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  
song_3134@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  
Beijing, China  
yingshan@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 a common hidden unit. 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 Persistence Interaction Detection (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...
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 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 homology. 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 $\mathbf{W}^T$ or $\mathbf{x}^T$. For a set $\mathcal{S}$, its cardinality is denoted by $|\mathcal{S}|$. We use $\mathcal{S} \backslash j$ 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 $\geq 3$. 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 $W_l^{(i)} \in \mathbb{R}^{p_{l-1} \times p_l}$. Given a FNN with weights $\{W_l^{(i)}\}_{l=1}^L$, its equivalent weighted directed acyclic graph $G(V,E)$ 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 $W_{l,j}^{(i)}$ 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.
In this paper, the object we studied is a weighted directed graph $G = (V, E, \lambda)$, where weights, which indicates stronger connectivity, will appear over a wide range of spatial scales. As the threshold causes the death of $i$ time and $d_i$, 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).

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

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

For example, in the function of $x^2 + \log(x_1 + x_2 + x_4)$, 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 $G(V, E)$ 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 $G$. To be specific, $(G, \varphi)$ is called a size pair [14], where $\varphi$ is a measuring function [15]. The role of $\varphi$ is to take into account the connectivity properties of $G$. The $\lambda$-threshold set of $(G, \varphi)$ is defined as follows:

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

where $x$ is the edge of $G$. The measuring function $\varphi: E \rightarrow \mathbb{R}$ 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 $\varphi$ is non-decreasing over $G$. Given $(G, \varphi)$, where $\varphi$ is non-decreasing over $G$, 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 $(G, \varphi)$.

We propose to build the filtration for FNNs and define the measuring function as follows. Let $W$ be the set of weights. Given $W$ of a trained feed-forward neural network such that $w_{\text{max}} = \max_{w \in W} |w|$ and $W^\prime := \{ |w| / w_{\text{max}} | w \in W \}$, where $W^\prime$ is indexed in non-ascending order, namely $1 = w_0^\prime \geq w_1^\prime \geq ... \geq w_n^\prime \geq 0$. The weights associated with edges reflect the connectivity between vertices in the networks. Similar to [17], the measuring function $\varphi$ for $G$ is defined as $\varphi((v_{i-1,i}, v_{i,j})) = |W_{i,j}| / w_{\text{max}}, (v_{i-1,i}, v_{i,j}) \in E$, which represents the edge strength. The sorted weights are used as $\lambda$ in Definition 2. Consequently, the filtration can be constructed as $G^{w_0^\prime} \subseteq G^{w_1^\prime} \subseteq ...$, where $G^{w_i^\prime} = (V, \{(u, v) | (u, v) \in E \land \varphi((u, v)) \geq w_i^\prime\})$. We remark that $G^{\lambda}$ is both a subgraph of $G$ and a $\lambda$-threshold set of size pair $(G, \varphi)$. $G^{w_i^\prime}$ 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 $G^{\lambda}$ is that, $G^{\lambda}$ is the image of $G$ 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 $G$. 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 $G$ [18]. Based on this, persistence diagram is a computational tool for quantifying the importance of these emerged connected components.

### Persistence Diagrams
Given the filtration of a size pair $(G, \varphi)$, 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 \( D = \{(b_i, d_i) | \forall i \text{ appears in the filtration} \} \) is called the persistence diagrams of \((G, \phi)\). 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 \( w_i \).](image)

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 \( I \), we cannot directly obtain the importance of \( I \) from the persistence of connected components that contain \( I \). Also, an interaction \( I \) is a subset of input features that is not within the scope of persistent homology. 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 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 \((G, \phi)\) be the corresponding size pair and \( G^{w_0} \subseteq G^{w_1} \subseteq \ldots \) be the filtration of a neural network, respectively; and \( \{0, \ldots, d - 1\} \) denotes the set of input features. For a feature subset \( I \) and a real-number threshold \( \lambda \), we call \( I \) and the final output units are \( \langle \phi = \lambda \rangle \)-connected if: first, there exists a connected component \( A \subseteq G^\phi \) containing \( I \) and the final output units; second, for any such connected component \( A, \forall i \in \{0, \ldots, d - 1\} \setminus I \), it satisfies \( i \notin A \).

**Persistence diagrams of interactions** Given the threshold \( \lambda_0 \), suppose: The feature subset \( I \) and the final output unit are \( \langle \phi = \lambda_0 \rangle \)-connected and, \( \forall \lambda_i \geq \lambda_0, I \) and the final output unit are not \( \langle \phi = \lambda_i \rangle \)-connected, then we call \( \lambda_0 \) the birth time of \( I \). Correspondingly, the death time \( \lambda_d \) of \( I \) is that \( \forall \lambda_i \leq \lambda_d, I \) and the outputs become not \( \langle \phi = \lambda_i \rangle \)-connected, i.e., interaction \( I \) no longer exists
due to the addition of other input features. The collection of the birth time and the death time tuple $D = \{(b_I, d_I)| I \subseteq \{0, ..., d-1\}\}$ is called the persistence diagrams of interactions.

After defining the birth time and the death time of an interaction $I$, 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 $x = [x_1, x_2, x_3, x_4]$, $y = x_1^2 + \frac{x_2}{100}$. 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 $w'_0 \geq ... \geq w'_6$ are the top ten largest weights in $W^p$. 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 $\{x_1, x_2, x_3\}$.

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 $w'_3$, then the input feature $x_2$ and $x_3$ will 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 $M^\lambda_{(l)} \in \mathbb{R}^{p_l \times d}$ for the $l$th layer as

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

(1)

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

$$M^\lambda = (M^\lambda_{(L)})^\top \cdot (M^\lambda_{(L-1)})^\top \cdots (M^\lambda_{(1)})^\top.$$  

(2)

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

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 $\mathbf{M}^l$ into $\mathbf{M}_{(l)}^{\text{up}} \in \mathbb{R}^{p_l \times p_l}$ and $\mathbf{M}_{(l)}^{\text{down}} \in \mathbb{R}^{p_l \times d}$, where
\[
\begin{cases}
\mathbf{M}_{(l)}^{\text{up}} = (\mathbf{M}_{(l)}^1)^\top \cdots (\mathbf{M}_{(l)}^{l})^\top, \\
\mathbf{M}_{(l)}^{\text{down}} = (\mathbf{M}_{(l-1)}^1)^\top \cdots (\mathbf{M}_{(1)}^{l})^\top.
\end{cases}
\]

We can obtain the connectivity between a particular neuron $r$ at layer $l$ and units in the final output layer from $\mathbf{M}_{(l)}^{\text{up}}$ (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 $\mathbf{M}_{(l)}^{\text{down}}$. 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 $w_0^*$. 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].

### 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 $G_f(V,E)$ and $G_g(V,E)$ be their corresponding weighted graph, respectively. We denote the measuring function for $f$ and $g$ as $\phi_f$ and $\phi_g$, 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 $\delta$ 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.
We note that PID considers connectivity of the entire NN. In contrast, NID leverages weights beyond the architectures (Appendix E.3) and regularization strength (Appendix E.3 Figure 8, 9). 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.

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

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 candidate and combine these synthetic features with the original feature set. The synthetic feature for interaction $I_i$ is the Cartesian product among features in $I_i$. 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 constructed 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.

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.](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 together 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


