A Unified Approach to Interpreting Self-supervised Pre-training Methods for 3D Point Clouds via Interactions

Supplementary Material

7. Properties of interactions

As mentioned in Sec. 3, the interactions in this paper refer to Harsanyi interaction, which is a standard metric to quantify the AND relationship between input variables in a DNN. In fact, the Harsanyi interaction $I(S)$ satisfies several desirable properties and axioms in game theory [17], making it a robust choice for faithfully explaining the inference scores of a DNN, as detailed below.

(1) *Efficiency axiom.* The output score of a model can be decomposed into interaction effects of different patterns, *i.e.* $v(x) = \sum_{S\subseteq N} I(S)$.

(2) *Linearity axiom.* If we merge output scores of two models v_1 and v_2 as the output of model v, *i.e.* $\forall S \subseteq N,\ v(x_S) = v_1(x_S) + v_2(x_S)$, then their interaction effects $I_{v_1}(S)$ and $I_{v_2}(S)$ can also be merged as $\forall S \subseteq N,\ I_v(S) = I_{v_1}(S) + I_{v_2}(S)$.

(3) *Dummy axiom.* If a variable $i \in N$ is a dummy variable, *i.e.* $\forall S \subseteq N \setminus \{i\},\ v(x_{S\cup\{i\}}) = v(x_S) + v(x_{\{i\}})$, then it has no interaction with other variables, *i.e.* $\forall \emptyset \neq T \subseteq N \setminus \{i\},\ I(T \cup \{i\}) = 0$.

(4) *Symmetry axiom.* If input variables $i, j \in N$ cooperate with other variables in the same way, *i.e.* $\forall S \subseteq N \setminus \{i, j\},\ v(x_{S\cup\{i\}}) = v(x_{S\cup\{j\}})$, then they have same interaction effects with other variables, *i.e.* $\forall S \subseteq N \setminus \{i, j\},\ I(S \cup \{i\}) = I(S \cup \{j\})$.

(5) *Anonymity axiom.* For any permutations π on N, we have $\forall S \subseteq N,\ I_v(S) = I_{\pi v}(\pi S)$, where $\pi S \triangleq \{\pi(i) \mid i \in S\}$, and the new model πv is defined by $(\pi v)(x_{\pi S}) = v(x_S)$. This indicates that interaction effects are not changed by permutation.

(6) *Recursive axiom.* The interaction effects can be computed recursively. For $i \in N$ and $S \subseteq N \setminus \{i\}$, the interaction effect of the pattern $S \cup \{i\}$ is equal to the interaction effect of S with the presence of i minus the interaction effect of S with the absence of i, *i.e.* $\forall S \subseteq N\setminus\{i\}, I(S\cup\{i\}) = I(S \mid i \text{ is always present}) - I(S)$. $I(S \mid i \text{ is always present})$ denotes the interaction effect when the variable i is always present as a constant context, *i.e.* $I(S \mid i \text{ is always present}) = \sum_{T\subseteq S}(-1)^{|S|-|T|} \cdot v(x_{T\cup\{i\}})$.

(7) *Interaction distribution axiom.* This axiom characterizes how interactions are distributed for "interaction functions" [28]. An interaction function v_T parameterized by a subset of variables T is defined as follows. $\forall S \subseteq N$, if $T \subseteq S,\ v_T(x_S) = c$; otherwise, $v_T(x_S) = 0$. The function v_T models pure interaction among the variables in T, because only if all variables in T are present, the output value will be increased by c. The interactions encoded in the function v_T satisfies $I(T) = c$, and $\forall S \neq T,\ I(S) = 0$.

The Harsanyi interaction $I(S)$ also serves as the basis for several game-theoretic attributions and interactions, including the Shapley value [22], the Shapley interaction index [8], and the Shapley Taylor interaction index [28].

(1) *Connection to the Shapley value* [22]. Given an input sample x, let $\phi(i)$ denote the Shapley value of an input variable i. Then, the Shapley value $\phi(i)$ can be explained as the result of uniformly assigning attributions of each Harsanyi interaction to each involving variable i, *i.e.* $\phi(i) = \sum_{S\subseteq N\setminus\{i\}} \frac{1}{|S|+1} I(S \cup \{i\})$. This also proves that the Shapley value is a fair assignment of attributions from the perspective of Harsanyi interaction.

(2) *Connection to the Shapley interaction index* [8]. Given a subset of variables $T \subseteq N$ in an input sample x, the Shapley interaction index $I^{\text{Shapley}}(T)$ can be represented as $I^{\text{Shapley}}(T) = \sum_{S\subseteq N\setminus T} \frac{1}{|S|+1} I(S \cup T)$. In other words, the index $I^{\text{Shapley}}(T)$ can be explained as uniformly allocating $I(S')$ *s.t.* $S' = S \cup T$ to the compositional variables of S', if we treat the coalition of variables in T as a single variable.

(3) *Connection to the Shapley Taylor interaction index* [28]. Given a subset of variables $T \subseteq N$ in an input sample x, the k-th order Shapley Taylor interaction index $I^{\text{Shapley-Taylor}}(T)$ can be represented as weighted sum of interaction effects, *i.e.*, $I^{\text{Shapley-Taylor}}(T) = I(T)$ if $|T| < k$; $I^{\text{Shapley-Taylor}}(T) = \sum_{S\subseteq N\setminus T} \binom{|S|+k}{k}^{-1} I(S \cup T)$ if $|T| = k$; and $I^{\text{Shapley-Taylor}}(T) = 0$ if $|T| > k$.

8. Experimental details

8.1. The point cloud samples for quantifying interactions

This section explains why each input point cloud sample is divided into n regions for quantifying interactions, and provides detailed information about the sampled data. As mentioned in Sec. 3, for an input sample x with n input variables, the DNN can potentially encode up to 2^n interactions. Quantifying all these interactions is computationally prohibitive if each point in a point cloud is treated as an input variable, as a typical point cloud contains 1024 to 4096 points. Moreover, the contribution of each individual point to the DNN's output score is minimal. Thus, treating individual points as input variables is unsuitable. Instead, we divide each input point cloud into n regions following [26]

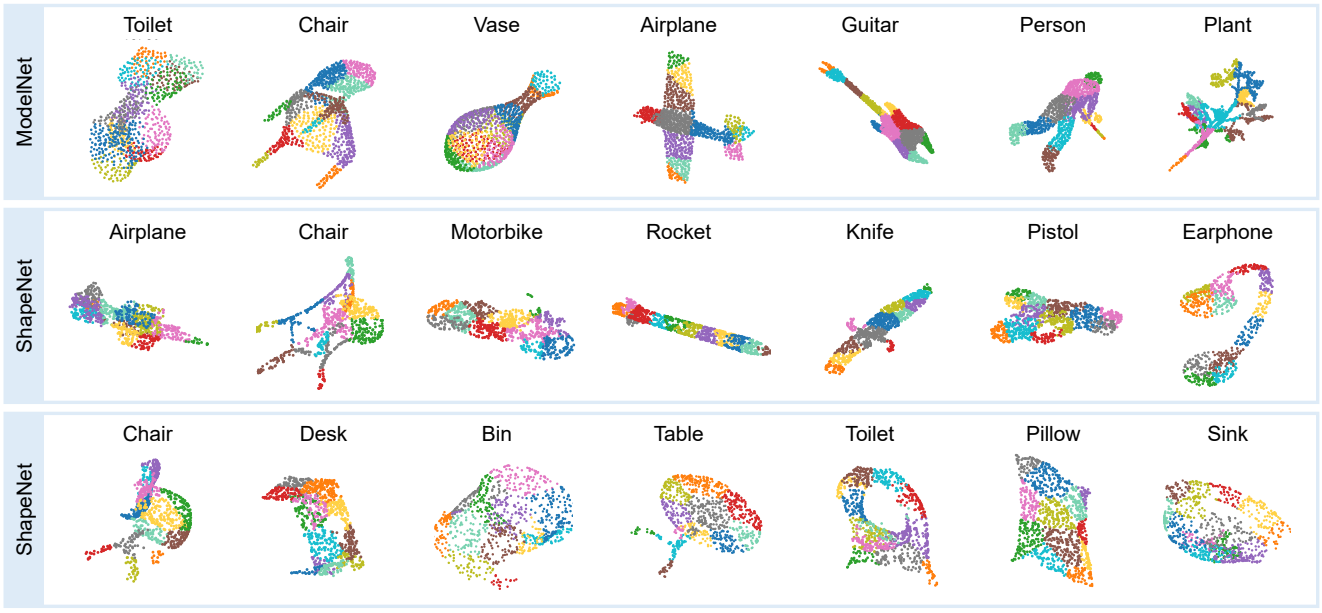

Figure 11. Visualization of the point cloud samples used for quantifying interactions, with each sample divided into 12 regions.

and treat each region as a "single" input variable. This approach ensures that each region carries meaningful semantic information while keeping the total number of interactions, 2^n, computationally manageable.

In addition, as mentioned in Sec. 3, we adopt the farthest point sampling (FPS) algorithm and the k-dimensional tree (KDTree) algorithm for region division. Semantic segmentation is not used for this purpose because its results depend on model performance, which often varies across categories and lacks consistency. Furthermore, semantic segmentation may produce an inconsistent number of regions across different categories, making it difficult to clearly define high-order and low-order interactions.

We randomly sample 10 point clouds from the test set of each category in ModelNet40, ShapeNet, and ScanObjectNN to quantify the interactions encoded by different DNNs, *i.e.*, we sample 400 point clouds from ModelNet40, 160 from ShapeNet, and 150 from ScanObjectNN. For the ScanObjectNN dataset, we quantify interactions using its OBJ_ONLY variant. Fig. 11 visualizes several samples from these datasets, each divided into 12 regions. We observe that the data distribution varies significantly across different datasets, which strongly demonstrates the universality of the common mechanism explored in our study for different pre-training methods across multiple datasets.

8.2. Implementation details for exploring the common mechanism of pre-training methods

In this section, we present the implementation details for exploring the common mechanism of different pre-training methods (corresponding to Conclusion 1). During the pre-training phase, to ensure a fair evaluation of the impact of different pre-training methods on DNNs, we primarily use the pre-trained weights provided in the respective official repositories instead of performing pre-training locally. Exceptionally, for the JigSaw method, since the authors do not provide an official implementation, we use the reproduced version available in the repository of the OcCo method.

We fine-tune the pre-trained DNNs on ModelNet40, ShapeNet, and ScanObjectNN datasets. For the ShapeNet dataset, due to the relatively low difficulty of the original dataset, we randomly sample 1% of the training data for fine-tuning. For the ScanObjectNN dataset, we fine-tune DNNs on its OBJ_ONLY variant. During the fine-tuning process, we randomly sample 1024 points from each point cloud as the input. Each model is fine-tuned for 100 epochs with a batch size of 32 to ensure convergence. For DGCNN, we follow the implementation in [35], setting k (the number of nearest neighbors) to 20, the dropout rate to 0.5, and the embedding dimension to 1024. The fine-tuning process uses the SGD optimizer with an initial learning rate of 0.1 and a cosine learning rate decay scheduler. For PointNet and PCN, we set the dropout rate to 0.3. The fine-tuning process uses the Adam optimizer with an initial learning rate of 0.001 and a step learning rate decay scheduler that reduces the learning rate by a factor of 0.5 every 20 epochs.

8.3. Implementation details for exploring the impact of different factors on the common mechanism

In this section, we present the implementation details for exploring the impact of different factors on the common

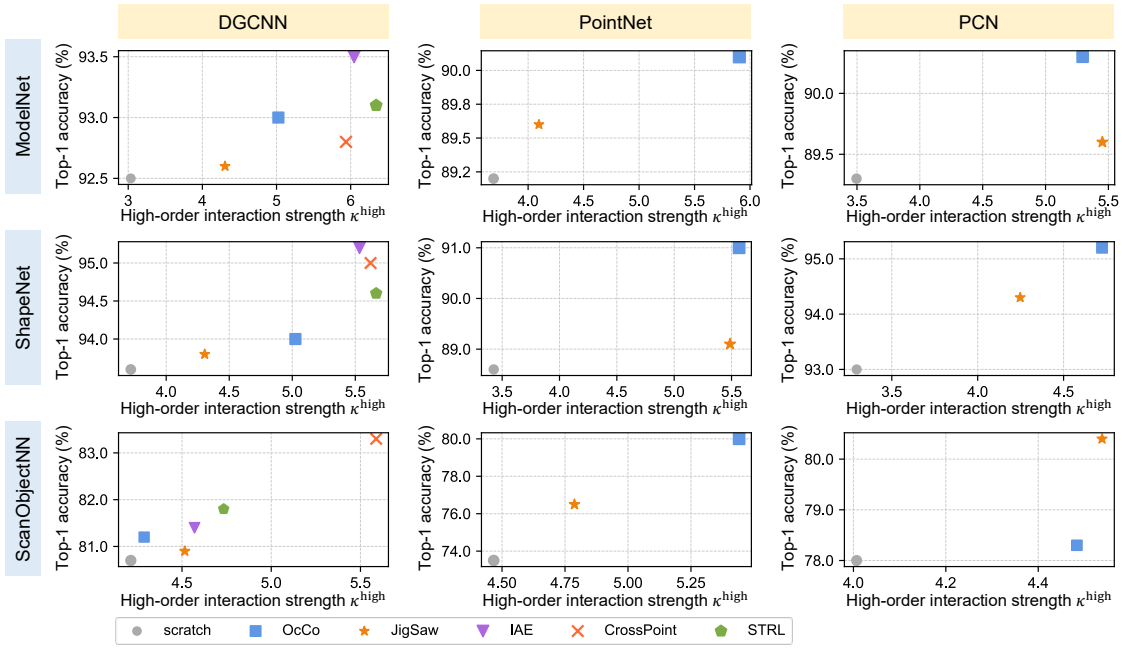

Figure 12. The relationship between the strength of high-order interactions encoded by different DNNs (including DGCNN, PointNet and PCN, trained from scratch or with pre-training methods) and their corresponding classification accuracy. Results show that DNNs encoding stronger high-order interactions tend to exhibit higher accuracy.

mechanism, including the extent of pre-training (corresponding to Conclusion 2(a)) and the amount of fine-tuning data (corresponding to Conclusion 2(b)).

When exploring the relationship between the extent of pre-training and the interactions encoded by DNNs, we conduct experiments on DGCNN using the IAE and CrossPoint methods. For the IAE method, we follow its open-source implementation and pre-train the DGCNN on the full ShapeNet dataset for 1200 epochs, selecting checkpoints at epochs 0, 240, 480, 720, 960, and 1200. These models are then fine-tuned on ModelNet40. For CrossPoint, we follow the implementation details outlined in its paper, pre-training the DGCNN on ShapeNet for 100 epochs, selecting checkpoints at epochs 0, 20, 40, 60, 80, and 100, and then fine-tuning these models on ModelNet40 as well. Next, we quantify the interactions encoded by these different DNNs.

When exploring the relationship between the amount of fine-tuning data for downstream tasks and the interactions encoded by DNNs, we generate varying amounts of fine-tuning data from ModelNet40 and conduct experiments on DGCNNs, trained either from scratch or pre-trained using the IAE or CrossPoint methods. We first randomly sample one instance per category from the ModelNet40 training set and then randomly select a specific proportion of the remaining instances to create fine-tuning datasets of varying sizes. Ultimately, we obtain a series of fine-tuning datasets containing 1%, 10%, 20%, 30%, 50%, 70%, and 100% of the data. To ensure convergence during fine-tuning with small-scale datasets, we fine-tune each DGCNN for 200 epochs. Other training configurations for DGCNN remain consistent with those described in Sec. 8.2.

	ModelNet40 classes	ShapeNet classes	# Testing
1	airplane	airplane	732
2	car	car	239
3	chair	chair	1100
4	guitar	guitar	237
5	lamp	lamp	429
6	laptop	laptop	127
7	cup	mug	54
8	table	table	1436

Table 5. The 8 common classes between ShapeNet and ModelNet40 and the corresponding number of test samples in ShapeNet.

8.4. Implementation details for exploring the potential risk of pre-training methods in reducing DNN's transferability

In this section, we present the implementation details for exploring the potential risk of pre-training methods in reducing DNN's transferability (corresponding to Conclusion 3). We fine-tune DGCNNs on varying amounts of data from ModelNet40, as described in Sec. 8.3, and evaluate their performance on the unseen ShapeNet dataset. Since the categories in ModelNet40 and ShapeNet do not completely overlap, we select eight common categories between the two datasets and evaluate the accuracy of DNNs on these

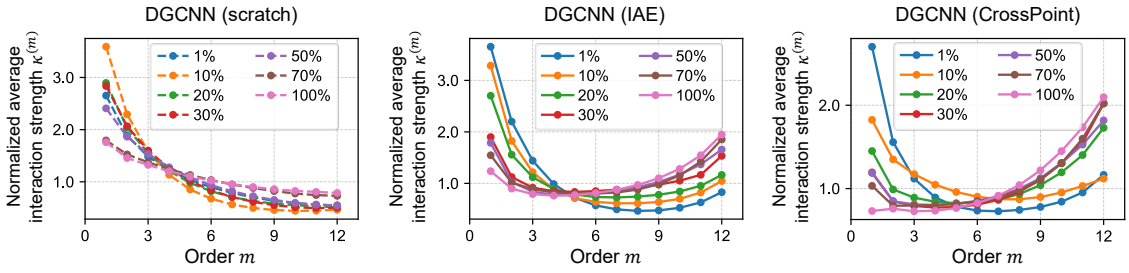

Figure 13. Comparing the normalized average strength of interactions encoded by DNNs fine-tuned with varying amounts of data, including the DGCNN trained from scratch and the DGCNN pre-trained using the IAE or CrossPoint methods. Results show that as the amount of fine-tuning data increases from 1% to 100%, the strength of high-order interactions encoded by the DNNs generally increases, while the strength of low-order interactions decreases. Additionally, for any given amount of fine-tuning data, pre-trained DGCNNs consistently encode stronger high-order interactions than the DGCNN trained from scratch.

categories from ShapeNet. Detailed information is provided in Tab. 5. The training configurations for DGCNN are consistent with those outlined in Sec. 8.3.

8.5. Implementation details for training DNNs with the proposed loss term

In this section, we explain the process of training DNNs with the proposed loss term and present the implementation details for training DNNs on classification and semantic segmentation tasks. As noted in Sec. 5, computing Eq. (6) is NP-hard. When the training dataset is large, such as ModelNet40 with 9843 training samples, quantifying all 2^n interactions for each sample becomes infeasible. To address this, we adopt a sampling-based approach, as described in Sec. 5, and derive the approximate formula for the interaction loss, presented in Eq. (7). However, calculating $\mathcal{L}'_{\text{interaction}}$ for all samples still incurs significant computational overhead. Therefore, we randomly select a subset of samples from each batch and compute $\mathcal{L}'_{\text{interaction}}$ only for this subset. Specifically, we sample $\frac{1}{8}$ of the data from a batch, *i.e.*, 4 samples from a batch of 32. This approach ensures that the computation of the proposed loss remains within a manageable range.

For the 3D point cloud classification task, we evaluate the effectiveness of our proposed loss term on the ModelNet40 and ScanObjectNN datasets. For the ScanObjectNN dataset, we conduct experiments on its most challenging variant, PB_T50_RS, which is derived from the original dataset through random bounding box shifts, rotations, and scaling. This variant includes 11,416 training samples and 2,882 testing samples. In our experiments, we randomly sample 1,024 points from each point cloud as the input for both training and testing. All training settings for DNNs remain consistent with those outlined in Sec. 8.2.

For the semantic segmentation task, we conduct experiments on the S3DIS dataset, as described in Sec. 5. S3DIS consists of 3D point clouds collected from six large-scale indoor environments, each annotated with per-point categorical labels. Specifically, following the implementation details in [38], we divide each room instance into 1m $\times$ 1m blocks and randomly sample 4,096 points from each block as input. We first train the DGCNN on the ModelNet40 dataset for the classification task, and then fine-tune it on the semantic segmentation task using six-fold cross-validation. All other training settings for DGCNN remain consistent with those in Sec. 8.2.

9. More experimental results

9.1. Additional Results for Conclusion 1

As a supplement to Fig. 3 (b), Section 4.2 of the paper, we present the relationship between the classification accuracy and the strength of high-order interactions encoded by PointNet and PCN, as shown in Fig. 12. The figure reveals a trend consistent with that observed for DGCNNs, *i.e.*, DNNs encoding stronger high-order interactions tend to achieve higher accuracy. This observation further reinforces Conclusion 1.

9.2. Additional Results for Conclusion 2(b)

As a supplement to Fig. 7 (b), Section 4.3 of the paper, we present the interactions encoded by the DGCNN trained from scratch with varying amounts of data, comparing them with the interactions encoded by the DGCNNs pre-trained using the IAE or the CrossPoint methods. As show in Fig. 13, the strength of high-order interactions encoded by the DGCNN trained from scratch increases as the amount of fine-tuning data grows from 1% to 100%, while the strength of low-order interactions generally decreases, similar to the pre-trained DGCNNs. This observation further supports Conclusion 2(b). Moreover, for any given amount of fine-tuning data, the pre-trained DGCNNs consistently encode stronger high-order interactions than the DGCNN trained from scratch, further validating the common mechanism discussed in Conclusion 1.