 \subseteq A}\sum_{C_2 \subseteq A}(-1)^{|C_1|+|C_2|} c_{I_1} c_{I_2} \cdot (x_{C_1} + y_{B_1})_{J_1} \cdot (x_{C_2} + y_{B_2})_{J_2}\Big] =$$

$$\sum_{J_1 \neq J_2 \subseteq A} \mathbb{E}_{x_{j_1}}[x_{j_1}] \cdot \mathbb{E}_{x-x_{j_1},y}\Big[\dots\Big] = \sum_{J_1 \neq J_2 \subseteq A} 0 \cdot \mathbb{E}_{x-x_{j_1},y}\Big[\dots\Big] = 0$$

Now, we are focusing on only a specific J and K, giving:

$$c^2_{(J+K)} \cdot \sum_{C_1 \subseteq A}\sum_{C_2 \subseteq A} \mathbb{E}_{x,y}\Big[(-1)^{|C_1|+|C_2|}(x_{C_1} + y_{B_1})_J \cdot (x_{C_2} + y_{B_2})_J\Big]$$

Writing the symmetric difference as $(C_1 \Delta C_2) := (C_1 - C_2) \cap (C_2 - C_1)$, our quantity is equal to:

$$c^2_{(J+K)} \cdot \sum_{C_1 \subseteq A}\sum_{C_2 \subseteq A}(-1)^{|C_1|+|C_2|} \cdot \mathbb{E}_{x,y}\Big[\prod_{j \in (J \cap (C_1 \Delta C_2))} x_j y_j\Big]$$

This expectation will be equal to zero if the product is nonempty, following the same logic as before by taking the expectation over one of the independent x_j or y_j. Otherwise, the expectation will be equal to one. Hence, it is only equal to one for the terms in which $(J \cap (C_1 \Delta C_2)) = \emptyset$ and our quantity of interest is equal to:

$$c^2_{(J+K)} \cdot \sum_{C_1, C_2 \subseteq A : (C_1 \Delta C_2) \subseteq (A-J)} (-1)^{|C_1|+|C_2|}$$

This sum will, however, be equal to zero unless $J = A$. Suppose instead that there is an $a \in A - J$. Consider the mapping from pairs (C_1, C_2) where $a \notin C_1$ to the pairs which have $a \in C_1$. These sets clearly partition $\{(C_1, C_2) \subseteq A \times A : (C_1 \Delta C_2) \subseteq (A - J)\}$ and the mapping is bijective inside the subsets. Hence, the two sets are of the same size and with odd parity of $(-1)^{|C_1|+|C_2|}$. Together, this implies the sum above will be equal to zero for such cases. In the alternate case that $J = A$, we know that $(C_1 \Delta C_2) = \emptyset$ or $C_1 = C_2$, meaning we instead consider $c^2_{(J+K)} \cdot \sum_{C \subseteq A} 1 = c^2_{(J+K)} \cdot 2^{|A|}$.

Altogether, our original quantity of interest has now become:

$$\sum_{K \subseteq A'}\sum_{J=A} c^2_{(J+K)} \cdot 2^{|A|} = 2^{|A|} \cdot \sum_{K \subseteq A'} c^2_{(A+K)} = 2^{|A|} \cdot \sum_{I \supseteq A} c^2_I$$

□

Proposition 1. *The FIS Algorithm 1 recovers the downwards closure of the true nonzero entries of the function. Let $S \subseteq \mathcal{P}([d])$ be the set of nonzero Fourier coefficients of size $|S| = s$. Define the closure or downwards closure of S as $\bar{S} := \{I \subseteq [d] : I \subseteq J \text{ for some } J \in S\}$. Algorithm 1 returns exactly $\bar{S}$.*

Proof. Using the lemma above, we know that the measurement on the psuedo-boolean function will be the upper cone. From this it fairly simply follows that any member of the downwards closure of S will have positive upper cone size and will be added to the candidate set of interactions, whereas nonmembers of the downwards closure will all have zero upper cone size and will not be added. □

A.2 Decomposition of a general function

Given a more general space $\mathcal{X} \subseteq \mathbb{R}^d$ and a function $f : \mathcal{X} \to \mathbb{R}$ we can define another decomposition similar to the Fourier decomposition from before. Given some fixed distribution on our input space $x \sim \mathcal{P}(\mathcal{X})$, we can define the conditional expectation functions as:

$$\tilde{f}_{\mathcal{I}}(x_{\mathcal{I}}) = \mathbb{E}_{x_{\setminus \mathcal{I}}}[f(x_{\mathcal{I}} + x_{\setminus \mathcal{I}})]$$

and inductively define $f_{\mathcal{I}}(x_{\mathcal{I}})$ with:

$$\sum_{\mathcal{J} \subseteq \mathcal{I}} f_{\mathcal{J}}(x_{\mathcal{J}}) = \tilde{f}_{\mathcal{I}}(x_{\mathcal{I}})$$

We can now decompose our original function f as:

$$f(x) = \sum_{\mathcal{I} \subseteq [d]} f_{\mathcal{I}}(x_{\mathcal{I}})$$

where each of these 2^d functions corresponds to a subset $\mathcal{I}$ of the entire feature space which focuses only on these sets of features.

Moreover, we see that these functions are orthogonal with respect to the distribution.

$$\mathbb{E}_{x \sim \mathcal{P}(\mathcal{X})}\left[(f(x))^2\right] = \sum_{\mathcal{I} \subseteq [d]} \mathbb{E}_{x \sim \mathcal{P}(\mathcal{X})}\left[(f_{\mathcal{I}}(x_{\mathcal{I}}))^2\right]$$

We can more enticingly write this as:

$$\|f\|^2 = \sum_{\mathcal{I} \subseteq [d]} \|f_{\mathcal{I}}\|^2$$

Example For instance, the function $f_\emptyset$ corresponds to the average value of the function f given the input distribution. The function f_{x_1} corresponds to the expectation of the function conditioned on knowing variable x_1 minus the mean value. As a running example, let's consider $f(x, y) = (1 + x + xy)$ using the uniform distribution over the square $x, y \sim U(-1, 1)^2$. We will have $\tilde{f}_\emptyset = 1$, $\tilde{f}_x = 1 + x$, $\tilde{f}_y = 1$, and $\tilde{f}_{x,y} = 1 + x + xy$, so that $f_\emptyset = 1$, $f_x = x$, $f_y = 0$, $f_{x,y} = xy$. In our example this means that:

$$\mathbb{E}_{x,y}[(f(x,y))^2] \;=\; \mathbb{E}[1^2] + \mathbb{E}_x[x^2] + \mathbb{E}_y[0^2] + \mathbb{E}_{x,y}[(xy)^2] \;=\; 1 + \frac{1}{3} + 0 + \frac{1}{9} \;=\; \frac{13}{3}$$

Formally, let us assume that our function f is measurable, L^1 and L^2 integrable with respect to our probability measure, and that additionally each such subfunction is well defined and similarly integrable for all product measures. In this case it should be that all of the following expectations (and integrations) that we write are well defined and exchangeable. Under these assumptions, we have the sum decomposition as just written above.

Proof. Let us first note that:

$$\|f\|^2 = \mathbb{E}[\sum_{I_1 \subseteq [d]} \sum_{I_2 \subseteq [d]} f_{I_1} f_{I_2}] = \sum_{I_1 \subseteq [d]} \sum_{I_2 \subseteq [d]} \mathbb{E}[f_{I_1} f_{I_2}]$$

and consider arbitrary $I_1, I_2 \subseteq [d]$. Let us define

$K = I_1 \cap I_2$, $K_1 = I_1 \setminus I_2$, $K_2 = I_2 \setminus I_1$, $L = I_1^C \cap I_2^C$.

Further, we see that:

$$\mathbb{E}[f_{I_1} f_{I_2}] = \mathbb{E}[\sum_{A_1 \subseteq K, B_1 \subseteq K_1} \sum_{A_2 \subseteq K, B_2 \subseteq K_2} (-1)^\epsilon \tilde{f}_{A_1 \cup B_1} \tilde{f}_{A_2 \cup B_2}]$$

$$\text{where} \quad \epsilon = (|I_1| + |I_2| - |A_1| - |B_1| - |A_2| - |B_2|)$$

Let us further focus on

$$\mathbb{E}[\tilde{f}_{A_1\cup B_1}\tilde{f}_{A_2\cup B_2}] = \mathbb{E}\Big[\mathbb{E}_{\setminus(A_1\cup B_1)}[f]\cdot\mathbb{E}_{\setminus(A_2\cup B_2)}[f]\Big]$$

We are now able to take $E = A_1 \cap A_2$ and we see that our outside expectation can be reduced to an expectation over E because for each feature not in the set E, at least one of the functional terms in the product is constant with respect to that variable. In more detail, anything outside of $(A_1 \cup B_1)$ will have the first term be constant and anything outside of $(A_2 \cup B_2)$ will have the second term be constant. With respect to any of these variables, we are able to use the linearity of expectation to show that our reduced expectation will have the same value as the original expectation. Consequently, we may reduce this expectation except for the variables in $(A_1 \cup B_1) \cap (A_2 \cup B_2) = (A_1 \cap A_2) = E$ and we have that our expectation reduces to:

$$\mathbb{E}_E[\mathbb{E}_{\setminus(E)}[f]\cdot\mathbb{E}_{\setminus(E)}[f]] = \mathbb{E}_E[(\mathbb{E}_{\setminus E}[f]^2)]$$

In greater detail, we have that when we define $A_1' = K \setminus A_1$, $A_2' = K \setminus A_2$, $B_1' = K_1 \setminus B_1$, $B_2' = K_2 \setminus B_2$, and $E' = K \setminus E$ that:

$$\begin{aligned}
&\mathbb{E}\Big[\mathbb{E}_{\setminus(A_1\cup B_1)}[f]\cdot\mathbb{E}_{\setminus(A_2\cup B_2)}[f]\Big]\\
=\ &\mathbb{E}_{A_1,B_1,A_2,B_2}\Big[\mathbb{E}_{A_1',B_1',K_2,L}[f]\cdot\mathbb{E}_{A_2',K_1,B_2',L}[f]\Big]\\
=\ &\mathbb{E}_{E,A_1\setminus E,A_2\setminus E}\Big[\mathbb{E}_{B_1,B_2}\Big[\mathbb{E}_{A_1',B_1',K_2,L}[f]\cdot\mathbb{E}_{A_2',K_1,B_2',L}[f]\Big]\Big]\\
=\ &\mathbb{E}_{E,A_1\setminus E,A_2\setminus E}\Big[\mathbb{E}_{B_1}\Big[\mathbb{E}_{A_1',B_1',K_2,L}[f]\cdot\mathbb{E}_{B_2}\Big[\mathbb{E}_{A_2',K_1,B_2',L}[f]\Big]\Big]\Big]\\
=\ &\mathbb{E}_{E,A_1\setminus E,A_2\setminus E}\Big[\mathbb{E}_{B_1}\Big[\mathbb{E}_{A_1',B_1',K_2,L}[f]\Big]\cdot\mathbb{E}_{A_2',K_1,K_2,L}[f]\Big]\\
=\ &\mathbb{E}_E\Big[\mathbb{E}_{A_1\setminus E,A_2\setminus E}\Big[\mathbb{E}_{A_1',K_1,K_2,L}[f]\cdot\mathbb{E}_{A_2',K_1,K_2,L}[f]\Big]\Big]\\
=\ &\mathbb{E}_E\Big[\mathbb{E}_{A_1\setminus E}\Big[\mathbb{E}_{A_1',K_1,K_2,L}[f]\cdot\mathbb{E}_{A_2\setminus E}\Big[\mathbb{E}_{A_2',K_1,K_2,L}[f]\Big]\Big]\Big]\\
=\ &\mathbb{E}_E\Big[\mathbb{E}_{A_1\setminus E}\Big[\mathbb{E}_{A_1',K_1,K_2,L}[f]\Big]\cdot\mathbb{E}_{E',K_1,K_2,L}[f]\Big]\\
=\ &\mathbb{E}_E\Big[\mathbb{E}_{E',K_1,K_2,L}[f]\cdot\mathbb{E}_{E',K_1,K_2,L}[f]\Big]\\
=\ &\mathbb{E}_E\Big[\Big(\mathbb{E}_{\setminus E}[f]\Big)^2\Big]
\end{aligned}$$

where each step is done by the linearity of expectation and the law of total expectation.

Since this is true for any $I_1, I_2, A_1, A_2, B_1, B_2$, we now focus on entire summation. Let us assume that $I_1 \neq I_2$ so that we will have that either K_1 or K_2 is nonempty. Let's assume without loss of generality that it is K_1 which is nonempty. It then follows that

$$\sum_{B_1\subseteq K_1}(-1)^{|K_1|-|B_1|}\mathbb{E}\Big[\tilde{f}_{A_1\cup B_1}\tilde{f}_{A_2\cup B_2}\Big] =$$

$$\sum_{B_1\subseteq K_1}(-1)^{|K_1|-|B_1|}\mathbb{E}_E\Big[\tilde{f}^2_{\setminus E}\Big] =$$

$$\mathbb{E}_E\Big[\tilde{f}^2_{\setminus E}\Big]\cdot\sum_{B_1\subseteq K_1}(-1)^{|K_1|-|B_1|} = \mathbb{E}_E\Big[\tilde{f}^2_{\setminus E}\Big]\cdot 0 = 0$$

Altogether, we can go back to our original reformulation to get our desired result:

$$\|f\|^2 = \mathbb{E}[\sum_{I_1\subseteq[d]}\sum_{I_2\subseteq[d]} f_{I_1}f_{I_2}] = \sum_{I_1\subseteq[d]}\sum_{I_2\subseteq[d]}\mathbb{E}[f_{I_1}f_{I_2}] =$$

$$\sum_{I\subseteq[d]}\mathbb{E}[f_I f_I] + \sum_{I_1\neq I_2\subseteq[d]}\mathbb{E}[f_{I_1}f_{I_2}] =$$

$$\sum_{I \subseteq [d]} \mathbb{E}[f_I^2] + \sum_{I_1 \neq I_2 \subseteq [d]} 0 = \sum_{I \subseteq [d]} \mathbb{E}[f_I^2]$$

□

Moreover, our model g approximating the function f decomposes in the same way and we divide our estimation problem into 2^d different signal estimation problems with:

$$\|f - g\|^2 = \sum_{\mathcal{I} \subseteq [d]} \|f_{\mathcal{I}} - g_{\mathcal{I}}\|^2$$

In the next two subsections, we will further develop this framework to layout why such a decomposition should succeed by only estimating a subset of all possible interactions. In particular, we will show that when $|\mathcal{I}|$ is large, it will become likely that $\|f_{\mathcal{I}}\|^2 < \|f_{\mathcal{I}} - g_{\mathcal{I}}\|^2$ as the effective signal-to-noise ratio shrinks as $|\mathcal{I}|$ grows. In other words, noisy estimates of the signal on average perform even worse than the baseline guess of zero signal.

A.3 Higher dimensional signals decay with dimension

For this section we will focus on a Fourier signal f on the d-dimensional rescaled torus $[-1, 1]^d$.

Proposition 2. *If we assume that our signal is in the class C^k of k-differentiable smooth functions, then we have that our Fourier coefficients decay like $(\prod_{j=1}^{n} \frac{1}{m_j})^k$. This is a known fact from Fourier analysis and can be easily derived using a Fourier series.*

Consequently, we will consider a random signal defined as follows: Draw coefficients $c_{\overrightarrow{m}}$ and $d_{\overrightarrow{m}}$ from $N(0, \sigma^2)$ independently for all $\overrightarrow{m} \in \mathbb{N}_0^d$. Rescale these coefficients according to k-smooth decay as $a_{\overrightarrow{m}} := c_{\overrightarrow{m}} \cdot (\prod_{j:m_j \neq 0} \frac{1}{m_j})^k$ and $b_{\overrightarrow{m}} := d_{\overrightarrow{m}} \cdot (\prod_{j:m_j \neq 0} \frac{1}{m_j})^k$. Define our function as the Fourier series:

$$f(\overrightarrow{x}) = \sum_{\overrightarrow{m} \in \mathbb{N}_0^d} a_{\overrightarrow{m}} \cos(2\pi \overrightarrow{m} \cdot \overrightarrow{x}) + \sum_{\overrightarrow{m} \in \mathbb{N}_0^d} b_{\overrightarrow{m}} \sin(2\pi \overrightarrow{m} \cdot \overrightarrow{x})$$

We know that with probability one this will correspond to a k-smooth function on $[-1, 1]^n$. Let us assume that our $\overrightarrow{x}$ are uniformly distributed on the cube/ torus.

Proposition 3. *When we consider all interactions of size K ($\mathbb{I}_K := \{\mathcal{I} \subseteq [d] : |\mathcal{I}| = K\}$), the norm of our function scales like:*

$$\sum_{\mathcal{I} \in \mathbb{I}_K} \|f_{\mathcal{I}}\|^2 = \binom{d}{K} \frac{(\zeta(2k)/2)^K}{(\zeta(2k)/2 + 1)^d}$$

Proof. Let us first introduce the 0-norm $\|\overrightarrow{m}\|_0$ which denotes how many nonzero entries the index $\overrightarrow{m}$. This will be useful because while our basis functions are orthogonal we are now considering a probabilistic space instead of a normally scaled L^2 space meaning we will be integrating against the uniform measure $dx/2$ We know that all of our basis functions are orthogonal in this space and so by Parseval's identity we see that:

$$\mathbb{E}[f^2] = (a_0^2 + 2a_0 b_0 + b_0^2) + \sum_{0 \neq \overrightarrow{m} \in \mathbb{N}_0^d} a_{\overrightarrow{m}}^2 \cdot 2^{-\|m\|_0}$$

$$+ \sum_{0 \neq \overrightarrow{m} \in \mathbb{N}_0^d} b_{\overrightarrow{m}}^2 \cdot 2^{-\|m\|_0}$$

Moreover, due to direct calculation or usage of our proposition from our previous section, we can calculate the value of the interactions of feature size K. First, let's calculate for $K = 0$. We see that $\mathbb{E}[(a_0^2 + 2a_0 b_0 + b_0^2)] = 2\sigma^2$ For $K > 0$, we can see that

$$\sum_{\mathcal{I} \in \mathbb{I}_K} \|f_{\mathcal{I}}\|^2 = 2^{-K} \mathbb{E}\left[\sum_{\overrightarrow{m} \in \mathbb{N}_0^d : \|\overrightarrow{m}\|_0 = K} a_{\overrightarrow{m}}^2 + b_{\overrightarrow{m}}^2 \right]$$

We can also see that there are $\binom{d}{K}$ regions where $\|\overrightarrow{m}\|_0 = K$ will hold corresponding to the $\binom{d}{K}$ feature subsets of size K. We will briefly calculate this sum for only one of such subsets, namely $I = \{1, \ldots, K\} \in \mathbb{I}_K$.

$$\sum_{\overrightarrow{m} \in \mathbb{N}_0^K} \mathbb{E}\left[a_{\overrightarrow{m}}^2 + b_{\overrightarrow{m}}^2\right] = \sum_{m_1, \ldots, m_K = 1}^{\infty} \frac{2\sigma^2}{m_1^{2k} \cdot \cdots \cdot m_K^{2k}}$$

Now consider the Riemann-Zeta function defined as $\zeta(a) = \sum_{j=1}^{\infty} \frac{1}{j^a}$ we see that our summation can be taken over each index as the Riemann-Zeta function and we get $2\sigma^2(\frac{\zeta(2k)}{2})^K$. Combining this with the fact that we know there are $\binom{d}{K}$ such sums we get that the overall contribution is indeed $\binom{d}{K} 2\sigma^2(\frac{\zeta(2k)}{2})^K$. Setting $\|f\| = 1$, we get that $\frac{1}{\sigma^2} = 2(\frac{\zeta(2k)}{2} + 1)^d$ and we are done. □

Consequently, we find that even for a very general signal, we still ultimately have a decay in the signal. The signal we consider is a completely generalized C^k Fourier signal and still decays to having zero strength for some K large enough. This is the first part of our argument and one could equally assume that the higher dimensional interactions are actually sparse or mostly zero. This is due to the structure we would expect out of real-world data unlike a completely arbitrary signal. One should further note that only a measure zero subset of C^k functions are C^{k+1} and so even when we assume k-times differentiable we implicitly consider a signal that it is not $(k+1)$ times differentiable. Thus, we have shown there is eventually decay for a very arbitrary signal.

A.4 Higher dimensional signals are intrinsically more difficult to estimate

We should imagine that the amount of data we need to estimate what happens in n-dimensional space should be something like r^n where r is the 'resolution' of the space that we require. For instance, if we use histograms, we could imagine that we only need 2^n different bins in a binary space but more like 10^n or higher bins in a continuous space.

We do some calculations that show in the 'high resolution' regime where we have enough resolution to be able to sufficiently capture the details of our function, we will ultimately need samples scaling like $\frac{r^n}{SNR}$.

Consider a signal with additive noise of the form:

$$y = \mu \cdot \exp[2\pi i m(x_1 + \cdots + x_n)] + \sigma \cdot \epsilon$$

where we define this function with x uniformly distributed on the n-dimensional torus $\mathbb{T}^n$ and $\epsilon \sim N(0, 1)$. Note that this is a complex-valued function and we will use the notation $\bar{y}$ to denote the complex conjugate.

If we consider a histogram estimation technique which divides $\mathbb{T}^n$ into r^n bins corresponding to a uniform binning $\{0, \frac{1}{r}, \ldots, \frac{r-1}{r}, 1\}$. We can then calculate exactly the 'value' of using our sample as part of our histogram. Moreover, if we imagine each bin having N samples we can estimate the overall impact of the N-sample estimate.

Using our sample is beneficial whenever:

$$(1 + \frac{\sigma^2}{\mu^2}) < (N + 1) \cdot c(r, m, n)$$

where our constant can be written

$$c(r, m, n) = \left(\frac{\sin(\pi \frac{m}{r})}{\pi \frac{m}{r}}\right)^{2n}$$

Proof. Let us take our histogram estimator to have resolution $r \in \mathbb{N}$ and with bins having endpoints $0 = a_0 < b_0 = \frac{1}{r} = a_1 < b_1 = \frac{2}{r} = a_2 < \cdots < \frac{r}{r} = b_r$. Our histogram estimator will use the estimate defined in each bin as $\hat{f} = \frac{1}{N} \sum_{j=1}^{N} (\mu \cdot \exp[2\pi i m \cdot x_j] + \sigma \cdot \epsilon_j)$ We will break our estimate of the mean-squared error of our function approximation over each bin of the histogram. We will then have our approximation error given by:

$$\int_{a^{(1)}}^{b^{(1)}} \cdots \int_{a^{(n)}}^{b^{(n)}} \|f - \hat{f}\|^2 = \int_{a^{(1)}}^{b^{(1)}} \cdots \int_{a^{(n)}}^{b^{(n)}} [f\bar{f} - 2f\bar{\hat{f}} + \hat{f}\bar{\hat{f}}] =$$

$$\|f\|^2 - 2\mathbb{E}[f]\mathbb{E}\left[\bar{f} + \frac{1}{N}\sum_{j=1}^{N}\epsilon_j\right] + \mathbb{E}\left[\frac{1}{N^2}\sum_{j=1}^{N}\sum_{k=1}^{N}(\mu\cdot\exp[2\pi im\cdot x^{(j)}] + \sigma\cdot\epsilon_j)(\mu\cdot\exp[-2\pi im\cdot x^{(k)}] + \sigma\cdot\epsilon_k)\right] =$$

$$\|f\|^2 - 2\mathbb{E}[f]\mathbb{E}[\bar{f}] + \frac{\sigma^2}{N^2}\mathbb{E}[\sum_j \epsilon_j^2] + \frac{\mu^2}{N^2}\sum_j \mathbb{E}[\exp[2\pi im\cdot(x^{(j)} - x^{(j)})]] +$$

$$\frac{\mu^2}{N^2}\sum_{j\neq k}\mathbb{E}[\exp[2\pi im\cdot(x^{(j)} - x^{(k)})]] =$$

$$\|f\|^2 - 2\mu^2\mathbb{E}[f/\mu]\mathbb{E}[\bar{f}/\mu] + \frac{\sigma^2}{N} + \frac{\mu^2}{N^2}(N\cdot 1) + \frac{\mu^2}{N^2}((N^2 - N)\cdot\mathbb{E}[f/\mu]\mathbb{E}[\bar{f}/\mu] =$$

In order to be a good estimate (better than zero), we need that everything but the first term is smaller than zero, because this will mean we are decreasing from the baseline of $\|f\|^2$. Hence, we desire that

$$-2Nc + \frac{\sigma^2}{\mu^2} + 1 + (N-1)\cdot c < 0$$

where we define c as the quantity $\mathbb{E}[f/\mu]\mathbb{E}[\bar{f}/\mu]$.

Hence, we need that:

$$(N+1)\cdot c > \frac{1}{SNR} + 1$$

We now calculate c.

$$\mathbb{E}[f/\mu] = \int_{a^{(1)}}^{b^{(1)}}\cdots\int_{a^{(n)}}^{b^{(n)}}\exp[2\pi im\cdot x] = \prod_{j=1}^{n}\frac{\exp[2\pi imb] - \exp[2\pi ima]}{2\pi im(b-a)}$$

So,

$$\mathbb{E}[f/\mu]\mathbb{E}[\bar{f}/\mu] = \prod_{j=1}^{n}(\frac{e^{[2\pi imb]} - e^{[2\pi ima]}}{2\pi im(b-a)})\cdot(\frac{e^{[-2\pi imb]} - e^{[-2\pi ima]}}{-2\pi im(b-a)}) = \frac{(-1)^d(-1)^d}{(2\pi)^{2n}}.$$

And so then because no matter our choice of a, b we always have that $b - a = \frac{1}{r}$(by our choice of uniform spacing) it follows that:

$$\prod_{j=1}^{n}\frac{e^{[2\pi im(b-b)]} - e^{[2\pi im(b-a)]} - e^{[2\pi im(a-b)]} - e^{[2\pi im(a-a)]}}{m(b-a)\cdot m(b-a)}$$

$$= \frac{r^{2n}}{(2\pi)^{2n}m^{2n}}\cdot\prod_{j=1}^{n}(1 - \cos(2\pi m/r) - \cos(2\pi m/r) + 1)$$

$$= (\frac{r}{2\pi m})^{2n}\cdot(2(1 - \cos(2\pi m/r))^n = (\frac{r}{2\pi m})^{2n}\cdot(2(2\sin(\pi m/r)^2))^n = \left(\frac{\sin[\pi\frac{m}{r}]}{\pi\frac{m}{r}}\right)^{2n}$$

Altogether, this yields:

$$1 + \frac{1}{SNR} < (N+1)\cdot\left(\frac{\sin[\pi\frac{m}{r}]}{\pi\frac{m}{r}}\right)^{2n}$$

□

We can consider the regime of 'high resolution' where we have that $r >> m$ meaning that the complexity of the histogram estimator (number of bins) is much higher than the signal complexity (frequency of oscillation). In this regime, we can see that $c(r, m, n) \approx 1$ meaning our equation reduces to :

$$(1 + \frac{\sigma^2}{\mu^2}) < (N+1)$$

If we further note that our N is a bin-wise sample size we will note that we actually need $N' = Nr^n$ total samples for our estimator and that $\frac{\sigma^2}{\mu^2}$ is just the signal to noise ratio. Finally, we get that $N' > \frac{1}{SNR}\cdot r^n$ is the requirement to get an improvement over alternatively assuming the signal is uniformly zero. At a fixed level of noise, this means that we need samples of the order $\gtrsim r^n$.

We should additionally consider, as we connect this argument with the section above, that we have previously implied that the strength of the signal μ also shrinks with growing dimension, hence raising the requirement of sample size N' even higher as we increase the dimension K.

A.5 Summary of Theory

In conclusion, we find multiple strands of theory which support that estimating the full functional form is both unnecessary and at risk of overfitting. Altogether, we have that $\|f_{\mathcal{I}}\|^2$ shrinks as $|\mathcal{I}|$ grows and that $\|f_{\mathcal{I}} - g_{\mathcal{I}}\|^2$ grows as $|\mathcal{I}|$ grows. Ultimately, we expect there to be an intersection between these for most datasets. Consequently, there will be an optimal cut-off point where higher-order feature interactions should be left unconsidered.

In practice, we find that it is moreover beneficial to cut out not only those signals which are too small because they are too high dimensional, but also those signals which we estimate to be small from the available data. Although touched upon in the first section for psuedo-boolean functions, a more fleshed out argument for continuous functions which follow a sparsity assumption of the type in this paper, leveraging work in high-dimensional statistics, would be a worthwhile direction. Additionally, a more thorough investigation of the distribution of feature interactions under theoretical setups beyond the Gaussian-Fourier setup we consider could be useful to gain further intuition about the empirical distribution of interactions found in real-world data. Such results could further enhance the practicality of the assumptions made in high-dimensional statistics arguments for sparse feature interactions.

B Discussion on GAMs

Generalized Additive Models (GAMs) have existed for decades since their conception in 1990 [16]. Originally formulated with only univariate functions and fit using splines, they were soon extended to 'Smoothing-Spline ANOVA' which used pairs of features to fit bivariate splines, being able to represent even more complex functions [46]. As the years have passed, the functional model being used to fit the nonlinear shape functions has changed alongside popular ML techniques of the era: random forests, boosting machines, kernel methods, and recently neural networks. In fact, it has only been in the last few years where neural additive models have grown in popularity as an interpretable alternative to deep neural networks [2, 8, 49]. Despite consistent usage over the years, the application of GAMs to fit trivariate functions has been extremely scant. Besides scattered works fitting trivariate GAMs on very small-scale synthetic data, only SALSA [21] seems to evaluate higher-order GAMs across multiple real-world datasets. Two key obstructions which have prevented higher-order GAMs from becoming popular are: (1) three-dimensional functions are fundamentally more challenging to visualize and interpret; (2) cubic and higher scaling of the number of interactions is prohibitively expensive to train.

In this work, we argue that higher-order additive models have greater flexibility than existing additive models while having greater interpretability than completely black-box approaches. We achieve the ability to model higher-order interactions by phrasing the model's interactions as a sparse selection problem. In the following two sections, we address the major concerns of additive models through a discussion on sparsity and a discussion on feature correlation.

B.1 Sparse Additive Models

Work in the late '90s and early '00s brought work on high-dimensional statistics like LASSO and LARS into the limelight. With the increasing popularity of sparse linear regression methods also came a body of work focusing on sparse nonlinear additive regression leveraging the same feature shrinkage techniques [24, 36, 29]. There have also been methods designed for sparse high-dimensional regression with bivariate models in mind [43, 25]. Even recently, new methods have explored sparse regression for neural additive models, using the same principles of shrinkage estimation and regularization [48, 49].

We highlight two key differences between the existing work in sparsity for additive models and our work in sparsity. First, we do not make the typical assumption of high-dimensional inference that the number of features (d) is much greater than the number of samples. In fact, the highest number of features in the datasets we explore is only 90. Rather, we make the assumption that the number of feature interactions (2^d) is much greater than the number of samples. In this light, we are doing sparse regression in the interaction space rather than the feature space. Second, we do not depend on shrinkage estimation and instead leverage heredity to provide sparse solutions. All existing work in sparse additive models depends on shrinking the estimates of unimportant features

through regularization. In our work, we find that this is an infeasible approach for dealing with the exponentially growing number of higher-order feature interactions. For this reason, we leverage the heuristics given by a partially trained neural network to guide our sparse selection algorithm. We find that in conjunction with the heredity assumption, we are able to vastly reduce the search space and effectively complete the sparse selection problem.

Let us also briefly discuss the additional implications of sparse selection. Fitting a model to fewer learned shape functions actually increases the interpretability of the model by needing to communicate fewer shape functions. Some existing work has focused explicitly on the cognitive load of additive models by reducing the number of shape functions which must be communicated [1]. Other work [9] notes that encouraging sparse solutions can have imbalanced accuracies across small subpopulations, possibly leading to unfair predictions. Many of these properties are a consequence of the underlying feature distribution, whereas GAMs are most effective in the presence of totally independent features.

B.2 Feature Correlation

When interpreting additive models, it is important to consider the underlying distribution of the features x. We focus on two simple but extreme examples to illustrate these points.

First, consider the function $f(x_1, x_2) = x_1 x_2$ where $x_1, x_2 \sim U(-1, 1)$ but assume that they are completely dependent on each other: $x_1 = x_2$. Although the full 2D function is the XOR function on the domain, conditioning on either x_1 or x_2 immediately gives the value of the other. Accordingly, each of the 1D conditional expectations are $\bar{f}_1(x_1) = x_1^2$ and $\bar{f}_2(x_2) = x_2^2$. These univariate trends are then subtracted from the bivariate conditional expectation after normalization/ purification. Consequently, on the domain of $x_1 = x_2$ each of the univariate functions looks like $x_1 x_2$ but the bivariate function actually looks like $-x_1 x_2$, the opposite of what it truly is. Although real-world data does not have such extreme correlations, similar phenomena can still make interpretation difficult.

Second, consider the function $f(x)$ which is $+1$ when $x = c_1$ and is -1 when $x = c_2$. Further, suppose that x is equal to c_1 99% of the time and is c_2 only 1% of the time. It follows that the average value of f is 0.98. The purified univariate value then becomes $+0.02$ at $x = c_1$ and -1.98 at $x = c_2$. This looks like having a value of $x = c_2$ is 99x more potent than having a value of $x = 1$. Accordingly, for visualization purposes, it might be more natural to normalize by the 'counterfactual' distribution that both $x = c_1$ and $x = c_2$ are equally likely.

Tackling these challenges behind feature correlations could be one of the key technical limitations of additive models. For example, on computer vision datasets, it becomes unreasonable to assume that one pixel is independent of its neighbor. Similarly in natural language, certain words have greater co-occurrence which bring structure to language.

C Feature Interaction Selection

We introduce an aggregation procedure using Archipelago in its original format using zero as a baseline as well as a contrastive version. We turn a local explanation into a global explanation by using our aggregation procedure. We use a feature interaction detection algorithm alongside heredity, further details are available in Algorithm 1.

C.1 Existing Feature Interaction Detection (FID) Techniques

Archipelago Archipelago estimates the feature interaction strength by approximating the Hessian of the model given a target data instance x^* to be explained and a baseline data instance x' to be compared against. The secant approximation of the Hessian which is used by Archipelago is defined:

$$\omega_{i,j}(x) = \Big(\tfrac{1}{h_i h_j}(f(x^\star_{\{i,j\}} + x_{\setminus\{i,j\}}) - f(x'_{\{i\}} + x^\star_{\{j\}} + x_{\setminus\{i,j\}}) \\ - f(x^\star_{\{i\}} + x'_{\{j\}} + x_{\setminus\{i,j\}}) + f(x'_{\{i,j\}} + x_{\setminus\{i,j\}}))\Big)^2 \quad (4)$$

where $h_i = x^*_i - x'_i$, $h_j = x^*_j - x'_j$. We further approximate the true expectation over all contexts with the two point average used in [42]: $\bar{\omega}_{i,j} := \frac{1}{2}(\omega_{i,j}(x^*) + \omega_{i,j}(x'))$. This generates a score $\bar{\omega}_{i,j}$ for every possible feature pair $\{(i,j) : i \neq j \in [d]\}$. Higher degree interactions are calculated in the same way using a secant approximation of the higher-order derivative.

Integrated Hessians The integrated gradient [40] attribution to the feature i given a target feature x^* and a baseline feature x' can be written as:

$$\phi_i(x^*) := (x^*_i - x'_i)\int_0^1 \frac{\partial f(x' - \alpha(x^* - x'))}{\partial x_i} d\alpha$$

This definition can be extended to two-dimensional interactions between features i and j and the integrated hessian [19] attribution is defined as:

$$\phi_{i,j}(x^*) := (x^*_i - x'_i)(x^*_i - x'_i)\int_{\beta=0}^1 \int_{\alpha=0}^1 \alpha\beta \frac{\partial^2 f(x' + \alpha\beta(x^* - x'))}{\partial x_i \partial x_j} d\alpha d\beta$$

While this method is able to give accurate feature attributions for differentiable models, the estimation of the integral makes it prohibitively slow, even for calculation of interaction pairs. Moreover, running this method on higher-order interactions is nearly impossible with current implementations.

SHAP SHapley Additive exPlanations [28] are of growing popularity because of their nice theoretical properites and applicability to any black-box model. They are defined by the following equation:

$$\phi_i(v) = \sum_{S \subseteq ([d] \setminus i)} \frac{|S|!(d - |S| - 1)!}{d!}\Big(v(S \cup j) - v(S)\Big)$$

Where $v : \mathcal{P}([d]) \to \mathbb{R}$ is the value function, typically defined as $v(S) := \mathbb{E}[f(X)|X_S = x^*_S]$.

This definition was originally inspired by game theory, attributing how much of the value v of a coalition S was created by each player i. Further extensions to interactions also exist like the Shapley-Taylor interaction index [13]. Unfortunately, estimating the sum in the definition of SHAP values is usually very expensive and fast implementations currently only exist for tree-based approaches [27].

C.2 Beyond Archipelago and ReLU

As mentioned in the previous section, there are other existing methods for feature interaction attribution, namely Taylor SHAP and integrated hessians. Unfortunately, both of these methods perform poorly on the main architecture of choice in this work: ReLU neural networks. There are currently no fast implementations of SHAP outside of the tree-based approaches and Integrated Hessians requires the second partial derivative to exist, which is not the case for piecewise linear ReLU networks. In theory, however, both of these techniques could be substituted into our Algorithm 1. In this way, one could imagine using (random forests + SHAP) or (tanh networks + integrated hessians), so long as the implementations are not prohibitively expensive.

In this direction, we pursue one such novel combination in NODE-GAM + Archipelago. In Section 5.2, we use Archipelago on trained NODE models to select a subset of feature triples using Algorithm 1. We next train an adjusted implementation NODE-GA3M which is able to focus on the feature triples we prescribe. As mentioned in the main body, this technique allows us to push state of the art on the Housing dataset and demonstrates the capability of our pipeline as a more general approach.

C.3 Further Visualization

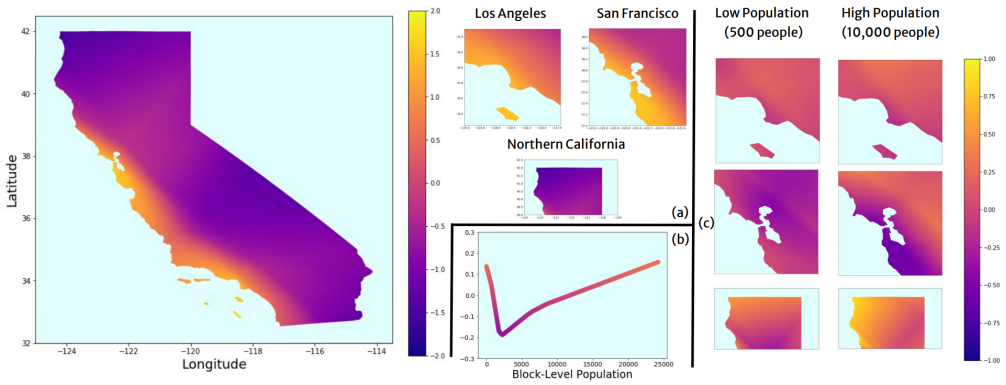

Figure 5: Reproduction of Figure 3. Visualization of the 3D feature interaction for California Housing.

One might be interested in knowing how the interpretability of SIAN compares with a regular DNN using post-hoc feature attribution. In an effort to illustrate the differences between the global interpretability of additive models and the local interpretability of feature attribution methods, we show the analogous interpretations coming from post-hoc attribution.

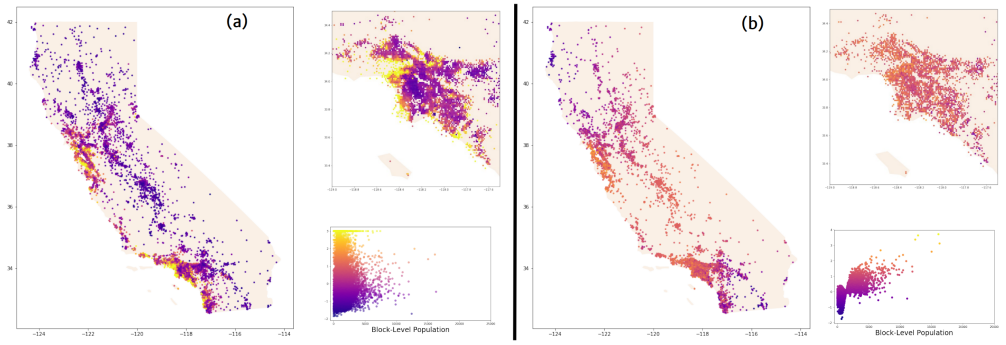

Figure 6: Feature Attribution Visualization of two feature interaction for California Housing. (a) Directly visualizing the dataset. (b) Visualizing the Archipelago feature attributions.

The main qualitative difference comes from the fact that these visualizations are using scatter plots of the training samples rather than the heatmap used in the original visualization. In Figure 6a, it is clear that there is too much noise to discern any serious trends from location or population, considering it is possible that other available features are creating the visible differences in the map. In Figure 6b, we can see a somewhat similar trend to the one in Figure 5a. There is still some noise in the feature attribution as we can see in the panel zoomed into Los Angeles. In the final plot of block-level population, we can see a more clear trend than the original data-level plot, however, ultimately it is still very difficult to discern a meaningful trend from this plot. Further, visualizing a noisy scatter plot like this means we have very minimal insight into how the model would interpolate or extrapolate into new regions.

D Visualizing MIMIC-III Shape Functions

In this section we fully depict all 45 shape functions required to make a prediction on a patient from the MIMIC-III dataset using the SIAN-2 model. Although the full model is not quite as interpretable as linear regression, the entire model which beats random forests, deep networks, and boosting machines fits on only a few pages. It may take some time to digest each of the shape functions, but it is easier than visualizing an arbitrary 30-dimensional function.

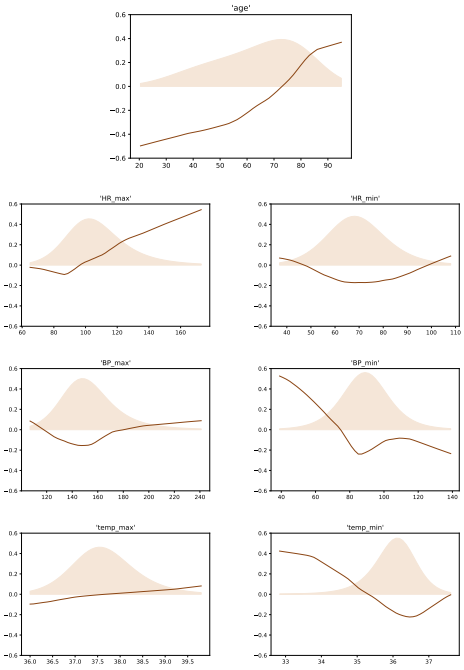

Figure 7: MIMIC 1D Shape Functions
(age, heart rate, blood pressure, temperature)

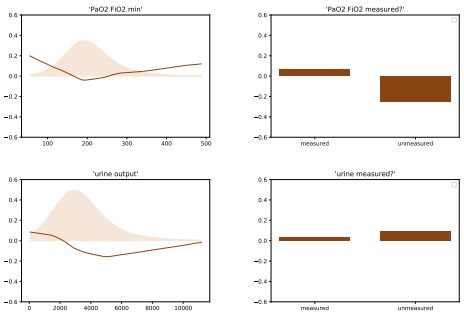

Figure 8: MIMIC 1D Shape Functions
(P/F oxygenation ratio, urine)

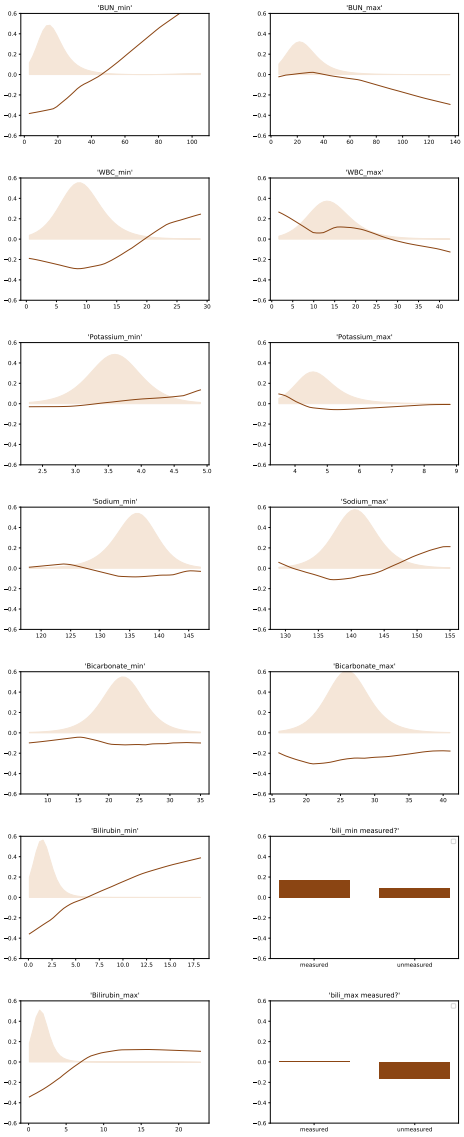

Figure 9: MIMIC 1D Shape Functions
(blood urea nitrogen, white blood cell count, potassium, sodium, bicarbonate, bilirubin)

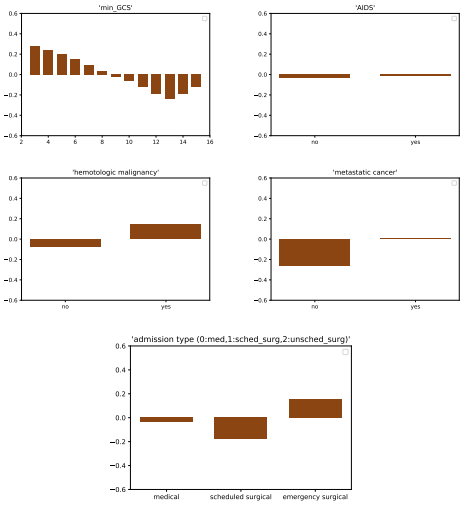

Figure 10: MIMIC 1D Shape Functions (glascow coma scale, AIDS, hematologic malignancy, metastatic cancer, admission visit type)

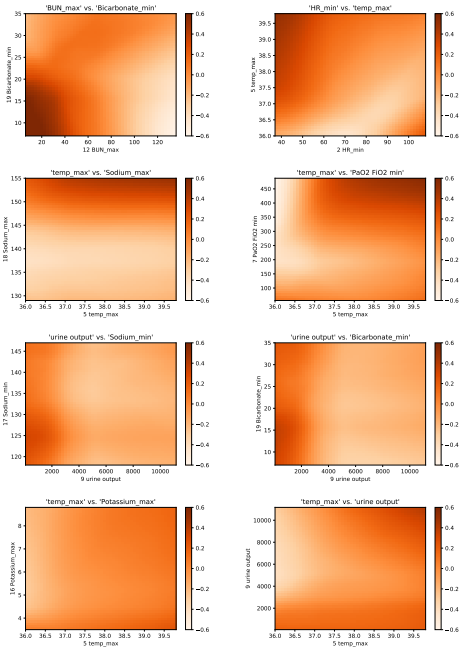

Figure 11: MIMIC 2D Shape Functions

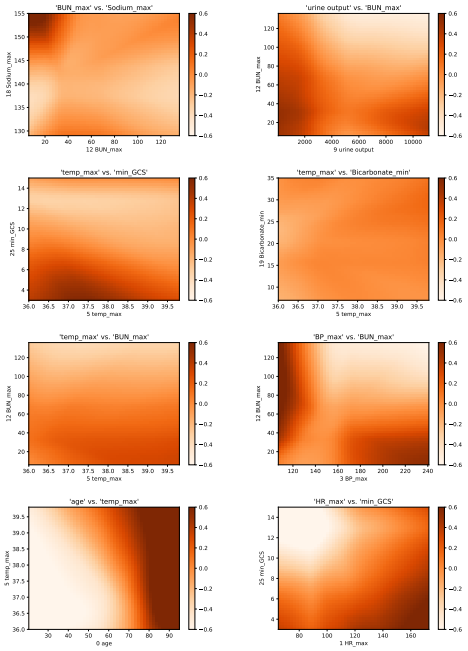

Figure 12: MIMIC 2D Shape Functions