DETECTING STATISTICAL INTERACTIONS FROM NEURAL NETWORK WEIGHTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We develop a method of detecting statistical interactions in data by directly interpreting the trained weights of a feedforward multilayer neural network. By structuring the neural network to statistical properties of data and applying sparsity regularization, we are able to leverage the weights to detect interactions with similar performance to the state-of-the-art *without* searching an exponential solution space of possible interactions. We obtain our computational savings by first observing that interactions between input features are created by the non-additive effect of nonlinear activation functions and that interacting paths are encoded in weight matrices. We use these observations to develop a way of identifying higher-order interactions with a simple traversal over the input weight matrix. In experiments on simulated and real-world data, we demonstrate the performance of our method and the importance of discovered interactions.

1 INTRODUCTION

Despite their predictive capability, neural networks have traditionally been difficult to interpret, preventing their adoption in many application domains. Healthcare and finance are examples of such domains, where understanding a machine learning model is paramount when using it to make critical decisions (Caruana et al., 2015; Goodman & Flaxman, 2016). This is because models can learn unintended patterns from data, and the risks associated with depending on these models can be consequential for stakeholders (Varshney & Alemzadeh, 2016).

Existing approaches to interpreting feedforward neural networks have focused on explanations of individual feature importance, for example by computing input gradients (Hechtlinger, 2016; Ross et al., 2017) or by using post-hoc means (Ribeiro et al., 2016). Owing to the importance of interpretation, we add to the existing approaches by introducing a way of finding feature groupings that neural networks model, in this case the statistical interactions.

Statistical interactions carry great importance in natural phenomena, where features often have joint effects with other features on predicting an outcome. This is different than correlation because correlations do not involve outcome variables. The discovery of interactions can be very useful for science, where for example, physicists may want to better understand what joint factors provide evidence for new elementary particles. Moreover, interpreting interactions can also be useful for validating machine learning models. For example, doctors may want to know what interactions are accounted for in risk prediction models, to compare against known interactions from scientific literature.

In this work, we developed a simple and efficient algorithm that proposes statistical interactions of variable order in data, by accounting for all weights of a feedforward network that is fully-connected across input features. Our approach is efficient because it avoids searching over an exponential solution space of interaction candidates, which is achieved by making an approximation of hidden unit importance at the first hidden layer via all weights above and doing a 2D traversal of the input weight matrix. We provide theoretical justification that interactions between features must be created at hidden units and that our hidden unit importance approximation satisfies bounds on hidden unit gradients. We propose our framework, *Neural Interaction Detector* (`NID`), which generates a ranking of interaction candidates solely by interpreting the weights of a feedforward network. Top-K true interactions are then determined by finding a cutoff on the ranking using a special form of generalized additive model, which accounts for interactions of variable order (Wood, 2006; Lou

et al., 2013). We show via proof that our NID framework satisfies desirable properties on the rank order of higher-order interactions. In experiments on simulated and real-world data, we evaluate the performance of our approach, the results of which show similar interaction detection performance compared to the state-of-the-art while taking orders of magnitude less time.

Our contributions are as follows: 1) we demonstrate a new interpretation of neural network weights on the statistical interactions that feedforward networks capture, and 2) we propose an accurate and efficient algorithm that detects statistical interactions of variable order in data, without searching an exponential solution space of interaction candidates.

Roadmap: We first review related works and define notations in Section 2. In Section 3, we examine and quantify the interactions encoded in a neural network, which leads to our framework for interaction detection detailed in Section 4. Finally, we study our framework empirically and demonstrate its practical utility on real-world datasets in Section 5.

2 Related Work and Notations

2.1 Interaction Detection

Statistical interaction detection has been a well-studied topic in statistics, dating back to the 1920s when two-way ANOVA was first introduced (Fisher, 1925). Since then, two general approaches emerged for conducting interaction detection. One approach has been to conduct individual tests for each combination of features (Lou et al., 2013). The other approach has been to pre-specify all interaction forms of interest, then use lasso to simultaneously select which are important (Tibshirani, 1996; Bien et al., 2013).

Notable approaches such as *ANOVA* and *Additive Groves* (Sorokina et al., 2008) belong to the first group. Two-way ANOVA has been a standard method of performing pairwise interaction detection that involves conducting hypothesis tests for each interaction candidate by checking each hypothesis with F-statistics (Wonnacott & Wonnacott, 1972). Besides two-way ANOVA, there is also three-way ANOVA that performs the same analyses but with interactions between three variables instead of two; however, four-way ANOVA and beyond are rarely done because of how computationally expensive such tests become. Specifically, the number of interactions to test grows exponentially with interaction order.

Additive Groves is another method that conducts individual tests for interactions and hence must face the same computational difficulties; however, it is special because the interactions it detects are not constrained to any functional form e.g. multiplicative interactions. The unconstrained manner by which interactions are detected is advantageous when the interactions are present in highly nonlinear data (Sorokina et al., 2007; 2008). *Additive Groves* accomplishes this by comparing two regression trees, one that fits all interactions, and the other that has the interaction of interest forcibly removed.

In interaction detection, lasso-based methods are popular in large part due to how quick they are at selecting interactions. One can construct an additive model with many different interaction terms and let lasso shrink the coefficients of unimportant terms to zero (Tibshirani, 1996). While lasso methods are fast, they require specifying all interaction terms of interest. For pairwise interaction detection, this requires $O(p^2)$ terms (where p is the number of features), and $O(2^p)$ terms for higher-order interaction detection. Still, the form of interactions that lasso-based methods capture is limited by which are pre-specified.

Our approach to interaction detection is unlike others in that it is both fast and capable of detecting interactions of variable order without limiting their functional forms. The approach is fast because it does not conduct individual tests for each interaction to accomplish higher-order interaction detection. This property has the added benefit of avoiding a high false positive-, or *false discovery rate*, that commonly arises from multiple testing (Benjamini & Hochberg, 1995).

2.2 Interpretability

The interpretability of neural networks has largely been a mystery since their inception; however, many approaches have been developed in recent years to interpret neural networks in their traditional feedforward form and as deep architectures. Feedforward neural networks have undergone

multiple advances in recent years, with theoretical works justifying the benefits of neural network depth (Telgarsky, 2016; Liang & Srikant, 2016) and new research on interpreting feature importance from input gradients (Hechtlinger, 2016; Ross et al., 2017). Deep architectures have seen some of the greatest breakthroughs, with the widespread use of attention mechanisms in both convolutional and recurrent architectures to show where they focus on for their inferences (Itti et al., 1998; Mnih et al., 2014; Xu et al., 2015). Methods such as feature map visualization (Yosinski et al., 2015; Zhang et al., 2017), de-convolution (Zeiler & Fergus, 2014), saliency maps (Simonyan et al., 2013), and many others have been especially important to the vision community for understanding how convolutional networks represent images.

Unlike previous works in interpretability, our approach interprets a multilayer feedforward network based on statistical groupings of features that are modeled, specifically statistical interactions between features with respect to an outcome variable, y.

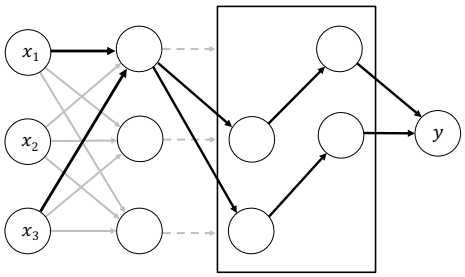


Figure 1: An illustration of an interaction within a fully connected feedforward neural network, where the box contains later layers in the network. The first hidden unit takes inputs from x_1 and x_3 with large weights and creates an interaction between them. The strength of the interaction is determined by both incoming weights and the outgoing paths between a hidden unit and the final output, y.

2.3 Notations

Vectors are represented by boldface lowercase letters, such as $\mathbf{x}, \mathbf{w}$; matrices are represented by boldface capital letters, such as $\mathbf{W}$. The i-th entry of a vector $\mathbf{w}$ is denoted by w_i, and element (i, j) of a matrix $\mathbf{W}$ is denoted by $W_{i,j}$. The i-th row and j-th column of $\mathbf{W}$ are denoted by $\mathbf{W}_{i,:}$ and $\mathbf{W}_{:,j}$, respectively. For a vector $\mathbf{w} \in \mathbb{R}^n$, let $\mathrm{diag}(\mathbf{w})$ be a diagonal matrix of size $n \times n$, where $\{\mathrm{diag}(\mathbf{w})\}_{i,i} = w_i$. For a matrix $\mathbf{W}$, let $|\mathbf{W}|$ be a matrix of the same size where $|\mathbf{W}|_{i,j} = |\mathbf{W}_{i,j}|$.

Let $[p]$ denote the set of integers from 1 to p. An *interaction*, $\mathcal{I}$, is a subset of all input features $[p]$ with $|\mathcal{I}| \geq 2$. For a vector $\mathbf{w} \in \mathbb{R}^p$ and $\mathcal{I} \subseteq [p]$, let $\mathbf{w}_{\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|}$ be the vector restricted to the dimensions specified by $\mathcal{I}$.

Feedforward Neural Network [1] Consider a feedforward neural network with L hidden layers. Let p_ℓ be the number of hidden units in the ℓ-th layer. We treat the input features as the 0-th layer and $p_0 = p$ is the number of input features. There are L weight matrices $\mathbf{W}^{(\ell)} \in \mathbb{R}^{p_\ell \times p_{\ell-1}}$ and L bias vectors $\mathbf{b}^{(\ell)} \in \mathbb{R}^{p_\ell}$, $\ell = 1, 2, \ldots, L$. Let $\phi(\cdot)$ be the activation function (non-linearity), and let $\mathbf{w}^y \in \mathbb{R}^{p_L}$ and $b^y \in \mathbb{R}$ be the coefficients and bias for the final output. Then, the hidden units $\mathbf{h}^{(\ell)}$ of the neural network and the output y with input $\mathbf{x} \in \mathbb{R}^p$ can be expressed as:

$$\mathbf{h}^{(0)} = \mathbf{x}, \quad y = (\mathbf{w}^y)^\top \mathbf{h}^{(L)} + b^y, \quad \text{and } \mathbf{h}^{(\ell)} = \phi\left(\mathbf{W}^{(\ell)}\mathbf{h}^{(\ell-1)} + \mathbf{b}^{(\ell)}\right), \quad \forall \ell = 1, 2, \ldots, L.$$

We can construct a directed acyclic graph $G = (V, E)$ based on non-zero weights, where we create vertices for input features and hidden units in the neural network and edges based on the non-zero entries in the weight matrices. See Appendix A for a formal definition.

3 Feature Interactions in Neural Networks

A statistical interaction describes a situation in which the joint influence of multiple variables on an output variable is not additive (Dodge, 2006; Sorokina et al., 2008). Let $x_i, i \in [p]$ be the features and y be the response variable, a statistical interaction $\mathcal{I} \subseteq [p]$ exists if and only if $\mathbb{E}[y|\mathbf{x}]$, which is a function of $\mathbf{x} = (x_1, x_2, \ldots, x_p)$, contains a non-additive interaction between variables $\mathbf{x}_{\mathcal{I}}$:

[1] In this paper, we mainly focus on the *multilayer perceptron* architecture with ReLU activation functions, while some of our results can be generalized to a broader class of feedforward neural networks.

Definition 1 (Non-additive Interaction). *Consider a function $f(\cdot)$ with input variables $x_i, i \in [p]$, and an interaction $\mathcal{I} \subseteq [p]$. Then $\mathcal{I}$ is a non-additive interaction of function $f(\cdot)$ if and only if there does not exist a set of functions $f_i(\cdot), \forall i \in \mathcal{I}$ where $f_i(\cdot)$ is not a function of x_i, such that*

$$f(\mathbf{x}) = \sum_{i \in \mathcal{I}} f_i\left(\mathbf{x}_{[p] \setminus \{i\}}\right).$$

For example, in $x_1 x_2 + \sin(x_2 + x_3 + x_4)$, there is a pairwise interaction $\{1, 2\}$ and a *higher-order* 3-way interaction $\{2, 3, 4\}$, where higher-order denotes $|\mathcal{I}| \geq 3$. Note that from the definition of statistical interaction, a *d-way interaction can only exist if all its corresponding $(d-1)$-interactions exist* (Sorokina et al., 2008). For example, the interaction $\{1, 2, 3\}$ can only exist if interactions $\{1, 2\}$, $\{1, 3\}$, and $\{2, 3\}$ also exist. We will often use this property in this paper.

In feedforward neural networks, statistical interactions between features, or feature interactions for brevity, are created at hidden units with non-linear activation functions, and the influences of the interactions are propagated layer-by-layer to the final output (see Figure 1). In this section, we propose a framework to identify and quantify interactions at a hidden unit for efficient interaction detection, then the interactions are combined across hidden units in Section 4.

3.1 Feature Interactions at Individual Hidden Units

In feedforward neural networks, any interacting features must follow strongly weighted connections to a common hidden unit before the final output. That is, in the corresponding directed graph, interacting features will share at least one common descendant. The key observation is that non-overlapping paths in the network are aggregated via weighted summation at the final output without creating any interactions between features. The statement is rigorized in the following proposition and a proof is provided in Appendix A. The reverse of this statement, that a common descendant will create an interaction among input features, holds true in most cases.

Proposition 2 (Interactions at Common Hidden Units). *Consider a feedforward neural network with input feature $x_i, i \in [p]$, where $y = \varphi(x_1, \ldots, x_p)$. For any interaction $\mathcal{I} \subset [p]$ in $\varphi(\cdot)$, there exists a vertex $v_{\mathcal{I}}$ in the associated directed graph such that $\mathcal{I}$ is a subset of the ancestors of $v_{\mathcal{I}}$ at the input layer (i.e., $\ell = 0$).*

In general, the weights in a neural network are nonzero, in which case Proposition 2 blindly infers that all features are interacting. For example, in a neural network with just a single hidden layer, any hidden unit in the network can imply up to $2^{\|\mathbf{W}_{j,:}\|_0}$ potential interactions, where $\|\mathbf{W}_{j,:}\|_0$ is the number of nonzero values in the weight vector $\mathbf{W}_{j,:}$ for the j-th hidden unit. Managing the large solution space of interactions based on nonzero weights requires us to characterize the relative importance of interactions, so we must mathematically define the concept of interaction strength. In addition, we limit the search complexity of our task by only quantifying interactions created at the first hidden layer, which is important for fast interaction detection and sufficient for high detection performance based on empirical evaluation (see evaluation in Section 5.2 and Table 2).

Consider a hidden unit in the first layer: $\phi\left(\mathbf{w}^\top \mathbf{x} + b\right)$, where $\mathbf{w}$ is the associated weight vector and $\mathbf{x}$ is the input vector. While having the weight w_i of each feature i, the correct way of summarizing feature weights for defining interaction strength is not trivial. For an interaction $\mathcal{I} \subseteq [p]$, we propose to use an average of the relevant feature weights $\mathbf{w}_{\mathcal{I}}$ as the surrogate for the interaction strength: $\mu(|\mathbf{w}_{\mathcal{I}}|)$, where $\mu(\cdot)$ is the averaging function for an interaction that represents the interaction strength due to feature weights.

We provide guidance on how μ should be defined by first considering representative averaging functions from the generalized mean family: maximum value, root mean square, arithmetic mean, geometric mean, harmonic mean, and minimum value (Bullen et al., 1988). These options can be narrowed down by accounting for intuitive properties of interaction strength : 1) interaction strength is evaluated as zero whenever an interaction does not exist (one of the features has zero weight); 2) interaction strength does not decrease with any increase in magnitude of feature weights; 3) interaction strength is less sensitive to changes in large feature weights.

While the first two properties place natural constraints on interaction strength behavior, the third property is subtle in its intuition. Consider the scaling between the magnitudes of multiple feature weights, where one weight has much higher magnitude than the others. In the worst case, there is

one large weight in magnitude while the rest are near zero. If the large weight grows in magnitude, then interaction strength may not change significantly, but if instead the smaller weights grow at the same rate, then interaction strength should strictly increase. As a result, maximum value, root mean square, and arithmetic mean should be ruled out because they do not satisfy either property 1 or 3.

3.2 Measuring the Influence of Hidden Units

Our definition of interaction strength at individual hidden units is not complete without considering their outgoing paths, because an outgoing path of zero weight cannot contribute an interaction to the final output. To propose a way of quantifying the influence of an outgoing path on the final output, we draw inspiration from Garson's algorithm (Garson, 1991; Goh, 1995), which instead of computing the influence of a hidden unit, computes the influence of features on the output. This influence is calculated via cumulative matrix multiplications of the absolute value of neural network weight matrices. In the following, we propose our definition of hidden unit influence, then prove that this definition upper bounds the gradient magnitude of the hidden unit. To represent the influence of a hidden unit i at the ℓ-th hidden layer, we define the *aggregated weight* $z_i^{(\ell)}$, where $\mathbf{z}^{(\ell)} \in \mathbb{R}^{p_\ell}$ and

$$\mathbf{z}^{(\ell)} = |\mathbf{w}^y|^\top \left|\mathbf{W}^{(L)}\right| \cdot \left|\mathbf{W}^{(L-1)}\right| \cdots \left|\mathbf{W}^{(\ell+1)}\right|.$$

We show that this definition upper bounds the gradient magnitudes of hidden units by proving that it computes Lipschitz constants for the corresponding units. Gradients have been commonly used as variable importance measures in neural networks, especially input gradients which compute directions normal to decision boundaries (Ross et al., 2017; Goodfellow et al., 2014; Simonyan et al., 2013). Thus, an upper bound on the gradient magnitude approximates how important the variable can be. The full proof is shown in Appendix C.

Lemma 3 (Neural Network Lipschitz Estimation)**.** *Let the activation function $\phi(\cdot)$ be a 1-Lipschitz function. Then the output y is $z_i^{(\ell)}$-Lipschitz with respect to $h_i^{(\ell)}$.*

3.3 Quantifying Interaction Strength

We now combine our definitions from Sections 3.1 and 3.2 to obtain the interaction strength $\omega_i(\mathcal{I})$ of a potential interaction $\mathcal{I}$ at the i-th unit in the first hidden layer $h_i^{(1)}$,

$$\omega_i(\mathcal{I}) = z_i^{(1)} \mu\left(\mathbf{W}_{i,\mathcal{I}}^{(1)}\right). \tag{1}$$

Note that $\omega_i(\mathcal{I})$ is defined on a single hidden unit, and it is agnostic to scaling ambiguity within a ReLU based neural network. In Section 4, we discuss our scheme of aggregating strengths across hidden units, so we can compare interactions of different orders.

4 Interaction Detection

In this section, we propose our feature interaction detection algorithm `NID`, which can extract interactions of all orders without individually testing for each of them. Our methodology for interaction detection is comprised of three main steps: 1) structure the feedforward network to model univariate effects explicitly, 2) interpret trained weights to obtain a ranked shortlist of interaction candidates, and 3) determine a cutoff for the top K interactions.

4.1 Architecture

Data often contains both statistical interactions and main effects (Winer et al., 1971). Main effects describe the univariate influences of variables on an outcome variable. In fully-connected networks, main effects must pass through hidden units, which may errantly create statistical interactions between what are exclusively main effect variables and other variables. We minimize the opportunity for such spurious interactions being modeled by first structuring a neural network to account for main effects, that is, we separately model main effects using networks with univariate inputs for each input variable, and we sum up these networks and the standard fully-connected network together at their respective outputs (see Figure 2). We refer to this additive model networks as *MLP-M*. When we jointly train *MLP-M* via backpropagation, we apply L1 regularization on the interaction network and not on the main effect networks to push the modeling of main effects away from the interaction network as much as possible.

Algorithm 1 NID Greedy Ranking Algorithm

Input: input-to-first hidden layer weights $\mathbf{W}^{(1)}$, aggregated weights $\mathbf{z}^{(1)}$
Output: ranked list of interaction candidates $\{\mathcal{I}_i\}_{i=1}^m$

```
d ← initialize an empty dictionary mapping interaction candidate to interaction strength
for each row w' of W^(1) indexed by r do
    for j = 2 to p do
        I ← sorted indices of top j weights in w'
        d[I] ← d[I] + z_r^(1) μ(|w'_I|)
{I_i}_{i=1}^m ← interaction candidates in d sorted by their strengths in descending order
```

4.2 Ranking Interactions of Variable Order

We design a greedy algorithm that generates a ranking of interaction candidates by only considering, at each hidden unit, the top-ranked interactions of every order, where $2 \leq |\mathcal{I}| \leq p$, thereby drastically reducing the search space of potential interactions while still considering all orders. We first train our architecture *MLP-M* with L1 regularization on the interaction network to suppress unimportant weights, after which we use our greedy strategy to traverse the regularized input weight matrix $\mathbf{W}^{(1)}$ (Figure 2). The greedy algorithm (Algorithm 1) selects only top-ranked interaction candidates per hidden unit based on their interaction strengths. By selecting top-ranked interactions of every order and summing their respective strengths across hidden units, we obtain final interaction strengths, allowing arbitrary-order interaction candidates to be ranked relative to each other. For this algorithm, we set the averaging function $\mu(\cdot) = \min(\cdot)$ based on its performance in experimental evaluation (Section 5.1).

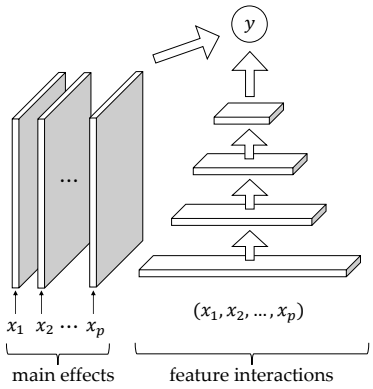


Figure 2: An illustration of the neural network architecture for interaction detection, *MLP-M*. Main effects are modeled by the lefthand networks, which are used to minimize the detection of spurious interactions in the righthand primary network.

Since a higher-order interaction only exists if its subset interactions also exist (Section 3), the subset interactions are redundant in the presence of the higher-order interaction. Thus, Algorithm 1 assumes that there are at least as many first-layer hidden units as there are the true number of such higher-order interactions and non-redundant pairwise interactions.[2][3]

In addition to efficiency, a benefit of Algorithm 1's greedy strategy is that it automatically improves the ranking of a higher-order interaction over its redundant subsets. This allows the higher-order interaction to have a better chance of ranking above any false positives and being captured in the cutoff stage. We justify this improvement by proving Theorem 4 under a mild assumption.

Theorem 4 (Improving the ranking of higher-order interactions)**.** *Let $\mathcal{R}$ be the set of interactions proposed by Algorithm 1, let $\mathcal{I} \in \mathcal{R}$ be a d-way interaction where $d \geq 3$, and let $\mathcal{S}$ be the set of subset $(d-1)$-way interactions of $\mathcal{I}$ where $|\mathcal{S}| = d$. Assume that for any hidden unit j which proposed $s \in \mathcal{S} \cap \mathcal{R}$, $\mathcal{I}$ will also be proposed at the same hidden unit, and $w_j(\mathcal{I}) > \frac{1}{d} w_j(s)$. Then, one of the following must be true: a) $\exists s \in \mathcal{S} \cap \mathcal{R}$ ranked lower than $\mathcal{I}$, i.e., $w(\mathcal{I}) > w(s)$, or b) $\exists s \in \mathcal{S}$ where $s \notin \mathcal{R}$.*

The full proof is included in Appendix D. Under the noted assumption, the theorem in part a) shows that a d-way interaction will improve over one its $d-1$ subsets in rankings as long as there is no sudden drop from the weight of the $(d-1)$-way to the d-way interaction at the same hidden units. We note that the improvement extends to b) as well, when $d = |\mathcal{S} \cap \mathcal{R}| > 1$.

[2]In practice, we use an arbitrarily large number of first layer hidden units because true interactions are initially unknown.

[3]The redundancy of the subset interactions allows us to prune them from the interaction ranking when corresponding superset interactions are higher ranked.

Table 1: Test suite of data-generating functions

$F_1(\mathbf{x})$	$\pi^{x_1 x_2}\sqrt{2x_3} - \sin^{-1}(x_4) + \log(x_3 + x_5) - \frac{x_9}{x_{10}}\sqrt{\frac{x_7}{x_8}} - x_2 x_7$
$F_2(\mathbf{x})$	$\pi^{x_1 x_2}\sqrt{2\lvert x_3\rvert} - \sin^{-1}(0.5x_4) + \log(\lvert x_3 + x_5\rvert + 1) + \frac{x_9}{1+\lvert x_{10}\rvert}\sqrt{\frac{x_7}{1+\lvert x_8\rvert}} - x_2 x_7$
$F_3(\mathbf{x})$	$\exp\lvert x_1 - x_2\rvert + \lvert x_2 x_3\rvert - x_3^{2\lvert x_4\rvert} + \log(x_4^2 + x_5^2 + x_7^2 + x_8^2) + x_9 + \frac{1}{1+x_{10}^2}$
$F_4(\mathbf{x})$	$\exp\lvert x_1 - x_2\rvert + \lvert x_2 x_3\rvert - x_3^{2\lvert x_4\rvert} + (x_1 x_4)^2 + \log(x_4^2 + x_5^2 + x_7^2 + x_8^2) + x_9 + \frac{1}{1+x_{10}^2}$
$F_5(\mathbf{x})$	$\frac{1}{1+x_1^2+x_2^2+x_3^2} + \sqrt{\exp(x_4 + x_5)} + \lvert x_6 + x_7\rvert + x_8 x_9 x_{10}$
$F_6(\mathbf{x})$	$\exp(\lvert x_1 x_2\rvert + 1) - \exp(\lvert x_3 + x_4\rvert + 1) + \cos(x_5 + x_6 - x_8) + \sqrt{x_8^2 + x_9^2 + x_{10}^2}$
$F_7(\mathbf{x})$	$(\arctan(x_1) + \arctan(x_2))^2 + \max(x_3 x_4 + x_6, 0) - \frac{1}{1+(x_4 x_5 x_6 x_7 x_8)^2} + \left(\frac{\lvert x_7\rvert}{1+\lvert x_9\rvert}\right)^5 + \sum_{i=1}^{10} x_i$
$F_8(\mathbf{x})$	$x_1 x_2 + 2^{x_3+x_5+x_6} + 2^{x_3+x_4+x_5+x_7} + \sin(x_7 \sin(x_8 + x_9)) + \arccos(0.9x_{10})$
$F_9(\mathbf{x})$	$\tanh(x_1 x_2 + x_3 x_4)\sqrt{\lvert x_5\rvert} + \exp(x_5 + x_6) + \log\left((x_6 x_7 x_8)^2 + 1\right) + x_9 x_{10} + \frac{1}{1+\lvert x_{10}\rvert}$
$F_{10}(\mathbf{x})$	$\sinh(x_1 + x_2) + \arccos(\tanh(x_3 + x_5 + x_7)) + \cos(x_4 + x_5) + \sec(x_7 x_9)$

4.3 Cutoff on Interaction Ranking

In order to predict the true top-K interactions $\{\mathcal{I}_i\}_{i=1}^K$, we must find a cutoff point on our interaction ranking from Section 4.2. We obtain this cutoff by constructing a Generalized Additive Model (*GAM*) with interactions:

$$c_K(\mathbf{x}) = \sum_{i=1}^{p} g_i(x_i) + \sum_{i=1}^{K} g_i'(\mathbf{x}_{\mathcal{I}}),$$

where $g_i(\cdot)$ captures the main effects, $g_i'(\cdot)$ captures the interactions, and both g_i and g_i' are small feedforward networks trained jointly via backpropagation. We refer to this model as *MLP-Cutoff*.

We gradually add top-ranked interactions to the *GAM*, increasing K, until *GAM* performance on a validation set plateaus. The exact plateau point can be found by early stopping or other heuristic means, and we report $\{\mathcal{I}_i\}_{i=1}^K$ as the identified feature interactions.

4.4 Pairwise Interaction Detection

A variant to our interaction ranking algorithm tests for all pairwise interactions. Pairwise interaction detection has been a standard problem in the interaction detection literature (Lou et al., 2013; Fan et al., 2016) due to its simplicity. Modeling pairwise interactions is also the *de facto* objective of many successful machine learning algorithms such as factorization machines (Rendle, 2010) and hierarchical lasso (Bien et al., 2013).

We rank all pairs of features $\{i, j\}$ according to their interaction strengths $\omega(\{i, j\})$ calculated on the first hidden layer, where again the averaging function is $\min(\cdot)$, and $\omega(\{i, j\}) = \sum_{s=1}^{p_1} \omega_s(\{i, j\})$. The higher the rank, the more likely the interaction exists.

5 Experiments

In this section, we discuss our experiments on both simulated and real-world datasets to study the performance of our approach on interaction detection.

5.1 Experimental Setup

Averaging Function Our proposed `NID` framework relies on the selection of an averaging function (Sections 3.1, 4.2, and 4.4). We experimentally determined the averaging function by comparing representative functions from the generalized mean family (Bullen et al., 1988): maximum value, root mean square, arithmetic mean, geometric mean, harmonic mean, and minimum value, intuitions behind which were discussed in Section 3.1. In order to compare the averaging functions, we used a test suite of 10 synthetic functions, which consist of a variety of interactions of varying order and overlap, as shown in Table 1. We applied our proposed greedy ranking algorithm, Algorithm

1, to each of the test functions in the test suite over 10 trials, and we counted the total number of correct interactions that are ranked before any false positives. In our evaluation, we ignore predicted interactions that are subsets of true higher-order interactions because the subset interactions are redundant (Section 3). As shown in Figure 3, the number of true top interactions we recover is highest with the averaging function, minimum value, which we will use in all of our experiments. A simple analytical study on a bivariate hidden unit, provided in Appendix B, also suggests that the minimum value is closely correlated with interaction strength.

Neural Network Configuration In our experiments, we trained feedforward neural networks that combined separate networks for capturing feature interactions and main effects for both *MLP-M* and *MLP-Cutoff* (Figure 2, Section 4.3). The interaction and main effect networks are summed at their outputs, allowing joint training via backpropagation.

In all of our experiments, the feature interaction component of our neural network consisted of four hidden layers with first-to-last layer sizes of: 140, 100, 60, and 20 units. The main effect components each had three hidden layers with sizes of: 10, 10 and 10 units. All networks used ReLU activation. The objective functions were mean-squared error for regression and cross-entropy for classification tasks. On the synthetic test suite, *MLP-M* was trained with L1 constants in the range of 5e-6 to 5e-4, based on parameter tuning on a validation set. On real-world datasets, L1 was fixed at 5e-5. *MLP-Cutoff* used a fixed L2 constant of 1e-4 in all experiments involving cutoff. Early stopping was used to prevent overfitting. In all experiments, we obtained interaction rankings from *MLP-M*, which was trained to fit data.

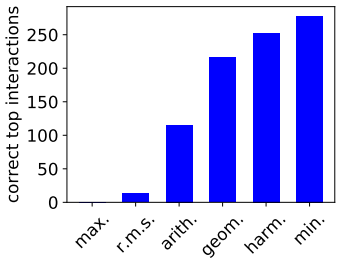


Figure 3: A comparison of averaging functions by the total number of correct interactions ranked before any false positives, evaluated on the test suite (Table 1) over 10 trials. x-axis labels are maximum, root mean square, arithmetic mean, geometric mean, harmonic mean, and minimum.

Datasets We study our interaction detection framework on both simulated and real-world experiments. For simulated experiments, we used a test suite of synthetic functions, as shown in Table 1. The test functions were designed to have a mixture of pairwise and higher-order interactions, with varying order, strength, nonlinearity, and overlap. F_1 is a commonly used function in interaction detection literature (Hooker, 2004; Sorokina et al., 2008; Lou et al., 2013). All features were uniformly distributed between -1 and 1 except in F_1, where we used the same variable ranges as reported in literature (Hooker, 2004). In all synthetic experiments, we used random train/valid/test splits of $1/3$ each on 30k data points.

We use four real-world datasets, of which two are regression datasets, and the other two are binary classification datasets. The datasets are a mixture of common prediction tasks in the cal housing and bike sharing datasets, a scientific discovery task in the higgs boson dataset, and an example of very-high order interaction detection in the letter dataset. Specifically, the cal housing dataset is a regression dataset with 21k data points for predicting California housing prices (Pace & Barry, 1997). The bike sharing dataset contains 17k data points of weather and seasonal information to predict the hourly count of rental bikes in a bikeshare system (Fanaee-T & Gama, 2014). The higgs boson dataset has 800k data points for classifying whether a particle environment originates from the decay of a Higgs Boson (Adam-Bourdarios et al., 2014). Lastly, the letter recognition dataset contains 20k data points of transformed features for binary classification of letters on a pixel display (Frey & Slate, 1991). For all real-world data, we use random train/valid/test splits of $80/10/10$.

Baselines We compare the performance of `NID` to that of three baseline interaction detection methods. Two-Way *ANOVA* (Wonnacott & Wonnacott, 1972) utilizes linear models to conduct significance tests on the existence of interaction terms. *Hierarchical lasso* (HierLasso) (Bien et al., 2013) applies lasso feature selection to extract pairwise interactions. *RuleFit* (Friedman & Popescu, 2008) contains a statistic to measure pairwise interaction strength using partial dependence functions. *Additive Groves* (AG) (Sorokina et al., 2008) is a nonparameteric means of testing for interactions by placing structural constraints on an additive model of regression trees. *AG* is a reference method for interaction detection because it directly detects interactions based on their non-additive definition.

Table 2: AUC of pairwise interaction strengths proposed by NID and baselines on a test suite of synthetic functions (Table 1). *ANOVA*, *HierLasso*, and *RuleFit* are deterministic.

	NID (proposed)	ANOVA	HierLasso	RuleFit	AG
$F_1(\mathbf{x})$	$0.995 \pm 4.4\text{e}{-3}$	0.992	1.00	0.754	1 ± 0.0
$F_2(\mathbf{x})$	$0.85 \pm 3.9\text{e}{-2}$	0.468	0.636	0.698	$0.88 \pm 1.4\text{e}{-2}$
$F_3(\mathbf{x})$	1 ± 0.0	0.657	0.556	0.815	1 ± 0.0
$F_4(\mathbf{x})$	$0.996 \pm 4.7\text{e}{-3}$	0.563	0.634	0.689	$0.999 \pm 1.4\text{e}{-3}$
$F_5(\mathbf{x})$	1 ± 0.0	0.544	0.625	0.797	$0.67 \pm 5.7\text{e}{-2}$
$F_6(\mathbf{x})$	$0.70 \pm 4.8\text{e}{-2}$	0.780	0.730	0.811	$0.64 \pm 1.4\text{e}{-2}$
$F_7(\mathbf{x})$	$0.82 \pm 2.2\text{e}{-2}$	0.726	0.571	0.666	$0.81 \pm 4.9\text{e}{-2}$
$F_8(\mathbf{x})$	$0.989 \pm 4.5\text{e}{-3}$	0.929	0.958	0.946	$0.937 \pm 1.4\text{e}{-3}$
$F_9(\mathbf{x})$	$0.93 \pm 3.3\text{e}{-2}$	0.783	0.681	0.584	$0.713 \pm 5.8\text{e}{-3}$
$F_{10}(\mathbf{x})$	$0.99 \pm 2.1\text{e}{-2}$	0.765	0.583	0.876	1 ± 0.0
average	$\mathbf{0.93} \pm 1.8\text{e}{-2}$	0.721	0.698	0.764	$0.87 \pm 1.4\text{e}{-2}$

5.2 Pairwise Interaction Detection

As discussed in Section 4.4, our framework NID can be used for pairwise interaction detection. To evaluate this approach, we used datasets generated by synthetic functions F_1-F_{10} (Table 1) that contain a mixture of pairwise and higher-order interactions, where in the case of higher-order interactions, we test for their pairwise subsets as in Sorokina et al. (2008); Lou et al. (2013). AUC scores of interaction strength proposed by NID and each baseline are shown in Table 2, and NID's interaction strengths are visualized in Figures 4 and 5 for all synthetic and real-world datasets.

For the AUC evaluation, we ran ten trials of NID and *AG* on each dataset and removed two trials with the highest and lowest AUC scores. When comparing the AUCs of NID and *AG*, we observe that their scores tend to close, except for F_5, F_6, F_8, and F_9, where NID performs significantly better than *AG*. This performance difference may be due to limitations on the model capacity of *AG*, which is tree-based. In comparison to *ANOVA*, *HierLasso*, and *RuleFit*, NID generally performs on par or better. This is expected for *ANOVA* and *HierLasso* because they are based on quadratic models, which can have difficulty approximating the interaction nonlinearities that are present in the test suite. There is one function, F_6, where *ANOVA*, *HierLasso*, and *RuleFit* outperform both NID and *AG*. We can understand this function better through visualizations of interaction strength.

In Figure 4, heat maps of synthetic datasets show the relative strengths of all possible pairwise interactions, and ground truth is indicated by red cross-marks. As shown, the interaction strengths are normally high at the cross-marks. An exception is F_6, where NID proposes weak or negligible interaction strengths at the cross-marks on the lower right region of the heat map. *ANOVA* and *HierLasso* are more sensitive to these interactions, perhaps because their underlying models are simpler. Besides F_6, F_7 also shows erroneous interaction strengths; however, comparative detection performance by the baselines is similarly poor.

Interaction strengths are also visualized on real-world datasets via heat maps (Figure 5). For example, in the cal housing dataset, there is a high-strength interaction between x_1 and x_2. These variables mean longitude and latitude respectively, and it is clear to see that the outcome variable, California housing price, should indeed strongly depend on geographical location. We further observe high-strength interactions appearing the the heat maps of the bike sharing, higgs boson dataset, and letter datasets. For example, all feature pairs appear to be interacting in the letter dataset. The binary classification task from the letter dataset is to distinguish letters A-M from N-Z using 16 pixel display features. Since the decision boundary between A-M and N-Z is not obvious, it would make sense that a neural network learns a highly interacting function to make the distinction.

5.3 Higher-Order Interaction Detection

We use our greedy interaction ranking algorithm (Algorithm 1) to perform higher-order interaction detection without an exponential search of interaction candidates. We first visualize our higher-

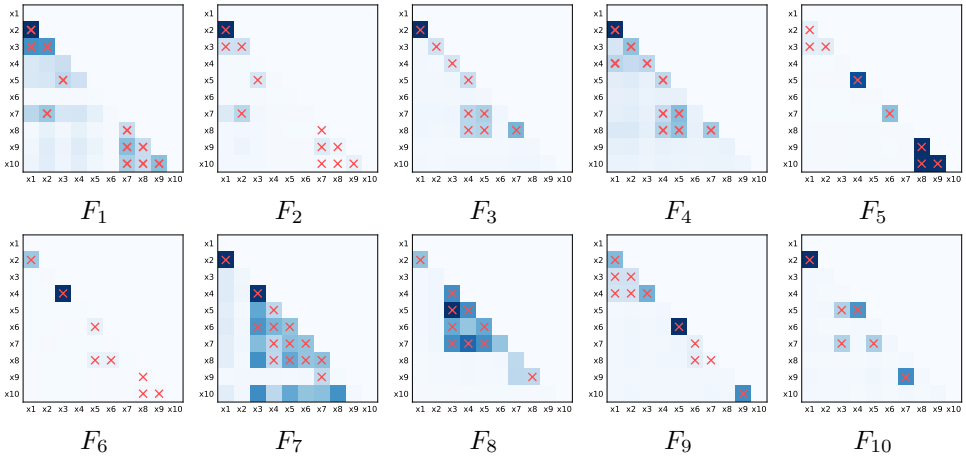


Figure 4: Heat maps of pairwise interaction strengths proposed by our NID framework on datasets generated by functions F_1-F_{10} (Table 1). Red cross-marks indicate ground truth interactions.

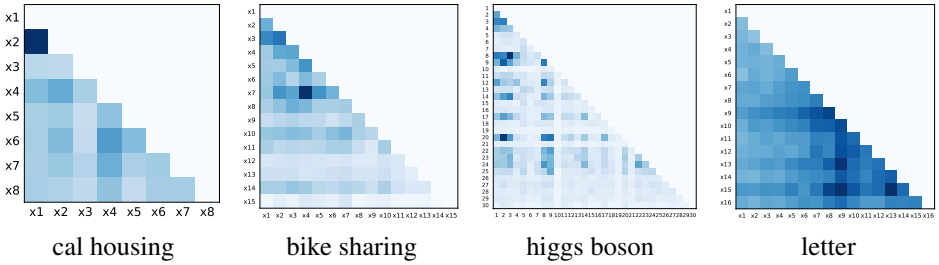


Figure 5: Heat maps of pairwise interaction strengths proposed by our NID framework on real-world datasets.

order interaction detection algorithm on synthetic and real-world datasets, then we show how the predictive capability of detected interactions closes the performance gap between *MLP-Cutoff* and *MLP-M*. Next, we discuss our experiments comparing NID and *AG* with added noise, and lastly we verify that our algorithm obtains significant improvements in runtime.

We visualize higher-order interaction detection on synthetic and real-world datasets in Figures 6 and 7 respectively. The plots correspond to higher-order interaction detection as the ranking cutoff is applied (Section 4.3). The interaction rankings generated by NID are shown on the x-axes, and the blue bars correspond to the validation performance of *MLP-Cutoff* as interactions are added. For example, the plot for cal housing shows that adding the first interaction significantly reduces RMSE. We keep adding interactions into the model until reaching a cutoff point. In our experiments, we use a cutoff heuristic where interactions are no longer added after *MLP-Cutoff*'s validation performance reaches or surpasses *MLP-M*'s validation performance (represented by horizontal dotted lines).

As seen with the red cross-marks, our method finds true interactions in the synthetic data of F_1-F_{10} before the cutoff point. Challenges with detecting interactions are again mainly associated with F_6 and F_7, which have also been difficult for baselines in the pairwise detection setting (Table 2). For the cal housing dataset, we obtain the top interaction $\{1, 2\}$ just like in our pairwise test (Figure 5, cal housing), where now the $\{1, 2\}$ interaction contributes a significant improvement in *MLP-Cutoff* performance. Similarly, from the letter dataset we obtain a 16-way interaction, which is consistent with its highly interacting pairwise heat map (Figure 5, letter). For the bike sharing and higgs boson datasets, we note that even when considering many interactions, *MLP-Cutoff* eventually reaches the cutoff point with a relatively small number of superset interactions. This is because many subset interactions become redundant when their corresponding supersets are found.

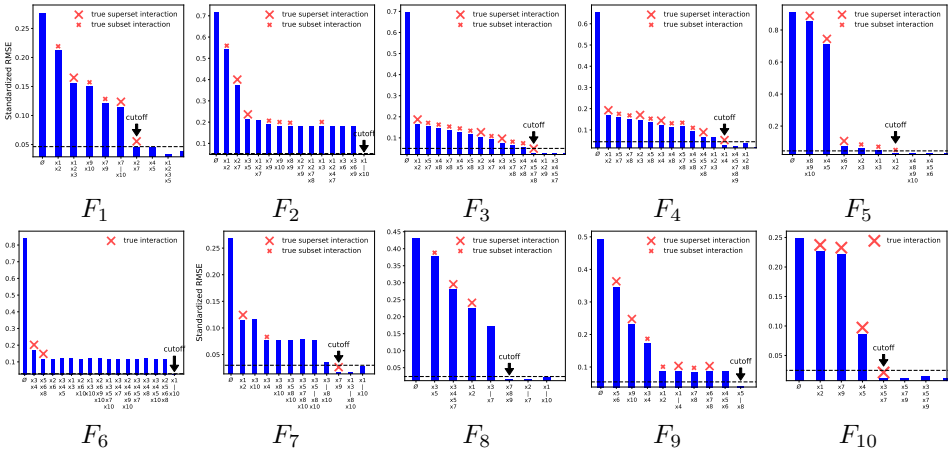


Figure 6: *MLP-Cutoff* error with added top-ranked interactions (along x-axis) of F_1-F_{10} (Table 1), where the interaction rankings were generated by the NID framework. Red cross-marks indicate ground truth interactions, and Ø denotes *MLP-Cutoff* without any interactions. Subset interactions become redundant when their true superset interactions are found.

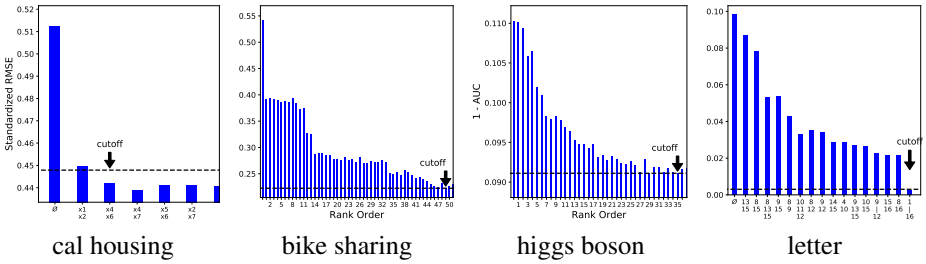


Figure 7: *MLP-Cutoff* error with added top-ranked interactions (along x-axis) of real-world datasets (Table 1), where the interaction rankings were generated by the NID framework. Ø denotes *MLP-Cutoff* without any interactions.

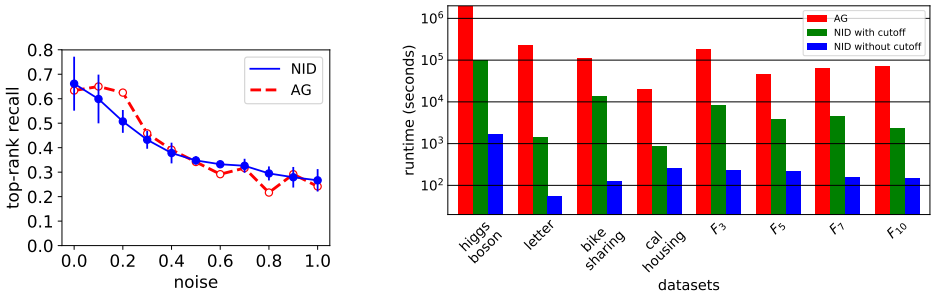
In our evaluation of interaction detection on real-world data, we study detected interactions via their predictive performance. By comparing the test performance of *MLP-Cutoff* and *MLP-M* with respect to *MLP-Cutoff* without any interactions (*MLP-Cutoff*$^{\emptyset}$), we can compute the relative test performance improvement obtained by including detected interactions. These relative performance improvements are shown in Table 3 for the real-world datasets as well as four selected synthetic datasets, where performance is averaged over ten trials per dataset. The results of this study show that a relatively small number of interactions of variable order are highly predictive of their corresponding datasets, as true interactions should.

We further study higher-order interaction detection of our NID framework by comparing it to *AG* in both interaction ranking quality and runtime. To assess ranking quality, we design a metric, *top-rank recall*, which computes a recall of proposed interaction rankings by only considering those interactions that are correctly ranked before any false positive. The number of top correctly-ranked interactions are then divided by the true number of interactions. Because subset interactions are redundant in the presence of corresponding superset interactions, only such superset interactions can count as true interactions, and our metric ignores any subset interactions in the ranking. We compute the *top-rank recall* of NID, averaged across all tests in the test suite of synthetic functions (Table 1) with 10 trials per test function. For each test, we remove two trials with max and min recall. We conduct the same tests using the state-of-the-art interaction detection method *AG*, except with only one trial per test because *AG* is very computationally expensive to run. In Figure 8a, we show *top-rank recall* of NID and *AG* at different Gaussian noise levels[4], and in Figure 8b, we show runtime comparisons on real-world and synthetic datasets. As shown, NID can obtain similar top-rank recall as *AG* while running orders of magnitude times faster.

[4]Gaussian noise was to applied to both features and the outcome variable after standard scaling all variables.

Table 3: Test performance improvement when adding top-K discovered interactions to *MLP-Cutoff* on real-world datasets and select synthetic datasets. Here, the median $\bar{K}$ excludes subset interactions, and $|\bar{\mathcal{I}}|$ denotes average interaction cardinality. RMSE values are standard scaled.

Dataset	p	Relative Performance Improvement	Absolute Performance Improvement	$\bar{K}$	$\lvert\bar{\mathcal{I}}\rvert$
cal housing	8	$99\% \pm 4.0\%$	$0.09 \pm 1.3\text{e}{-2}$ RMSE	2	2.0
bike sharing	12	$98.8\% \pm 0.89\%$	$0.331 \pm 4.6\text{e}{-3}$ RMSE	12	4.7
higgs boson	30	$98\% \pm 1.4\%$	$0.0188 \pm 5.9\text{e}{-4}$ AUC	11	4.0
letter	16	$101.1\% \pm 0.58\%$	$0.103 \pm 5.8\text{e}{-3}$ AUC	1	16
$F_3(\mathbf{x})$	10	$104.1\% \pm 0.21\%$	$0.672 \pm 2.2\text{e}{-3}$ RMSE	4	2.5
$F_5(\mathbf{x})$	10	$102.0\% \pm 0.30\%$	$0.875 \pm 2.2\text{e}{-3}$ RMSE	6	2.2
$F_7(\mathbf{x})$	10	$105.2\% \pm 0.30\%$	$0.2491 \pm 6.4\text{e}{-4}$ RMSE	3	3.7
$F_{10}(\mathbf{x})$	10	$105.5\% \pm 0.50\%$	$0.234 \pm 1.5\text{e}{-3}$ RMSE	4	2.3



(a) (b)

Figure 8: Comparisons between *AG* and NID in higher-order interaction detection. (a) Comparison of top-ranked recall at different noise levels on the synthetic test suite (Table 1), (b) comparison of runtimes, where NID runtime with and without cutoff are both measured. NID detects interactions with top-rank recall close to the state-of-the-art *AG* while running orders of magnitude times faster.

5.4 Limitations

In higher-order interaction detection, our NID framework can have difficulty detecting interactions from certain forms of functions. Specifically, functions in which interacting variables have interlinked structure can cause problems for our framework to separate such structure. For example, a clique $x_1x_2 + x_1x_3 + x_2x_3$ or a ring $x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_1$ only contains pairwise interactions. When detecting pairwise interactions (Section 5.2), NID often obtains an AUC of 1. However, in higher-order interaction detection, the interlinked pairwise interactions are often confused for single higher-order interactions. This issue could mean that our higher-order interaction detection algorithm fails to separate interlinked pairwise interactions encoded in a neural network, or the network approximates interlinked low-order interactions as higher-order interactions.

In experiments on real-world datasets, our framework sometimes detected spurious interactions or missed interactions as a result of correlations between features; however, correlations are known to cause these problems for any interaction detection method (Sorokina et al., 2008; Lou et al., 2013).

6 Conclusion

We presented our NID framework, which detects statistical interactions in data without searching an exponential solution space of interactions candidates. The framework detects interactions by interpreting the weights of a feedforward neural network, which is trained to fit data. Our core insight is that interactions between features must be modeled at common hidden units, and we directly decode the weights according to this insight.

In future work, we plan to detect feature interactions by accounting for common units in intermediate hidden layers of feedforward networks. We would also like to use the perspective of interaction detection to interpret weights in other neural architectures, including deep networks.

REFERENCES

Claire Adam-Bourdarios, Glen Cowan, Cecile Germain, Isabelle Guyon, Balazs Kegl, and David Rousseau. Learning to discover: the higgs boson machine learning challenge. *URL https://higgsml.lal.in2p3.fr/documentation/*, 2014.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pp. 289–300, 1995.

Jacob Bien, Jonathan Taylor, and Robert Tibshirani. A lasso for hierarchical interactions. *Annals of statistics*, 41(3):1111, 2013.

PS Bullen, DS Mitrinović, and PM Vasić. Means and their inequalities, mathematics and its applications, 1988.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie 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*, pp. 1721–1730. ACM, 2015.

Yadolah Dodge. *The Oxford dictionary of statistical terms*. Oxford University Press on Demand, 2006.

Yingying Fan, Yinfei Kong, Daoji Li, and Jinchi Lv. Interaction pursuit with feature screening and selection. *arXiv preprint arXiv:1605.08933*, 2016.

Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2-3):113–127, 2014.

Ronald Aylmer Fisher. *Statistical methods for research workers*. Genesis Publishing Pvt Ltd, 1925.

Peter W Frey and David J Slate. Letter recognition using holland-style adaptive classifiers. *Machine learning*, 6(2):161–182, 1991.

Jerome H Friedman and Bogdan E Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, pp. 916–954, 2008.

G David Garson. Interpreting neural-network connection weights. *AI Expert*, 6(4):46–51, 1991.

ATC Goh. Back-propagation neural networks for modeling complex systems. *Artificial Intelligence in Engineering*, 9(3):143–151, 1995.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*, 2016.

Yotam Hechtlinger. Interpretation of prediction models using the input gradient. *arXiv preprint arXiv:1611.07634*, 2016.

Giles Hooker. Discovering additive structure in black box functions. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 575–580. ACM, 2004.

Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

Shiyu Liang and R Srikant. Why deep neural networks for function approximation? 2016.

Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 623–631. ACM, 2013.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pp. 2204–2212, 2014.

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

Sanjay Purushotham, Martin Renqiang Min, C-C Jay Kuo, and Rachel Ostroff. Factorized sparse learning models with interpretable high order feature interactions. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 552–561. ACM, 2014.

Steffen Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pp. 995–1000. IEEE, 2010.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144. ACM, 2016.

Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Daria Sorokina, Rich Caruana, and Mirek Riedewald. Additive groves of regression trees. *Machine Learning: ECML 2007*, pp. 323–334, 2007.

Daria Sorokina, Rich Caruana, Mirek Riedewald, and Daniel Fink. Detecting statistical interactions with additive groves of trees. In *Proceedings of the 25th international conference on Machine learning*, pp. 1000–1007. ACM, 2008.

Matus Telgarsky. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*, 2016.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

Kush R Varshney and Homa Alemzadeh. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *arXiv preprint arXiv:1610.01256*, 2016.

Ben James Winer, Donald R Brown, and Kenneth M Michels. *Statistical principles in experimental design*, volume 2. McGraw-Hill New York, 1971.

Thomas H Wonnacott and Ronald J Wonnacott. *Introductory statistics*, volume 19690. Wiley New York, 1972.

Simon Wood. *Generalized additive models: an introduction with R*. CRC press, 2006.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pp. 77–81, 2015.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. *arXiv preprint arXiv:1710.00935*, 2017.

A PROOF AND DISCUSSION FOR PROPOSITION 2

Given a trained feedforward neural network as defined in Section 2.3, we can construct a directed acyclic graph $G = (V, E)$ based on non-zero weights as follows. We create a vertex for each input features and hidden units in the neural network: $V = \{v_{\ell,i} | \forall i, \ell\}$, where $v_{\ell,i}$ be the vertex corresponding to the i-th hidden unit in the ℓ-th layer. Note that the final output y is not included. We create edges based on the non-zero entries in the weight matrices, i.e., $E = \{(v_{\ell-1,i}, v_{\ell,j}) \,|\, \mathbf{W}^{\ell}_{j,i} \neq 0, \forall i, j, \ell\}$. Note that under the graph representation, the value of any hidden unit is a function of parent hidden units. We will also use vertices and hidden units interchangeably.

Proposition 2 (Interactions at Common Hidden Units)**.** *Consider a feedforward neural network with input feature $x_i, i \in [p]$, where $y = \varphi(x_1, \ldots, x_p)$. For any interaction $\mathcal{I} \subset [p]$ in $\varphi(\cdot)$, there exists a vertex $v_{\mathcal{I}}$ in the associated directed graph such that $\mathcal{I}$ is a subset of the ancestors of $v_{\mathcal{I}}$ at the input layer (i.e., $\ell = 0$).*

Proof. We prove Proposition 2 by contradiction.

Let $\mathcal{I}$ be an interaction where there is no vertex in the associated graph which satisfies the condition. Then, for any vertex $v_{L,i}$ at the L-th layer, the value f_i of the corresponding hidden unit is a function of its ancestors at the input layer $\mathcal{I}_i$ where $\mathcal{I} \not\subset \mathcal{I}_i$.

Next, we group the hidden units at the L-th layer into non-overlapping subsets by the first missing feature with respect to the interaction $\mathcal{I}$. That is, for element i in $\mathcal{I}$, we create a index set $\mathcal{S}_i \in [p_L]$:

$$\mathcal{S}_i = \{j \in [p_L] | i \notin \mathcal{I}_j \text{ and } \forall i' < i, j \notin \mathcal{S}_{i'}\}.$$

Note that the final output of the network is a weighed summation over the hidden units at the L-th layer:

$$\varphi(\mathbf{x}) = b^y + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{S}_i} w^y_j f_j\left(\mathbf{x}_{\mathcal{I}_j}\right),$$

Since that $\sum_{j \in \mathcal{S}_i} w^y_j f_j\left(\mathbf{x}_{\mathcal{I}_j}\right)$ is not a function of x_i, we have that $\varphi(\cdot)$ is a function without the interaction $\mathcal{I}$, which contradicts with our assumption. □

The reverse of this statement, that a common descendant will create an interaction among input features, holds true in most cases. The existence of counterexamples is manifested when early hidden layers capture an interaction that is negated in later layers. For example, the effects of two interactions may be directly removed in the next layer, as in the case of the following expression: $\max\{w_1x_1 + w_2x_2, 0\} - \max\{-w_1x_1 - w_2x_2, 0\} = w_1x_1 + w_2x_2$. Such an counterexample is legitimate; however, due to random fluctuations, it is highly unlikely in practice that the w_1s and the w_2s from the left hand side are exactly equal.

B PAIRWISE INTERACTION STRENGTH VIA QUADRATIC APPROXIMATION

We can provide a finer interaction strength analysis on a bivariate ReLU function: $\max\{\alpha_1x_1 + \alpha_2x_2, 0\}$, where x_1, x_2 are two variables and α_1, α_2 are the weights for this simple network. We quantify the strength of the interaction between x_1 and x_2 with the cross-term coefficient of the best quadratic approximation. That is,

$$\beta_0, \ldots, \beta_5 = \underset{\beta_i, i=0,\ldots,5}{\operatorname{argmin}} \iint_{-1}^{1} \Big[\beta_0 + \beta_1x_1 + \beta_2x_2 + \beta_3x_1^2 + \beta_4x_2^2 + \beta_5x_1x_2 - \max\{\alpha_1x_1 + \alpha_2x_2, 0\}\Big]^2 dx_1\, dx_2.$$

Then for the coefficient of interaction $\{x_1, x_2\}$, β_5, we have that,

$$|\beta_5| = \frac{3}{4}\left(1 - \frac{\min\{\alpha_1^2, \alpha_2^2\}}{5\max\{\alpha_1^2, \alpha_2^2\}}\right)\min\{|\alpha_1|, |\alpha_2|\}. \tag{2}$$

Note that the choice of the region $(-1, 1) \times (-1, 1)$ is arbitrary: for larger region $(-c, c) \times (-c, c)$ with $c > 1$, we found that $|\beta_5|$ scales with c^{-1}. By the results of Proposition B, the strength of the interaction can be well-modeled by the minimum value between $|\alpha_1|$ and $|\alpha_2|$. Note that the factor before $\min\{|\alpha_1|, |\alpha_2|\}$ in Equation (2) is almost a constant with less than 20% fluctuation.

C PROOF FOR LEMMA 3

Lemma 3 (Neural Network Lipschitz Estimation)**.** *Let the activation function $\phi(\cdot)$ be a 1-Lipschitz function. Then the output y is $z_i^{(\ell)}$-Lipschitz with respect to $h_i^{(\ell)}$.*

Proof. For non-differentiable $\phi(\cdot)$ such as the ReLU function, we can replace it with a series of differentiable 1-Lipschitz functions that converges to $\phi(\cdot)$ in the limit. Therefore, without loss of generality, we assume that $\phi(\cdot)$ is differentiable with $|\partial_x \phi(x)| \leq 1$. We can take the partial derivative of the final output with respect to $h_i^{(\ell)}$, the i-th unit at the ℓ-th hidden layer:

$$\begin{aligned}\frac{\partial y}{\partial h_i^{(\ell)}} =& \sum_{j_{\ell+1},\ldots,j_L} \frac{\partial y}{\partial h_{j_L}^{(L)}} \frac{\partial h_{j_L}^{(L)}}{\partial h_{j_{L-1}}^{(L-1)}} \cdots \frac{\partial h_{j_{\ell+1}}^{(\ell+1)}}{\partial h_i^{(\ell)}} \\ =& \mathbf{w}^{y\top} \mathrm{diag}(\dot{\boldsymbol{\phi}}^{(L)}) \mathbf{W}^{(L)} \cdots \mathrm{diag}(\dot{\boldsymbol{\phi}}^{(\ell+1)}) \mathbf{W}^{(\ell+1)},\end{aligned}$$

where $\dot{\boldsymbol{\phi}}^{(\ell)} \in \mathbb{R}^{p_\ell}$ is a vector that

$$\dot{\phi}_k^{(\ell)} = \partial_x \phi\left(\mathbf{W}_{k,:}^{(\ell)} \mathbf{h}^{(\ell-1)} + b_k^{(\ell)}\right).$$

We can conclude the Lemma by proving the following inequality:

$$\left|\frac{\partial y}{\partial h_i^{(\ell)}}\right| \leq |\mathbf{w}^y|^\top \left|\mathbf{W}^{(L)}\right| \cdots \left|\mathbf{W}_{:,i}^{(\ell+1)}\right| = z_i^{(\ell)}.$$

The left-hand side can be re-written as

$$\sum_{j_{\ell+1},\ldots,j_L} w_{j_L}^y \dot{\phi}_{j_L}^{(L)} W_{j_L,j_{L-1}}^{(L)} \dot{\phi}_{j_{L-1}}^{(L-1)} \cdots \dot{\phi}_{j_{\ell+1}}^{(\ell+1)} W_{j_{\ell+1},i}^{(\ell+1)}.$$

The right-hand side can be re-written as

$$\sum_{j_{\ell+1},\ldots,j_L} \left|w_{j_L}^y\right| \left|W_{j_L,j_{L-1}}^{(L)}\right| \cdots \left|W_{j_{\ell+1},i}^{(\ell+1)}\right|.$$

We can conclude by noting that $|\partial_x \phi(x)| \leq 1$. □

D PROOF FOR THEOREM 4

Theorem 4 (Improving the ranking of higher-order interactions)**.** *Let $\mathcal{R}$ be the set of interactions proposed by Algorithm 1, let $\mathcal{I} \in \mathcal{R}$ be a d-way interaction where $d \geq 3$, and let $\mathcal{S}$ be the set of subset $(d-1)$-way interactions of $\mathcal{I}$ where $|\mathcal{S}| = d$. Assume that for any hidden unit j which proposed $s \in \mathcal{S} \cap \mathcal{R}$, $\mathcal{I}$ will also be proposed at the same hidden unit, and $w_j(\mathcal{I}) > \frac{1}{d} w_j(s)$. Then, one of the following must be true: a) $\exists s \in \mathcal{S} \cap \mathcal{R}$ ranked lower than $\mathcal{I}$, i.e., $w(\mathcal{I}) > w(s)$, or b) $\exists s \in \mathcal{S}$ where $s \notin \mathcal{R}$.*

Proof. Suppose for the purpose of contradiction that $\mathcal{S} \subseteq \mathcal{R}$ and $\forall s \in \mathcal{S}$, $w(s) \geq w(\mathcal{I})$. Because $w_j(\mathcal{I}) > \frac{1}{d} w_j(s)$,

$$w(\mathcal{I}) = \sum_{s \in \mathcal{S} \cap \mathcal{R}} \sum_{j \text{ propose } s} z_j w_j(\mathcal{I}) > \frac{1}{d} \sum_{s \in \mathcal{S} \cap \mathcal{R}} \sum_{j \text{ propose } s} z_j w_j(s) = \frac{1}{d} \sum_{s \in \mathcal{S} \cap \mathcal{R}} w(s).$$

Since $\forall s \in \mathcal{S}$, $w(s) \geq w(\mathcal{I})$,

$$\frac{1}{d} \sum_{s \in \mathcal{S} \cap \mathcal{R}} w(s) \geq \frac{1}{d} \sum_{s \in \mathcal{S} \cap \mathcal{R}} w(\mathcal{I})$$

Since $\mathcal{S} \subseteq \mathcal{R}$, $|\mathcal{S} \cap \mathcal{R}| = d$. Therefore,

$$\frac{1}{d} \sum_{s \in \mathcal{S} \cap \mathcal{R}} w(\mathcal{I}) \geq \frac{1}{d} w(\mathcal{I}) d \geq w(\mathcal{I}),$$

which is a contradiction.

□

E ROC CURVES

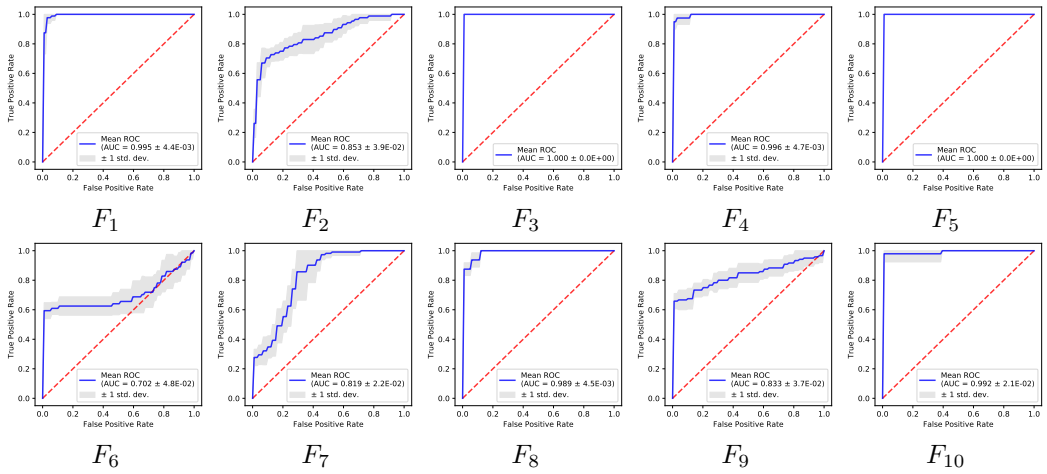


Figure 9: ROC curves of NID corresponding to Table 2

F LARGE p EXPERIMENT

We evaluate our approach in a large p setting with pairwise interactions using the same synthetic function as in Purushotham et al. (2014). Specifically, we generate a dataset of n samples and p features $\{(\mathbf{X}^{(i)}, y^{(i)})\}$ using the function

$$y^{(i)} = \boldsymbol{\beta}^\top \mathbf{X}^{(i)} + \mathbf{X}^{(i)\top} \mathbf{W} \mathbf{X}^{(i)} + \epsilon^{(i)},$$

where $\mathbf{X}^{(i)} \in \mathbb{R}^p$ is the ith instance of the design matrix $\mathbf{X} \in \mathbb{R}^{p \times n}$, $y^{(i)} \in \mathbb{R}$ is the ith instance of the response variable $\mathbf{y} \in \mathbb{R}^{n \times 1}$, $\mathbf{W} \in \mathbb{R}^{p \times p}$ contains the weights of pairwise interactions, $\boldsymbol{\beta} \in \mathbb{R}^p$ contains the weights of main effects, $\epsilon^{(i)}$ is noise, and $i = 1, \ldots, n$. $\mathbf{W}$ was generated as a sum of K rank one matrices, $\mathbf{W} = \sum_{k=1}^{K} \boldsymbol{a}_k \boldsymbol{a}_k^\top$.

In this experiment, we set $p = 1000$, $n = 1\text{e}4$, and $K = 5$. $\mathbf{X}$ is normally distributed with mean 0 and variance 1, and $\epsilon^{(i)}$ is normally distributed with mean 0 and variance 0.1. Both $\boldsymbol{a}_k$ and $\boldsymbol{\beta}$ are sparse vectors of 2% nonzero density and are normally distributed with mean 0 and variance 1. We train *MLP-M* with the same hyperparameters as before (Section 5.1) but with a larger architecture of five hidden layers, with first-to-last layers sizes of 500, 400, 300, 200, and 100.

From this experiment, we obtain a pairwise interaction strength AUC of 0.984, measured in the same way as the AUCs in Table 2. The corresponding ROC curve is shown in Figure 10.

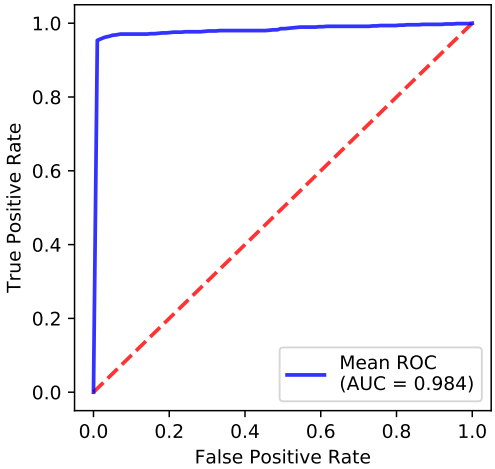


Figure 10: ROC curve for large p experiment