DISCOVERING AND EXPLAINING THE REPRESENTATION BOTTLENECK OF DNNS

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper explores the bottleneck of feature representations of deep neural networks (DNNs), from the perspective of the complexity of interactions between input variables encoded in DNNs. To this end, we focus on the multi-order interaction between input variables, where the order represents the complexity of interactions. We discover that a DNN is more likely to encode both too simple interactions and too complex interactions, but usually fails to learn interactions of intermediate complexity. Such a phenomenon is widely shared by different DNNs for different tasks. This phenomenon indicates a cognition gap between DNNs and human beings, and we call it a *representation bottleneck*. We theoretically prove the underlying reason for the representation bottleneck. Furthermore, we propose a loss to encourage/penalize the learning of interactions of specific complexities, and analyze the representation capacities of interactions of different complexities.

1 INTRODUCTION

The revolution from shallow to deep models is a crucial step in the development of artificial intelligence. Deep neural networks (DNNs) usually exhibit superior performance to shallow models, which is generally believed as a result of the improvement of the representation power (Pascanu et al., 2013; Montúfar et al., 2014). To this end, instead of considering previous issues of the accuracy and the generalization ability of DNNs, we focus on the following two questions about the representation capacity:

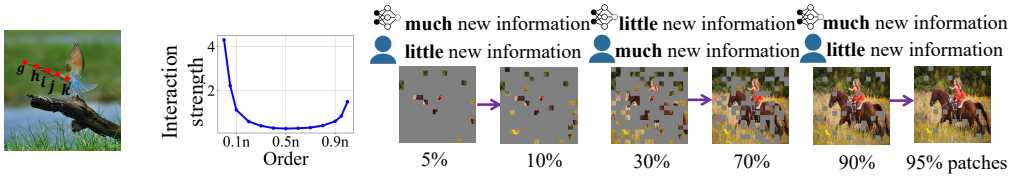
• Are there any common tendencies of DNNs in representing specific types of features?
• Does a DNN encode similar visual concepts as human beings for image classification?

In order to answer the above two questions, our research first investigates the bottleneck of knowledge representation, *i.e.*, which types of knowledge points (*e.g.*, certain visual concepts) are likely to be encoded by a DNN, and which types of knowledge points are difficult to be learned. To this end, we discover that the interaction between input variables is an effective tool to analyze the knowledge representation. It is because the DNN does not consider input variables working independently, but it encodes the interaction between input variables to form an interaction pattern for inference. For example, the inference of a face image can be explained as the interaction between left and right eyes, the interaction between nose and mouth, and etc.

As the answers to the above questions, we discover a common representation bottleneck of DNNs in encoding interactions, *i.e.*, a DNN is more likely to encode both too complex and too simple interactions, instead of encoding interactions of intermediate complexity. This also indicates a dramatic difference from human cognition.

The interaction can be understood as follows. Let us take the face recognition task for example. Let $\phi_{i=\text{mouth}}$ measure the numerical importance of the mouth region i to the classification score. Then, the interaction utility between the mouth region i and the nose region j is measured as the change of the $\phi_{i=\text{mouth}}$ value by the presence or absence of the nose region j. If the presence of j increases the importance $\phi_{i=\text{mouth}}$ by 0.1, then, we consider 0.1 as the utility of the interaction between i and j.

Multi-order interactions. In order to represent the interaction complexity mentioned in the representation bottleneck, we use the multi-order interaction utility between variables i, j proposed by Zhang et al. (2020). The interaction of the m-th order $I^{(m)}(i, j)$ measures the average interaction utility between variables i, j on all contexts consisting of m variables. In this way, the order m



(a) Interaction pattern (b) Representation bottleneck (c) Whether humans/DNNs extract new information from patches.

Figure 1: (a) Five pixels (g, h, i, j, k) interact with each other, forming an edge pattern for classification. (b) Representation bottleneck. A DNN is likely to encode low-order and high-order interactions, but usually fails to learn middle-order interactions. (c) The cognition gap between DNNs and humans. Humans extract little information from a few image patches (*e.g.*, 5% patches). Also, given almost all patches (*e.g.*, 90% patches), the information is already redundant for human cognition. Then, people learn little new information from the additional 5% patches. In comparison, the DNN encodes most interactions when the DNN is given very few patches or most patches.

reflects the contextual complexity of the interaction. A low-order $I^{(m)}(i, j)$ measures the relatively simple collaboration between variables i, j and a few m contextual variables, while a high-order $I^{(m')}(i, j)$ corresponds to the complex collaboration between i, j and m' massive contextual variables, where $m' \gg m$.

Moreover, we prove that the multi-order interaction is a trustworthy tool to analyze the representation capacity of DNNs. Specifically, the output score of a DNN can be decomposed into utilities of compositional multi-order interactions between different pairs of variables, *i.e.*, *model output* $= \sum_{m=0}^{n-2} \sum_{i,j \in N, i \neq j} w^{(m)} I^{(m)}(i, j)$ + $\sum_{i \in N}$ *local utility of* i + *bias*. For example, the inference score of a face can be decomposed into the interaction utility between left and right eyes, between mouth and nose, and etc. Therefore, we can take the utility $I^{(m)}(i, j)$ as the underlying reason to explain the DNN, because each interaction makes a compositional contribution to the output.

Representation bottleneck of DNNs. Surprisingly, the above decomposition of multi-order interactions enables us to discover a representation bottleneck of DNNs. As Figure 1(b) shows, low-order and high-order interaction utilities $I^{(m)}(i, j)$ usually have high absolute values, while middle-order interaction utilities $I^{(m)}(i, j)$ usually have low absolute values. In other words, a DNN is more likely to encode the interaction between variables i and j, when i, j interact with a few contextual variables. Similarly, it is also easy for the DNN to learn the interaction, when i, j interact with most contextual variables. However, it is difficult for the DNN to learn the interaction, when i, j cooperate with a medium number of contextual variables. The difficulty of learning middle-order interactions reflects a representation bottleneck of DNNs.

Cognitive gap between DNNs and humans. Such a representation bottleneck also indicates a significant gap between the knowledge representation of DNNs and the visual cognition of humans. As Figure 1(c) shows, people usually cannot extract meaningful information from a few image patches. Besides, if people are given almost all patches, then the information is already too redundant and inserting additional patches will bring in little new information. In contrast, the DNN encodes most information, when the DNN is given only a few patches or is given most patches.

Theoretical proof. In this paper, we theoretically prove the mechanism that is responsible for the representation bottleneck. Such a proof also enables us to simulate the distribution of interactions of different orders, which well matches the distribution of interactions in real applications.

Beyond the theoretical proof, another important issue is how to guide the learning of knowledge representation in DNNs by learning interactions of specific orders. We propose two losses to encourage/penalize the DNN to make inferences by interactions of specific orders, thereby boosting/preventing the learning of such interactions. Experimental results have validated the effectiveness of the two losses. Next, we investigate the representation capacities of several DNNs which encoded interactions of different orders. We find that the DNNs mainly encoding high-order interactions represent more structural information than the normally trained DNNs. In addition, high-order interactions were vulnerable to adversarial attacks.

In summary, this paper makes three contributions:
• This study discovers a representation bottleneck phenomenon of DNNs, *i.e.*, it is difficult for a DNN to learn middle-order interactions. It also clearly proves that DNNs and humans use different

types of visual concepts for inference.
• We theoretically prove the underlying reason for the representation bottleneck.
• We design two losses to encourage/penalize the DNN to learn interactions of specific orders. Experiments have validated the effectiveness of the proposed losses. Besides, we investigate the representation capacities of DNNs which encode interactions of different orders.

2 RELATED WORK

The representation capacity of DNNs. The evaluation of the representation capacity of DNNs provides a new perspective to explain and analyze DNNs. Pascanu et al. (2013) and Montúfar et al. (2014) used the number of linear response regions in a deep rectifier MLP to evaluate its representation capacity. The information bottleneck theory (Shwartz-Ziv & Tishby, 2017) used the mutual information to explain how DNNs gradually learned the information during the training process. Achille & Soatto (2018), Amjad & Geiger (2019) and Hjelm et al. (2019) further improved the representation capacity by optimizing mutual information. Arpit et al. (2017) studied the memorization behavior of DNNs during training to analyze the feature representations. Xu (2018) proposed Fourier analysis to understand the generalization. In addition, several metrics were proposed to analyze the generalization capacity or robustness of DNNs, including the stiffness (Fort et al., 2019), the sensitivity (Novak et al., 2018), and the CLEVER score (Weng et al., 2018). Neyshabur et al. (2017) examined whether existing complexity measures can guarantee generalization.

Previous researches mainly studied the theoretically maximum complexity, generalization ability, and robustness of DNNs. In comparison, our research focuses on the limitation of DNNs in feature representations, *i.e.*, which types of interactions are unlikely to be encoded.

Interactions. Interactions between input variables of a DNN have been widely investigated in recent years. Based on the Shapley value (Shapley, 1951), Grabisch & Roubens (1999) proposed the Shapley interaction index to define the interaction in a cooperative game. Lundberg et al. (2018) used the interaction effect to build tree ensemble explanations for the decision of DNNs. Janizek et al. (2021) extended Integrated Gradients (Sundararajan et al., 2017) to explain the pairwise feature interaction in DNNs. Sundararajan et al. (2020) proposed the Shapley Taylor interaction to measure interactions among multiple variables. Tsang et al. (2020) and Tsang et al. (2017) interpreted DNNs by detecting statistical interactions between input variables and interactions between network weights respectively. Peebles et al. (2020) and Tsang et al. (2018) achieved the disentanglement of features by restricting interactions. Song et al. (2019) and Lian et al. (2018) designed network architectures to effectively learn feature interactions in the recommendation system. Lengerich et al. (2020) applied the ANOVA technique to model interaction effects and further explored the relationship between dropout and interactions. Zhang et al. (2020) proposed the multi-order interaction, and used it to understand and boost dropout. In this paper, we used the multi-order interaction to study the complexity of interactions encoded in DNNs.

3 REPRESENTATION BOTTLENECK

Before the analysis of the representation bottleneck, let us first introduce multi-order interactions between input variables, which are encoded in a DNN. Given a pre-trained DNN v and an input sample with a set of n variables $N = \{1, \dots, n\}$ (*e.g.*, an input image with n pixels), $v(N)$ denotes the network output of all input variables. Input variables of DNNs usually interact with each other to make inferences, instead of working individually. In this study, we mainly discuss the pairwise interactions. For example, as Figure 1(a) shows, pixels $i, j \in N$ interact with each other, forming an edge pattern for classification. If the existence of this pattern increases the network output by 0.01, we consider the *utility* of this pattern as 0.01. Similarly, if the existence of this pattern decreases the network output, we consider this pattern has a negative utility.

Furthermore, the multi-order interaction $I^{(m)}(i, j)$ between two input variables $i, j \in N$, $0 \leq m \leq n-2$, is proposed to measure interactions of different complexities (Zhang et al., 2020). Specifically, the m-th order interaction $I^{(m)}(i, j)$ measures the average interaction utility between variables i, j under all possible contexts consisting of m variables. Therefore, the order m can be considered to represent the contextual complexity of the interaction. For example, as Figure 1(a) shows, five pixels (i, j, g, h, k) collaborate with each other and form an edge pattern for classification. Thus, the pair-

wise interaction between pixels i, j also depends on three contextual pixels g, h, k. Mathematically, the multi-order interaction $I^{(m)}(i, j)$ is defined as follows:

$$I^{(m)}(i, j) = \mathbb{E}_{S \subseteq N \setminus \{i,j\}, |S|=m}[\Delta v(i, j, S)], \tag{1}$$

where $\Delta v(i, j, S) = v(S \cup \{i, j\}) - v(S \cup \{i\}) - v(S \cup \{j\}) + v(S)$. Here, $v(S)$ is the output score when we keep variables in $S \subseteq N$ unchanged but replace variables in $N \setminus S$ by the baseline value. The baseline value follows the widely-used setting in Ancona et al. (2019), which is set as the average value of the variable over different samples. Let us take the multi-category image classification for example. Given an input image x, $v(S) = v(x_S)$ can be implemented as any scalar output of the DNN (*e.g.*, $\log \frac{P(\hat{y}=y^{\text{truth}}|x_S)}{1-P(\hat{y}=y^{\text{truth}}|x_S)}$ of the true category), where we replace the pixel values in $N \setminus S$ of original input x by the baseline value (the average pixel value over images) to construct a masked image x_S. Then, $\Delta v(i, j, S) = [v(S \cup \{i, j\}) - v(S \cup \{i\})] - [v(S \cup \{j\}) - v(S)]$ measures the marginal interaction effects of the collaboration between i, j in a context S.

Generic metric. The proposed multi-order interaction $I^{(m)}(i, j)$ is a generic metric, which has a strong connection with the Shapley value (Shapley, 1951) and the Shapley interaction index (Sundararajan et al., 2020) in game theory. In addition, it has been proven that $I^{(m)}(i, j)$ satisfies the following five desirable properties, *i.e.*, *linearity*, *nullity*, *commutativity*, *symmetry*, and *efficiency* properties. The connections with existing metrics and five properties are introduced in Appendix A.

3.1 Representation bottleneck

According to the ***efficiency*** property of $I^{(m)}(i, j)$, we find that the output of a DNN can be explained as the sum of all interaction utilities of different orders between different pairs of variables.

$$v(N) = v(\emptyset) + \sum\nolimits_{i \in N} \mu_i + \sum\nolimits_{i,j \in N, i \neq j} \sum\nolimits_{m=0}^{n-2} w^{(m)} I^{(m)}(i, j) \tag{2}$$

where $\mu_i = v(\{i\}) - v(\emptyset)$, and $w^{(m)} = \frac{n-1-m}{n(n-1)}$. Because $I^{(m)}(i, j)$ measures the interaction between variables i and j encoded in DNNs with m contextual variables, we can consider the interaction utility $I^{(m)}(i, j)$ as a specific reason for the inference, which makes a compositional contribution $w^{(m)} I^{(m)}(i, j)$ to the output.

In this way, we can categorize all underlying reasons for the network output into different complexities. Low-order interactions can be considered as simple underlying reasons, relying on very few variables. High-order interactions can be regarded as complex underlying reasons, depending on massive variables. In order to measure the reasoning complexity of the DNN, we measure the relative interaction strength $J^{(m)}$ of the encoded m-th order interaction as follows:

$$J^{(m)} = \frac{\mathbb{E}_{x \in \Omega}[\mathbb{E}_{i,j}[|I^{(m)}(i, j|x)|]]}{\mathbb{E}_{m'}[\mathbb{E}_{x \in \Omega}[\mathbb{E}_{i,j}[|I^{(m')}(i, j|x)|]]]} \tag{3}$$

where Ω denotes the set of all samples. $J^{(m)}$ is computed over all pairs of input variables in all samples. $J^{(m)}$ is normalized by the average value of interaction strength. The distribution of $J^{(m)}$ measures the distribution of the complexity of interactions encoded in DNNs.

Representation bottleneck. Based on the above metric, we discover an interesting phenomenon: *a DNN usually encodes strong low-order and high-order interactions, but encodes weak middle-order interactions*. Such a phenomenon is shared by different DNN architectures trained on different datasets, which is illustrated by the $J^{(m)}$ curves in Figure 2. Specifically, when the order m is smaller than $0.1n$ or greater than $0.9n$, its interaction strength $J^{(m)}$ is usually high. In comparison, $J^{(m)}$ is usually low when the order m approximates $0.5n$. Moreover, as shown in Figure 3(a), we find that such a phenomenon does not only exists in the well-trained DNNs, but also exists in the entire network training process.

The above phenomenon indicates that a DNN is more likely to learn simple interactions where a few variables (*e.g.*, less than $0.1n$ variables) interact with each other. Similarly, it is also easy for the DNN to encode the complex interaction where massive variables (*e.g.*, more than $0.9n$ variables) interact with each other. However, it is difficult for the DNN to learn the middle-complex interaction in which a medium number of variables (*e.g.*, about $0.5n$ variables) participate. Let us take Figure 1(c) for an example. When a DNN is given very few patches sparsely distributed on the horse image,

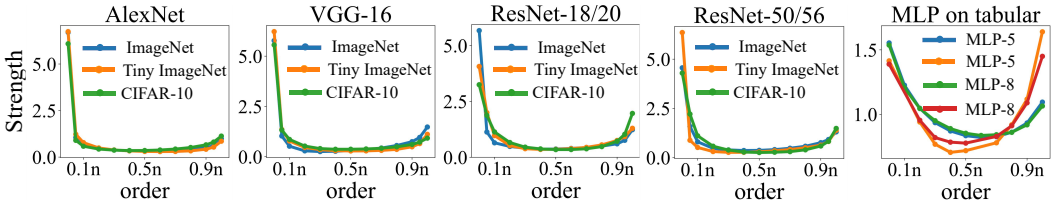


Figure 2: The distributions of interaction strength $J^{(m)}$ of different DNNs trained on various image datasets and tabular datasets. Noted that the relative strength $J^{(m)}$ of high-order interactions on two tabular datasets is higher than that on image datasets. This may be because that the classification on images usually depends on low-order interactions, but the classification on tabular data mainly depends on high-order interactions, thereby forcing the DNN to learn high-order interactions.

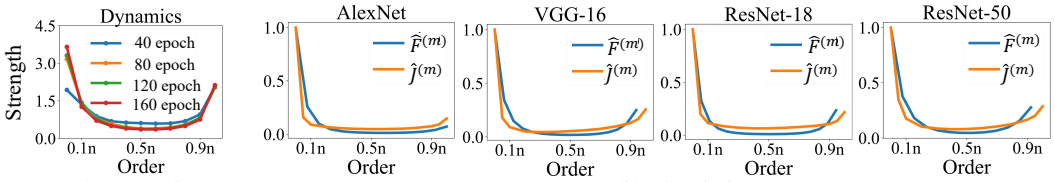
the DNN can successfully extract the interaction between the few patches. Similarly, when the DNN is given almost all patches of the horse image, then the insertion of two new patches will make the DNN trigger strong collaboration between the two patches and massive existing patches. However, when the DNN is just given a half patches, it is difficult for the DNN to encode collaborations between two patches. In a word, the interaction pattern encoded by the DNN is either too simple or too complex. The difficulty of learning interaction patterns of moderate complexity reflects a common tendency in the feature representation in the DNN.

Such a representation bottleneck also indicates that DNNs and human beings encode different types of visual patterns for inference. As Figure 1(c) shows, (i) Given very few patches, a DNN can encode much information from low-order interaction patterns between patches. However, it is difficult for people to recognize such low-order interactions. (ii) Given almost all patches of an image, any additional patches are already too redundant for human cognition, so people do not obtain much new information from additional patches. (iii) Given a medium number of patches, the DNN usually extracts little information, while people can extract much information for recognition.

Implementation details. In order to measure $J^{(m)}$, we conducted experiments on three image datasets including the ImageNet dataset (Russakovsky et al., 2015), the Tiny-ImageNet dataset (Le & Yang, 2015) and the CIFAR-10 dataset (Krizhevsky et al., 2009). We mainly analyzed several DNNs trained on these datasets for image classification, including Alexnet (Krizhevsky et al., 2012), VGG-16 (Simonyan & Zisserman, 2014) and Resnet-18/20/50/56 (He et al., 2016). Due to the high dimension of input variables ($n = 224 \times 224$ for ImageNet), the computational cost of $J^{(m)}$ is intolerable. To reduce the computational cost, we split the input image into 16×16 patches, and considered each patch as an input variable. To compute $J^{(m)}$, we set $v(S|x) = \log \frac{P(\hat{y}=y^*|x_S)}{1-P(\hat{y}=y^*|x_S)}$ given the masked sample x_S, where y^* was the true label and $P(\hat{y} = y^*|x_S)$ was the probability of classifying the masked sample x_S to the true category. In the masked sample x_S, pixel values in image patches in $N \setminus S$ were replaced by the average pixel value over different patches in all images, just like in Ancona et al. (2019). Note that $J^{(m)}$ is an average over all possible contexts S, all pairs of variables (i, j), and all samples x, which is computationally infeasible. Therefore, we approximated $J^{(m)}$ using a sampling strategy (Zhang et al., 2020). Please see Appendix C for sampling details. In addition, we conducted experiments on two tabular datasets, including the UCI census income dataset (census) and the UCI TV news channel commercial detection dataset (commercial) (Dua et al., 2017). Each sample in the two datasets contained $n = 12$ and $n = 10$ input variables, respectively. We analyzed a five-layer MLP (namely, MLP-5) and an eight-layer MLP (namely, MLP-8) network. Each layer except for the output layer contained 100 neurons. For each MLP network, we trained two models for the classification on the two datasets, respectively. In the computation of $J^{(m)}$, $P(\hat{y} = y^*|x_S)$ was also computed by setting the baseline value of a variable to the average value of the variable. Please see Appendix C for details.

3.2 Explaining the representation bottleneck

In this subsection, we theoretically prove the underlying reason for the representation bottleneck. Let W denote network parameters of a DNN. We focus on the change ΔW of network parameters, which also represents the strength of training the DNN. The change of weights is calculated by $\Delta W = -\eta \frac{\partial L}{\partial W} = -\eta \frac{\partial L}{\partial v(N)} \frac{\partial v(N)}{\partial W}$. L denotes the loss function, and η is the learning rate. According to Eq. (2), the network output $v(N)$ of the DNN can be decomposed into the sum of multi-order



(a) Dynamics (b) Simulations

Figure 3: (a) Distributions of the interaction strength $J^{(m)}$ of a ResNet-20 model over different orders, which are measured after different training epoches. The DNN is trained on the CIFAR-10 dataset. (b) Simulations of the $\hat{J}^{(m)}$ distributions based on $\hat{F}^{(m)}$ curves on the ImageNet dataset.

interactions $I^{(m)}(i,j)$. Therefore, ΔW can be further represented as the sum of gradients $\frac{\partial I^{(m)}(i,j)}{\partial W}$ of multi-order interactions.

$$\Delta W = -\eta \frac{\partial L}{\partial v(N)} \frac{\partial v(N)}{\partial W} = \Delta W_U + \sum\nolimits_{m=0}^{n-2} \sum\nolimits_{i,j\in N, i\neq j} \Delta W^{(m)}(i,j), \tag{4}$$

where $U = v(\emptyset) + \sum_{i\in N} \mu_i$. Specifically,

$$\Delta W_U \overset{\text{def}}{=} -\eta \frac{\partial L}{\partial v(N)} \frac{\partial v(N)}{\partial U} \frac{\partial U}{\partial W}, \quad \Delta W^{(m)}(i,j) \overset{\text{def}}{=} R^{(m)} \frac{\partial I^{(m)}(i,j)}{\partial W},$$

where $R^{(m)} = -\eta \frac{\partial L}{\partial v(N)} \frac{\partial v(N)}{\partial I^{(m)}(i,j)}$. Here, ΔW_U represents the component of ΔW *w.r.t.* $\frac{\partial U}{\partial W}$, and $\Delta W^{(m)}(i,j)$ represents the component of ΔW *w.r.t.* $\frac{\partial I^{(m)}(i,j)}{\partial W}$. Therefore, besides ΔW_U, we can consider there are additional $\frac{n(n-1)^2}{2}$ paths *w.r.t.* different pairs of i,j and different orders m in the backpropagation, and the weight change through each propagation path is $\Delta W^{(m)}(i,j)$. In this way, we can consider $\Delta W^{(m)}(i,j)$ measures the strength of learning the interaction between variables i and j under contexts of m variables.

Theorem 1. (*Proof in Appendix B*) Assume $\mathbb{E}_{i,j,S}[\frac{\partial \Delta v(i,j,S)}{\partial W}] = 0$. Let σ^2 denote the variance of $\frac{\partial \Delta v(i,j,S)}{\partial W}$. Then for $\Delta W^{(m)}(i,j) = R^{(m)} \frac{\partial I^{(m)}(i,j)}{\partial W}$, $\mathbb{E}_{i,j}[\Delta W^{(m)}(i,j)] = 0$ and the variance of $\Delta W^{(m)}(i,j)$ is $(\eta \frac{\partial L}{\partial v(N)} w^{(m)})^2 \sigma^2 / \binom{n-2}{m}$, where $w^{(m)} = \frac{n-m-1}{n(n-1)}$.

Theorem 1 shows that the strength $|\Delta W^{(m)}(i,j)|$ of learning m-order interactions is proportional to $F^{(m)} = w^{(m)} / \sqrt{\binom{n-2}{m}} = \frac{n-m-1}{n(n-1)} / \sqrt{\binom{n-2}{m}}$. Therefore, when the order m is small or large (*e.g.*, $m = 0.05n$ or $0.95n$), the training strength of the m-order interaction is relatively higher. In contrast, when the order m is medium (*e.g.*, $m = 0.5n$), the training strength of the m-order interaction is much lower. The above analysis explains why it is easy for a DNN to learn low-order and high-order interactions, but difficult for a DNN to learn middle-order interactions.

Simulation of the curve of the interaction strength. We find that the above training strength $|\Delta W^{(m)}(i,j)|$ can be used to simulate the distribution of interaction strengths $J^{(m)}$ in real applications, which verifies our theory. Based on Theorem 1, the training strength *w.r.t.* the order m is proportional to the aforementioned $F^{(m)}$, so we can use $F^{(m)}$ to simulate $J^{(m)}$. For fair comparison, we normalize $F^{(m)}$ and $J^{(m)}$ by $\hat{F}^{(m)} = F^{(m)}/F^{(0)}$ and $\hat{J}^{(m)} = J^{(m)}/J^{(0)}$ respectively, such that $\hat{F}^{(0)} = \hat{J}^{(0)} = 1$. As Figure 3(b) shows, the curves of normalized $\hat{F}^{(m)}$ can well match the distributions of $\hat{J}^{(m)}$ in real applications. Due to the redundancy of feature representations in DNNs, we usually consider that the actual dimension n' of the latent space of DNNs is much lower than the number n of input variables. Therefore, instead of directly using the number n of input variables, we adopted a smaller n' (*i.e.*, $n' < n$) in $\hat{F}^{(m)}$ for the simulation.

3.3 Method to control interactions of specific orders

The representation bottleneck is widely shared by DNNs of different architectures for various tasks, when DNNs are normally trained. In this section, we mainly explore methods, which force the DNN to learn interactions of specific orders. In this way, we can investigate the properties of feature representations of such DNNs.

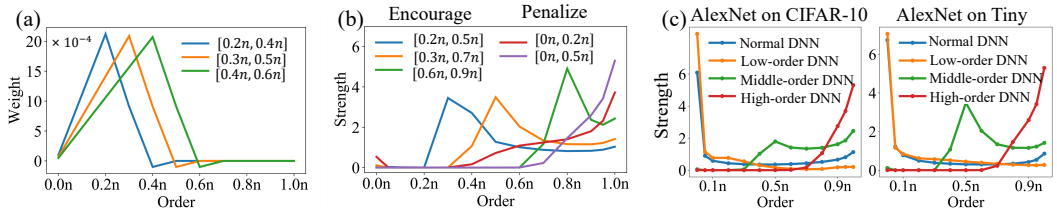


Figure 4: (a) Weight coefficients $\tilde{w}^{(m)}$ of different orders when we set different pairs of (r_1, r_2). (b) Distributions of the interaction strength $J^{(m)}$ over different orders. Each curve indicates a DNN (using AlexNet) whose interactions were encouraged/penalized by $L^+(r_1, r_2)$ and $L^-(r_1, r_2)$ with certain pairs of (r_1, r_2). (c) Distributions of the interaction strength $J^{(m)}$ of four types of DNNs using AlexNet. The Appendix C provides more results.

In order to force the DNN to learn interactions of specific orders, we propose two simple-yet-efficient losses in the training process. The two losses encourage and penalize interactions of specific orders, respectively. Before designing the two losses, let us focus on the output change $\Delta u(r_1, r_2)$ between two subsets of variables S_1 and S_2:

$$\Delta u(r_1, r_2) = \mathbb{E}_{S_1, S_2 : \emptyset \subseteq S_1 \subsetneq S_2 \subseteq N}[v(S_2) - \frac{r_2}{r_1} v(S_1)], \tag{5}$$

where the subsets S_1 and S_2 are randomly sampled from all input variables N, such that $\emptyset \subseteq S_1 \subsetneq S_2 \subseteq N$, $|S_1| = r_1 n$, $|S_2| = r_2 n$, and $0 \leq r_1 < r_2 \leq 1$.

Theorem 2. (*Proof in Appendix B*) The output change $\Delta u(r_1, r_2)$ can be decomposed into the sum of multi-order interactions between different pairs of variables.

$$\Delta u(r_1, r_2) = (1 - \frac{r_2}{r_1}) v(\emptyset) + \sum\nolimits_{m=0}^{n-2} \sum\nolimits_{i,j \in N, i \neq j} \tilde{w}^{(m)} I^{(m)}(i, j)$$
$$\text{where} \quad \tilde{w}^{(m)} = \begin{cases} (\frac{r_2}{r_1} - 1)(m+1)/[n(n-1)], & m \leq r_1 n - 2 \\ (r_2 n - m - 1)/[n(n-1)], & r_1 n - 2 < m \leq r_2 n - 2 \\ 0, & r_2 n - 2 < m \leq n - 2 \end{cases} \tag{6}$$

Interestingly, as Figure 4(a) shows, we can consider that the output change $\Delta u(r_1, r_2)$ mainly encodes interactions whose orders are in the range of $[0, r_2 n]$. The weight coefficient $\tilde{w}^{(m)}$ of the m-th order interaction reached a peak at the $r_1 n$-th order in $\Delta u(r_1, r_2)$.

Encourage/penalize interactions of specific orders. Based on the above analysis, $\Delta u(r_1, r_2)$ only contains partial interactions of the $[0, r_2 n]$-th orders. Hence, we propose two losses based on $\Delta u(r_1, r_2)$, which encourage and penalize the DNN to use interactions of specific orders for inference, respectively. The first loss $L^+(r_1, r_2)$ forces the DNN to mainly use interactions encoded in $\Delta u(r_1, r_2)$ for inference, thereby boosting the learning of these interactions.

$$L^+(r_1, r_2) = -\frac{1}{|\Omega|} \sum\nolimits_{x \in \Omega} \sum\nolimits_{c=1}^{C} P(y^* = c|x) \log P(\hat{y} = c|\Delta u_c(r_1, r_2|x)), \tag{7}$$

where $L^+(r_1, r_2)$ is the proposed cross entropy that uses $\Delta u(r_1, r_2)$ for classification. Here, Ω is the training set, and C denotes the number of classes. Given an input image $x \in \Omega$, y^* is the true label, and $\hat{y}$ denotes the predicted label. Here, $\Delta u_c(r_1, r_2|x) = v_c(S_2|x) - \frac{r_2}{r_1} v_c(S_1|x)$ denotes the change of the logits of the category c, where the logit $v_c(S|x)$ denotes the feature dimension corresponding to the c-th category before the softmax layer. The two subsets S_1, S_2 are randomly sampled. In this way, we compute $P(\hat{y} = c|\Delta u_c(r_1, r_2|x))$ as the probability of using the C-dimensional vector $\{\Delta u_c(r_1, r_2|x)|1 \leq c \leq C\}$ to classify the sample to the category c. We input the C-dimensional vector $\{\Delta u_c(r_1, r_2|x)|1 \leq c \leq C\}$ into the softmax layer to compute the probability.

Besides, the second loss $L^-(r_1, r_2)$ is designed to prevent the DNN from encoding interactions of the $[0, r_2 n]$-th orders. Specifically, we maximize the entropy of classification based on $\Delta u(r_1, r_2)$, in order to make $\Delta u(r_1, r_2)$ non-discriminative.

$$L^-(r_1, r_2) = \frac{1}{|\Omega|} \sum\nolimits_{x \in \Omega} \sum\nolimits_{c=1}^{C} P(\hat{y} = c|\Delta u_c(r_1, r_2|x)) \log P(\hat{y} = c|\Delta u_c(r_1, r_2|x)), \tag{8}$$

where $L^-(r_1, r_2)$ denotes the minus entropy of the classification probability based on $\Delta u_c(r_1, r_2)$.

In this way, we can train a DNN using the following loss,

$$\text{Loss} = \text{Loss}_{\text{classification}} + \lambda_1 L^+(r_1, r_2) + \lambda_2 L^-(r_1, r_2) \tag{9}$$

Table 1: (left) Classification accuracies of four types of DNNs, including the normally trained DNNs, and the other three types of DNNs mainly encoding low-order, middle-order, and high-order interactions. (right) Comparison of adversarial accuracies between normally trained DNNs and DNNs mainly encoding high-order interactions on the census dataset and the commercial dataset.

	CIFAR-10			Tiny-ImageNet		
Model	AlexNet	VGG16	VGG19	AlexNet	VGG16	VGG19
Normal training	88.52	90.50	90.61	56.00	56.16	52.56
Low interaction	86.97	89.99	89.74	57.52	55.04	55.44
Mid interaction	86.65	90.29	90.03	53.88	55.84	53.36
High interaction	88.68	90.84	90.79	56.12	55.36	53.28

Model	Normal training	Penalize low-order & boost high-order
MLP-5 on census	38.22	**7.31**
MLP-8 on census	39.33	**2.02**
MLP-5 on commer	27.01	**22.00**
MLP-8 on commer	25.92	**20.58**

where $\lambda_1 \geq 0, \lambda_2 \geq 0$ are two constants to balance the three terms.

Effects of the two losses. We find that the loss $L^-(r_1, r_2)$ can successfully penalize interactions of $[0, r_2 n]$-th orders. $L^+(r_1, r_2)$ can encourage interactions of the $[r_1 n, r_2 n]$-th orders, instead of encouraging interactions of the $[0, r_2 n]$-th orders. Specifically, we conducted the following experiments. We trained AlexNet networks on the Tiny-ImageNet dataset to encourage interactions of specific orders without penalizing any interactions by setting $\lambda_1 = 1, \lambda_2 = 0$. In experiments, we set $[r_1 = 0.2, r_2 = 0.5]$, $[r_1 = 0.3, r_2 = 0.7]$, and $[r_1 = 0.6, r_2 = 0.9]$ in the $L^+(r_1, r_2)$ loss to learn three AlexNet models, respectively. We also trained AlexNet models to penalize interactions of specific orders by setting $\lambda_1 = 0, \lambda_2 = 1$. Two DNNs were trained by setting $[r_1 = 0, r_2 = 0.2]$ and $[r_1 = 0, r_2 = 0.5]$ in the $L^-(r_1, r_2)$ loss, respectively. Figure 4(b) shows the interaction strength $J^{(m)}$ of these DNNs. When we encouraged the DNN to encode interactions of the $[r_1 n, r_2 n]$-th orders, the interaction strength $J^{(m)}$ of the $[r_1 n, r_2 n]$-th orders significantly increased, compared to the normally trained DNNs. Figure 4(b) also shows that the loss $L^-(r_1, r_2)$ could successfully remove interactions of the $[0, r_2 n]$-th orders.

3.4 INVESTIGATION OF THE REPRESENTATION CAPACITIES

In the previous subsection, we introduce two losses, which force the DNN to encode interactions of different orders. In this subsection, we investigate the representation capacities of such DNNs. Thus, we conducted experiments to train four types of DNNs. The first DNN was normally trained. The other three types of DNNs were trained to mainly encode low-order, middle-order, and high-order interactions, respectively. Specifically, the second DNN was trained to penalize interactions of the $[0.7n, n]$-th orders by minimizing the $L^-(r_1, r_2)$ loss with $\lambda_1 = 0, \lambda_2 = 1, r_1 = 0.7, r_2 = 1.0$. The third DNN was learned to boost interactions of the $[0.3n, 0.7n]$-th orders by minimizing the $L^+(r_1, r_2)$ loss with $\lambda_1 = 1, \lambda_2 = 0, r_1 = 0.3, r_2 = 0.7$. The fourth DNN was trained to penalize interactions of the $[0, 0.5n]$-th orders by minimizing the $L^-(r_1, r_2)$ loss with $\lambda_1 = 0, \lambda_2 = 1, r_1 = 0, r_2 = 0.5$. The second DNN, the third DNN, and the fourth DNN were termed *the low-order DNN*, *the middle-order DNN* and *the high-order DNN*, respectively. In experiments, we trained three versions of each DNN based on the above four settings. We applied architectures of AlexNet and VGG-16/19 architectures on the CIFAR-10 and the Tiny-ImageNet dataset.

Figure 4(c) and Figure 7 (in the appendix) show that the trained DNNs successfully learned interactions as expected. In other words, interactions of the $[0.7n, n]$-th orders were penalized in the low-order DNN. Interactions of the $[0.3n, 0.7n]$-th orders were boosted in the middle-order DNN. Interactions of the $[0, 0.5n]$-th orders were penalized in the high-order DNN.

Classification accuracy. Firstly, Table 1 shows classification performance of the above four types of DNNs. In general, the four types of DNNs achieved similar accuracies. The similar performance indicated that it was not necessary for a DNN to encode low-order interactions and high-order interactions to make inferences. Middle-order interactions can also provide discriminative information.

Bag-of-words representations vs. structural representations. Theoretically, high-order interactions usually represent the global structure of objects, which requires the complex collaborations of massive input variables. In comparison, low-order interactions learn local patterns from local and simple collaborations of a few input variables.

Therefore, we conducted two experiments to examine whether the high-order DNN encoded more structural information than the normally trained DNN. Specifically, as Figure 5 shows, in the first experiment, we tested the DNN on images where m patches in each image were randomly masked. In

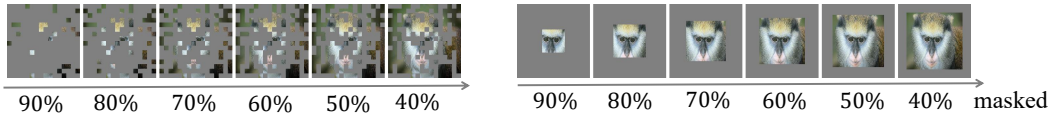


Figure 5: Tested images by random masking (left) and centrally-surrounding masking (right).

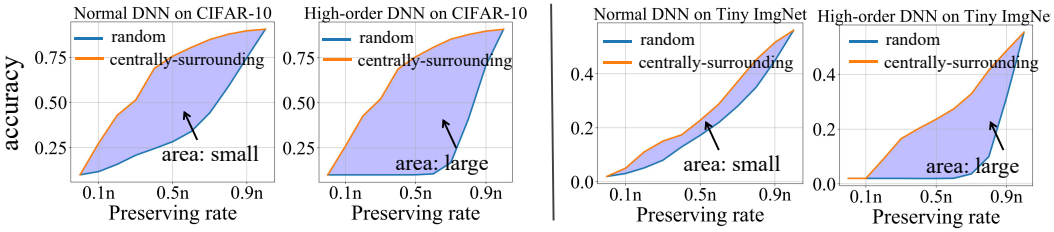


Figure 6: Classification accuracies using VGG-16 on images with different numbers of patches being masked. The Appendix C provides more results.

the second experiment, we tested the DNN on images where m patches in each image on the image boundary were masked, and the patches in the center were preserved. In this way, we consider the structural information in tested images was destroyed in the first experiment, but such information was maintained in the second experiment. Here, in each sub-figure in Figures 6 and 8, we computed the area between the accuracy curve on samples generated by the random masking method and the accuracy curve on samples generated by the centrally-surrounding masking method. The area indicated the sensitivity to the structural destruction. We found that such area of high-order DNNs were much larger than the area of normally trained DNNs. This phenomenon indicated that normally trained DNN usually encoded local patterns, just like the bag-of-words representations, which were robust to the structural destruction. However, high-order DNN encoded more structural information.

Adversarial robustness. Ren et al. (2021) have demonstrated that adversarial attacks mainly affect high-order interactions. Therefore, we conducted experiments to train DNNs mainly encoding high-order interactions based on the proposed losses, in order to verify whether such DNNs were more vulnerable to adversarial attacks. To train DNNs mainly encoding high-order interactions, we set $\lambda_1 = 1, \lambda_2 = 1$. Specifically, the DNN was trained to encourage interactions of the $[0.6n, n]$-th orders by setting $r_1 = 0.6, r_2 = 1$ for $L^+(r_1, r_2)$ and simultaneously penalize interactions of the $[0, 0.5n]$-th orders by setting $r_1 = 0, r_2 = 0.5$ for $L^-(r_1, r_2)$ simultaneously. We used the aforementioned MLP-5 and MLP-8 networks in Section 3.1. Each MLP was trained on the census and commercial dataset, respectively. Figure 9 in the appendix shows distributions of interaction strength of these DNNs. Then, we compared the adversarial robustness between normally trained DNNs and the DNNs whose high-order interactions were boosted. We adopted the untargeted PGD attack (Madry et al., 2018) based on L_∞ norm. We set the attack strength $\epsilon = 0.6$ with 100 steps for the census dataset, and set $\epsilon = 0.2$ with 50 steps for the commercial dataset. The step size was uniformly set to 0.01 for all attacks. Please see more details in Appendix C. Table 1 shows that DNNs with boosted high-order interactions exhibited significantly lower adversarial accuracies than normally trained DNN, especially on the census dataset. These results verified that high-order interactions were vulnerable to adversarial attacks.

4 Conclusion

In this paper, we have discovered and theoretically proved the representation bottleneck of DNNs, from a new perspective of the complexity of interactions encoded in DNNs. We adopt the multi-order interaction, and we use the order to represent the complexity of interactions. It is a common phenomenon that a DNN usually encodes very simple interactions and very complex interactions, but rarely learns interactions of intermediate complexity. We have theoretically proved the underlying reason for the representation bottleneck. Furthermore, we propose two losses to learn DNNs which encode interactions of specific complexities. Experimental results have shown that it is not necessary for a DNN to encode or avoid encoding interactions of specific orders, in terms of classification performance. However, high-order interactions usually encode more structural information than low-order interactions, and are usually vulnerable to adversarial attacks.

5 REPRODUCIBILITY STATEMENT

This research discovered and theoretically explained the representation bottleneck phenomenon, based on the multi-order interaction. Appendix A shows the trustworthiness of the multi-order interaction, by introducing its five desirable properities and its connections with existing typical metrics in game theory. Appendix B and Section 3.2 provide proofs for all theoretical results in the paper. Section 3 and Appendix C have discussed all experimental details, including the computation of interaction strength and how to train DNNs by the proposed two losses, which ensure the reproducibility. Furthermore, we will release the code when the paper is accepted.

REFERENCES

Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40 (12):2897–2905, 2018.

Rana Ali Amjad and Bernhard C Geiger. Learning representations for neural network-based classification using the information bottleneck principle. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2225–2239, 2019.

Marco Ancona, Cengiz Oztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In *International Conference on Machine Learning*, pp. 272–281. PMLR, 2019.

Devansh Arpit, Stanislaw Jastrzkebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242. PMLR, 2017.

Dheeru Dua, Casey Graff, et al. Uci machine learning repository. 2017.

Stanislav Fort, Paweł Krzysztof Nowak, Stanislaw Jastrzebski, and Srini Narayanan. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 2019.

Michel Grabisch and Marc Roubens. An axiomatic approach to the concept of interaction among players in cooperative games. *International Journal of game theory*, 28(4):547–565, 1999.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

Joseph D Janizek, Pascal Sturmfels, and Su-In Lee. Explaining explanations: Axiomatic feature interactions for deep networks. *Journal of Machine Learning Research*, 22(104):1–54, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pp. 1097–1105, 2012.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Benjamin Lengerich, Eric P Xing, and Rich Caruana. On dropout, overfitting, and interaction effects in deep neural networks. *arXiv preprint arXiv:2007.00823*, 2020.

Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1754–1763, 2018.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pp. 4768–4777, 2017.

Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, 2014.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, 2017.

Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

William Peebles, John Peebles, Jun-Yan Zhu, Alexei Efros, and Antonio Torralba. The hessian penalty: A weak prior for unsupervised disentanglement. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pp. 581–597. Springer, 2020.

Jie Ren, Die Zhang, Yisen Wang, Lu Chen, Zhanpeng Zhou, Xu Cheng, Xin Wang, Yiting Chen, Jie Shi, and Quanshi Zhang. Game-theoretic understanding of adversarially learned features. *arXiv preprint arXiv:2103.07364*, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

LS Shapley. Notes on the n-person gameii: The value of an n-person game, the rand corporation, the rand corporation. *Research Memorandum*, 670, 1951.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2014.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1161–1170, 2019.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

Mukund Sundararajan, Kedar Dhamdhere, and Ashish Agarwal. The shapley taylor interaction index. In *International Conference on Machine Learning*, pp. 9259–9268. PMLR, 2020.

Michael Tsang, Dehua Cheng, and Yan Liu. Detecting statistical interactions from neural network weights. In *International Conference on Learning Representations*, 2017.

Michael Tsang, Hanpeng Liu, Sanjay Purushotham, Pavankumar Murali, and Yan Liu. Neural interaction transparency (nit): Disentangling learned interactions for improved interpretability. *Advances in Neural Information Processing Systems*, 31:5804–5813, 2018.

Michael Tsang, Dehua Cheng, Hanpeng Liu, Xue Feng, Eric Zhou, and Yan Liu. Feature interaction interpretability: A case for explaining ad-recommendation systems via neural interaction detection. In *International Conference on Learning Representations*, 2020.

Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018.

Zhiqin John Xu. Understanding training and generalization in deep learning by fourier analysis. *arXiv preprint arXiv:1808.04295*, 2018.

Hao Zhang, Sen Li, Yinchao Ma, Mingjie Li, Yichen Xie, and Quanshi Zhang. Interpreting and boosting dropout from a game-theoretic view. In *International Conference on Learning Representations*, 2020.

A THE MULTI-ORDER INTERACTION

Zhang et al. (2020) proposed the multi-order interaction between input variables i, j as follows:

$$I^{(m)}(i,j) = \mathbb{E}_{|S|\subseteq N\setminus\{i,j\},|S|=m}\Delta v(i,j,S)$$

where $\Delta v(i,j,S)=v(S\cup\{i,j\})-v(S\cup\{i\})-v(S\cup\{j\})+v(S)$. $I^{(m)}(i,j)$ denotes the interaction between variables $i, j \in N$ of the m-th order, which measures the average interaction utility between i, j under contexts of m variables. It has been proven that $I^{(m)}(i,j)$ satisfies the following five desirable properties.

•*Linear property.* If two independent games u and v are combined, $w(S) = u(S) + v(S)$, then the multi-order interaction of the combined game equals to the sum of multi-order interactions derived from u and v. *I.e.*, $I_w^{(m)}(i,j) = I_u^{(m)}(i,j) + I_v^{(m)}(i,j)$.

•*Nullity property.* A dummy variable $i \in N$ satisfies $\forall S \subseteq N \setminus \{i\}, v(S \cup \{i\}) = v(S) + v(\{i\})$. Then, variable i has no interactions with other variables, *i.e.*, $\forall m, \forall j \in N \setminus \{i\}, I^{(m)}(i,j) = 0$.

• *Commutativity property.* $\forall i, j \in N, I^{(m)}(i,j) = I^{(m)}(j,i)$.

• *Symmetry property.* Assume two variables i, j are equivalent in the sense that i, j have same cooperations with other variables, $\forall S \subseteq N \setminus \{i,j\}, v(S\cup\{i\}) = v(S\cup\{j\})$. Then, for any variable $k \in N$, $I^{(m)}(i,k) = I^{(m)}(j,k)$.

• ***Efficiency property.*** The network output of a DNN can be decomposed into the sum of interactions of different orders between different pairs of variables.

$$v(N) - v(\emptyset) = \sum\nolimits_{i\in N} \mu_i + \sum\nolimits_{i,j\in N, i\neq j} \sum\nolimits_{m=0}^{n-2} w^{(m)} I^{(m)}(i,j).$$

where $\mu_i = v(\{i\}) - v(\emptyset)$ represents the independent effect of variable i, and $w^{(m)} = \frac{n-1-m}{n(n-1)}$.

Connection with Shapley value and Shapley interaction index.

Shapley value. Shapley (1951) proposed the Shapley value to measure the numerical importance of each player to the total reward in a cooperative game. The Shapley value has been widely used to explain the decision of DNNs in recent years (Lundberg & Lee, 2017; Ancona et al., 2019). For a given DNN and an input sample with a set of input variables $N = \{1, \dots, n\}$, we use $2^N = \{S|S \subseteq N\}$ to denote all possible variable subsets of N. Then, the DNN v can be considered as $v : 2^N \to \mathbb{R}$ that calculates the output $v(S)$ of each specific subset $S \subseteq N$. Each input variable i is regarded as a player, and the network output $v(N)$ of all input variables can be considered as the total reward of the game. The Shapley value aims to fairly distribute the network output to each individual variables as follows:

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

where $v(S)$ denotes the network output when we keep variables in S unchanged while mask variables in $N \setminus S$ by following the setting in Ancona et al. (2019). It has been proven that the Shapely value is an unique method to fairly allocate overall reward to each player that satisfies *linearity*, *nullity*, *symmetry*, and *efficiency* properties.

Connections between the Shapley interaction index and the Shapely value. Input variables of a DNN usually interact with each other, instead of working individually. Based on the Shapley value, Grabisch & Roubens (1999) further proposed the Shapley interaction index to measure the interaction utility between input variables. The Shapley interaction index is the only axiomatic extension of the Shapley value, which satisfies *linearity*, *nullity*, *symmetry*, and *recursive* properties. For two variables $i, j \in N$, the Shapley interaction index $I(i,j)$ can be considered as the change of the numerical importance of variable i by the presence or absence of variable j.

$$I(i,j) = \tilde{\phi}(i)_{j \text{ always present}} - \tilde{\phi}(i)_{j \text{ always absent}},$$

where $\tilde{\phi}(i)_{j \text{ always present}}$ denotes the Shapley value of the variable i computed under the specific condition that variable j is always present. $\tilde{\phi}(i)_{j \text{ always absent}}$ is computed under the specific condition that j is always absent.

Connections between the multi-order interaction and the Shapley interaction index. Based on the Shapley interaction index, Zhang et al. (2020) further defined the order of interaction, which represents the contextual complexity of interactions. It has been proven that the above Shapley interaction index $I(i,j)$ between variables i,j can be decomposed into multi-order interactions as follows:

$$I(i,j) = \frac{1}{n-1}\sum\nolimits_{m=0}^{n-2} I^{(m)}(i,j).$$

B PROOF OF THEOREMS

B.1 PROOF OF THEOREM 1

Motivation of Theorem 1. Let W denote network parameters of the DNN. Let L and η denote the loss function and learning rate of training, respectively. According to the efficiency property of $I^{(m)}(i,j)$ in Eq. (2), the network output $v(N)$ of a DNN can be decomposed into the sum of multi-order interactions $I^{(m)}(i,j)$. Therefore, the change of weights ΔW can be computed by:

$$\Delta W = -\eta \frac{\partial L}{\partial v(N)}\frac{\partial v(N)}{\partial W} = \Delta W_U + \sum\nolimits_{m=0}^{n-2}\sum\nolimits_{i,j\in N, i\neq j} \Delta W^{(m)}(i,j)$$

where $U = v(\emptyset) + \sum_{i\in N}\mu_i$. Specifically,

$$\Delta W_U \overset{\text{def}}{=} -\eta \frac{\partial L}{\partial v(N)}\frac{\partial v(N)}{\partial U}\frac{\partial U}{\partial W}, \quad \Delta W^{(m)}(i,j) \overset{\text{def}}{=} R^{(m)}\frac{\partial I^{(m)}(i,j)}{\partial W}.$$

where $R^{(m)} = -\eta\frac{\partial L}{\partial v(N)}\frac{\partial v(N)}{\partial I^{(m)}(i,j)}$. Based on the analysis in Section 3.2, $\Delta W^{(m)}(i,j) = R^{(m)}\frac{\partial I^{(m)}(i,j)}{\partial W}$ represents the strength of learning the average interaction between variables i and j under contexts of m variables. Therefore, in Theorem 1, we aim to prove that the variance (strength) of learning the m-order interaction is $(\eta\frac{\partial L}{\partial v(N)}w^{(m)})^2\sigma^2/\binom{n-2}{m}$.

Proof skeleton in Theorem 1. Actually, the essence for the bottleneck is that the strength (*i.e.*, the gradient) of learning middle-order interactions $\Delta W^{(m)}(i,j)$ is much smaller than low-order and high-order interactions. Such a small gradient is caused by the following reasons. First, the updating gradient of $I^{(m)}(i,j)$ is a sample average of the gradient of $\Delta v(i,j,S)$ over contexts S. Second, the number of possible contexts S is given by $\binom{n-2}{m}$, and the middle-order interactions corresponds to massive possible contexts. Third, combining the first two points, we derive that the gradient of learning middle-order interactions is significantly lower than the gradient of learning low-order interactions and the gradient of learning high-order interactions. Therefore, DNNs usually fail to encode middle-order interactions.

Proof of Theorem 1. The m-order interaction $I^{(m)}(i,j)$ between variables i,j is defined as,

$$\begin{aligned} I^{(m)}(i,j) &= \mathbb{E}_{S\subseteq N\setminus\{i,j\},|S|=m}\Delta v(i,j,S) \\ &= \frac{1}{\binom{n-2}{m}}\sum\nolimits_{\substack{S\subseteq N\setminus\{i,j\}\\|S|=m}} \Delta v(i,j,S). \end{aligned} \tag{10}$$

Assume $\mathbb{E}_{i,j,S}[\frac{\partial\Delta v(i,j,S)}{\partial W}] = 0$. Let σ^2 denote the variance of the gradient $\frac{\partial\Delta v(i,j,S)}{\partial W}$. Based on Eq. (10), we have,

$$\begin{aligned} \frac{\partial I^{(m)}(i,j)}{\partial W} &= \sum\nolimits_{\substack{S\subseteq N\setminus\{i,j\}\\|S|=m}} \frac{\partial I^{(m)}(i,j)}{\partial\Delta v(i,j,S)}\frac{\partial\Delta v(i,j,S)}{\partial W} \\ &= \frac{1}{\binom{n-2}{m}}\sum\nolimits_{\substack{S\subseteq N\setminus\{i,j\}\\|S|=m}} \frac{\partial\Delta v(i,j,S)}{\partial W}. \end{aligned}$$

Because the gradients $\frac{\partial\Delta v(i,j,S)}{\partial W}$ on different contexts are independent with each other, we have $\mathbb{E}_{i,j}[\frac{\partial I^{(m)}(i,j)}{\partial W}] = 0$, and the variance of $\frac{\partial I^{(m)}(i,j)}{\partial W}$ is $\sigma^2/\binom{n-2}{m}$.

Furthermore, because $\Delta W^{(m)}(i,j) = -\eta\frac{\partial L}{\partial v(N)}w^{(m)}\frac{\partial I^{(m)}(i,j)}{\partial W}$, then $\mathbb{E}_{i,j}[\Delta W^{(m)}(i,j)] = 0$ and the variance of $\Delta W^{(m)}(i,j)$ is $(\eta\frac{\partial L}{\partial v(N)}w^{(m)})^2\sigma^2/\binom{n-2}{m}$, where $w^{(m)} = \frac{n-m-1}{n(n-1)}$.

Therefore, the conclusion of Theorem 1 holds.

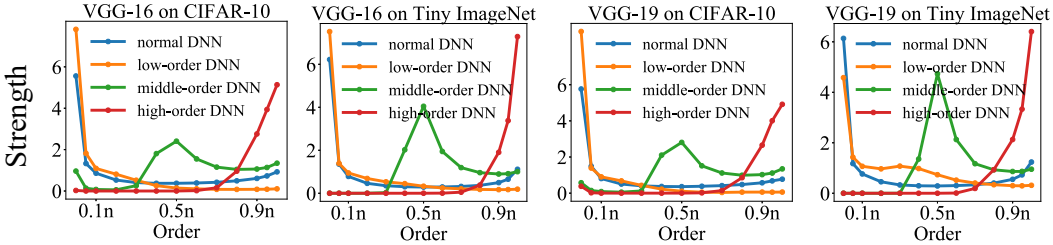


Figure 7: Distributions of the interaction strength $J^{(m)}$ of four types of DNNs

B.2 Proof of Theorem 2

Let S_1, S_2 denote the two variable subsets randomly sampled from all input variables N, where $|S_1| = r_1 n, |S_2| = r_2 n$, and $0 \leq r_1 < r_2 \leq 1$. According to the efficiency property of the multi-order interaction, we have,

$$\begin{aligned}
\mathbb{E}_{S_1}[v(S_1)] &= v(\emptyset) + \mathbb{E}_{S_1}[\sum_{i \in S_1} \mu_i] + \mathbb{E}_{S_1}[\sum_{i,j \in S_1, i \neq j} [\sum_{m=0}^{r_1 n - 2} \frac{r_1 n - 1 - m}{r_1 n (r_1 n - 1)} I_{S_1}^{(m)}(i,j)]] \\
&= v(\emptyset) + r_1 n \mathbb{E}_i(\mu_i) + \mathbb{E}_{S_1}[\sum_{m=0}^{r_1 n - 2} \frac{r_1 n - 1 - m}{r_1 n (r_1 n - 1)} \sum_{i,j \in S_1, i \neq j} I_{S_1}^{(m)}(i,j)] \\
&= v(\emptyset) + r_1 n \mathbb{E}_i(\mu_i) + \sum_{m=0}^{r_1 n - 2} (r_1 n - 1 - m) \mathbb{E}_{S_1}[\mathbb{E}_{i,j}[(I_{S_1}^{(m)}(i,j))]]
\end{aligned}$$

where $I_{S_1}^{(m)}(i,j) \overset{\text{def}}{=} \mathbb{E}_{S' \subseteq S_1 \setminus \{i,j\}, |S'| = m}(\Delta v(i,j,S'))$. Similarly,

$$\begin{aligned}
\mathbb{E}_{S_2}[v(S_2)] &= v(\emptyset) + \mathbb{E}_{S_2}[\sum_{i \in S_2} \mu_i] + \mathbb{E}_{S_2}[\sum_{i,j \in S_2, i \neq j} [\sum_{m=0}^{r_2 n - 2} \frac{r_2 n - 1 - m}{r_2 n (r_2 n - 1)} I_{S_2}^{(m)}(i,j)]] \\
&= v(\emptyset) + r_2 n \mathbb{E}_i(\mu_i) + \sum_{m=0}^{r_2 n - 2} (r_2 n - 1 - m) \mathbb{E}_{S_2}[\mathbb{E}_{i,j}[(I_{S_2}^{(m)}(i,j))]]
\end{aligned}$$

where $I_{S_2}^{(m)}(i,j) \overset{\text{def}}{=} \mathbb{E}_{S' \subseteq S_2 \setminus \{i,j\}, |S'| = m}(\Delta v(i,j,S'))$. Note that the contexts when computing $I_{S_1}^{(m)}(i,j)$ and $I_{S_2}^{(m)}(i,j)$ are different. When $m < r_1 n < r_2 n$, we obtain that $\mathbb{E}_{S_1}[\mathbb{E}_{i,j}(I_{S_1}^{(m)}(i,j))] = \mathbb{E}_{S_2}[\mathbb{E}_{i,j}(I_{S_2}^{(m)}(i,j))] = \mathbb{E}_{i,j}(I^{(m)}(i,j))$. Therefore, $\Delta u(r_1, r_2)$ can be rewritten as follows:

$$\begin{aligned}
\Delta u(r_1, r_2) &= \mathbb{E}_{(S_1, S_2)}[v(S_2) - \frac{r_2}{r_1} v(S_1)] \\
&= \mathbb{E}_{S_2}[v(S_2)] - \frac{r_2}{r_1} \mathbb{E}_{S_1}[v(S_1)] \\
&= (1 - \frac{r_2}{r_1}) v(\emptyset) + \sum_{m=0}^{n-2} \sum_{i,j \in N, i \neq j} \tilde{w}^{(m)} I^{(m)}(i,j)
\end{aligned}$$

$$\text{where} \quad \tilde{w}^{(m)} = \begin{cases} (\frac{r_2}{r_1} - 1)(m+1)/[n(n-1)], & m \leq r_1 n - 2 \\ (r_2 n - m - 1)/[n(n-1)], & r_1 n - 2 < m \leq r_2 n - 2 \\ 0, & r_2 n - 2 < m \leq n - 2 \end{cases}$$

Then, the conclusion holds.

C EXPERIMENTAL DETAILS AND RESULTS

C.1 IMPLEMENTATION DETAILS

The sampling strategy when computing $J^{(m)}$. The interaction strength $J^{(m)}$ is defined as,

$$J^{(m)} = \frac{\mathbb{E}_{x\in\Omega}[\mathbb{E}_{i,j}[|I^{(m)}(i,j|x)|]]}{\mathbb{E}_{m'}[\mathbb{E}_{x\in\Omega}[\mathbb{E}_{i,j}[|I^{(m')}(i,j|x)|]]]}, \text{ where } I^{(m)}(i,j|x) = \mathbb{E}_{\substack{S\subseteq N\setminus\{i,j\}\\|S|=m}}[\Delta v(i,j,S)].$$

To precisely compute $J^{(m)}$, we need to average all possible contexts $S \subseteq N$, all pairs of variables $i, j \in N$, and all samples $x \in \Omega$, which is usually computationally infeasible. Therefore, we computed $J^{(m)}$ approximately by a sampling strategy (Zhang et al., 2020).

On each image dataset, we first sampled 50 correctly classified images in total. On the ImageNet and Tiny-ImageNet dataset, these images were sampled from different 50 classes. On the CIFAR-10 dataset, we sampled 5 images from each class. Then, for each image, we sampled 200 pairs of patches $i, j \in N$. Since DNNs usually encodes stronger interactions between neighbor patches, we restricted that patch i is located at the neighbor patch j with a radius of two patches. Next, for each pair of patches i, j and each order m, we randomly sampled 100 contexts S from all possible contexts, where $|S| = m$. To plot the curve of interaction strength, we computed 13 orders for $J^{(m)}$, where $m = \frac{r}{10}n, 0 \leq r \leq n$ and $m = 0.05n, 0.95n$. On each tabular dataset, since there are only a few input variables ($n = 12$ on census and $n = 10$ on commercial), we sampled all 100 instances and all pairs of variables for computation. For each pair of patches $i, j \in N$ and each order m, we randomly sampled 100 contexts S from all possible contexts.

Considering the randomness in the sampling process, we also conducted an experiment to evaluate the stability of the estimated interaction strength $J^{(m)}$ via the sampling-based approximation method. Specifically, given an input image x and a sampling number q, we measured the instability of $J^{(m)}(x)$, when we repeatedly computed the interaction for q times. The instability *w.r.t.* $J^{(m)}(x)$ was computed as $\frac{\mathbb{E}_{u,v;u\neq v}|J_u^{(m)}(x)-J_v^{(m)}(x)|}{\mathbb{E}_w|J_w^{(m)}(x)|}$, where $J_u^{(m)}(x), J_v^{(m)}(x)$ denotes the estimated $J^{(m)}(x)$ at the u-th sampling time and the v-th sampling time, respectively. Then, the instability value was computed by averaging over 50 images and all sampled orders, *i.e.*, *instability* $= \mathbb{E}_{x\in\Omega}\mathbb{E}_m[\frac{\mathbb{E}_{u,v;u\neq v}|J_u^{(m)}(x)-J_v^{(m)}(x)|}{\mathbb{E}_w|J_w^{(m)}(x)|}]$. We used the trained AlexNet on the Tiny-ImageNet dataset and VGG-16 on the CIFAR-10 dataset to compute the instability.

We aim to evaluate how the estimated $J^{(m)}$ value will be affected by the randomness in the sampling process. During the sampling process to compute $J^{(m)}(x)$, we need to sample over multiple pairs of variables (i, j) and multiple contexts S. Therefore, we first fixed 100 pairs of (i, j), and evaluated the instability of $J^{(m)}(x)$ *w.r.t* the sampling of the contexts S. Figure 9(b) shows that on the two datasets, when the sampling number of S increased, the instability decreased. Furthermore, when the sampling number of S was greater than 100 (*i.e.,* our setting), the instability value was less than 0.05, which indicated a stable approximation. Secondly, we evaluated the instability *w.r.t* the sampling processes of (i, j), where the sampling number of S was set to 100 as mentioned above. As Figure 9(b) shows, when the sampling number of (i, j) pairs increased, the instabilities decreased. When the sampling number of (i, j) pairs was greater than 200 (*i.e.,* our setting), the instability was less than 0.1, which indicated a stable approximation of $J^{(m)}(x)$. These results demonstrated that our sampling strategy can well approximate the interaction strength $J^{(m)}$.

Implementation details of adversarial attacks. Here, we introduce how to measure the adversarial robustness in Section 3.4. We adopted the untargeted PGD attack (Madry et al., 2018) with the L_∞ constraint $\|\Delta x\|_\infty \leq \epsilon$ to generate adversarial examples. For the census dataset, we set $\epsilon = 0.6$ and the attack was conducted with 100 steps. For the commercial dataset, we set $\epsilon = 0.2$ and the attack was conducted with 50 steps. The step size was set to 0.01 for all attacks.

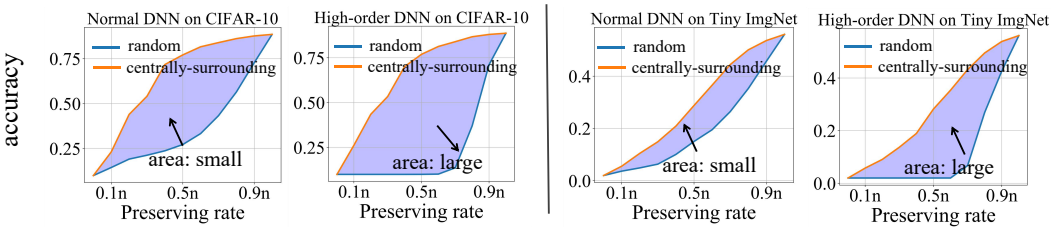


Figure 8: Classification accuracies using AlexNet architecture on images with different numbers of patches being masked.

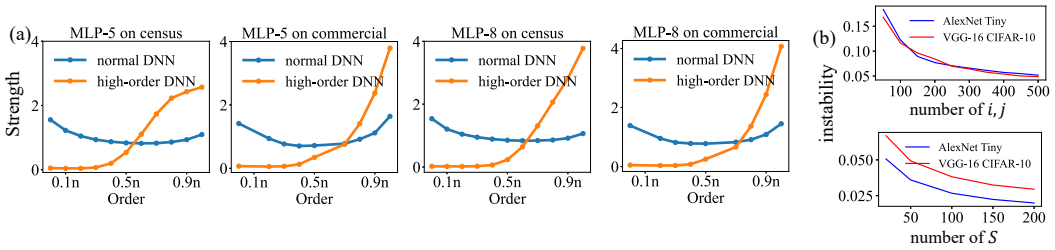


Figure 9: (a) Distributions of the interaction strength $J^{(m)}$ of normal DNNs and high-order DNNs, where high-order DNNs are trained by encouraging high-order interactions and penalizing low-order interactions simultaneously. (b) The instability of $J^{(m)}(x)$ *w.r.t.* the sampling number of pairs of variables (i, j) and the sampling number of contexts S.

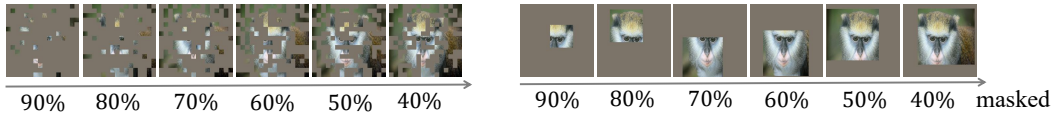


Figure 10: Images generated by random masking (left) and randomly-surrounding masking (right).

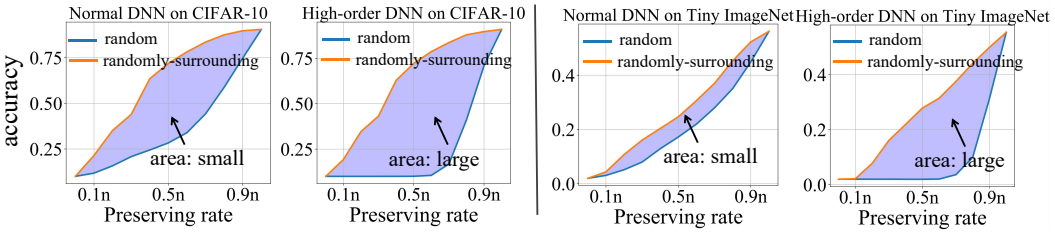


Figure 11: Classification accuracies using VGG-16 architecture on images with different numbers of patches being masked (by random masking and randomly-surrounding masking).

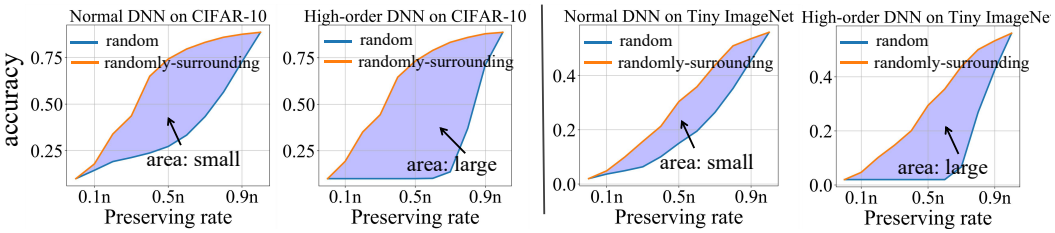


Figure 12: Classification accuracies using AlexNet architecture on images with different numbers of patches being masked (by random masking and randomly-surrounding masking).

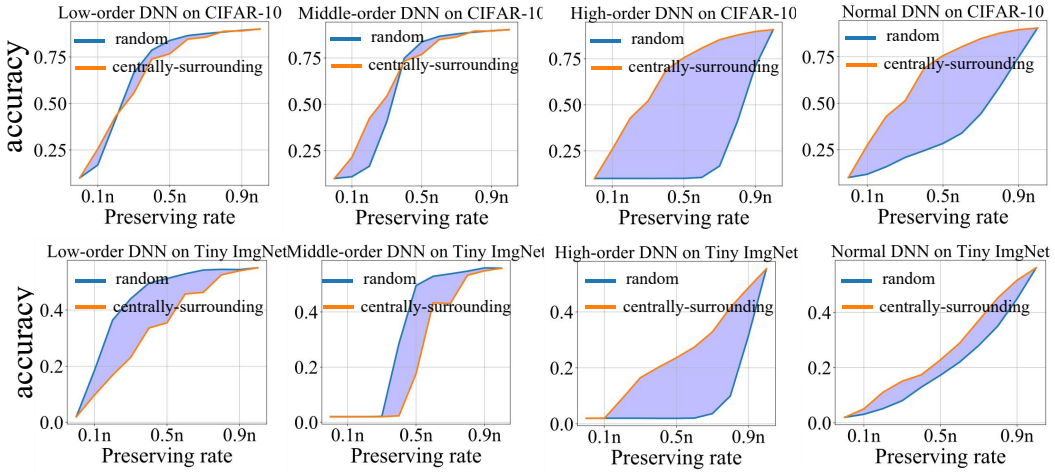


Figure 13: Classification accuracies of low-order DNNs, middle-order DNNs, high-order DNNs, and normally trained DNNs (using VGG16 architecture) on images with different numbers of patches being masked. Here, we used the random masking method and the centrally-surrounding masking method.

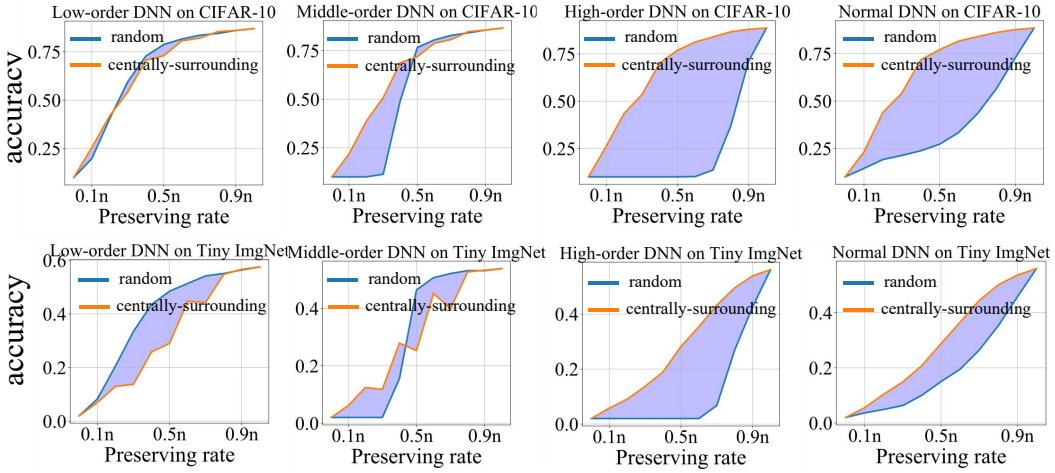


Figure 14: Classification accuracies of low-order DNNs, middle-order DNNs, high-order DNNs, and normally trained DNNs (using AlexNet architecture) on images with different numbers of patches being masked. Here, we used the random masking method and the centrally-surrounding masking method.

C.2 More experimental results.

In this subsection, we provide more experimental results besides the results in the main paper. Due to the computational cost, when we trained DNNs on the Tiny ImageNet dataset, we selected 50 object classes from 200 classes at equal intervals for training and testing. In the computation of $\Delta u(r_1, r_2)$, we set $\Delta u(r_1, r_2) = \mathbb{E}_{S_2}[v(S_2)] - v(\emptyset)$ when $S_1 = \emptyset$. Figure 7 shows the distributions of $J^{(m)}$ of four types of DNNs (*i.e.*, normally trained DNNs, low-order DNNs, middle-order DNNs, and high-order DNNs) mentioned in Section 3.4. The distributions of $J^{(m)}$ in Figure 7 were consistent with that of AlexNet model (see Figure 4(c)), which further demonstrates that these DNNs could successfully learn interactions as expected. Figure 8 shows classification accuracies using AlexNet network on images with different numbers of patches being masked. These results further verifies that high-order DNNs were more sensitive to the destruction of structural information. Figure 9 shows the $J^{(m)}$ distributions of normal DNNs and high-order DNNs on two tabular datasets.

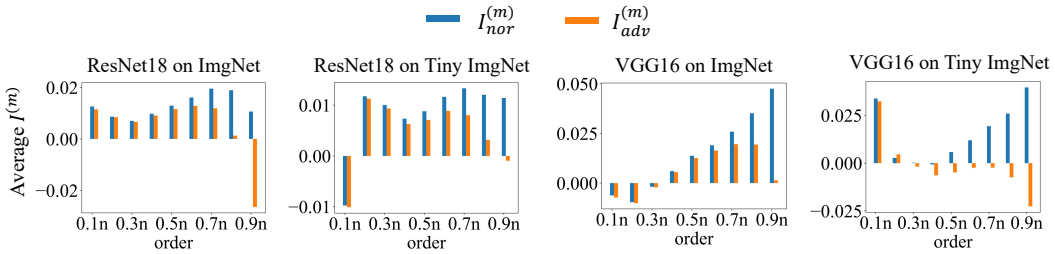
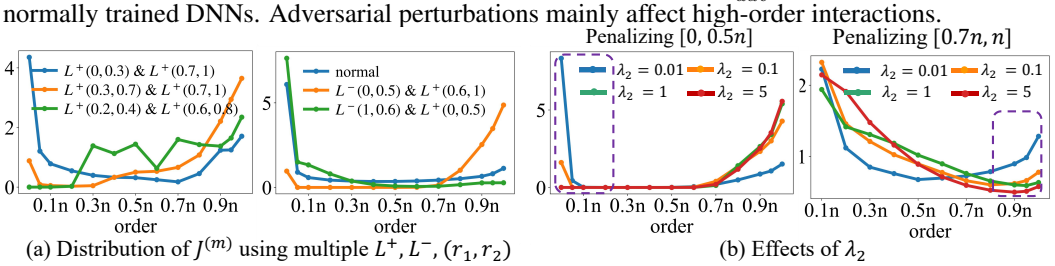


Figure 15: The multi-order interaction $I_{nor}^{(m)}$ on normal samples and $I_{adv}^{(m)}$ on adversarial samples for normally trained DNNs. Adversarial perturbations mainly affect high-order interactions.



(a) Distribution of $J^{(m)}$ using multiple L^+, L^-, (r_1, r_2) (b) Effects of λ_2

Figure 16: (a) Distributions of the interaction strength $J^{(m)}$ of normally trained DNN and DNNs trained by using multiple L^+, L^-, and (r_1, r_2) pairs. (b) Distributions of the interaction strength $J^{(m)}$ of DNNs trained by using different λ_2s.

C.2.1 More experiments on structural representation

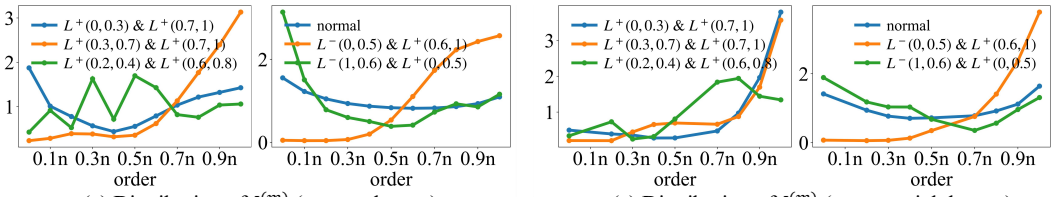
More masking methods to verify structural representations. Besides, we conducted experiments to investigate a DNN's capacity of encoding structural information by using a new masking method. Specifically, we masked m patches in each image and only preserved a single large region of $n - m$ patches, as shown in Figure 10. The position of the preserved region was randomly determined, instead of being fixed in the center of the image previously. We called the new masking method as *randomly-surrounding masking method*. By doing so, the structural information within the large region was maintained.

Figures 11 and 12 show experimental results. In each sub-figure, the x-axis represents the ratio of preserved patches, and the y-axis represents the classification accuracy when only the masked images were fed as input. In addition, the blue curve denotes the accuracy curve on the samples generated by the random masking method, and the orange curve shows the accuracy on the samples generated by the randomly-surrounding masking method. The area between the blue curve and the orange curve measures the sensitivity of the DNN to the structural destruction.

Based on the accuracy on masked samples in Figures 11 and 12, we found that the aforementioned area of the high-order DNN were much larger than the area of the normally trained DNN. This experimental results verify that high-order DNNs encoded more structural information than normally trained DNNs. Such results were also consistent with results obtained in our previous experiments in Figures 6 and 8.

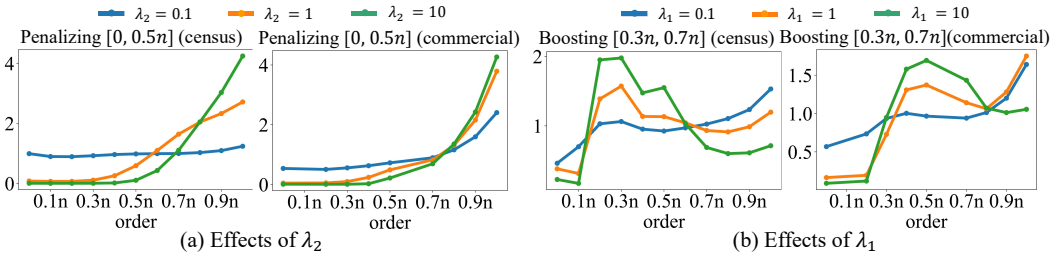
Comparisons of structural representations between four types of DNNs. We further compared the capacity of encoding structural representations among four types of DNNs (*i.e.*, normally trained DNNs, low-order DNNs, middle-order DNNs, and high-order DNNs). To this end, we followed the experimental setting in Section 3.4. We trained low-order DNNs by penalizing interactions of the $[0.7n, n]$-th orders based on $L^-(r_1, r_2)$ loss, and trained middle-order DNNs by boosting interactions of the $[0.3n, 0.7n]$-th orders based on $L^+(r_1, r_2)$ loss.

Figure 13 compares classification accuracies on images generated by the random masking method (which were considered as bag-of-words features without structures) and images generated by the centrally-surrounding masking method (with structural patterns). The large gap between the two classification accuracies indicates that the learned DNN encoded rich structural information for inference. *I.e.*, the larger area between the accuracy curve on images generated by the random masking method and the accuracy curve on images generated by the centrally-surrounding masking method means the richer structural information in the DNN. In this way, Figure 13 compares the capacity



(a) Distribution of $J^{(m)}$ (census dataset) (a) Distribution of $J^{(m)}$ (commercial dataset)

Figure 17: Distributions of the interaction strength $J^{(m)}$ of normally trained DNN and DNNs trained with L^+ and/or L^- *w.r.t.* different pairs of (r_1, r_2).



(a) Effects of λ_2 (b) Effects of λ_1

Figure 18: Distributions of the interaction strength $J^{(m)}$ of DNNs trained by using different pairs (λ_1, λ_2) on the census dataset and the commercial dataset.

of encoding structural information between four types of VGG-16 networks. Similarly, Figure 14 compares the capacity of encoding structural information between four types of AlexNets.

To this end, we obtained three conclusions.

(i) Low-order DNNs trained on the CIFAR-10 dataset encoded little structural information, because such DNNs did not exhibit much difference in classification accuracy on the above two types of samples. Furthermore, for low-order DNNs trained on the Tiny ImageNet dataset, the accuracies on images generated by the random masking method were even higher than the accuracies on images generated by the centrally-surrounding masking method. This result further verifies that low-order DNNs preferred bag-of-words representations, but failed to encode structural information.

(ii) High-order DNNs showed the highest accuracy difference between the above two types of samples, which indicates that high-order DNNs encoded most structural information.

(iii) The capacity of Middle-order DNNs to encode structural information was somewhere in between low-order DNNs and high-order DNNs. It indicates that middle-order DNNs encoded more structural information than low-order DNNs, but middle-order DNNs encoded less structural information than high-order DNNs.

Experimental results demonstrated that it was the high-order interaction that was mainly responsible for encoding structural information.

C.2.2 More experiments on adversarial robustness

Comparisons on adversarial robustness between four types of DNNs. We further compared the adversarial robustness between four types of DNNs, including low-order DNNs, middle-order DNNs, high-order DNNs, and normally trained DNNs. To this end, we followed the experimental setting in Section 3.4. Here, the low-order DNNs were trained by encouraging interactions of the $[0,\ 0.5n]$-th orders based on $L^+(r_1, r_2)$ and simultaneously penalizing interactions of the $[0.6n,\ n]$-th orders based on $L^-(r_1, r_2)$. We set $\lambda_1 = 10, \lambda_2 = 10$. The middle-order DNNs were trained to encourage interactions of the $[0.3n,\ 0.7n]$-th orders based on $L^+(r_1, r_2)$. We set $\lambda_1 = 10, \lambda_2 = 0$. We used the MLP-5 and MLP-8 networks in Section 3.1. Each MLP was trained on the census dataset and commercial dataset, respectively. Adversarial attacks were conducted by following experimental settings in Section 3.4.

Table 2 reports the adversarial robustness of four types of DNNs. We obtained following results. (i) Low-order DNNs usually exhibited the highest adversarial robustness, which outperformed normally trained DNNs. (ii) High-order DNNs exhibited the lowest robustness, which were significantly lower

Table 2: Comparison of the adversarial accuracy between normally trained DNNs, low-order DNNs, middle-order DNNs, and high-order DNNs on the census dataset and the commercial dataset.

Model	Normal training	penalize high-order & boost low-order	boost middle-order	Penalize low-order & boost high-order
MLP-5 on census	38.22	42.40	15.93	7.31
MLP-8 on census	39.33	44.65	18.10	2.02
MLP-5 on commercial	27.01	29.76	23.70	22.00
MLP-8 on commercial	25.92	28.86	22.55	20.58

Table 3: Comparison of the running time between normally training DNNs by 200 epochs and training DNNs by 200 epochs with the proposed losses.

	CIFAR-10		Tiny-ImageNet	
Model	AlexNet	VGG16	AlexNet	VGG16
Normally training DNNs	0.46 h	0.83 h	0.89 h	9.25 h
Training using $L^+(r_1, r_2)$	1.36 h	2.35 h	1.87 h	27.28 h
Training using $L^-(r_1, r_2)$	1.27 h	2.12 h	1.82 h	26.68 h

than normally trained DNNs. (iii) The robustness of middle-order DNNs was somewhere in between low-order and high-order DNNs.

The above results showed a strong connection between adversarial robustness and the orders of the encoded interactions.

More explorations on Adversarial robustness. We further explored the relationship between adversarial robustness and the order of interactions. We conducted experiments on images in the ImageNet dataset and the Tiny ImageNet dataset, in order to explore interactions of which orders were vulnerable to adversarial attacks.

Specifically, let x denote a normal sample, and let $x' = x + \epsilon$ denote an adversarial sample. Then, according to the efficiency property of $I^{(m)}(i, j)$, the significant output change caused by adversarial perturbations, $v(N|x) - v(N|x')$, can be represented as the sum of differences in multi-order interactions as follows.

$$v(N|x) - v(N|x') = \text{ignorable bias} + \textstyle\sum_m w^{(m)} \sum_{i,j} \Delta I^{(m)}(i, j),$$

where $\Delta I^{(m)}(i, j) = I^{(m)}(i, j|x) - I^{(m)}(i, j|x')$ is the compositional adversarial effect on the interaction utility $I^{(m)}(i, j|x)$ caused by adversarial perturbations. In this way, the overall attacking utility (the output change) can be decomposed into adversarial effects on massive compositional interactions $I^{(m)}(i, j|x)$.

Therefore, Figure 15 uses the change of multi-order interactions $\Delta I^{(m)}(i, j)$ as specific reasons for adversarial vulnerability. This figure shows the difference between interactions of normal samples and interactions of adversarial samples. Adversarial perturbations mainly affected high-order interactions, and rarely affected low-order interactions. This phenomenon has been showed in some previous studies. The above experiments further verified the connection between adversarial robustness and the order of the encoded interactions.

Although experimental results showed a strong connection between adversarial robustness and the order of the encoded interactions, the order of interactions does not fully determine the adversarial robustness. Previous studies have shown that interactions encoded in adversarially-trained DNNs were usually more robust to attacks.

C.2.3 More experiments on different hyper-parameters

Training DNNs with both L^+ and L^- *w.r.t.* different pairs of (r_1, r_2). We investigated the performances of DNNs when we simultaneously used multiple L^+ and L^- with different pairs of (r_1, r_2) in the loss function. We used the following five sets of experimental settings, including (i) $L^+(0, 0.3)$ and $L^+(0.7, 1.0)$; (ii) $L^+(0.3, 0.7)$ and $L^+(0.7, 1.0)$; (iii) $L^+(0.2, 0.4)$ and

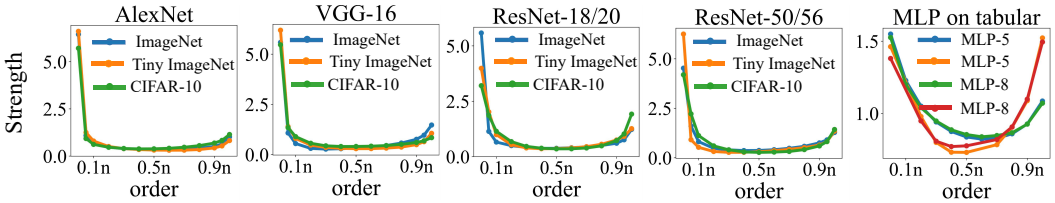


Figure 19: The distributions of interaction strength $\bar{J}^{(m)}$ of different DNNs using new metric. These DNNs were trained on various image datasets and tabular datasets.

$L^+(0.6, 0.8)$; (iv) $L^-(0, 0.5)$ and $L^+(0.6, 1)$; (v) $L^+(0, 0.5)$ and $L^-(0.6, 1)$ to learn DNNs. DNNs were trained on the CIFAR-10 dataset, the census dataset, and the commercial dataset, respectively.

Figure 16(a) shows that the trained DNNs successfully learned interactions as expected. For example, the DNN trained with $L^+(0, 0.3)$ & $L^+(0.7, 1.0)$ simultaneously boosted both interactions of the $[0, 0.3n]$-orders and interactions of the $[0.7n, n]$-orders. These results further verified the effectiveness of the proposed losses.

Furthermore, DNNs trained with L^+ and/or L^- *w.r.t.* different pairs of (r_1, r_2) further verified the conclusion obtained in Table 1 (left). *I.e.*, although the interactions of different orders performed differently in both adversarial robustness (Table 1 (right)) and the capacity of encoding structural information (Figures 6, 8, 13, and 14), interactions of different orders had similar classification accuracies (the difference was within $\pm 1\%$ accuracy). It showed that although interactions of different orders had their distinctive properties, such difference did not necessarily affect their classification accuracies on normal images.

Effects of hyper-parameters. We further investigated the effects of different hyper-parameters.

First, the effects of a single (r_1, r_2) pair had been investigated. Figure 4 (b) and Figure 4 (c) showed that using the $L^+(r_1, r_2)$/$L^-(r_1, r_2)$ loss boosted/penalized interactions of the $[r_1 n, r_2 n]$-th orders.

Second, in spite of that, we conducted new experiments to further investigate the effects of multiple (r_1, r_2) pairs. We used the following five sets of experimental settings, including (i) $L^+(0, 0.3)$ and $L^+(0.7, 1.0)$; (ii) $L^+(0.3, 0.7)$ and $L^+(0.7, 1.0)$; (iii) $L^+(0.2, 0.4)$ and $L^+(0.6, 0.8)$; (iv) $L^-(0, 0.5)$ and $L^+(0.6, 1)$; (v) $L^+(0, 0.5)$ and $L^-(0.6, 1)$. We used these parameters to train DNNs on the CIFAR-10 dataset. Figure 16(a) shows that $L^+(r_1, r_2)$ and $L^+(r'_1, r'_2)$ loss simultaneously boosted interactions of the $[r_1 n, r_2 n]$-th orders and the $[r'_1 n, r'_2 n]$-th orders. For example, by using $L^+(0, 0.3)$ and $L^+(0.7, 1.0)$, the trained DNN simultaneously boosted both interactions of the $[0, 0.3n]$-th orders and interactions of the $[0.7n, n]$-th orders.

In addition, we conducted experiments on two tabular datasets, *i.e.*, the census dataset and the commercial dataset. We extended above five different experimental settings (*w.r.t.* r_1 and r_2) for image data to the training of DNNs in tabular data. Thus, we trained five MLP-5 networks on each tabular dataset following each experimental setting. The architecture of the MLP-5 has been introduced in Section 3.1. Figure 17 shows that we can use $L^+(r_1, r_2)$ and $L^+(r'_1, r'_2)$ loss to simultaneously boost interactions of the $[r_1 n, r_2 n]$-th orders and interactions of the $[r'_1 n, r'_2 n]$-th orders.

Third, we conducted new experiments to investigate the effects of the parameter λ. Specifically, we fixed $\lambda_1 = 0$, and adjusted $\lambda_2 = 0.01, 0.1, 1, 5$ to study the effects of λ_2. We trained DNNs on the Tiny ImageNet dataset. First, we trained high-order DNNs by penalizing interactions of the $[0, 0.5n]$-th orders. Figure 16(b) shows that the interaction strength of the $[0, 0.5n]$-th orders decreased as λ_2 increased. The results indicated that the larger λ_2 value penalized interactions of the $[0, 0.5n]$-th orders more significantly. Second, we trained low-order DNNs by penalizing interactions of the $[0.7n, n]$-th orders. Similarly, we found that the strength of these high-order interactions also decreased as λ_2 increased. This further indicates that the larger λ_2 value penalized the high-order interactions more significantly. We also conducted experiments on the census dataset and the commercial dataset. Specifically, we fixed $\lambda_1 = 0$, and adjusted $\lambda_2 = 0.1, 1, 10$. We trained high-order DNNs by penalizing interactions of the $[0, 0.5n]$-th orders. Here, we used the MLP-5 networks mentioned in Section 3.1. Figure 18(a) also shows the same pattern that the larger λ_2 value penalized interactions of the $[0, 0.5n]$-th orders more significantly. These results showed that the parameter λ_2 controlled the strength of penalization.

Table 4: Expectations and standard deviations ($\mu \pm \sigma$) of the projections at the first five principal directions of the gradient (at initialization). These projections were computed under different sizes of contexts S.

	$\lvert S\rvert = 0.05n$	$\lvert S\rvert = 0.5n$	$\lvert S\rvert = 0.95n$
The first direction	-0.0024±0.1004	0.0025±0.1740	0.0103±0.2277
The second direction	0.0089±0.0890	0.0003±0.1434	0.0076±0.1906
The third direction	-0.0005±0.0855	-0.0012±0.1350	0.0139±0.1819
The forth direction	-0.0029±0.0802	-0.0012±0.1250	-0.0010±0.1543
The fifth direction	0.0000±0.0766	0.0005±0.1161	0.0022±0.1453

Table 5: Expectations and standard deviations ($\mu \pm \sigma$) of the projections at the first five principal directions of the gradient (at 40 epochs). These projections were computed under different sizes of contexts S.

	$\lvert S\rvert = 0.05n$	$\lvert S\rvert = 0.5n$	$\lvert S\rvert = 0.95n$
The first direction	-0.0015±0.4037	0.0003±0.8675	0.0423±1.5470
The second direction	-0.0209±0.3042	0.0042±0.5653	0.0802±0.9754
The third direction	-0.0284±0.2817	0.0070±0.5164	0.0269±0.7845
The forth direction	0.0208±0.2315	0.0003±0.5057	0.0281±0.7231
The fifth direction	-0.0018±0.1853	0.0082±0.4178	-0.0358±0.6715

Furthermore, we investigated the effects of λ_1 on two tabular datasets. Specifically, we fixed $\lambda_2 = 0$, and adjusted $\lambda_1 = 0.1, 1, 10$ to study the effects of λ_1. We trained middle-order DNNs by boosting interactions of the $[0.3n, 0.7n]$-th orders. Here, we used the MLP-5 networks mentioned in Section 3.1. Figure 18(b) shows that the parameter λ_1 controlled the strength of boosting interactions.

C.2.4 TIME COMPLEXITY ANALYSIS.

We compared the running time between training DNNs by 200 epochs using the proposed losses and normally training DNNs by 200 epochs. We trained these models with a mini-batch size of 128 on a single NVIDIA GeForce RTX 3090 GPU and used 4 subprocesses in data loading. Table 3 shows that the time cost of training DNNs with the proposed losses was about three times as much as the time cost of normally training DNNs, which was acceptable in real applications.

In fact, a more reasonable experiment should compare the time of training the DNN to convergence, *i.e.* the time cost of training a DNN with a certain decrease of the classification loss. However, due to the time limit of the rebuttal, this experiment has not been finished. Nevertheless, the current loss curve has indicated that the learning with additional losses does not cause a too long training time. We will report such results after the acceptance of the paper.

C.2.5 OTHER VALIDATION EXPERIMENTS

Using new metric to validate bottleneck. Besides, to further remove effects of magnitudes of $|I^{(m)}(i, j|x)|$ on different samples for fair comparison, we adopted the following new metric.

$$\bar{J}^{(m)} = \mathbb{E}_{x\in\Omega} \frac{\mathbb{E}_{i,j}[|I^{(m)}(i, j|x)|]}{\mathbb{E}_{m'}[\mathbb{E}_{i,j}[|I^{(m')}(i, j|x)|]]} \tag{11}$$

where Ω denotes the set of all samples. We found that using the new metric resulted in few changes to the distribution of the interaction strength. Figure 19 shows a similar representation bottleneck phenomenon when we used the new metric, *i.e.*, the middle-order interaction was less likely to be encoded in DNNs.

Validation of zero-mean assumption in Theorem 1. We further verified the reliability of the zero-mean assumption in Theorem 1. To this end, we analyzed the expectation $\mathbb{E}_{i,j,S}[\frac{\partial \Delta v(i,j,S)}{\partial W}]$ of the gradient *w.r.t.* the parameter W at the first convolutional layer when training ResNet-18 on the Tiny ImageNet dataset.

Since the above gradient $\frac{\partial \Delta v(i,j,S)}{\partial W}$ is a high-dimensional vector, it is difficult to visualize the expectation of all dimensions. Alternatively, we inferred and analyzed the projections on the first five

principal directions of the gradient. In this way, the gradient can be represented as a weighted sum of these projections. Then, the validation was simplified as analyzing the expectations of the five projections.

Specifically, we sampled 10 pairs of (i, j) and 1000 contexts S, and calculated the above gradient for each set (i, j, S), respectively. We used the principal component analysis (PCA) technique to represent each high-dimensional $\frac{\partial \Delta v(i,j,S)}{\partial W}$ as a five-dimensional projection vector on the principal directions. *I.e.*, $\frac{\partial \Delta v(i,j,S)}{\partial W} \approx \sum_{k=1}^{5} G_k(i, j, S) v_k$, where v_k represents the k-th principal directions and $G_k(i, j, S)$ denotes the corresponding projection of $\frac{\partial \Delta v(i,j,S)}{\partial W}$ on the direction v_k. Therefore, if $\mathbb{E}_{i,j,S}[G_k(i, j, S)] = 0$ for $k = 1, \ldots, 5$, then the zero-mean assumption that $\mathbb{E}_{i,j,S}[\frac{\partial \Delta v(i,j,S)}{\partial W}] = 0$ can be validated.

To this end, we computed the expectation $\mathbb{E}_{i,j,S} G_k(i, j, S)$ by averaging over 10 pairs of (i, j) and 1000 contexts S. We computed the expectation when $|S| = 0.05n, |S| = 0.5n$, and $|S| = 0.95n$. Tables 4 and 5 report expectations and standard deviations ($\mu \pm \sigma$) of the five projections at initialization and at 40 epochs, respectively. Tables 4 and 5 show that these expectations were almost 0, which validated the zero-mean assumption in Theorem 1.

In addition, for simplicity, we used σ^2 to denote the variance of the gradient $\frac{\partial \Delta v(i,j,S)}{\partial W}$ in Theorem1, without distinguishing different sizes of contexts S. Tables 4 and 5 show that there existed some small difference between variances of the gradient when we considered different sizes of contexts S. However, such difference is negligible in the proof and does not affect the conclusion of Theorem 1. This is because the learning strength of the m-th order interaction is proportional to $\frac{n-m-1}{n(n-1)} / \sqrt{\binom{n-2}{m}}$, whose difference between different sizes of contexts was much larger than the above difference in variance.