Towards Interaction Detection Using Topological Analysis on Neural Networks

Zirui Liu
Dept. of Computer Science
Texas A&M University
College Station, TX
tradigrada@tamu.edu

Qingquan Song
Dept. of Computer Science
Texas A&M University
College Station, TX
song3134@tamu.edu

Kaixiong Zhou
Dept. of Computer Science
Texas A&M University
College Station, TX
zkxiong@tamu.edu

Ting-Hsiang Wang
Dept. of Computer Science
Texas A&M University
College Station, TX
thwang1231@tamu.edu

Ying Shan
Tencent Company
Beijing, China
yingsshan@tencent.com

Xia Hu
Dept. of Computer Science
Texas A&M University
College Station, TX
hu@cse.tamu.edu

Abstract

Detecting statistical interactions between input features is a crucial and challenging task. Recent advances demonstrate that it is possible to extract learned interactions from trained neural networks. It has also been observed that, in neural networks, any interacting features must follow a strongly weighted connection to common hidden units. Motivated by the observation, in this paper, we propose to investigate the interaction detection problem from a novel topological perspective by analyzing the connectivity in neural networks. Specially, we propose a new measure for quantifying interaction strength, based upon the well-received theory of persistent homology. Based on this measure, a **P**ersistence **I**nteraction **D**etection (PID) algorithm is developed to efficiently detect interactions. Our proposed algorithm is evaluated across a number of interaction detection tasks on several synthetic and real world datasets with different hyperparameters. Experimental results validate that the PID algorithm outperforms the state-of-the-art baselines.

1 Introduction

Statistical interaction describes a subset of input features that interact with each other to have an effect on outcomes. For example, using Phenelzine together with Fluoxetine may lead to serotonin syndrome [1]. Interaction detection problem is to quantify the influence of any subset of input features that may potentially be an interaction. The quantified influence in the problem is called interaction strength. With detected interactions, we may formulate hypotheses that could lead to new data collection and experiments. Traditional methods often need to conduct individual tests for all interaction candidates [2, 3] or pre-specify all functional forms of interests [4, 5]. Recent efforts have been dedicated to extracting learned interactions in neural networks by designing measures for quantifying interaction strength based on predefined conditions in a heuristic way [6]. It has been shown to be an effective way to detect interactions and avoid the drawbacks of traditional methods.

One key observation in the state-of-the-art methods is that any interacting features must follow strongly weighted connections to a common hidden unit before reaching the final output layer [6, 7]. Based on this, the strength of interactions can be modeled by the connectivity between these interacting features and output units of a trained neural network. This motivates us to solve the problem from a novel topological perspective. Specifically, our framework builds upon computational

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

techniques from algebraic topology, specially the persistent homology, which has been shown beneficial for several deep learning models [8–10]. The main advantages of utilizing persistent homology are twofold. First, it provides us a rigorous mathematical framework for analyzing the connectivity in a trained neural network. Second, persistent homology can be used to quantify the importance of each connected component in the neural network, and the connectivity between interacting features and output units are characterized by these connected components.

However, persistent homology cannot be directly applied to quantify the interaction strength from the importance of connected components that link interacting features to units in the final output layers. Also, an interaction is a subset of input features, which is not within the scope of persistent homolgy. The key challenge remains to define a measure for quantifying the interaction strength, which should provide meaningful insights while maintaining theoretical generality.

In this paper, we show that the key concepts of *persistence diagrams* in persistent homology theory can be extended to interactions for tackling the challenges. Specifically, we propose a new measure for quantifying interaction strength, which is computed to reflect the connectivity between interacting features and output units in a neural network. Based on the measure, we propose Persistence Interaction Detection (PID), a framework that can efficiently extract interactions from neural networks. We also prove that our framework is locally stable, meaning that PID is not sensitive to the perturbation of weights in neural networks. Formally, our contributions are as follows:

- We formulate the interaction detection problem as a topology problem. Based on the persistent homology theory, we propose a new measure for quantifying interaction strength by analyzing the topology of neural networks. We then provide analysis for the measure from different perspectives.
- We derive an efficient algorithm to calculate the proposed interaction strength measure. Also, we theoretically analyze the local stability of our proposed framework.
- The proposed PID framework demonstrates strong performance across different tasks, network architectures, hyperparameter settings, and datasets.

2 Preliminaries

We first introduce the notations and give the formal definition of feature interactions. Based on the notations, we introduce concepts of the filtration and persistence diagrams. We then show how to build filtration for neural networks in 2.2, serving as the preliminary of our proposed method.

2.1 Problem Formulation and Notations

We denote vectors with boldface lowercase letters (e.g., $\mathbf{x}$, $\mathbf{w}$), matrices with boldface capital letters (e.g. $\mathbf{W}$), and scalars with lowercase letters (e.g., a). We use x_i to represent the i-th entry of vector $\mathbf{x}$, and W_{ij} to denote the entry in the i^{th} row and j^{th} column of $\mathbf{W}$. The transpose of a matrix or a vector is denoted as $\mathbf{W}^\top$ or $\mathbf{x}^\top$. For a set $\mathcal{S}$, its cardinality is denoted by $|\mathcal{S}|$. We use $\mathcal{S}\backslash i$ to denote the set $\{j | j \in \mathcal{S} \text{ and } j \neq i\}$. Let $\mathbf{x} \in \mathbb{R}^d$ be the feature vector. An interaction $\mathcal{I}$ is a set of interacting features, where $|\mathcal{I}| \geq 2$. A K-order interaction $\mathcal{I}$ satisfies $|\mathcal{I}| = K$. A high-order interaction is an interaction whose order ≥ 3. We will write $\mathbf{x}^{\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|}$ as the feature vector selected by $\mathcal{I}$.

Consider a feed-forward neural network (FNN) with L hidden layers (e.g., an MLP). Let p_l be the number of hidden units at the l^{th} layer. The input features are treated as the 0^{th} layer and $p_0 = d$ is the number of input features. The l^{th} layer weight matrix is denoted by $\mathbf{W}^{(l)} \in \mathbb{R}^{p_{l-1} \times p_l}$. Given a FNN with weights $\{\mathbf{W}^{(i)}\}_{i=1}^{L}$, its equivalent weighted directed acyclic graph $\mathcal{G}(V, E)$ can be constructed as follows: We create a vertex for each hidden unit in the neural network and consequently the set of all vertices: $V = \{v_{l,i} | \forall\, l, i\}$, where $v_{l,i}$ represents the i^{th} hidden unit at the l^{th} layers; and we assign weight $W_{i,j}^{(l)}$ to each edge in $E = \{(v_{l-1,i}, v_{l,j}) | \forall l, i, j\}$.

In this work, we focus on detecting non-additive interactions. The non-additive interaction is formally defined in Definition 1. We remark that detecting "additive interactions" is a trivial task because any "additive interactions" can be decomposed to the sum of two terms, and non-additive interactions are those which cannot be further decomposed.

Definition 1 (Non-additive interactions [3, 11]). *Let $\{0, ..., d-1\}$ denotes the input feature set. Given a function f: $\mathbb{R}^d \to \mathbb{R}$ and a feature vector $\mathbf{x} = (x_1, ..., x_d)$, f shows no non-additive interaction of $\{x_i, x_j\}$ if f can be expressed as the sum of two functions, $f_{\setminus i}$ and $f_{\setminus j}$, where $f_{\setminus i}$ is a function which does not depend on x_i and $f_{\setminus j}$ is a function which not depend on x_j:*

$$f(\mathbf{x}) = f_{\setminus i}(\mathbf{x}^{\{0,...,d-1\}\setminus i}) + f_{\setminus j}(\mathbf{x}^{\{0,...,d-1\}\setminus j}).$$

For example, in the function of $\pi^{x_0 x_1} + \log(x_1 + x_2 + x_4)$, there is a pairwise interaction $\{0, 1\}$ and a 3-order interaction $\{x_1, x_2, x_4\}$. In contrast, $\{x_0, x_1, x_2, x_4\}$ is a spurious interaction. The goal of interaction detection algorithms is to map models into a set of their learned interaction candidates associated with interaction strength. Ideally, a larger value of interaction strength should indicate the true interaction instead of a spurious interaction.

2.2 Persistent Homology on Neural Networks

Persistent homology is an algebraic method for identifying the most prominent connectivity characterizing a geometric object, which is widely used in medical imaging and geometric modeling [12, 13]. In this paper, the object we studied is a weighted directed graph $\mathcal{G}(V, E)$ corresponding to a trained feed-forward neural network. In topology, connected components represent the connectivity of the graph. We can apply persistent homology theory to quantify the importance of each connected component in $\mathcal{G}$. To be specific, $(\mathcal{G}, \phi)$ is called a *size pair* [14], where ϕ is a *measureing function* [15]. The role of ϕ is to take into account the connective properties of $\mathcal{G}$. The λ-threshold set of $(\mathcal{G}, \phi)$ is defined as follows:

$$L^\lambda = \{x | x \in E, \phi(x) \geq \lambda\},$$

where x is the edge of $\mathcal{G}$. The measuring function $\phi : E \to \mathbb{R}$ maps a specific edge to a real number.

Definition 2 (Filtration [16]). *Without loss of generality, suppose $\lambda_1 > \lambda_2$, if the corresponding threshold sets satisfy $L^{\lambda_1} \subseteq L^{\lambda_2}$, then ϕ is non-decreasing over $\mathcal{G}$. Given $(\mathcal{G}, \phi)$, where ϕ is non-decreasing over $\mathcal{G}$, and a set of thresholds follow $\lambda_0 \geq \lambda_1 \geq ... \geq \lambda_n$, the collections of threshold sets $L^{\lambda_0} \subseteq L^{\lambda_1} \subseteq ... \subseteq L^{\lambda_n}$ is called a **filtration** of $(\mathcal{G}, \phi)$.*

We propose to build the filtration for FNNs and define the measuring function as follows. Let $\mathcal{W}$ be the set of weights. Given $\mathcal{W}$ of a trained feed-forward neural network such that $w_{max} := \max_{w \in \mathcal{W}} |w|$ and $\mathcal{W}' := \{|w|/w_{max} | w \in \mathcal{W}\}$, where $\mathcal{W}'$ is indexed in non-ascending order, namely $1 = w'_0 \geq w'_1 \geq ... \geq w'_n \geq 0$. The weights associated with edges reflect the connectivity between vertices in the networks. Similar to [17], the measuring function ϕ for $\mathcal{G}$ is defined as $\phi((v_{l-1,i}, v_{l,j})) = |W^{(l)}_{i,j}|/w_{max}, \forall (v_{l-1,i}, v_{l,j}) \in E$, which represents the edge strength. The sorted weights are used as λ in Definition 2. Consequently, the filtration can be constructed as $\mathcal{G}^{w'_0} \subseteq \mathcal{G}^{w'_1} \subseteq ...$, where $\mathcal{G}^{w'_i} = (V, \{(u, v) | (u, v) \in E \wedge \phi((u, v)) \geq w'_i\})$. We remark that $\mathcal{G}^\lambda$ is both a subgraph of $\mathcal{G}$ and a λ-threshold set of size pair $(\mathcal{G}, \phi)$. $\mathcal{G}^{w'_0}$ is the sub-graph with exact one edge which has greatest weight. As shown in Figure 1, when the thresholds are decreased, edges are added into the sub-graph and vertices will be connected. It is summarized in Figure 1.

The interpretation of $\mathcal{G}^\lambda$ is that, $\mathcal{G}^\lambda$ is the image of $\mathcal{G}$ at different spatial resolution. Edges with larger weights, which indicates stronger connectivity, will appear over a wide range of spatial scales. As the threshold of filters decreases, edges with smaller weights, which indicates weaker connectivity, will start to pass the filter and provide detailed information of $\mathcal{G}$. In the filtration process, these gradually added edges will form different connected components. From persistent homology theory, persistent connected components, which are detected over a wide range of spatial scales, are more representative for the connectivity pattern of $\mathcal{G}$ [18]. Based on this, persistence diagram is a computational tool for quantifying the importance of these emerged connected components.

Given the size pair $(\mathcal{G}, \phi)$, when we decrease λ, connected components can be *created* (new edges are added, forming new components) or *destroyed* (two connected components joining together). For each connected component i, the threshold causes the birth of i is called the *birth* time b_i and the threshold causes the death of i is called the *death* time d_i. The persistence diagram tracks these changes and represents creation and destruction of i as a tuple (b_i, d_i). It quantifies the importance of each connected component by its lifetime (persistence).

Persistence Diagrams Given the filtration of a size pair $(\mathcal{G}, \phi)$, with the *birth* time b_i and the *death* time d_i of each connected component i appearing in the filtration, the collection of the *birth* time and

the *death* time tuple $\mathcal{D} = \{(b_i, d_i)|\forall i \text{ appears in the filtration}\}$ is called the **persistence diagrams** of $(\mathcal{G}, \phi)$. The **persistence** of i is $\text{per}(i) = |b_i - d_i|$.

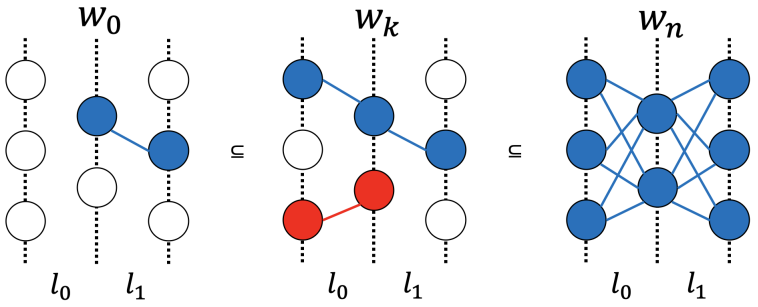


Figure 1: The filtration of a network with two layers. The color scheme illustrates the connected components. The filtration process is represented by colouring connected components that are created or merged when the respective weights are greater than or equal to the threshold w_i.

3 Persistence Interaction Detection

In this section, we present the proposed PID framework for detecting feature interactions in neural networks. The key intuition of our proposed method is to formulate the interaction detection as a topology problem. That is to find the long-lived connectivity between interactions and outputs in the neural network over a wide range of spatial scales. Based on this, we propose a new measure for quantifying interaction strength (section 3.1). Subsequently, we derive our proposed Persistence Interaction Detection algorithm for calculating the measure efficiently (section 3.2). We then give the stability analysis for our proposed algorithm (section 3.3). To avoid creating confusion in terminology, by "interaction", we mean a subset of input features that satisfies Definition 1 in the rest of the paper.

3.1 Persistence As a Measure For Interaction Strength

The concepts of the *birth* time and the *death* time are originally defined for connected components in persistent homology theory. However, the persistence of connected components only implies the importance of themselves. For a particular interaction $\mathcal{I}$, we cannot directly obtain the importance of $\mathcal{I}$ from the persistence of connected components that contain $\mathcal{I}$. Also, an interaction $\mathcal{I}$ is a subset of input features that is not within the scope of persistent homolgy. In this subsection, we will extend these concepts to interactions for deriving a new measure for quantifying interaction strength.

From Definition 1, an interaction is a set of associate features that have an effect on the output. Inspired by persistent homology, we can model this effect by the connectivity between interactions and output units in neural networks. Informally, the *birth* time of an interaction is when there exists a path connecting it to the final output layer, and the *death* time is when the path is also connected to any additional input feature in the filtration. After extending concepts of the $birth$ time and the $death$ time to interactions, we can obtain persistence diagrams of interactions and the interaction strength can be quantified from the lifetime of the connectivity. We first give the definition for the connectivity strength between the interactions and the units in the final output layer in Definition 3. Based on the quantified connectivity, we formally define the *birth* time and the *death* time of an interaction.

Definition 3 ($\langle\phi = \lambda\rangle$-connected)**.** *Let $(\mathcal{G}, \phi)$ be the corresponding size pair and $\mathcal{G}^{w'_0} \subseteq \mathcal{G}^{w'_1} \subseteq ...$ be the filtration of a neural network, respectively; and $\{0, ..., d-1\}$ denotes the set of input features. For a feature subset $\mathcal{I}$ and a real-number threshold λ, we call $\mathcal{I}$ and the final output units are $\langle\phi = \lambda\rangle$-connected if: first, there exists a connected component $A \subseteq \mathcal{G}^{\lambda}$ containing $\mathcal{I}$ and the final output units; second, for any such connected component A, $\forall i \in \{0, ..., d-1\}\backslash\mathcal{I}\}$, it satisfies $i \notin A$.*

Persistence diagrams of interactions Given the threshold λ_b, suppose: The feature subset $\mathcal{I}$ and the final output unit are $\langle\phi = \lambda_b\rangle$-connected and, $\forall\lambda_i \geq \lambda_b$, $\mathcal{I}$ and the final output unit are not $\langle\phi = \lambda_i\rangle$-connected, then we call λ_b the *birth* time of $\mathcal{I}$. Correspondingly, the *death* time λ_d of $\mathcal{I}$ is that $\forall\lambda_i \leq \lambda_d$, $\mathcal{I}$ and the outputs become not $\langle\phi = \lambda_i\rangle$-connected, i.e., interaction $\mathcal{I}$ no longer exists

due to the addition of other input features. The collection of the *birth* time and the *death* time tuple $\mathcal{D} = \{(b_{\mathcal{I}}, d_{\mathcal{I}}) | \forall \mathcal{I} \subseteq \{0, ..., d-1\}\}$ is called the **persistence diagrams** of interactions.

After defining the *birth* time and the *death* time of an interaction $\mathcal{I}$, we can quantify its interaction strength by its persistence. We remark that the aforementioned process creates new interaction candidates by associating new features with existing interaction candidates. Some interaction candidates might never born. An example of the persistence of interactions is illustrated in Figure 2. Let $\mathbf{x} = [x_1, x_2, x_3, x_4]$. $y = x_1^{x_2} + \frac{x_3 x_4}{1000}$. We train a neural network f to minimize the loss $\mathcal{L}(f(x), y) + \mathcal{R}(f)$, where $\mathcal{L}$ is the mean square error and $\mathcal{R}$ is the regularization term. Suppose $w_0' \geq ... \geq w_9'$ are the top ten largest weights in $\mathcal{W}'$. Then the interaction $\{x_1, x_2\}$ and y are $\langle \phi = w_3' \rangle$-connected because of the connected component marked in red. The birth time and the death time of $\{x_1, x_2\}$ are w_3' and w_6', respectively. And the death time of $\{x_1, x_2\}$ marks the birth time of $\{x_1, x_2, x_3\}$.

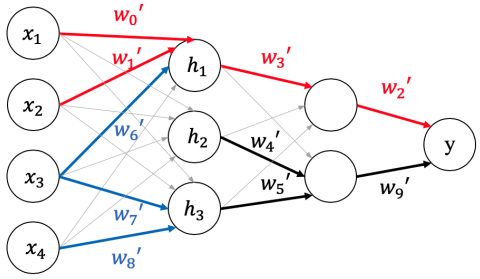


Figure 2: An example for illustrating persistence of interactions.

Intuitively, strength can be quantified in terms of the minimal "amount of change" necessary to eliminate a learned interaction in NNs. The "amount of change" is referred to the distance between changed weights and original weights. From this perspective, the proposed measure is the minimal "amount of change" to eliminate the $\langle \phi = \lambda \rangle$-connectivity between interactions and outputs. For example, if we change w_6' to be as large as w_1', then the input feature x_2 and x_3 will be simultaneously added to $\{x_1\}$ to form $\{x_1, x_2, x_3\}$, thus $\{x_1, x_2\}$ will never born. The persistence of $\{x_1, x_2\}$ is the gap between the red connected component's smallest weight and w_6' (i.e., $|w_3' - w_6'|$). This gap is the minimal amount of change to eliminate the $\langle \phi = w_3' \rangle$-connectivity between $\{x_1, x_2\}$ and y.

3.2 Ranking Interactions Using PID

In this subsection we present our PID framework that calculates the proposed measure efficiently. To detect these $\langle \phi = \lambda \rangle$-connectivity between interactions and outputs in Definition 3. We define the mask matrix $\mathbf{M}_{(l)}^{\lambda} \in \mathbb{R}^{p_{l-1} \times p_l}$ for the l^{th} layer as

$$[\mathbf{M}_{(l)}^{\lambda}]_{i,j} = \begin{cases} 1, & \text{if} \quad \phi((v_{l-1,i}, v_{l,j})) \geq \lambda. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The aggregated mask matrix $\mathbf{M}^{\lambda} \in \mathbb{R}^{p_L \times d}$ are defined as:

$$\mathbf{M}^{\lambda} = (\mathbf{M}_{(L)}^{\lambda})^{\top} \cdot (\mathbf{M}_{(L-1)}^{\lambda})^{\top} \cdots (\mathbf{M}_{(1)}^{\lambda})^{\top}. \tag{2}$$

Lemma 1 (Proof in Appendix B). *Let $\{0, ..., d-1\}$ denotes the input feature set, and $\mathbf{M}^{\lambda}$ denotes the aggregated mask matrix corresponding to threshold λ, where the r^{th} row of $\mathbf{M}^{\lambda}$ is denoted as $\mathbf{m}_r^{\lambda} \in \mathbb{R}^d$. The feature subset $\mathcal{I}$ and the corresponding r^{th} unit at the final output layer are $\langle \phi = \lambda \rangle$-connected if all elements in $[\mathbf{m}_r^{\lambda}]^{\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|}$ are non-zero and all other elements in $[\mathbf{m}_r^{\lambda}]^{\{0,...,d-1\} \setminus \mathcal{I}}$ are zero, where $[\mathbf{m}_r^{\lambda}]^{\mathcal{I}}$ is the subvector of $\mathbf{m}_{\mathbf{r}}^{\lambda}$ selected by $\mathcal{I}$.*

As pointed out in [19], different neurons are activated by different patterns (patterns are exactly interactions of raw input features). This indicates that we should generate interaction candidates for each neuron separately. With Lemma 1, we can detect the $\langle \phi = \lambda \rangle$-connectivity between interactions and units in the output layer. However, only care the $\langle \phi = \lambda \rangle$-connectivity between them will ignore the difference between neurons. For example, in Figure 2, all neurons share common interaction candidates in the aforementioned process. The edges gradually added by the filtration process sequentially create the interaction candidates $\{x_1, x_2\}$, $\{x_1, x_2, x_3\}$ and $\{x_1, x_2, x_3, x_4\}$. $\{x_3, x_4\}$ will not be considered because x_3 has been merged with $\{x_1, x_2\}$ when they meet at h_1. But clearly, x_3 and x_4 might be a potential interaction candidate because the activation pattern of h_3 is largely determined by x_3 and x_4. To generate interaction candidates for each neuron at a particular layer l,

we decompose $\mathbf{M}^{\lambda}$ into $\mathbf{M}_{(l)}^{\lambda_{up}} \in \mathbb{R}^{p_L \times p_l}$ and $\mathbf{M}_{(l)}^{\lambda_{down}} \in \mathbb{R}^{p_l \times d}$, where

$$\begin{cases} \mathbf{M}_{(l)}^{\lambda_{up}} = (\mathbf{M}_{(L)}^{\lambda})^{\top} \cdots (\mathbf{M}_{(l)}^{\lambda})^{\top}. \\ \mathbf{M}_{(l)}^{\lambda_{down}} = (\mathbf{M}_{(l-1)}^{\lambda})^{\top} \cdots (\mathbf{M}_{(1)}^{\lambda})^{\top}. \end{cases} \tag{3}$$

We can obtain the connectivity between a particular neuron r at layer l and units in the final output layer from $\mathbf{M}_{(l)}^{\lambda_{up}}$ (by viewing the layer l as the input layer in the Lemma 1). Similarly, the connectivity between the neuron r and input features can be inferred from $\mathbf{M}_{(l)}^{\lambda_{down}}$. For each neuron r at layer l, we generate interaction candidate $\mathcal{I}$ for r only if: first, $\mathcal{I}$ and r are connected; second, r and units in output layer are connected. It is summarized in Figure 3. For example, in Figure 2, if the layer l is set to the first layer, then PID will first generate $\{x_3, x_4\}$ for h_3 because $\{x_3, x_4\} - h_3 - y$ are connected once the threshold achieves w_9'. For the same interaction candidate generated at different neurons, we aggregate their persistence. We list the full algorithm in Appendix A. From the topological perspective, we model how interactions influence a particular neuron at layer l, as well as how this neuron influences units in the final output layer by the quantified connectivity between them. p in algorithm 1 is the norm of the persistence diagram. The p-norm is known to be a stable summary for persistence diagrams [20].

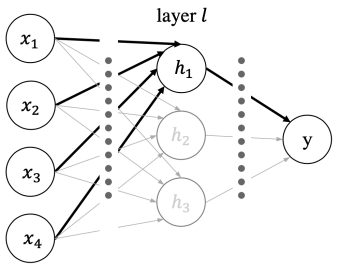


Figure 3: Illustration of PID.

3.3 Stability of Persistence Interaction Detection

An interaction detection algorithm should not be sensitive to the perturbation of weights, e.g., training neural networks with one or two extra epochs should only change the proposed interaction strength a little. We call that these insensitive algorithms are locally stable. It should be noted that local stability is a necessary condition for fidelity. If the algorithm gives totally different results after training one extra epoch, we cannot tell which one is correct, especially concerning that there are no ground truth labels for interactions in real world datasets. We will show our method is theoretically locally stable.

For two feed forward neural networks f and g with exactly same architecture, let $\mathcal{G}_f(V, E)$ and $\mathcal{G}_g(V, E)$ be their corresponding weighted graph, respectively. We denote the measuring function for f and g as ϕ_f and ϕ_g, respectively. For all interaction candidate $\mathcal{I}$ that are both detected in f and g by Algorithm 1, we denote the interaction strength of $\mathcal{I}$ corresponding to f and g as $\rho_f(\mathcal{I})$ and $\rho_g(\mathcal{I})$ respectively. We propose the following Theorem:

Theorem 1 (Proof and empirical analysis in Appendix C)**.** *Let $\delta = \max_{e \in E} |\phi_f(e) - \phi_g(e)|$ be the magnitude of perturbation. For all interaction candidate $\mathcal{I}$ that are both detected in f and g by Algorithm 1, it satisfies $|\rho_f(\mathcal{I}) - \rho_g(\mathcal{I})| \leq C\delta$.*

Theorem 1 only states the stability for interaction candidates that are detected by both f and g. We note that this is the common situation when the perturbation magnitude is small. However, there might exists the corner case where there are interaction candidates that are only detected in one network, but not the other. We also show the proof that this case only happens if the perturbation magnitude δ is greater than a threshold in Appendix C.

4 Experiments

Our experiments attempt to answer the following research questions: (**Q1**) How effective is PID in detecting true interactions (Section 4.1)? (**Q2**) Is the algorithm sensitive to hyperparameters and different architectures (Appendix E)? (**Q3**) Is considering these detected interactions beneficial for machine learning models (Section 4.2)? (**Q4**) Can PID detect extremely high-order interactions (Section 4.3)? We remark the norm p in Algorithm 1 is set to 2 across all experiments, which captures the Euclidean distance of points in persistence diagrams [20]. The other experiment-specific settings are described in respective sections.

Table 1: AUC of pairwise interaction strengths proposed by PID and baselines on the synthetic functions (Table 4). ANOVA, HierLasso, and RuleFit are deterministic.

	ANOVA	HierLasso	RuleFit	AG	NID	PID
$F_1(x)$	0.992	1.00	0.754	**1±0.0**	0.985±6.3e−3	0.986±4.1e−3
$F_2(x)$	0.468	0.636	0.698	**0.88±1.4e−2**	0.776±4.3e−2	0.804±5.7e-2
$F_3(x)$	0.657	0.556	0.815	**1±0.0**	**1.0±0.0**	**1.0±0.0**
$F_4(x)$	0.563	0.634	0.689	**0.999±1.4e-3**	0.916±6.3e−2	0.935±3.9e−2
$F_5(x)$	0.544	0.625	0.797	0.67±5.7e-2	0.997±8.9e−3	**1.0±0.0**
$F_6(x)$	0.780	0.730	0.811	0.64±1.4e-2	0.999±3.3e−3	**1.0±0.0**
$F_7(x)$	0.726	0.571	0.666	0.81±4.9e-2	0.880±2.6e−2	**0.888±2.8e−2**
$F_8(x)$	0.929	0.958	0.946	0.937±1.4e-3	**1.0±0.0**	**1.0±0.0**
$F_9(x)$	0.783	0.681	0.584	0.808±5.7e-3	0.968±2.3e−2	**0.972±2.9e−2**
$F_{10}(x)$	0.765	0.583	0.876	1.0±0.0	**0.989±3.0e−2**	0.987±3.5e−2
average	0.721	0.698	0.764	0.87±1.4e-2	0.951±7.0e−2	**0.957±6.2e−2**

4.1 Pairwise Interaction Detection on Synthetic Data

Since there are no ground-truth labels for interactions in real world datasets, to answer **Q1** and **Q2**, we utilize ten synthetic datasets that contain a mixture of pairwise interactions and higher-order interactions, as shown in the Appendix E.1. For higher-order interactions, we tested their pairwise subsets as in [3, 6, 21]. All ten datasets and MLP structures are the same as those in [6]. The detailed experimental settings can be found in Appendix E.1. The pairwise interaction strength of $\{i, j\}$ is obtained by aggregating the strength of all interaction candidates proposed by PID which contain $\{i, j\}$. The layer l in Algorithm 1 is set to the first layer because the neural network naturally separates different interactions in the first hidden layer [6, 7] (see Figure 6 and Figure 7 in Appendix E.2).

We compared the proposed PID with several strong existing algorithms in the interaction detection literature, including ANOVA [2], Hierarchical lasso (HierLasso) [4], RuleFit [11], Additive Groves (AG) [3], and Neural Interaction Detection (NID) [6]. Because both PID and NID detect learned interactions from MLPs in a post-hoc way, we apply the NID and PID on the same MLPs for fair comparison. We ran ten trials of AG, NID, and PID on each dataset and removed two trials with the highest and lowest AUC scores. The AUC scores of interaction strength proposed by baseline methods and PID are shown in Table 1. The heat map of pairwise interaction strength and a detailed analysis about main effects are in Appendix E.2. Here we provide only the general results.

In general, the AUCs of AG and PID are close, except for F_5, F_6, and F_8, where PID significantly outperforms AG. This may be caused by the limitations in the AG's model capacity, which is tree-based [6]. When comparing the AUCs of PID and NID, the AUCs of PID are comparable or better. We note that PID considers connectivity of the entire NN. In contrast, NID leverages weights beyond the first hidden layer to obtain the maximum gradient magnitude of the hidden units in the first hidden layer, loosing some information encoded in latter layers in the process. Hence, the similar results of NID and PID are likely because the latter layers played lesser roles in this specific setting. However, we remark PID constantly outperformed NID with various settings, as shown in Appendix E.3, Figure 8, 9, and 10. To answer **Q2**, we also compare the result based on MLPs with different architectures (Appendix E.3 Figure 8) and regularization strength (Appendix E.3 Figure 10, Figure 9). In general, both NID and PID are insensitive to the architecture of MLPs, and both are sensitive to the regularization strength. A possible reason is that the connectivity between hidden units of a trained MLP is significantly influenced by regularization strength. We show that the AUCs of PID are better than those of NID under all different settings (Appendix E.3).

4.2 Automatic Feature Engineering

Intricate feature engineering often plays deterministic roles in winning solutions of Kaggle competitions [22]. In this regard, interaction detection algorithms are invaluable in that they reveal knowledge about the data. A reasonable question is, can different machine learning models benefit from the knowledge to alleviate the need for hand-crafted feature engineering (**Q3**)? We try to answer it by integrating these detected interactions with the original input features and then check the performance gain of models trained on this augmented data.

Table 2: Comparing the quality of features automatically generated by interaction detection algorithms. The "Original" column shows the results of random forest built without using synthetic features.

Dataset	Original	Random	NID	PID
Amazon Employee	0.8378±0.0046	0.7780±0.0575	0.8321±0.0299	**0.8460±0.0079**
Higgs Boson	0.7421±0.0019	0.7421±0.0192	**0.7422±0.0017**	**0.7422±0.0017**
Creditcard	0.9555±0.0390	0.9579±0.0377	0.9607±0.0333	**0.9625±0.0354**
Spambase	0.9680±0.0085	0.9692±0.0076	0.9724±0.0065	**0.9738±0.0063**
Diabetes	0.8077±0.0334	0.8078±0.0335	0.8044±0.0335	**0.8101±0.0349**

We compare our PID and NID on five real world binary classification datasets. The statistics of these datasets are shown in Appendix F.1 Table 6. Following [23, 24], we explicitly construct synthetic features for each detected interaction candidates and combine these synthetic features with the original feature set. The synthetic feature for interaction $\mathcal{I}$ is the Cartesian product among features in $\mathcal{I}$. The details are described in Appendix F.1. We construct synthetic features for the top ten interactions candidates according to interaction strength. Because of the excellent performance and efficiency of random forest on tabular datasets, similar to [25, 26], we choose the random forest as our learning algorithm. A more detailed experiment setting can be found in Appendix F.1. In Table 2, we also report results of the baseline "Random", where the ten randomly generated synthetic features are used to combine with the original data and then construct the random forest. The AUCs of the random forest construed with different synthetic features are summarized in Table 2.

From Table 2, we remark that incorporating synthetic features generally boost the performance of the random forest model. Namely, NID outperforms Original in the Higgs Boson, Creditcard, and Spambase datasets, while the proposed PID method outperforms all the compared methods. The statistics of detected interactions by different methods are shown in Appendix F.2. Furthermore, we remark that the feature interactions discovered by PID highly coincide with the top solution of the Amazon Employee Challenge [1](Appendix F.2).

4.3 High-order Interaction Detection on Image Datasets

For image data, input features are raw pixels and interactions are patterns that represent visual cues characterizing the object in the image. To answer **Q4**, we apply PID to find out the contributing pattern in a particular image that lead the CNNs to make the prediction. In Section 3, the proposed framework is used for detecting global interactions. Global (or model-level) interaction means the learned interactions for making predictions on the entire dataset. Specifically, the only input to global interaction detection algorithms is the model to be analyzed, without any information about the position or the scale of the object in an image. Local (or instance-level) interaction detection, however, tries to answer what interactions of a data sample lead the model to make a special prediction. We remark that global interaction detection is meaningless for image data because it is not invariant to the position or scales of objects. We show how to extend PID to the CNNs and local interaction detection in Appendix D.

Detecting interactions in a specific image is a more challenging task for the following reasons: First, the order of interactions is extremely high in image data; second, image data is high dimensional by nature. The number of possible interaction candidates grows exponentially with respect to the number of input features, e.g., for a $1 \times 28 \times 28$ image from MNIST dataset, the number of interaction candidates within the search space is $2^{784} \approx 10^{235}$. We note that interactions in an image is similar to the Superpixels [27], which is originally proposed for solving the image segmentation task. However, it is not straight-forward to show detected interaction by considering them as superpixels: First, the interaction in an image is a group of pixels that are not necessarily connected; second, theoretically, each interaction is associated with the interaction

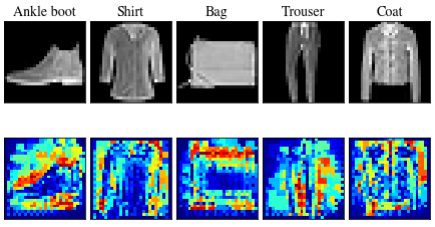


Figure 4: Saliency maps of interaction strength found from applying PID on the CNN trained on FashionMNIST dataset.

[1]https://www.kaggle.com/c/amazon-employee-access-challenge

strength, which cannot be shown by simply breaking the image into different segmentation. To evaluate the detected interactions with better representation of images, instead of connecting pixels that are in the same interaction, we build the saliency map for each input images by visualizing the importance of raw pixels. Specifically, the importance of the raw pixel i is obtained by aggregating the interaction strength of each candidate set which contains i.

We trained a simple CNN to classify images on the MNIST dataset [28] and FashionMNIST dataset [29]. The detailed experiment setting can be found in Appendix G. Here we only present the saliency maps of FashionMNIST images. The saliency maps of MNIST images are available in Appendix G. We observe PID is capable of detecting high-order interactions that represent visual cues. From Figure 4, the CNN acquired complex knowledge about the shapes associated with each category. For example, the interpretations of the "Ankle boot" classification show that interaction detection finds the shape of the boot instead of the boot texture. This is indicated by the fact that pixels with higher importance (warm colors) essentially trace the contour of an ankle boot.

5 Limitation and Discussion

In this paper, we extend concepts of the *birth* time and *death* time to the interaction for proposing a new measure that quantifies the interaction strength. These concepts are originally proposed for identifying true topology features (e.g., connected components and loops). Rigorously speaking, the structure of interest for interaction detection is a sub-network that connects a group of input features with output neurons, which is not a well-defined algebraic topology concept. Therefore, a lot of theoretical properties of the subject across the filtration is lost. However, to the best of our knowledge, by extending these concepts from Persistent Homology, we propose the first NN specific interaction strength measure with stability guarantee (Theorem 1). Furthermore, we derived a topology-motivated algorithm to compute the interaction strength efficiently (Lemma 1).

We note that as NNs contain only 1-simplex, many of these topological properties degenerate to the field of graph theory. The proposed filtration process is equivalent to building maximum spanning trees (MSTs) of NNs using the Kruskal algorithm. The proposed persistence of feature groups is the gap length between MSTs of two sub-networks. It would be interesting to consider the theoretical benefit of our proposed measure from the perspective of graph theory. We leave it as future work.

Also, we want to emphasize that our image experiment in section 4.3 is exploratory. This experiment is designed to illustrate that the proposed PID is capable of detecting extreme high-order interactions in a specific input. Moreover, the saliency map obtained by utilizing PID could also provide visual cues for understanding how CNNs make decisions. We note that PID is complementary to most existing explainable-CV works. Especially, the saliency map in section 4.3 is obtained only from the interaction effects between raw pixels. In contrast, most explainable-CV works (e.g., Grad-CAM [30]) only consider how a specific raw-pixel influence the decision of models, and the interaction effects are ignored by them because most of these works do not access to Hessian Matrix or compute the approximation of Hessian Matrix.

6 Conclusion

In this work, we propose a theoretically well-defined measure for quantifying interaction strength by investigating the topology of neural networks. We show that this measure captures topological information that pertains to learned interactions in neural networks. Based on this measure, we derive the PID algorithm to detect interactions. We also give the theoretical analysis for it and show how to extend our method to local interaction detection. We demonstrate our proposed method has the practical utility of accurately detecting feature interactions without the need to prespecify interaction types or to search an exponential solution space of interaction candidates.

Statement of Broader Impact

The proposed PID algorithm can be applied in various fields because it provides knowledge about a domain. Any researcher who needs to design experiments might benefit from our proposed algorithm in the sense that it can help researchers formulate hypotheses that could lead to new data collection and experiments. For example, PID can help us discover the combined effects of drugs on human body: By utilizing PID on patients' records, we might find using Phenelzine togther with Fluoxetine has a strong interaction effect towards serotonin syndrome. Thus, PID has great potential in helping the development of new therapies for saving lives.

Also, this project will lead to effective and efficient algorithms for finding useful any-order crossing features in an automated way. Finding useful crossing features is one of the most crucial task in the Recommender Systems. Engineers and Scientists in E-commerce companies may benefit from our results that our algorithm can alleviate the human effect on finding these useful patterns in the data.

Acknowledgements

We would like to sincerely thank everyone who has provided their generous feedback for this work. Thank the anonymous reviewers for their thorough comments and suggestions. The authors thank the Texas A&M College of Engineering and Texas A&M University.

References

[1] Christina Sun-Edelstein, Stewart J Tepper, and Robert E Shapiro. "Drug-induced serotonin syndrome: a review". In: *Expert opinion on drug safety* 7.5 (2008), pp. 587–596.

[2] Ronald Aylmer Fisher. "Statistical methods for research workers". In: *Breakthroughs in statistics*. Springer, 1992, pp. 66–70.

[3] Daria Sorokina et al. "Detecting statistical interactions with additive groves of trees". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1000–1007.

[4] Jacob Bien, Jonathan Taylor, and Robert Tibshirani. "A lasso for hierarchical interactions". In: *Annals of statistics* 41.3 (2013), p. 1111.

[5] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[6] Michael Tsang, Dehua Cheng, and Yan Liu. "Detecting statistical interactions from neural network weights". In: *International Conference on Learning Representations*. 2018.

[7] Michael Tsang et al. "Neural interaction transparency (nit): Disentangling learned interactions for improved interpretability". In: *Advances in Neural Information Processing Systems*. 2018, pp. 5804–5813.

[8] Christoph Hofer et al. "Deep learning with topological signatures". In: *Advances in Neural Information Processing Systems*. 2017, pp. 1634–1644.

[9] Valentin Khrulkov and Ivan Oseledets. "Geometry score: A method for comparing generative adversarial networks". In: *arXiv preprint arXiv:1802.02664* (2018).

[10] Christoph D Hofer, Roland Kwitt, and Marc Niethammer. "Learning representations of persistence barcodes". In: *Journal of Machine Learning Research* 20.126 (2019), pp. 1–45.

[11] Jerome H Friedman, Bogdan E Popescu, et al. "Predictive learning via rule ensembles". In: *The Annals of Applied Statistics* 2.3 (2008), pp. 916–954.

[12] Herbert Edelsbrunner. "Surface tiling with differential topology". In: *Proceedings of the third Eurographics symposium on Geometry processing*. Eurographics Association. 2005, p. 9.

[13] Jos BTM Roerdink and Arnold Meijster. "The watershed transform: Definitions, algorithms and parallelization strategies". In: *Fundamenta informaticae* 41.1, 2 (2000), pp. 187–228.

[14] Michele d'Amico, Patrizio Frosini, and Claudia Landi. "Natural pseudo-distance and optimal matching between reduced size functions". In: *Acta applicandae mathematicae* 109.2 (2010), pp. 527–554.

[15] Patrizio Frosini and Claudia Landi. "Size theory as a topological tool for computer vision". In: *Pattern Recognition and Image Analysis* 9.4 (1999), pp. 596–603.

[16] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[17] Bastian Rieck et al. "Neural persistence: A complexity measure for deep neural networks using algebraic topology". In: *International Conference on Learning Representations*. 2018.

[18] Gunnar Carlsson. "Topology and Data". In: *Bulletin of The American Mathematical Society - BULL AMER MATH SOC* 46 (Apr. 2009), pp. 255–308. DOI: 10.1090/S0273-0979-09-01249-X.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[20] David Cohen-Steiner et al. "Lipschitz functions have L p-stable persistence". In: *Foundations of computational mathematics* 10.2 (2010), pp. 127–139.

[21] Yin Lou et al. "Accurate intelligible models with pairwise interactions". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 623–631.

[22] Rebecca Roelofs et al. "A Meta-Analysis of Overfitting in Machine Learning". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 9179–9189. URL: http://papers.nips.cc/paper/9117-a-meta-analysis-of-overfitting-in-machine-learning.pdf.

[23] Michael Tsang et al. "Feature Interaction Interpretability: A Case for Explaining Ad-Recommendation Systems via Neural Interaction Detection". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=BkgnhTEtDS.

[24] Yuanfei Luo et al. "AutoCross: Automatic Feature Crossing for Tabular Data in Real-World Applications". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 1936–1945.

[25] Udayan Khurana, Horst Samulowitz, and Deepak Turaga. "Feature engineering for predictive modeling using reinforcement learning". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[26] Fatemeh Nargesian et al. "Learning Feature Engineering for Classification." In: *IJCAI*. 2017, pp. 2529–2535.

[27] Radhakrishna Achanta et al. "SLIC superpixels compared to state-of-the-art superpixel methods". In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2274–2282.

[28] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[29] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[30] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[31] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. "Stability of persistence diagrams". In: *Discrete & Computational Geometry* 37.1 (2007), pp. 103–120.

[32] Chen Kong and Simon Lucey. "Take it in your stride: Do we need striding in CNNs?" In: *arXiv preprint arXiv:1712.02502* (2017).

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

[34] Michael Lim and Trevor Hastie. "Learning interactions via hierarchical group-lasso regularization". In: *Journal of Computational and Graphical Statistics* 24.3 (2015), pp. 627–654.

[35] Yinfei Kong et al. "Interaction pursuit in high-dimensional multi-response regression via distance correlation". In: *The Annals of Statistics* 45.2 (2017), pp. 897–922.

[36] Matthias Feurer et al. "OpenML-Python: an extensible Python API for OpenML". In: *arXiv preprint arXiv:1911.02490* (2019).

[37] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. "CatBoost: gradient boosting with categorical features support". In: *arXiv preprint arXiv:1810.11363* (2018).

[38] Guolin Ke et al. “Lightgbm: A highly efficient gradient boosting decision tree”. In: *Advances in neural information processing systems*. 2017, pp. 3146–3154.

Appendices

A The Persistence Interaction Detection Algorithm

Algorithm 1: The proposed Persistence Interaction Detection (PID) algorithm

Input: A trained feed-forward neural network, target layer l, norm p.
Output: ranked list of interaction candidates $\{\mathcal{I}_i\}$.

Construct size pair $(\mathcal{G}, \phi)$ and its filtration $\mathcal{G}^{w_0'} \subseteq ... \subseteq \mathcal{G}^{w_n'}$
$\mathcal{K} \leftarrow$ initialize an empty dictionary mapping interaction candidate to persistence
for i=0:n **do**
    $\lambda \leftarrow w_i'$; $\mathcal{G}^{\lambda} \leftarrow \mathcal{G}^{w_i'}$
    Calculate $\mathbf{M}_{(l)}^{\lambda_{up}}$ and $\mathbf{M}_{(l)}^{\lambda_{down}}$ according to Equation (3)
    for each row $\mathbf{m}$ of $\mathbf{M}_{(l)}^{\lambda_{down}}$ indexed by r **do**
        if all elements in r^{th} column of $\mathbf{M}_{(l)}^{\lambda_{up}}$ are 0 **then**
            continue `// r-th unit in l-th layer is not connected with any final output units`
        end
        $\mathcal{I} \leftarrow$ initialize an empty set;
        for j=0:d-1 **do**
            if m_j == 0 **then**
                continue `// r-th unit is not connected with feature` j
            end
            $d_{\mathcal{I}} \leftarrow \lambda$ `//` $\mathcal{I}$ `merged with` j
            ;
            $b_{\mathcal{I} \cup j} \leftarrow \lambda$;
            $\mathcal{K}[\mathcal{I}] \leftarrow \mathcal{K}[\mathcal{I}] + |b_{\mathcal{I}} - d_{\mathcal{I}}|^p$;
            $\mathcal{I} \leftarrow \mathcal{I} \cup j$;
        end
    end
end
$\{\mathcal{I}_i\} \leftarrow$ interaction candidates in $\mathcal{K}$ sorted by their strengths in descending order.

Our PID framework is presented in Algorithm 1. Besides the $\langle \phi = \lambda \rangle$-connectivity between $\mathcal{I}$ and final outputs, we also consider: First, whether $\mathcal{I}$ and a particular neuron r are connected; second, whether the neuron r and final outputs are connected under the threshold λ. Recall that the measuring function ϕ is non-decreasing over $\mathcal{G}$ (Definition 2), and the birth time and death time of each interaction candidates can be determined through one pass of all thresholds. As shown in Figure 3, calculating the interaction strength of $\mathcal{I}$ at neuron r is equivalent to running Algorithm 1 on a neural network whose l^{th} layer is only composed by neuron r.

The time complexity of PID is $\mathcal{O}(Ndp_l)$, where N denotes the total number of weights used as thresholds in the filtration and p_l is the number of neurons at target layer l. One possible way to reduce the time complexity is that, we can change the $\mathcal{W}' := \{|w|/w_{max} | w \in \mathcal{W}\}$ in section 2.2 to $\mathcal{W}' := \{|w|/w_{max} | w \in \mathcal{W} \wedge w \geq \eta w_{max}\}$, where η is a hyperparameter which controls total number of weights used as thresholds in Algorithm 1. We do not utilize this method to accelerate PID in all experiments of this paper (i.e., set η as 0).

B Proof of Lemma 1

Lemma 1 (Proof in Appendix B)**.** *Let $\{0, ..., d-1\}$ denotes the input feature set, and $\mathbf{M}^{\lambda}$ denotes the aggregated mask matrix corresponding to threshold λ, where the r^{th} row of $\mathbf{M}^{\lambda}$ is denoted as $\mathbf{m}_r^{\lambda} \in \mathbb{R}^d$. The feature subset $\mathcal{I}$ and the corresponding r^{th} unit at the final output layer are $\langle \phi = \lambda \rangle$-connected if all elements in $[\mathbf{m}_r^{\lambda}]^{\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|}$ are non-zero and all other elements in $[\mathbf{m}_r^{\lambda}]^{\{0,...,d-1\} \setminus \mathcal{I}}$ are zero, where $[\mathbf{m}_r^{\lambda}]^{\mathcal{I}}$ is the subvector of $\mathbf{m}_{\mathbf{r}}^{\lambda}$ selected by $\mathcal{I}$.*

We obtain Lemma 1 following from the theoretical analysis in Appendix E of [7].

Proof. If the network has exactly one layer, $\mathbf{M}^\lambda = (\mathbf{M}^\lambda_{(1)})^\top$ directly gives the connectivity between input features and output units in the final output layer.

In cases when $\mathbf{M}^\lambda$ has more than one hidden layer, first consider the weight connectivity between input features and the second hidden layer. Since a feed-forward neural network is a directed acyclic graph and a hop is a transition from one layer to the next, we can view the connectivity from input features to the second hidden layer as two hops or two applications of an adjacency matrix, $\mathbf{A}$, comprising of $\mathbf{M}^\lambda_{(2)}$ and $\mathbf{M}^\lambda_{(1)}$ as:

$$\mathbf{A} = \begin{bmatrix} 0 & (\mathbf{M}^\lambda_{(1)})^\top & 0 \\ 0 & 0 & (\mathbf{M}^\lambda_{(2)})^\top \\ 0 & 0 & 0 \end{bmatrix}.$$

Therefore, the adjacency matrix for two hops is:

$$\mathbf{A}^2 = \begin{bmatrix} 0 & 0 & (\mathbf{M}^\lambda_{(2)})^\top (\mathbf{M}^\lambda_{(1)})^\top \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Since the elements of $\mathbf{A}^2$ are the number of paths between graph vertices in two hops, the non-zero elements of $(\mathbf{M}^\lambda_{(2)})^\top (\mathbf{M}^\lambda_{(1)})^\top$ represent the existence of paths from features to the second hidden layer, and the zero elements represent the lack of such paths. We can therefore repeatedly apply hops up to the L^{th} hidden layer, yielding $(\mathbf{M}^\lambda_{(L)})^\top \cdot (\mathbf{M}^\lambda_{(L-1)})^\top \cdots (\mathbf{M}^\lambda_{(1)})^\top$ to represent the zero and non-zero paths from input features to the neurons in the L^{th} layer. Thus, if all elements in $[\mathbf{m}^\lambda_r]^{\mathcal{I}}$ are non-zero and all other elements in $[\mathbf{m}^\lambda_r]^{\{0,\ldots,d-1\}\setminus\mathcal{I}}$ are zero, $\mathcal{I}$ and unit r are $\langle \phi = \lambda \rangle - connected$ by Definition 3.

□

C Proof of Theorem 1

In this subsection, we will prove Theorem 1 and evaluate it empirically. We first give the stability lemma for connected components and then utilize it to derive Theorem 1.

Definition 4 (Hausdorff distance)**.** *For points $p = (p_1, p_2)$ and $q = (q_1, q_2)$ in $\mathbb{R}^2$, let $\|p - q\|_\infty$ be the maximum of $|p_1 - q_1|$ and $|p_2 - q_2|$. Let $\|f - g\|_\infty = \sup_x |f(x) - g(x)|$. Let X and Y be multisets of points. The Hausdorff distance is defined as*

$$d_H(X, Y) = \max\{\sup_{x\in X} \inf_{y\in Y} \|x - y\|_\infty, \sup_{y\in Y} \inf_{x\in X} \|x - y\|_\infty\},$$

For two feed forward neural networks f and g with the exact same architecture, let g be a neural network that is obtained by perturbing the weights of f. The corresponding size pairs $(\mathcal{G}_f, \phi_f)$ and $(\mathcal{G}_g, \phi_g)$ are constructed following instructions in Section 2.2. Let $\delta = \max_{e\in E} |\phi_f(e) - \phi_g(e)|$ be the magnitude of the perturbation, i.e., $\|\phi_f - \phi_g\|_\infty = \delta$. Persistence diagrams of $(\mathcal{G}_f, \phi_f)$ and $(\mathcal{G}_g, \phi_g)$ are denoted as $\mathcal{D}[(\mathcal{G}_f, \phi_f)]$ and $\mathcal{D}[(\mathcal{G}_g, \phi_g)]$, respectively. We note that ϕ_f and ϕ_g are piecewise linear functions on simplicial complexes, where a simplicial complex is a high-dimensional generalization of a graph in topological space. Piecewise linear functions satisfy the following Lemma:

Lemma 2 (Proof in [31])**.** $d_H(\mathcal{D}[(\mathcal{G}_f, \phi_f)], \mathcal{D}[(\mathcal{G}_g, \phi_g)]) \leq \delta$.

When weights in the networks are perturbed, the birth time and death time of connected components are also changed. Lemma 2 shows that the Hausdorff distance between the persistence diagrams is bounded by the magnitude of the perturbation, i.e., for the set of all connected components $\mathcal{J}$, suppose its birth time $b_\mathcal{J}$ and death time $d_\mathcal{J}$ changes to $b'_\mathcal{J}$ and $d'_\mathcal{J}$, then $\max(|b_\mathcal{J} - b'_\mathcal{J}|, |d_\mathcal{J} - d'_\mathcal{J}|) \leq \delta$.

For any interaction candidate $\mathcal{I}$ that are detected in both f and g by Algorithm 1, we denote the birth time of $\mathcal{I}$ in f and g as $b_\mathcal{I}$ and $b'_\mathcal{I}$, respectively. Similarly, we use $d_\mathcal{I}$ and $d'_\mathcal{I}$ for the death time of

$\mathcal{I}$ in f and g, respectively. Suppose the connected component $\mathcal{J}$ and the connected component $\mathcal{J}'$ cause the birth of interaction $\mathcal{I}$ in f and g, respectively. We have the following corollary:

Corollary 1. $|b_{\mathcal{I}} - b'_{\mathcal{I}}| \leq 3\delta$.

Proof. From Definition 3, we have $|b_{\mathcal{I}} - b'_{\mathcal{I}}| = |b_{\mathcal{J}} - b_{\mathcal{J}'}| = |\min_{e\in\mathcal{J}} \phi_f(e) - \min_{e\in\mathcal{J}'} \phi_g(e)|$. If $\mathcal{J}$ and $\mathcal{J}'$ are composed of identical set of edges, then we can directly prove Corollary 1 following Lemma 2. If $\mathcal{J}$ and $\mathcal{J}'$ contain different edges, without loss of generality, let $\mathcal{J}'\backslash\mathcal{J} \neq \emptyset$. $\forall e, \phi_f(e) \leq \min_{e'\in\mathcal{J}} \phi_f(e') - 2\delta$, we have the following inequality:

$$\phi_g(e) \leq \min_{e'\in\mathcal{J}} \phi_g(e'). \tag{4}$$

Inequality (4) follows from the fact that $\forall e, |\phi_f(e) - \phi_g(e)| \leq \delta$, which implies that, $\forall e, \phi_f(e) \leq \min_{e'\in\mathcal{J}} \phi_f(e') - 2\delta$, e has to wait for all edges in $\mathcal{J}$ to be added to the filtration before being added itself. Namely, $\mathcal{I}$ is born before the threshold arrives at $\phi_g(e)$ and, consequently, $e \notin \mathcal{J}'$. Thus, $\forall e, e \in \mathcal{J}'\backslash\mathcal{J}$, e satisfies $\phi_f(e) \geq \min_{e'\in\mathcal{J}} \phi_f(e') - 2\delta$. Following this fact, we have

$$\begin{aligned}
\min_{e\in\mathcal{J}'} \phi_g(e) &= \min\{\min_{e\in\mathcal{J}'\cup\mathcal{J}} \phi_g(e), \min_{e\in\mathcal{J}'\backslash\mathcal{J}} \phi_g(e)\} \\
&\geq \min\{\min_{e\in\mathcal{J}'\cup\mathcal{J}} \phi_f(e) - \delta, \min_{e\in\mathcal{J}'\backslash\mathcal{J}} \phi_f(e) - \delta\} &(5)\\
&\geq \min\{\min_{e\in\mathcal{J}'\cup\mathcal{J}} \phi_f(e) - \delta, \min_{e\in\mathcal{J}} \phi_f(e) - 3\delta\} &(6)\\
&\geq \min\{\min_{e\in\mathcal{J}} \phi_f(e) - \delta, \min_{e\in\mathcal{J}} \phi_f(e) - 3\delta\} \\
&= \min_{e\in\mathcal{J}} \phi_f(e) - 3\delta. &(7)
\end{aligned}$$

Inequality (5) follows from $\forall e \in E, |\phi_f(e) - \phi_g(e)| \leq \delta$. The inequality (6) follows from the fact that $\forall e \in \mathcal{J}'\backslash\mathcal{J}, \phi_f(e) \geq \min_{e'\in\mathcal{J}} \phi_f(e') - 2\delta$. By equation (7), we have $b_{\mathcal{J}} - b_{\mathcal{J}'} \leq 3\delta$.

By exchanging f with g, we have $b_{\mathcal{J}'} - b_{\mathcal{J}} \leq 3\delta$. Combining them together finishes the proof.

□

It is trivial to show that Corollary 1 can be extended to the death time, i.e., we also have $|d_{\mathcal{I}} - d'_{\mathcal{I}}| \leq 3\delta$.

After proving Corollary 1, we return to prove the theorem.

Theorem 1 (Proof and empirical analysis in Appendix C)**.** *Let $\delta = \max_{e\in E} |\phi_f(e) - \phi_g(e)|$ be the magnitude of perturbation. For all interaction candidate $\mathcal{I}$ that are both detected in f and g by Algorithm 1, it satisfies $|\rho_f(\mathcal{I}) - \rho_g(\mathcal{I})| \leq C\delta$.*

Proof. In Algorithm 1, interaction candidates are generated at each neuron r of a particular layer l. As shown in Figure 3, calculating the interaction strength of $\mathcal{I}$ at neuron r is equivalent to running Algorithm 1 on a neural network whose l^{th} layer is only composed by neuron r. Thus Corollary 1 also holds for interaction candidate $\mathcal{I}$ generated at each neuron. We use $\text{per}_f^{(r)}(\mathcal{I})$, $b^{(r)}(\mathcal{I})$, and $d^{(r)}(\mathcal{I})$ to represent the persistence, the birth time, and the death time of $\mathcal{I}$ generated at neuron r corresponding to f, respectively. Similarly, for g, the persistence, the birth time, and the death time of $\mathcal{I}$ generated at neuron r are denoted as $\text{per}_g^{(r)}(\mathcal{I})$, $b'^{(r)}(\mathcal{I})$, and $d'^{(r)}(\mathcal{I})$, respectively.

$$\begin{aligned}
\text{per}_f^{(r)}(\mathcal{I}) &= |b_{\mathcal{I}}^{(r)} - d_{\mathcal{I}}^{(r)}| \\
&= |b_{\mathcal{I}}^{(r)} - b_{\mathcal{I}}^{\prime(r)} + b_{\mathcal{I}}^{\prime(r)} - d_{\mathcal{I}}^{\prime(r)} + d_{\mathcal{I}}^{\prime(r)} - d_{\mathcal{I}}^{(r)}| \\
&\leq \text{per}_g^{(r)}(\mathcal{I}) + 6\delta,
\end{aligned}$$

By exchanging f with g, we have $\mathrm{per}_g^{(r)}(\mathcal{I}) \leq 6\delta + \mathrm{per}_f^{(r)}(\mathcal{I})$. Combining them together, we have $|\mathrm{per}_f^{(r)}(\mathcal{I}) - \mathrm{per}_g^{(r)}(\mathcal{I})| \leq 6\delta$. Then it follows

$$\begin{aligned} |\rho_f(\mathcal{I}) - \rho_g(\mathcal{I})| &= |\sum_{r \in l^{th}\text{layer}} [\mathrm{per}_f^{(r)}(\mathcal{I})]^p - [\mathrm{per}_g^{(r)}(\mathcal{I})]^p| \\ &\leq p|\sum_{r \in l^{th}\text{layer}} [\mathrm{per}_f^{(r)}(\mathcal{I}) - \mathrm{per}_g^{(r)}(\mathcal{I})] \max\{\mathrm{per}_f^{(r)}(\mathcal{I}), \mathrm{per}_f^{(r)}(\mathcal{I})\}^{p-1}| \\ &\leq 6pN_l\delta. \end{aligned} \tag{8}$$

Where N_l is the number of units in layer l. The inequality (8) follows from the fact that $\max\{\mathrm{per}_f^{(r)}(\mathcal{I}), \mathrm{per}_f^{(r)}(\mathcal{I})\} \leq 1$.

□

Beyond Theorem 1, there exists the corner case that there are interaction candidates only detected in one neural network, but not the other. We will show that this corner case only happens if δ is greater than a threshold.

Let $[d] := \{0, \cdots, d-1\}$ be the input feature set. Without loss of generality, suppose interaction candidate $\mathcal{I} \subset [d]$ only born in f, but not in g; and the connected component $\mathcal{J}$ cause the birth of $\mathcal{I}$ in f. Let g be a neural network that is obtained by perturbing the weights of f. According to Definition 3, if $\mathcal{I}$ only born in f, it means that there exists some edges corresponding to the connection between input features and hidden units in the first layer, which satisfy the following:

$$\begin{aligned} &\exists e' \in [d] \backslash \mathcal{I}, \\ s.t. \quad &\phi_f(e') \geq \min_{e \in \mathcal{J}} \phi_f(e) - 2\delta \end{aligned} \tag{9}$$

The above inequality follows from the fact that $\forall e, |\phi_f(e) - \phi_g(e)| \leq \delta$, which implies that, $\forall e, \phi_f(e) \leq \min_{e' \in \mathcal{J}} \phi_f(e') - 2\delta$, e has to wait for all edges in $\mathcal{J}$ to be added to the filtration before being added itself. Therefore, if $\mathcal{I}$ does not born in g, there must $\exists e' \in [d] \backslash \mathcal{I}$ such that $\phi_f(e') \geq \min_{e \in \mathcal{J}} \phi_f(e) - 2\delta$. In conclusion, if $\mathcal{I}$ only detected in f, the perturbation magnitude δ must satisfy:

$$\delta \geq \frac{\min_{e \in \mathcal{J}} \phi_f(e) - \max_{e \in [d] \backslash \mathcal{I}} \phi_f(e)}{2} \tag{10}$$

Table 3: Perturbation results.

δ	$\lvert\rho_f(\mathcal{I}) - \rho_g(\mathcal{I})\rvert$
0.001	0.0935
0.01	0.1645
0.1	0.1990
1	0.3321

Here we randomly perturb the weights of an MLP trained on synthetic dataset F_1, which has architecture of 64-32-16 first-to-last hidden layer sizes. The layer l in Algorithm 1 is set to the first layer, and the norm p in Algorithm 1 is set to 2. The results are shown in Table 3.

D Extensibility

In this section, first, we show how to extend PID to CNNs. Second, we introduce how to extend our method to local interaction detection.

Let $\mathbf{H} \in \mathbb{R}^{height \times width}$ be a convolution kernel and $\mathbf{X} \in \mathbb{R}^{H \times W}$ be a tensor. Let $*$ refer to the convolution operation. Suppose the height and width of the $\mathbf{H} * \mathbf{X}$ are H_{out} and W_{out}, respectively. We define $\mathcal{H} \in \mathbb{R}^{H_{\text{out}} \times W_{\text{out}} \times H \times W}$ as the corresponding four dimensional convolution tensor such that:

$$\mathcal{H}(i, j, i : i + height, j : j + width) = \mathbf{H},$$

for $\forall i \in [0, H_{\text{out}}), \forall j \in [0, W_{\text{out}})$. Then we have the following equation:

$$\mathbf{H} * \mathbf{X} = \mathcal{H} \otimes \mathbf{X}, \tag{11}$$

where $\mathcal{H} \otimes \mathbf{X}$ is the tensor product such that $[\mathcal{H} \otimes \mathbf{X}]_{i,j} = \sum_{k=0}^{H} \sum_{l=0}^{W} \mathcal{H}_{i,j,k,l} X_{k,l}$. Generally, for $\mathbf{H}' \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times height \times width}$ and $\mathbf{X}' \in \mathbb{R}^{C_{\text{in}} \times H \times W}$, we can convert the convolution between $\mathbf{H}'$ and $\mathbf{X}'$ into $\mathbf{H}' * \mathbf{X}' = \mathcal{H}'' \mathbf{X}''$ [32] using equation (11), where $\mathcal{H}'' \in \mathbb{R}^{C_{\text{out}} H_{\text{out}} W_{\text{out}} \times C_{\text{in}} HW}$ is the flattened matrix of $\mathcal{H}'$ with multi-channels, and $\mathbf{X}'' \in \mathbb{R}^{C_{in} HW}$ is the flattened vector of $\mathbf{X}'$. Then we can build the filtration of CNNs just as MLPs.

Given a data point $\mathbf{x} \in \mathbb{R}^d$, the feed forward neural network f with ReLU activation function is a linear model in a region surrounding $\mathbf{x}$:

$$f(\mathbf{x}) = \mathbf{W}_{\mathbf{x}}^{(L)\top} \cdots \mathbf{W}_{\mathbf{x}}^{(1)\top} \mathbf{x}, \tag{12}$$

where $\mathbf{W}_{\mathbf{x}}^{(L)}$ is the equivalent weight matrix which combines the resulted activation pattern with $\mathbf{W}^{(L)}$, e.g., $\text{ReLU}(\mathbf{W}^{(1)\top}\mathbf{x}) = \mathbf{W}_{\mathbf{x}}^{(1)\top}\mathbf{x}$, where $\mathbf{W}_{\mathbf{x}}^{(1)}$ is modified from $\mathbf{W}^{(1)}$ by setting the columns, whose corresponding activation patterns are 0, to be all zero vectors. We denote the output value of the i^{th} neuron in the l^{th} layer, before activation, as z_i^l. From equation (12), for local interaction detection, the measuring function ϕ can be revised to

$$\phi((v_{l-1,i}, v_{l,j})) = \frac{|W_{i,j}^{(l)}| \text{ReLU}(z_i^{l-1})}{\Phi}, \tag{13}$$

where $\Phi = \max_{i,j,l} |W_{i,j}^{(l)}| \text{ReLU}(z_i^{l-1})$.

E Supplemental Material for the Synthetic Data Experiments

E.1 Experiment Setting

Table 4: Test suite of data-generating functions.

$F_1(x)$	$\pi^{x_0 x_1}\sqrt{2x_2} - \sin^{-1}(x_3) + \log(x_2 + x_4) - \frac{x_8}{x_9}\sqrt{\frac{x_6}{x_7}} - x_1 x_6$
$F_2(x)$	$\pi^{x_0 x_1}\sqrt{2\lvert x_2\rvert} - \sin^{-1}(0.5x_3) + \log(\lvert x_2 + x_4\rvert + 1) + \frac{x_8}{1+\lvert x_9\rvert}\sqrt{\frac{x_6}{1+\lvert x_7\rvert}} - x_1 x_6$
$F_3(x)$	$e^{\lvert x_0 - x_1\rvert} + \lvert x_1 x_2\rvert - x_2^{2\lvert x_3\rvert} + \log(x_3^2 + x_4^2 + x_6^2 + x_7^2) + x_8 + \frac{1}{1+x_9^2}$
$F_4(x)$	$e^{\lvert x_0 - x_1\rvert} + \lvert x_1 x_2\rvert - x_2^{2\lvert x_3\rvert} + \log(x_3^2 + x_4^2 + x_6^2 + x_7^2) + x_8 + \frac{1}{1+x_9^2} + x_0^2 x_3^2$
$F_5(x)$	$\frac{1}{1+x_0^2+x_1^2+x_2^2} + \sqrt{e^{x_3+x_4}} + \lvert x_5 + x_6\rvert + x_7 x_8 x_9$
$F_6(x)$	$e^{\lvert x_0 x_1 + 1\rvert} - e^{\lvert x_2 + x_3\rvert + 1} + \cos(x_4 + x_5 - x_7) + \sqrt{x_7^2 + x_8^2 + x_9^2}$
$F_7(x)$	$(\tan^{-1}x_0 + \tan^{-1}x_1)^2 + \max(x_2 x_3 + x_5, 0) - \frac{1}{1+(x_3 x_4 x_5 x_6 x_7)^2} + (\frac{\lvert x_6\rvert}{1+\lvert x_8\rvert})^5 + \sum_{i=0}^{9} x_i$
$F_8(x)$	$x_0 x_1 + 2^{x_2+x4+x_5} + 2^{x_2+x_3+x_4+x_6} + \sin(x_6 \sin(x_7 + x_8)) + \cos^{-1}(0.9x_9)$
$F_9(x)$	$\tanh(x_0 x_1 + x_2 x_3)\sqrt{\lvert x_4\rvert} + e^{x_4+x_5} + \log(x_5^2 x_6^2 x_7^2 + 1) + x_8 x_9 + \frac{1}{1+\lvert x_9\rvert}$
$F_{10}(x)$	$\sinh(x_1 + x_2) + \cos^{-1}(\tanh(x_2 + x_4 + x_6)) + \cos(x_3 + x_4) + \sec(x_6 x_8)$

The synthetic datasets in section 4.1 are shown in Table 4. F_1 is a commonly used function in interaction detection literature [3, 21, 33]. All features were uniformly distributed between -1 and 1 except in F_1, where we used the same variable ranges as those reported in [21]. In all synthetic experiments, we evenly split train set, validation set and test set on 30k data points. All networks consisted of four hidden layers with first-to-last layer sizes of: 140, 100, 60, and 20 units. All networks employed ReLU activation and were trained using Adam optimizer with a $5e-3$ learning rate cross all ten datasets. The L1 regularization strength was set to $5e-5$. The early stopping round was set to 100 to prevent overfitting. The mean square error of all trained MLPs are less than $3e-3$ on test data.

E.2 Detailed Analysis

Main effects describe the univariate influences of features on outcomes [11], e.g., $\sin^{-1}(0.5x_3)$ in the synthetic dataset F_2. Main effects might entangle with true interactions, resulting in spurious interactions. For example, in F_2, $\{0, 1, 2\}$ is true interaction and $\{0, 1, 2, 3\}$ is a spurious interaction, which

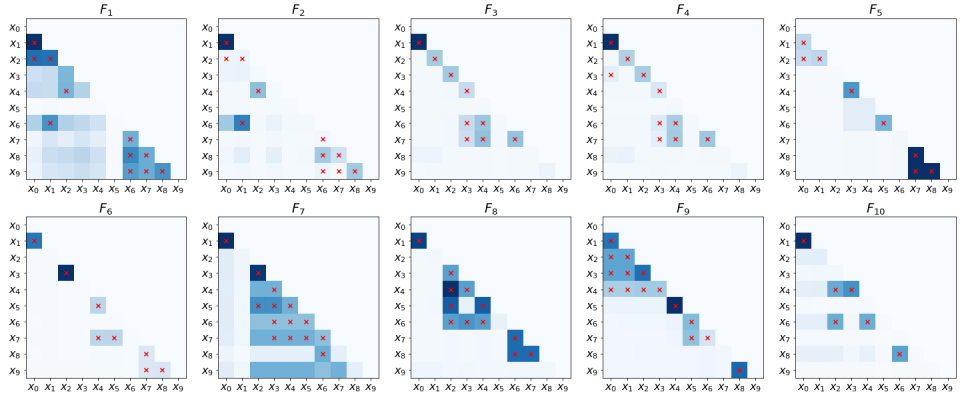


Figure 5: Heat maps of pairwise interaction strengths proposed by our PID corresponding to Table 1. Cross-marks indicate ground truth interactions.

is an entanglement between true interaction $\{0, 1, 2\}$ and main effect $\{3\}$. Handling main effects is an important problem in interaction detection [4, 34, 35]. We remark that in synthetic experiments, higher AUCs indicates the interaction detection algorithms can more thoroughly disentangle main effects from true interactions.

In Figure 5, heat maps of synthetic functions show the relative strengths of all possible pairwise interactions proposed by PID, and the ground truth is indicated by red cross-marks. In general, the interaction strengths are higher at the cross-marks. Although most of the synthetic functions contain main effects, from Figure 5 and Table 1, the influence of main effects is limited: only the AUCs of F_2 and F_7 are under 0.9. We hypothesize that if a overparameterized neural network is trained with proper regularization, the neural network will push the modeling of main effect to a small portion of neurons at the first layer.

To confirm our hypothesis, here we analyze the MLP trained on synthetic dataset F_3. For F_3, main effects are x_8 and $\frac{1}{1+x_9^2}$. Let $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times p_1}$ be the weight matrix of the first layer. The weights corresponding to input feature r are the r^{th} row of $\mathbf{W}^{(1)}$, which is denoted as $\mathbf{W}^{(1)}_{r,:}$. For convince, we mark different neurons at the first layer by their indices. In Figure 6, we show the statistics of magnitudes of $\mathbf{W}^{(1)}_{8,:}$ and $\mathbf{W}^{(1)}_{9,:}$ of an MLP trained on synthetic dataset F_3. In general, only a few neurons have large weights connecting to x_8 and x_9, which are corresponding to the peaks in Figure 6. We plot the weights of all input features to these neurons in Figure 7. To be specific, given a representative neuron c, we plot the weight statistics of input features to that neuron, which is denoted as $\mathbf{W}^{(1)}_{:,c}$. For $\mathbf{W}^{(1)}_{9,:}$, two peaks in Figure 6 have identical patterns. Here we only show statistics for one of them. For $\mathbf{W}^{(1)}_{8,:}$, we show weights statistics of all input features to neuron 36; For $\mathbf{W}^{(1)}_{9,:}$, we show weights statistics of all input features to neuron 53. This result is consistent with our hypothesis: neural networks will naturally separate different interactions in the first hidden layer.

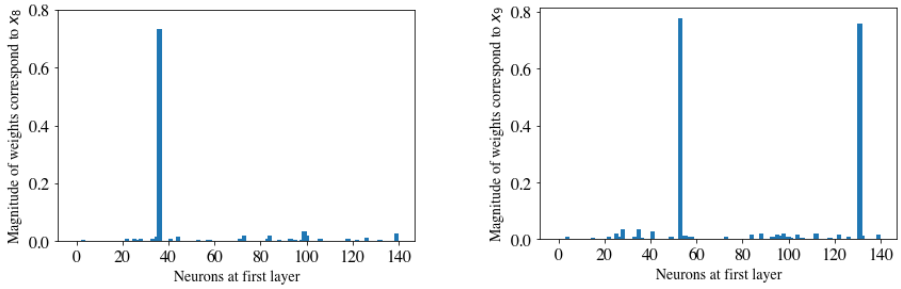


Figure 6: Statistics of the magnitudes of $\mathbf{W}^{(1)}_{8,:}$ and $\mathbf{W}^{(1)}_{9,:}$ (the MLP is trained on F_3).

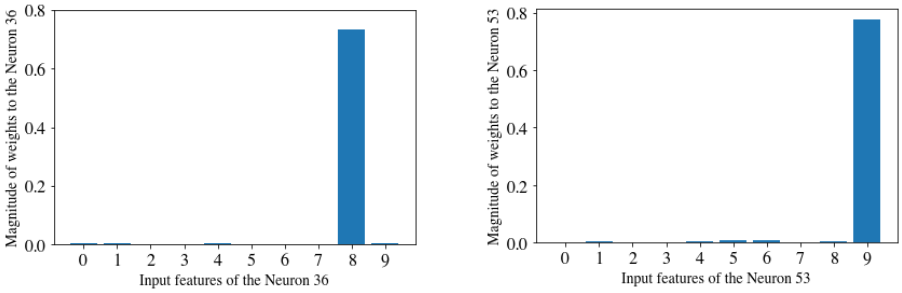
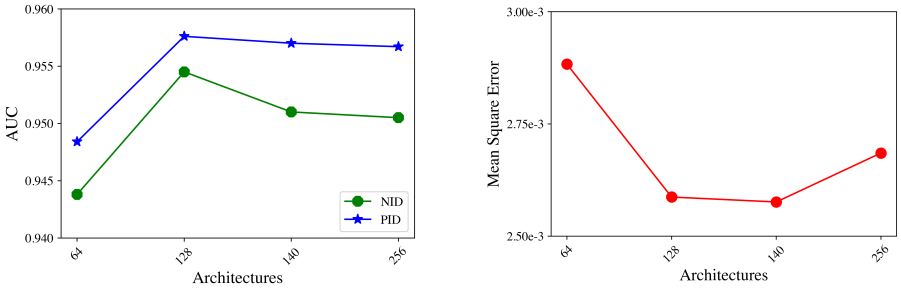


Figure 7: The magnitude of weights corresponding to different input features at the selected representative neurons in the first layer (these neurons are corresponding to the peaks in Figure 6).

E.3 Sensitivity to the Architecture and Regularization Strength

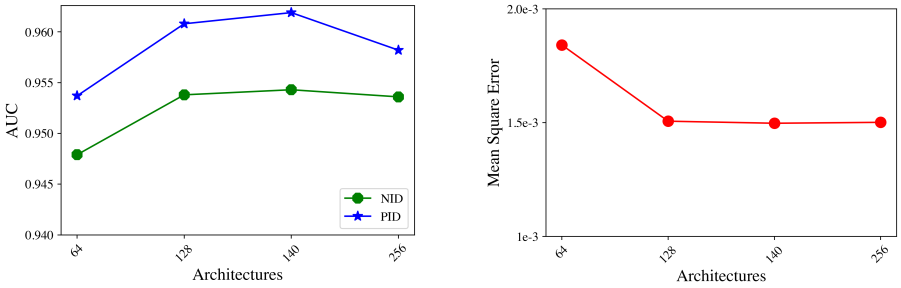
We try to analyze the sensitivity of interaction detection algorithms to the architecture of MLPs. In Figure 8, 64 represents an MLP with first-to-last layer sizes of 64-32-16; 128 represents an MLP with the 128-64-32 architecture; 140 represents an MLP with the 140-100-60-20 architecture; and 256 represents an MLP with the 256-128-64 architecture. The training hyperparameters of these MLPs are identical to those reported in Appednix E.1. We ran ten trials of NID and PID on each dataset and removed two trials with the highest and the lowest AUC scores. The mean square errors of all MLP models used for detecting interactions are less than 3e−3 on test data.



(a) Average AUCs of pairwise interaction detected by NID and PID using MLPs with different architectures.

(b) Average Mean Square Error (MSE) of MLPs with different architectures on test data.

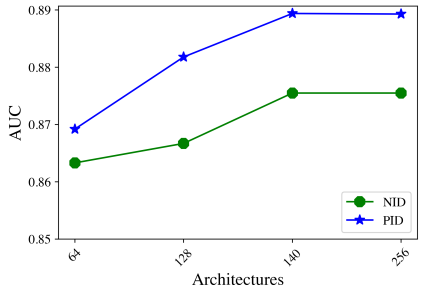
Figure 8: The sensitivity analysis of interaction detection algorithms to the architecture of MLPs (L1 is set to $5e-5$).



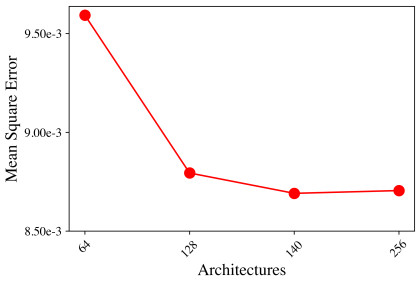
(a) Average AUCs of pairwise interaction detected by NID and PID using MLPs with different architectures.

(b) Average Mean Square Error of MLPs with different architectures on test data.

Figure 9: The sensitivity analysis of interaction detection algorithms to the regularization strength (L1 is set to $5e-6$).



(a) Average AUCs of pairwise interaction detected by NID and PID using MLPs with different architectures.



(b) Average Mean Square Error of MLPs with different architectures on test data.

Figure 10: The sensitivity analysis of interaction detection algorithms to the regularization strength (L1 is set to $5e-4$).

Table 5: AUC of pairwise interaction strengths proposed by PID and NID on the synthetic functions. The L1 regularization strength is set to $5e-4$ here.

	NID	PID
$F_1(x)$	0.898 ± 0.0145	$\mathbf{0.915 \pm 0.0144}$
$F_2(x)$	0.700 ± 0.0419	$\mathbf{0.717 \pm 0.0349}$
$F_3(x)$	0.964 ± 0.0318	$\mathbf{0.966 \pm 0.0342}$
$F_4(x)$	0.928 ± 0.0649	$\mathbf{0.938 \pm 0.0585}$
$F_5(x)$	1.000 ± 0.0000	1.000 ± 0.0000
$F_6(x)$	0.740 ± 0.0531	$\mathbf{0.769 \pm 0.0669}$
$F_7(x)$	$\mathbf{0.807 \pm 0.0318}$	0.806 ± 0.0385
$F_8(x)$	0.996 ± 0.0085	$\mathbf{0.997 \pm 0.0084}$
$F_9(x)$	0.785 ± 0.0778	$\mathbf{0.811 \pm 0.0475}$
$F_{10}(x)$	$\mathbf{0.937 \pm 0.0285}$	0.927 ± 0.0383
average	0.876 ± 0.1033	$\mathbf{0.885 \pm 0.0954}$

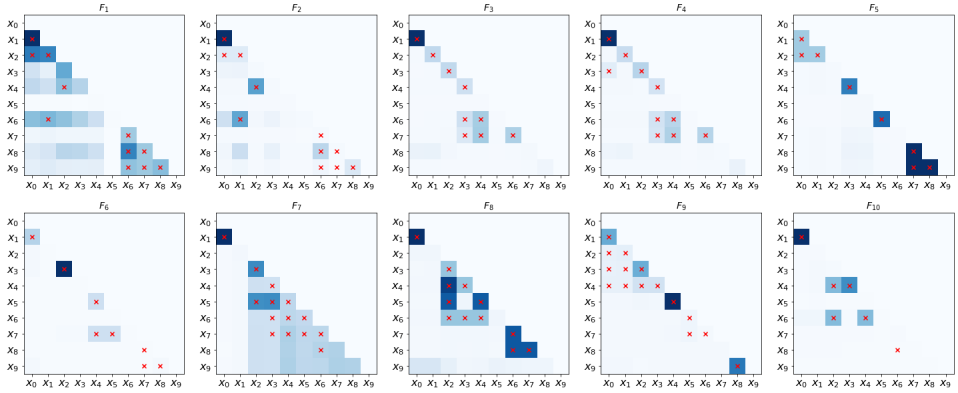


Figure 11: Heat maps of pairwise interaction strengths proposed by our PID corresponding to Table 1. Cross-marks indicate ground truth interactions. (L1 is set to $5e-4$).

The regularization strength controls the weight sparsity in neural networks. Intuitively, it significantly influences the interaction detection results because it will change the connectivity in networks. Here we change the L1 strength to 5e−4 and 5e−6, and all other experiment settings are identical.

Figure 9 shows the results using MLP with L1 set to $5e-6$. The average MSE of all MLP models used here is less than $2e-3$ on test data. Similar to Figure 8, PID can achieve better performance than NID but the gap is small. Figure 10 shows the results using MLP with L1 set to $5e-4$. The average MSE of all MLP models used here is less than $1e-2$ on test data. Comparing Figure 10

with Figure 8, the mean square error is worse but is acceptable. However, the false discovery rate increases dramatically. To better understand the impact of regularization strength, we further analyze the MLP of 140-100-60-20 architecture. Similar to Figure 5, we plot the heat map in Figure 11, and the detailed results are shown in Table 5. Comparing Table 5 with Table 1, both the performances of PID and NID dropped. Moreover, the AUCs of F_6 and F_9 dropped more than 0.1. Here we provide a detailed case study for MLPs trained on synthetic dataset F_6. Comparing Figure 11 with Figure 5, it should be noted that the interaction strength between $\{x_7, x_8, x_9\}$ is very small (near 0 actually). As [6] Appendix I points out, in synthetic dataset F_6, $\{x_7, x_8, x_9\}$ can be approximated as

$$\sqrt{x_7^2 + x_8^2 + x_9^2} \approx c + x_7^2 + x_8^2 + x_9^2.$$

In [6], the authors show that $\{x_7, x_8, x_9\}$ are modeled as spurious main effects in the MLP-M (the MLP-M is an MLP with optional univariate networks, which details can be found in [6] Figure 2). Here we hypothesize that, under strong regularization strength, they are also modeled as spurious main effects in MLPs. Figure 12 shows the weight statistics of the magnitudes of $\mathbf{W}^{(1)}_{7,:}$, $\mathbf{W}^{(1)}_{8,:}$, and $\mathbf{W}^{(1)}_{9,:}$ of an MLP trained on F_6. There is a similar pattern between Figure 12 and Figure 6. Similar to Figure 7, we further plot the weights of input features to the representative neurons corresponding to the peaks in Figure 13. We remark that, for all these neurons corresponding to peaks in Figure 12, they share a similar pattern. Therefore, we show only one of their statistics for illustrative purposes. In Figure 13, we select neuron 5, 131, and 19 for $\mathbf{W}^{(1)}_{7,:}$, $\mathbf{W}^{(1)}_{8,:}$, and $\mathbf{W}^{(1)}_{9,:}$, respectively. From Figure 13, it can be seen that MLP do not model the interaction $\{x_7, x_8, x_9\}$. Instead, they are modeled as spurious main effects.

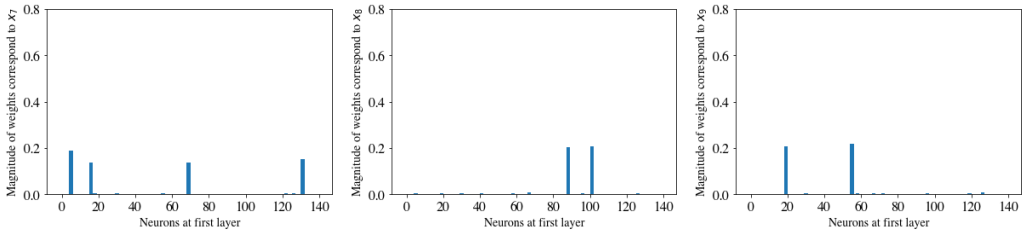


Figure 12: Statistics of the magnitudes of weights corresponding to x_7, x_8 and x_9 at different neurons of the first layer (the MLP is trained on F_6 with L1 regularization strength set to $5e-4$).

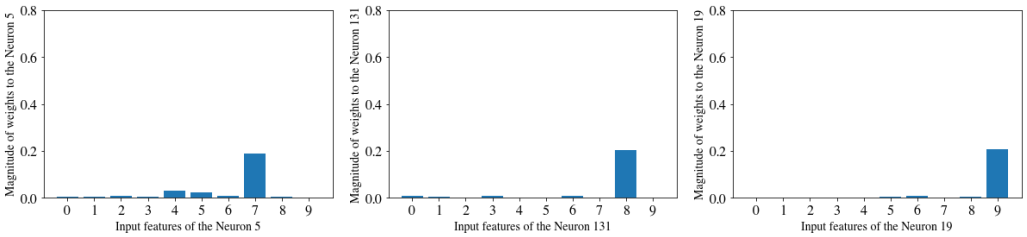


Figure 13: The magnitude of weights corresponding to different input features at the selected representative neurons of x_7, x_8 and x_9 (L1 is set to $5e-4$).

In conclusion, both NID and PID are insensitive to the architecture of MLPs and both of them are sensitive to the regularization strength. A detailed case study for the impact of regularization strength is shown in Figure 11, Figure 12, and Figure 13. This suggests that we should carefully choose the regularization strength. From Figure 8, Figure 10, and Figure 9, PID always achieves better performance. Also, we observe PID is more resilient to changes in regularization strength. Generally speaking, interaction detection algorithms have better AUC when the MLP has better performance. It makes sense that, when the MLP fits the true distribution, the interactions encoded in the networks are more accurate.

F Details for the Automatic Feature Engineering Experiments

F.1 Experiment Setting

Table 6: Statistics of datasets. "# Dense"and "# Sparse" are the number of numerical features and the number of categorical features, respectively. "# Samples" is total available samples in each dataset.

Dataset	#Samples	# Features	
		# Dense	# Sparse
Amazon Employee	32769	0	9
Higgs Boson	98050	28	0
Creditcard	284807	30	0
Spambase	4601	57	0
Diabetes	768	8	0

We perform most of the experiments on five open-source tabular datasets from different domains: **Amazon Employee**[2], **Higgs Boson**[3], **Creditcard** [4], **Spambase**[5] and **Diabetes**[6]. For the ease to reproduce our results, we use OpenML [36] to obtain all these datasets and adopt standard cross validation provided by OpenML. The statistics of datasets we used in Section 4.2 is described in Table 6.

The MLPs for NID and PID have architectures of 256-128-64 first-to-last hidden layer sizes, and they are trained with learning rate of $5e-3$, batchsize of 100, and the Adam optimizer. As pointed out in Appendix E.3, the regularization strength significantly influences the results of NID and PID. We tune the L1 regularization strength with a search space $[1e-6, 1e-1]$ for each dataset. The early stopping round is set to 20 to prevent overfitting.

The synthetic feature $\mathbf{x}_{\mathcal{L}_i}$ is created by explicitly crossing sparse features indexed in $\mathcal{L}_i$. If interaction $\mathcal{L}_i$ involves dense features, we bucketize the dense features before crossing them. The bucket size is set to 100 across all experiments. Let $|\mathcal{L}_i| = t$ and $\{0, ..., t-1\}$ is the interaction candidate specified by $\mathcal{L}_i$. A synthetic feature $\mathbf{x}_{\mathcal{L}_i}$ is an t-ary Cartesian product among t features, which means $\mathbf{x}_{\mathcal{L}_i}$ takes on all possible values in $\{(x_1, ..., x_t)|\forall x_i \in \mathbf{x}_i, i = 0, ..., t-1\}$.

Concerning the cardinality of synthetic features can be extremely large, yet many combinations do not exist in the training data, we limit the order of crossing features up to 4 over all five datasets. For sparse categorical features, like CatBoost [37], we apply target encoding to make them applicable to the random forest.

We run five trials of PID and NID on each dataset to obtain five different sets of top ten interactions. For each set of top ten interactions, we construct synthetic features and integrate them with original input features, and then we split the concatenated data into five folds. Subsequently, five random forest models are trained and evaluated with each fold given a chance to be the test set. Totally, we trained 25 random forest models on each dataset and removed two models with the highest and the lowest performance. We implement the random forest via LightGBM [38]. The hyperparameters of random forest is summerized in Table 7.

F.2 Additional Experiment Results

The statistics of detected interaction orders by PID and NID are shown in Table 8. Interaction orders are averaged over 5 folds of cross-validation.

Here we present the case study for the "Amazon Employee" dataset in Table 9 and Table 10. The main reasons for choosing "Amazon Employee" are as follows: first, it is a dataset used for Kaggle challenges and, thus, the top solution is available. Second, the key technique in the top solution is to construct synthetic features for 2-order and 3-order interactions, so we can compare our detected interactions against the best hand-crafted interactions.

[2]https://www.kaggle.com/c/amazon-employee-access-challenge

[3]https://archive.ics.uci.edu/ml/datasets/HIGGS

[4]https://www.openml.org/d/1597

[5]https://archive.ics.uci.edu/ml/datasets/spambase

[6]https://www.openml.org/d/37

Table 7: Hyperparameters of the random forest.

Name	Value
early_stopping_rounds	50
num_boost_round	5000
learning_rate	0.05
lambda_1	0.2
lambda_2	0.2
bagging_raction	0.85
bagging_req	3

Table 8: Interaction order statistics.

Method		Amazon Employee	Higgs Boson	Creditcard	Spambase	Diabetes
NID	max	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	3.60 ± 0.80
	mean	3.30 ± 0.06	2.50 ± 0.11	2.70 ± 0.17	2.62 ± 0.12	2.30 ± 0.17
	min	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00
PID	max	4.00 ± 0.00	3.80 ± 0.40	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00
	mean	3.30 ± 0.14	2.64 ± 0.20	2.84 ± 0.31	2.94 ± 0.24	3.48 ± 0.35
	min	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00	2.00 ± 0.00	2.40 ± 0.49

Table 9: Top ten interaction candidates proposed by PID for Amazom Employee dataset.

Interaction Candidates	Interaction Strength
{RESOURCE, MGR_ID, ROLE_FAMILY_DESC}	2.206
{RESOURCE, MGR_ID}	1.456
{RESOURCE, MGR_ID, ROLE_DEPTNAME, ROLE_FAMILY_DESC}	1.333
{MGR_ID, ROLE_FAMILY_DESC}	0.418
{RESOURCE, MGR_ID, ROLE_DEPTNAME}	0.393
{RESOURCE, MGR_ID, ROLE_TITLE, ROLE_FAMILY}	0.385
{RESOURCE, MGR_ID, ROLE_ROLLUP_2, ROLE_FAMILY_DESC}	0.315
{RESOURCE, MGR_ID, ROLE_TITLE, ROLE_FAMILY_DESC}	0.270
{RESOURCE, MGR_ID, ROLE_FAMILY}	0.220
{MGR_ID, ROLE_DEPTNAME}	0.190

Table 10: Top ten interaction candidates proposed by NID for Amazom Employee dataset.

Interaction Candidates	Interaction Strength
{RESOURCE, MGR_ID, ROLE_FAMILY_DESC}	26.757
{RESOURCE, MGR_ID}	22.060
{RESOURCE, MGR_ID, ROLE_DEPTNAME, ROLE_FAMILY_DESC}	10.423
{MGR_ID, ROLE_FAMILY_DESC}	7.713
{RESOURCE, MGR_ID, ROLE_TITLE, ROLE_FAMILY_DESC}	2.697
{RESOURCE, MGR_ID, ROLE_FAMILY_DESC, ROLE_CODE}	2.448
{RESOURCE, ROLE_FAMILY_DESC}	2.436
{RESOURCE, MGR_ID, ROLE_ROLLUP_2, ROLE_FAMILY_DESC}	2.316
{RESOURCE, MGR_ID, ROLE_FAMILY_DESC, ROLE_FAMILY}	1.187
{ROLE_CODE, MGR_ID, ROLE_TITLE, ROLE_FAMILY_DESC}	1.070

In general, the interaction candidates detected by NID and PID are similar. However, there exists some interaction candidates only detected by PID or NID, respectively. For example, "{MGR_ID, ROLE_FAMILY_DESC}" are only detected by NID. We note that the scale of the interaction strength proposed by PID and NID are different and only the rankings of interaction candidates are comparable. From Table 9, most of the interaction candidates proposed by PID for Amazon Employee are 3-order interactions. None of the top ranked interactions contain the input feature ROLE_CODE. This result is consistent with the top solution: "Transform the data to higher degree features by considering all

pairs and triples of the original data ignoring ROLE_CODE"[7]. In contrast, "ROLE_CODE" are contained in the interaction candidates proposed by NID. And our top ranked interactions are also consistent with the hand-designed synthetic features built from interactions,[8] such as {RESOURCE, MGR_ID} corresponding to "The number of unique resources that a MGR_ID received requests for".

G Details for High-order Interaction Detection on Image Datasets

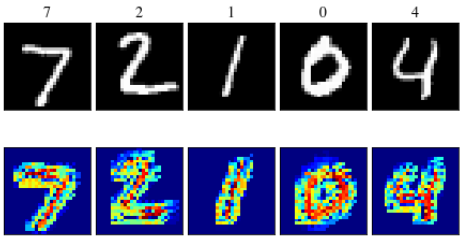


Figure 14: Saliency maps of interaction strength found from applying PID on the CNN trained on MNIST dataset.

The neural network is composed of two convolutional layers of kernel size 5 and stride 1, followed by a max pooling layer and ReLU activation, and ended with a dense layer. The two convolutional layers contain 8 and 16 filters, respectively. It is trained with learning rate of $5e-3$, batchsize of 100, the Adam optimizer, L1 regularization of $5e-4$, and train epochs of 5.

Similar to Figure 4, Figure 14 also shows that PID are capable of detecting high-order interactions that represent object shapes.

[7] https://www.kaggle.com/c/amazon-employee-access-challenge/discussion/4838
[8] https://www.kaggle.com/c/amazon-employee-access-challenge/discussion/5283