Rotation invariance vs non-invariance: Why not both in one network?

Editors: List of editors' names

Abstract

The lack of robustness of many state-of-the-art computer vision models to geometric transformations (rotation, scaling, etc.) has long been a recognized problem. While this issue is tackled with data augmentation, geometric deep learning offers an elegant augmentation-free solution at the cost of lower performance on standard benchmarks. We investigate whether we can obtain the best of both worlds.

We explore the idea of combining rotation-dependent and rotation-invariant features for point-cloud classification. A simple ensemble of regular and rotation-invariant deep point-cloud networks with a joint classifier head and a simple loss function boosts point-cloud classification and increases robustness to the arbitrary rotations. The presented approach works without rotation augmentation during training and is applicable to both pose-aligned and non-aligned datasets.

Keywords: Point-Cloud Classification, Rotation-Invariant Networks, Rotation-Equivariant Deep Learning, Geometric Deep Learning, Ensemble Learning, Data-Augmentation-Free Training, Robustness to Geometric Transformations

1. Introduction

General algorithms for 3D understanding of the world have many applications, including autonomous driving, robotics, or augmented reality. Approximating the continuous and infinite world of 3D geometries with discrete and sparse point clouds is currently the most popular approach, primarily because of data availability and the universality of this representation. Point clouds can be recorded using LiDAR or 3D scanners and digitally generated from meshes and CAD models. A general 3D understanding of the world should incorporate properties analogous to human visual perception. For example, this includes being able to recognize objects regardless of their pose, distance, or scale from the sensor. These variations can be described by geometric transformations, and indifference to these transformations can be formalized through the concepts of equivariance and invariance.

Nowadays, transformer-based architectures (Vaswani et al., 2017) are popular because they have low geometric priors (or biases). This enables them to be highly expressive and completely data-driven, i.e. to learn almost anything from a large corpus of data. However, a large number of parameters makes these models data-hungry and computationally intensive during training. Finally, there is no guarantee that they will correctly learn geometric symmetries from the data. This results in a poor generalization to unseen (out-of-distribution) geometric transformations.

Geometric deep learning (Bronstein et al., 2021) addresses these problems by replacing attention or convolutional layers with functions that are equivariant or invariant to specific geometric transformations (e.g. rotations, translations). Generally, equivariant methods require fewer data and fewer parameters to learn meaningful representations (Gerken et al., 2022), but they are less expressive than established deep learning methods (e.g. transformers) and often show lower performance on benchmarks.

Invariance and equivariance: Let $\Phi : \mathcal{X} \to \mathcal{Y}$ be a function on the high dimensional spaces $\mathcal{X}, \mathcal{Y}$ and let $(G, \cdot)$ be a group that is defined on both, $\mathcal{X}$ and $\mathcal{Y}$. The function Φ is *equivariant* to group operations from G if $\Phi(g \cdot X) = g \cdot \Phi(X)$, and *invariant* to group operations from G if $\Phi(g \cdot X) = \Phi(X)$, for every $X \in \mathcal{X}$ and $g \in G$. Here, the action of the group on the corresponding space is denoted by $g\cdot$ and the actual group action is implemented by a representation of the group on the corresponding space.

In this work, X is a point cloud in $\mathbb{R}^3$, G is a 3D rotation group $SO(3)$, g denotes the rotation matrix, and $\cdot$ is a matrix multiplication. The function Φ is a neural network with trainable parameters and $\mathcal{Y}$ is a high dimensional feature vector space representing a point cloud. Hence, interpretation of rotation invariance for Φ is the following: Φ gives the same output irrespective of the rotation of the input.

Note that the 3D space $\mathbb{R}^3$ introduces additional complexity compared to the $\mathbb{R}^2$ space, especially with respect to rotations. One example are the 2D and 3D rotation groups $SO(2)$ and $SO(3)$. $SO(3)$ is *non-commutative* (non-abelian) in contrast to $SO(2)$, i.e. in 3D it matters in which order one applies rotations! For that reason, our approach focuses solely on rotation invariance in 3D space for point clouds. Finally, point clouds are more suitable for analyzing rotation invariance than images, as rotating 2D images on a discrete grid introduces interpolation errors.

Why is the rotation invariance an important property? Its consequence is that no rotation augmentation is needed. This reduces the effort of data preparation or pose alignment for many point cloud applications. Rotation-invariant neural networks have proven to be more data-efficient and achieve comparable results without intensive data augmentation (Zhang et al., 2024). Some tasks, such as finding duplicates in a database, require matching two identical objects independent of their pose in the 3D space.

We aim at combining highly expressive data-driven architectures and equivariant architectures that are robust to geometric transformations. We propose a simple architecture that combines features from a rotation-invariant backbone and from a rotation-dependent backbone. Joint features are then concatenated and transformed via a multi layer perceptron (MLP) to obtain more robust and expressive representations for downstream tasks. Our experiments show:
1. Joint architecture consistently improves classification across six datasets (Section 4.2.1).
2. Features from invariant and non-invariant networks are somewhat complementary: one can use frozen pretrained networks and train only MLP heads to improve segmentation or classification across five datasets (Section 4.2.2).
3. Joint architecture is significantly more robust to random $SO(3)$ rotations for several rotation augmentation strategies during training (Section 4.2.3 and Appendix E).

2. Related work

Point clouds sparsely describe 3d geometry of an object. Alternative ways to process and analyze 3d geometries are via voxelization (Riegler et al., 2017; Wang et al., 2017), meshing (triangulation) (Shakibajahromi et al., 2024) or multi-view images (Hamdi et al., 2021). In the rest of the section, we give an overview of the most important ideas of deep networks with *point clouds* as input.

2.1. Standard (rotation-dependent) point cloud architectures

MLP/CNN-based architectures: One of the first works on deep learning exclusively for point clouds was a simple pointwise multilayer perceptron known as PointNet (Qi et al., 2017a) that was equivariant to point permutations in the point cloud. PointNet++ (Qi et al., 2017b) extends this architecture to extract features from local neighbourhoods of the points. This inspired further development and extension of convolutions (Thomas et al., 2019), graph convolutions (Wang et al., 2019) or residual MLPs (Ma et al., 2022; Qian et al., 2022) to point clouds.

Transformer-based architectures: First approaches using transformer-based neural networks (Vaswani et al., 2017) for point cloud classification were purely generalization of language transformers in a supervised setting (Guo et al., 2021; Zhao et al., 2021; Wu et al., 2022; Zhang et al., 2022a). The subsequent works improved the performance of transformer-based architectures by adapting a self-supervised training to large corpora of 3D/point cloud data, e.g. masked autoencoding with PointMAE (Pang et al., 2022), PointBERT (Yu et al., 2022), and PointM2AE (Zhang et al., 2022b). Recent state-of-the-art models are indeed transformer-based and include PointGPT (Chen et al., 2024), GPSFormer (Wang et al., 2024), and Point Transformer v3 (Wu et al., 2024).

Multimodal-based architectures: Many point cloud datasets are extracted from meshes or CAD models. Rendering tools enable the creation of 2D counterparts, i.e. images of 3D models. Hence, in these situations corresponding 2D and 3D spaces exist which can be aligned through, e.g. unsupervised multimodal contrastive learning (Zhang et al., 2022c; Xue et al., 2023, 2024; Qi et al., 2023) or autoencoders (Dong et al., 2022). Naturally, using pretrained text-image models also allows to align another modality, namely language, to 3D models, e.g. see (Xue et al., 2024).

2.2. How to achieve rotation invariance / equivariance?

1) *Alignment to canonical pose* refers to the standardization of the object poses within the same class. This pose can be chosen to reflect how this object usually appears around us. For that purpose, objects can be rotated so that its principal axes of inertia match x, y, z axis (Sun et al., 2021). Alternatively, we can train the network to align the objects without supervision, e.g. using capsule networks (Sun et al., 2021), spherical CNNs (Spezialetti et al., 2020), or ConDor (Sajnani et al., 2022). A pose alignment module can be incoporated in existing architectures to align poses prior to downstream tasks (Fang et al., 2020). Furthermore, one can align point clouds on local neighbourhoods via local reference frames (LRFs) (Kim et al., 2020b) or centrifugal reference frames (Lou et al., 2023).

2) *Rotation invariant/equivariant features.* Building blocks of PointNet can be adjusted to be rotation equivariant by extending neurons from 1D scalars to 3D vectors (Deng et al., 2021). Another way is to extract low-level rotation invariant features (angles, distances) from local neighbourhoods on several scales (Zhang et al., 2019) and combine them via residual MLPs (Zhang et al., 2022d) or transformers (Zhang et al., 2024).

3) *Group equivariant deep learning* (Cohen and Welling, 2016) is a general theoretical framework that builds equivariant representations on generic groups. Its applications to the $SE(3)$ (rotation-translation) group found its use for point clouds (Thomas et al., 2018; Chen et al., 2021; Finkelshtein et al., 2022; Fuchs et al., 2020).

4) *Training with data augmentation* is a standard technique to improve the robustness of (rotation-dependent) neural networks against object rotations. This, however, leads only to an approximate equivariance only. Studies also include comparisons between learning equivariance from data vs using equivariant models (Gerken et al., 2022; Müller et al., 2021; Gandikota et al., 2021; Gerken et al., 2022; Moskalev et al., 2023; Brehmer et al., 2024). The results imply that learning equivariance from data requires larger model capacity and more data. This highlights the necessity of directly incorporating the equivariance property in the model architecture.

3. Method

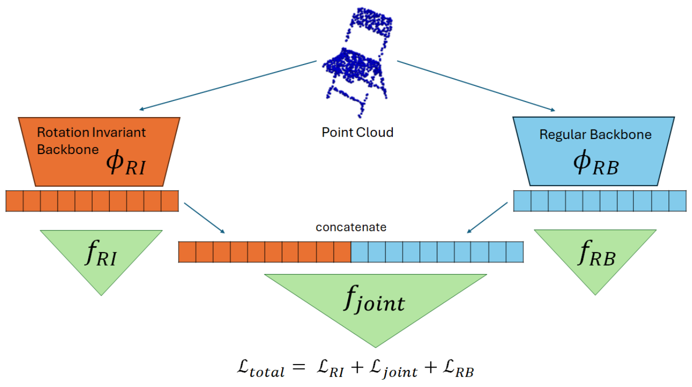
Point Cloud
Rotation Invariant Backbone
ϕ_{RI}
Regular Backbone
ϕ_{RB}
concatenate
f_{RI}
f_{joint}
f_{RB}
$\mathcal{L}_{total} = \mathcal{L}_{RI} + \mathcal{L}_{joint} + \mathcal{L}_{RB}$

Figure 1: Architecture overview

The main idea of our approach is to apply two point cloud networks in parallel. The resulting feature vectors are, subsequently, concatenated into one larger feature vector, and finally mapped to a class probability vector using a multi-layer perceptron (MLP) (Figure 1). The architecture consists of one rotation-invariant and one rotation-dependent network. Additionally, we introduce two MLPs for feature vectors from each network separately. This addition encourages both networks to learn meaningful feature representations individually.

3.1. Rotation-Invariant backbone

In principle, one can use arbitrary rotation-invariant backbones. In this paper, we use a simplified version of the RIConv++ network (Zhang et al., 2022d) (Figure 3 in Appendix B). RIConv++ requires point cloud with normals as input, i.e. $\mathbf{X} \in \mathbb{R}^{N\times 6}$. Its elements are denoted as $(x, n_x) \in \mathbf{X}$, where $x \in \mathbb{R}^3$ contains 3d coordinates and $n_x \in \mathbb{R}^3$ is a normal direction in point x. We use a subset of the original RI features from RIConv++ which consists of three simple features $\{d_0, \alpha_0, \alpha_1\}$ between reference point (p, n_p) and point in its local neighbourhood (x, n_x):

- $d_0 = ||x - p||$ distance between points x and p,
- $\alpha_0 = \angle(\overrightarrow{xp}, n_p)$ angle between normal n_p and line segment $\overrightarrow{xp}$,
- $\alpha_1 = \angle(\overrightarrow{xp}, n_x)$ angle between normal n_x and line segment $\overrightarrow{xp}$.

This simplified version of RIConv++ with 4 layers has 0.6M parameters. Rotation-invariant backbone is denoted as $\Phi_{RI} : \mathbb{R}^{N\times 6} \rightarrow \mathbb{R}^{d_{RI}}$, where d_{RI} is the dimension of the embedding vector.

3.2. Regular backbone

Similar to the rotation-invariant network, the rotation-dependent network can be any arbitrary model operating on point clouds. In this work, we investigate two network architectures, namely PointMLP Elite (Ma et al., 2022) and PointBERT (Yu et al., 2022). PointMLP (Elite) is MLP-based architecture inspired by the PointNet++ and introduces residual (i.e. skip-connections) MLP layers. PointBERT is an adaptation of the bidirectional encoder representations from transformers (BERT) architecture (Devlin et al., 2019) generalized to point clouds.

PointMLP Elite is a lightweight network with 0.6M parameters, while PointBERT has 22M parameters. Hence, with the backbone choice we cover both sides of the complexity spectrum. Regular rotation-dependent backbone is denoted as $\Phi_{RB} : \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^{d_{RB}}$, where d_{RB} is the dimension of the embedding vector.

3.3. Network architecture and loss function

Both, the rotation-dependent (Φ_{RB}) and rotation-invariant (Φ_{RI}) backbones take a given point cloud $\mathbf{X} \in \mathbb{R}^{N\times 3}$ as an input and estimate the two separate embedding vectors $\mathbf{y}_{RB} \in \mathbb{R}^{d_{RB}}$ and $\mathbf{y}_{RI} \in \mathbb{R}^{d_{RI}}$, respectively. These vectors are, subsequently, concatenated into one embedding vector $\mathbf{y}_{joint} = [\mathbf{y}_{RB}, \mathbf{y}_{RI}] \in \mathbb{R}^{d_{RB}+d_{RI}}$ and passed to the classification head $f_{joint} : \mathbb{R}^{d_{RB}+d_{RI}} \rightarrow \mathbb{R}^{d_{cls}}$ in order to obtain the class probability vector $\mathbf{c}_{joint} \in \mathbb{R}^{d_{cls}}$, where d_{cls} denotes the dimension of the vector, i.e. number of classes. To evaluate the discrepancy between the desired network output $\mathbf{c} \in \mathbb{R}^{d_{cls}}$ and the actual network output $\hat{\mathbf{c}} \in \mathbb{R}^{d_{cls}}$, we use the well-known cross-entropy loss function $\mathcal{L}_{ce}$.

Furthermore, we investigate whether classification performance benefits from additional optimization criteria. Therefore, we extend the loss function so that both backbone networks are forced to independently estimate the correct class for a given point cloud input $\mathbf{X}$ (Figure 1). Consequently, the overall criteria[1] (i.e. total loss) is defined as the sum of the individual cross-entropy losses as shown in the following equation:

$$\mathcal{L}_{total}\big(\mathbf{c}, \mathbf{X}\big) = \mathcal{L}_{joint}\big(\mathbf{c}, \mathbf{X}\big) + \mathcal{L}_{RB}\big(\mathbf{c}, \mathbf{X}\big) + \mathcal{L}_{RI}\big(\mathbf{c}, \mathbf{X}\big). \qquad (1)$$

4. Experiments

4.1. Datasets and training setup

Table 1 gives the overview of the datasets used. For more details on datasets and training setups, we refer to Appendix A. We used standard train/test splits from the literature. For classification task we report overall accuracy (OA) without voting procedure for every

1. More details on this construction is given in Appendix C, as well as ablation study on loss function in Appendix D.

dataset (Qi et al., 2017a). For evaluation of segmentation task we report both class and instance mean Intersection-over-Union (mIoU), as well as mean IoU for every class.

Dataset	Size	Task	Class number	Pose aligned
ModelNet40 (Wu et al., 2015)	12.3k	classification	40	✓
MCB-B (Kim et al., 2020a)	18k	classification	25	×
ScanObjectNN (Uy et al., 2019)	2.9k	classification	15	×
Protein data (Yacoub et al., 2025)	9.2k	classification	97	×
ShapeNetPart (Yi et al., 2016)	16.9k	segmentation	50	×

Table 1: Dataset summary. ScanObjectNN consists of three variations of dataset with respect to difficulty: OBJ-ONLY, OBJ-BG and PB-T50-RS.

4.2. Results

4.2.1. Benchmarking on several datasets

First, we analyze incremental improvement of composite model from individual models for two different regular backbones: PointMLP Elite and PointBERT (Table 2). In both cases our composite model brings improvements compared to the individual models on both pose-aligned and pose-non-aligned datasets. Note that for PointBERT baseline we fine-tune pretrained version of the model, while our composite model is trained completely from scratch. Interestingly, the smaller composite model using the PointMLP Elite as the backbone achieves higher performance compared to the larger model using the PointBERT backbone, despite having almost 20× less parameters.

Table 3 shows a comparison between our composite model using the PointMLP Elite backbone and results achieved by the major approaches from the literature. Our composite model, having only 1.2M parameters, shows higher performance than three competitors (Qi et al., 2017a,b; Wang et al., 2019), while being outperformed by the larger version of the PointMLP (Ma et al., 2022) and the GPSFormer (Wang et al., 2024). For unsupervised methods that were pretrained with ShapeNet (Chang et al., 2015) or even larger datasets, only larger variants of PointGPT-B/PointGPT-L (Chen et al., 2024) with 120M/360M parameters and pretrained on dataset with 0.5M point clouds clearly outperform our composite model. Note that ShapeNet with 50,000 objects is roughly 4 times larger than ModelNet40.

Method	ModelNet40	MCB-B	OBJ-ONLY / OBJ-BG / PB-T50-RS	Protein data
RI Baseline	90.6%	95.13%	83.4% / 81.2% / 79.2%	84.9%
PointMLP Elite	93.03%	91.26%	87.44% / 86.58% / 85.74%	84.3%
Composite (w PointMLP Elite)	93.6% (+0.57%)	95.51% (+4.25%)	91.39% / 91.05% / 86.40% (+2.95%/+4.47%/+0.66%)	89.15% (+4.9%)
PointBERT (pretrained)	92.86%	-	87.3% / 85.91% / 81.32%	-
Composite (w PointBERT)	93.64% (+0.78%)	-	88.78% / 88.78% / 83.87% (+1.48%/+2.86%/+2.55%)	-

Table 2: Quantifying the effect of combining rotation invariant and rotation dependent point cloud networks on standard benchmarks.

Method	Parameters	Supervised / Extra data	ModelNet40	OBJ-ONLY / OBJ-BG / PB-T50-RS
PointNet (Qi et al., 2017a)	3.5M	✓/ ×	90.3%	79.2% / 73.3% / 68.0%
PointNet++ (Qi et al., 2017b)	1.5M	✓/ ×	92.4%	84.3% / 82.3% / 77.9%
DGCNN (Wang et al., 2019)	1.8M	✓/ ×	92.1%	86.2% / 82.8% / 78.1%
DeltaConv (Wiersma et al., 2022)	2M	✓/ ×	93.8%	89.5% / 89.4% / 84.7%
PointMLP (Ma et al., 2022)	12.6M	✓/ ×	94.0%	- / - / 85.4%
GPSFormer (Wang et al., 2024)	2.4M	✓/ ×	95.0%	- / - / 93.8%
PointBERT (Yu et al., 2022)	22.1M	× / ✓	92.7%	88.12% / 87.43% / 83.07%
Point-MAE (Pang et al., 2022)	22.1M	× / ✓	93.2%	88.29% / 90.02% / 85.18%
Point-M2AE (Zhang et al., 2022b)	12.9M	× / ✓	93.4%	88.8% / 91.2% / 86.4%
ACT (Dong et al., 2022)	22.1M	× / ✓/ mutimodal	93.7%	89.16% / 91.22% / 85.81%
PointGPT-S (Chen et al., 2024)	29M	× / ✓	93.51%	90.0% / 91.6% / 86.9%
PointGPT-B (Chen et al., 2024)	120M	× / ✓	93.75%	95.2% / 95.8% / 91.9%
PointGPT-L (Chen et al., 2024)	360M	× / ✓	93.83%	96.6% / 97.2% / 93.4%
Composite (w PointMLP Elite)	1.2M	✓/ ×	93.6%	91.39% / 91.05% / 86.4%

Table 3: Benchmarking of our method with PointMLP Elite as backbone on 3 classification datasets. All accuracies are reported without voting.

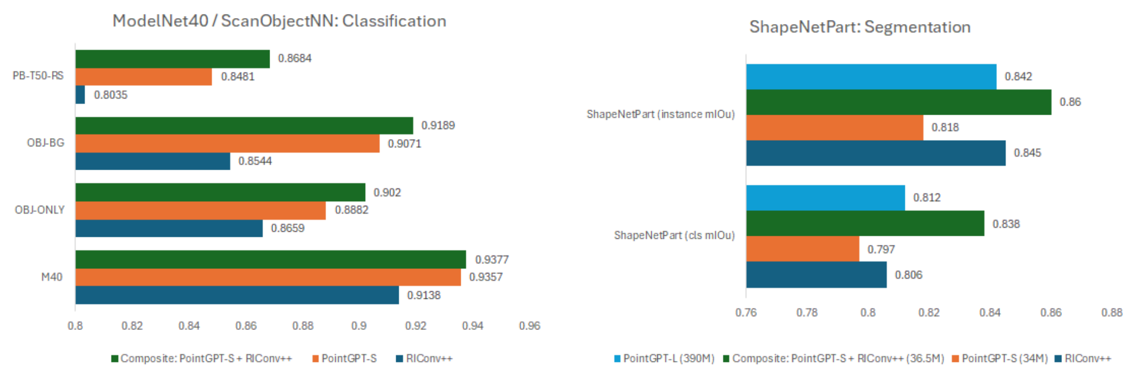

Figure 2: Training only classification heads on pretrained frozen backbones.

4.2.2. Training only MLP heads on pretrained frozen backbones

Finally, we test whether we can use pretrained features from both rotation-invariant and rotation-dependent backbones and combine them without fine-tuning the whole models. Here, we keep both backbones frozen and train only MLP heads for 100 epochs. For rotation-invariant backbone we use RIConv++ (Zhang et al., 2022d), while rotation-dependent backbone is PointGPT-S (Chen et al., 2024), the most recent state-of-the-art model. The results are shown in Figure 2, including segmentation task on ShapeNetPart, and support the claim that the frozen pretrained features from both networks are already enough to improve point-cloud classification/segmentation.

4.2.3. Robustness to rotations

We evaluate the robustness with respect to rotations on the ModelNet40 dataset for four rotation augmentation settings used for training. These include no rotation augmentation (Scenario $\emptyset$), augmentation with random rotation from $SO(3)$ group (Scenario $SO(3)$), as well rotation augmentation with fewer discrete rotations (Scenario $k\frac{\pi}{2}$ and Scenario $k\frac{\pi}{4}$). In comparison to PointBERT, composite model shows better overall accuracy and increased robustness to random $SO(3)$ rotations in every scenario, as shown in Table 4. Additional experiments on PB-T50-RS dataset are shown in Appendix E.

Scenario $\emptyset$: Generalization to unseen rotations This scenario trains the model in aligned poses and tests on randomly rotated objects. Results are shown in Table 4(a). This setting favors rotation-invariant methods. Our method shows increased robustness compared to all rotation-dependent methods, including all models from PointGPT series.

Setting: Train augmentation: Test augmentation:	(a) $\emptyset$ $\emptyset/SO(3)$	(b) $k\frac{\pi}{2}$ $\emptyset/SO(3)$	(c) $k\frac{\pi}{4}$ $\emptyset/SO(3)$	(d) $SO(3)$ $\emptyset/SO(3)$
PointBERT (pretrained)	92.86%/11.15%	88.22%/23.44%	84.76%/83.49%	85.93%/86.49%
Composite (w PointBERT)	93.64%/50.9% (+0.78%/ + 39.75%)	91.14%/87.06% (+2.92%/ + 63.62%)	91.14%/91.39% (+6.38%/ + 7.9%)	90.78%/90.97% (+4.85%/ + 4.48%)
PointGPT-S (29M)	93.51%/14.15%	–	–	–
PointGPT-B (120M)	93.75%/16.35%	–	–	–
PointGPT-L (360M)	93.83%/17.25%	–	–	–
Pt Transformer v2 (Wu et al., 2022)	–	–	–	–/88.3%
RotPredictor (Fang et al., 2020)	–	–	–	–/90.2%

Table 4: Evaluation of rotation augmentation strategies in training $\{\emptyset, k\frac{\pi}{2}, k\frac{\pi}{4}, SO(3)\}$. Evaluation is done on ModelNet40 test set under two conditions: no rotation augmentation ($\emptyset$ in Test) and random $SO(3)$ rotation ($SO(3)$ in Test).

Scenario $SO(3)$: Learning invariance with rotation augmentation In this scenario we attempt to train rotation-dependent models to "learn" rotation invariance from data by means of rotation data augmentation. Table 4(d) shows that PointBERT is able to only learn approximate rotation invariance at the expense of reduced overall accuracy: down from 92.86% on aligned training data to 85.93% on non-aligned training data. Although accuracy of the composite model also degrades, it still performs better than other rotation-dependent architectures.

Scenario $k\frac{\pi}{2}$: The first scenario covers the case where the training set consists of pose aligned data, while the second scenario reflects the case when pose is completely random in the training set. This scenario is somewhere in between: models are in aligned pose at the beginning but can be orthogonally rotated by $\{0, \frac{\pi}{2}, \pi, \frac{3}{2}\pi\}$ around x,y or z axis. In this way, the network has to learn to be rotation-invariant for a finite number of poses (or on a finite subgroup of $SO(3)$). Table 4(b) shows results for this scenario. **Scenario $k\frac{\pi}{4}$** samples more finely the rotation group $SO(3)$ with rotations in $\{k\frac{\pi}{4} | k \in \{0, 1, \cdots, 7\}\}$. Table 4(c) shows results for this scenario. In both scenarios composite model clearly outperforms PointBERT baseline.

4.3. Ablation Study

Ablation study is performed on ModelNet40 with PointBERT as a backbone. Additional ablations are given in Appendix D.

4.3.1. Testing other regular backbone

Table 5 shows the results when changing regular backbone to the well-known DGCNN (Wang et al., 2019). Combined with the findings for PointMLP Elite and PointBERT, this suggests that our method works for arbitrary rotation-dependent backbones.

	∅	+PointBERT	+ DGCNN	+RI
PointBERT (pretrained)	92.86%	–	92.60% (-0.26%)	93.64% (+0.78%)
DGCNN	91.61%	92.60% (+1%)	–	93.45% (+1.84%)

Table 5: Ablation study: varying combinations of RI and RB backbones.

4.3.2. Combining two regular backbones

Next, we explore whether replacing the rotation-invariant backbone (RI) by the rotation-dependent backbone DGCNN improves the classification performance. The results in Table 5 suggest that the improvement is indeed the consequence of the complementarity of the rotation-dependent and rotation-invariant features.

5. Conclusion

In this paper, we explore the idea of combining rotation-invariant and rotation-dependent features for boosting point cloud classification. We have shown that simple ensemble architecture of rotation-invariant and rotation-dependent network achieves the goal, while significantly increasing the robustness to rotations. Additionally, we explored several strategies for rotation augmentation and studied their effect on standard rotation-dependent point cloud network PointBERT, as well as on our composite model.
Our results imply that composite architectures have a lot of unexplored potential, not necessarily limited to point cloud domain. However, further developments in both areas separately should be encouraged because new progress will likely benefit the composite architectures, too. Main limitation our work is that model and data scaling has not been explored in conjuction with rotation invariance due to the lack of large labelled datasets in 3d point domain.

References

Johann Brehmer, Sönke Behrends, Pim De Haan, and Taco Cohen. Does equivariance matter at scale? *arXiv preprint arXiv:2410.23179*, 2024.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue. Pointgpt: Auto-regressively generative pre-training from point clouds. *Advances in Neural Information Processing Systems*, 36, 2024.

Haiwei Chen, Shichen Liu, Weikai Chen, Hao Li, and Randall Hill. Equivariant point network for 3d point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14514–14523, 2021.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

Runpei Dong, Zekun Qi, Linfeng Zhang, Junbo Zhang, Jianjian Sun, Zheng Ge, Li Yi, and Kaisheng Ma. Autoencoders as cross-modal teachers: Can pretrained 2d image transformers help 3d representation learning? *arXiv preprint arXiv:2212.08320*, 2022.

Jin Fang, Dingfu Zhou, Xibin Song, Shengze Jin, Ruigang Yang, and Liangjun Zhang. Rotpredictor: Unsupervised canonical viewpoint learning for point cloud classification. In *2020 international conference on 3D vision (3DV)*, pages 987–996. IEEE, 2020.

Ben Finkelshtein, Chaim Baskin, Haggai Maron, and Nadav Dym. A simple and universal rotation equivariant point-cloud network. In *Topological, Algebraic and Geometric Learning Workshops 2022*, pages 107–115. PMLR, 2022.

Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in neural information processing systems*, 33:1970–1981, 2020.

Kanchana Vaishnavi Gandikota, Jonas Geiping, Zorah Lähner, Adam Czapliński, and Michael Moeller. Training or architecture? how to incorporate invariance in neural networks. *arXiv preprint arXiv:2106.10044*, 2021.

Jan Gerken, Oscar Carlsson, Hampus Linander, Fredrik Ohlsson, Christoffer Petersson, and Daniel Persson. Equivariance versus augmentation for spherical images. In *International Conference on Machine Learning*, pages 7404–7421. PMLR, 2022.

Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational visual media*, 7:187–199, 2021.

Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1–11, 2021.

Sangpil Kim, Hyung-gun Chi, Xiao Hu, Qixing Huang, and Karthik Ramani. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 175–191. Springer, 2020a.

Seohyun Kim, Jaeyoo Park, and Bohyung Han. Rotation-invariant local-to-global representation learning for 3d point cloud. *Advances in Neural Information Processing Systems*, 33:8174–8185, 2020b.

Yujing Lou, Zelin Ye, Yang You, Nianjuan Jiang, Jiangbo Lu, Weiming Wang, Lizhuang Ma, and Cewu Lu. Crin: rotation-invariant point cloud analysis and rotation estimation via centrifugal reference frame. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 1817–1825, 2023.

Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*, 2022.

Artem Moskalev, Anna Sepliarskaia, Erik J Bekkers, and Arnold WM Smeulders. On genuine invariance learning without weight-tying. In *Topological, Algebraic and Geometric Learning Workshops 2023*, pages 218–227. PMLR, 2023.

Philip Müller, Vladimir Golkov, Valentina Tomassini, and Daniel Cremers. Rotation-equivariant deep learning for diffusion mri. *arXiv preprint arXiv:2102.06942*, 2021.

Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European conference on computer vision*, pages 604–621. Springer, 2022.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017b.

Zekun Qi, Runpei Dong, Guofan Fan, Zheng Ge, Xiangyu Zhang, Kaisheng Ma, and Li Yi. Contrast with reconstruct: Contrastive 3d representation learning guided by generative pretraining. In *International Conference on Machine Learning*, pages 28223–28243. PMLR, 2023.

Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in neural information processing systems*, 35:23192–23204, 2022.

Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017.

Rahul Sajnani, Adrien Poulenard, Jivitesh Jain, Radhika Dua, Leonidas J Guibas, and Srinath Sridhar. Condor: Self-supervised canonicalization of 3d pose for partial shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16969–16979, 2022.

Bahareh Shakibajahromi, Edward Kim, and David E Breen. Rimeshgnn: A rotation-invariant graph neural network for mesh classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3150–3160, 2024.

Riccardo Spezialetti, Federico Stella, Marlon Marcon, Luciano Silva, Samuele Salti, and Luigi Di Stefano. Learning to orient surfaces by self-supervised spherical cnns. *Advances in Neural information processing systems*, 33:5381–5392, 2020.

Weiwei Sun, Andrea Tagliasacchi, Boyang Deng, Sara Sabour, Soroosh Yazdani, Geoffrey E Hinton, and Kwang Moo Yi. Canonical capsules: Self-supervised capsules in canonical pose. *Advances in Neural information processing systems*, 34:24993–25005, 2021.

Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Changshuo Wang, Meiqing Wu, Siew-Kei Lam, Xin Ning, Shangshu Yu, Ruiping Wang, Weijun Li, and Thambipillai Srikanthan. Gpsformer: A global perception and local structure fitting-based transformer for point cloud understanding. In *European Conference on Computer Vision*, pages 75–92. Springer, 2024.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11, 2017.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019.

Ruben Wiersma, Ahmad Nasikun, Elmar Eisemann, and Klaus Hildebrandt. Deltaconv: anisotropic operators for geometric deep learning on point clouds. *ACM Transactions on Graphics (TOG)*, 41(4):1–10, 2022.

Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems*, 35:33330–33342, 2022.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4840–4851, 2024.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1179–1189, 2023.

Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, et al. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27091–27101, 2024.

Taher Yacoub, Camille Depenveiller, Atsushi Tatsuma, Tin Barisin, Eugen Rusakov, Udo Göbel, Yuxu Peng, Shiqiang Deng, Yuki Kagaya, Joon Hong Park, et al. Shrec 2025: Protein surface shape retrieval including electrostatic potential. 2025.

Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016.

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19313–19322, 2022.

Cheng Zhang, Haocheng Wan, Xinyi Shen, and Zizhao Wu. Pvt: Point-voxel transformer for point cloud learning. *International Journal of Intelligent Systems*, 37(12):11985–12008, 2022a.

Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training. *Advances in neural information processing systems*, 35:27061–27074, 2022b.

Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by clip. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8552–8562, 2022c.

Zhiyuan Zhang, Binh-Son Hua, David W Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *2019 International conference on 3d vision (3DV)*, pages 204–213. IEEE, 2019.

Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Riconv++: Effective rotation invariant convolutions for 3d point clouds deep learning. *International Journal of Computer Vision*, 130(5):1228–1243, 2022d.

Zhiyuan Zhang, Licheng Yang, and Zhiyu Xiang. Risurconv: Rotation invariant surface attention-augmented convolutions for 3d point cloud classification and segmentation. In *European Conference on Computer Vision*, pages 93–109. Springer, 2024.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021.

Appendix A. Details on datasets and training setup

ModelNet40 (Wu et al., 2015) contains 12,311 mesh-objects of man-made objects such as tables, chairs, cars, airplanes, etc. It is divided in 40 categories and comes with train/test split of 9,843/2,468. All objects within the same class are aligned in the same pose, which naturally favors rotation-dependent methods.

MCB-B (Kim et al., 2020a) is a variant of Mechanical Components Benchmark (MCB) which contains 18,038 mechanical components divided in 25 classes. To name a few: classes include gears, handles, springs, discs, plates, etc. This dataset is collected from public CAD repositories and hence comes as a pose non-aligned dataset. Theoretically, pose can be arbitrary but usually CAD designers construct the objects in one of the canonical poses.

ScanObjectNN (Uy et al., 2019) contains 2,902 unique object instances from 15 categories extracted from real-world 3D scans of 700 unique indoor scenes. It is more challenging than mesh/CAD-based datasets due to the presence of the background noise, non-uniform sampling and reconstruction errors from scanning. Evaluation is usually performed on three variants of the dataset OBJ-BG, OBJ-ONLY, and PB-T50-RS. OBJ-ONLY consists of "clean" segmented objects extracted from the scenes. OBJ-BG covers under-segmentation situation, i.e. objects are segmented together with background elements or noise. PB-T50-RS introduces additional perturbations of the objects such as partiality by shifting the bounding box, rotation and scaling.

Protein Data (Yacoub et al., 2025) contains 11,565 protein surfaces with calculated electrostatic potential divided into 97 unbalanced classes. A class is defined by a protein with different conformations (non-rigid deformations of protein surfaces). The training set included 9,244 protein surfaces with their corresponding ground truth. The test set included 2,311 protein surfaces without ground truth. As test set labels were not available, we used 80/20 split of the training set for train-test evaluation.

ShapeNetPart (Yi et al., 2016) is a point cloud segmentation benchmark. It contains 16,881 shapes from 16 classes with 50 part labels. Classes are man-made objects such as bag, car, guitar, laptop, etc. Each class has between 2 and 6 parts.

If not otherwise specified: we trained all our models using Adam optimizer with a default learning rate 0.001 and step learning rate decay of 0.7 every 20 epochs on a single GPU NVIDIA RTX 3500 Ada. For PointMLP Elite we use SGD optimizer as suggested in the original paper. For benchmarking and ablation study we trained for 150 epochs. Applying rotation augmentations to the training data results in a slower convergence and requires more training iterations. For this reason, we double the number of training epochs to 300. Batch size is always set to be between 8 and 32, depending on the dataset.

Appendix B. Details on RI Baseline

Here, we give a brief description of the RI baseline model. As shown in Figure 3, one rotation-invariant layer of network consists of several steps:

1. FPS (furthest point sampling) algorithm to subsample the point cloud,
2. K-nearest neighbour algorithm to create local neighbourhoods,
3. rotation-invariant feature extraction for every point in the local neighbourhood,
4. MLP projector of RI features to higher dimension,
5. combination with the features from the previous layer (if any) with MLP projection,
6. and finally max pooling over local neighbourhood.

Appendix C. Construction of the loss function

In analogy to f_{joint}, the classification heads for both backbones are defined as $f_{RB} : \mathbb{R}^{d_{RB}} \to \mathbb{R}^{d_{cls}}$ and $f_{RI} : \mathbb{R}^{d_{RI}} \to \mathbb{R}^{d_{cls}}$, respectively. The additional classification heads f_{RB} and f_{RI} takes their respective feature vectors $\mathbf{y}_{RB}$ and $\mathbf{y}_{RI}$ as inputs and map them to the class vectors $\mathbf{c}_{RB} \in \mathbb{R}^{d_{cls}}$ and $\mathbf{c}_{RI} \in \mathbb{R}^{d_{cls}}$, respectively. All three heads (f_{joint}, f_{RB}, f_{RI}) are desired to estimate the same given class vector $\mathbf{c}$:

$$\mathcal{L}_{joint}(\mathbf{c}, \mathbf{X}) = \mathcal{L}_{ce}\Big(\mathbf{c},\, f_{joint}\big(\Phi_{RB}(\mathbf{X}), \Phi_{RI}(\mathbf{X})\big)\Big),$$

$$\mathcal{L}_{RB}(\mathbf{c}, \mathbf{X}) = \mathcal{L}_{ce}\Big(\mathbf{c},\, f_{RB}\big(\Phi_{RB}(\mathbf{X})\big)\Big),$$

$$\mathcal{L}_{RI}(\mathbf{c}, \mathbf{X}) = \mathcal{L}_{ce}\Big(\mathbf{c},\, f_{RI}\big(\Phi_{RI}(\mathbf{X})\big)\Big).$$

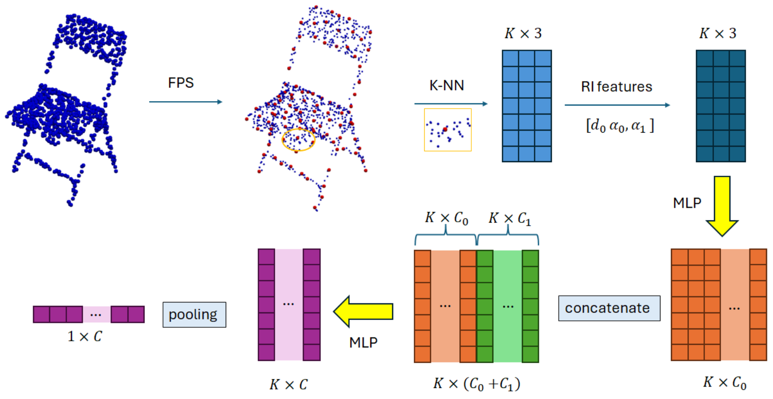

Figure 3: Rotation-invariant layer in rotation-invariant network. $K \times C_1$ feature matrix is forwarded from the previous layer (if there is any).

Appendix D. Additional ablation study

D.1. Loss function design

Table 6 shows modifications in training loss from eq. (1). The results support the hypothesis that classification performance benefits from both additional optimization criteria $\mathcal{L}_{RI}$ and $\mathcal{L}_{RB}$ used to train our model.

Loss function	ModelNet40
$\mathcal{L}_{joint}(\mathbf{c}, \mathbf{X})$	93.32%
$\mathcal{L}_{joint}(\mathbf{c}, \mathbf{X}) + \frac{1}{2}\mathcal{L}_{RB}(\mathbf{c}, \mathbf{X}) + \frac{1}{2}\mathcal{L}_{RI}(\mathbf{c}, \mathbf{X})$	93.57%
$\frac{1}{2}\mathcal{L}_{joint}(\mathbf{c}, \mathbf{X}) + \mathcal{L}_{RB}(\mathbf{c}, \mathbf{X}) + \mathcal{L}_{RI}(\mathbf{c}, \mathbf{X})$	93.41%
$\mathcal{L}_{joint}(\mathbf{c}, \mathbf{X}) + \mathcal{L}_{RB}(\mathbf{c}, \mathbf{X}) + \mathcal{L}_{RI}(\mathbf{c}, \mathbf{X})$	93.64%

Table 6: Ablation study: variations on the design of loss function from eq. (1).

D.2. Linear vs two-layer classification heads

Table 7 shows that increasing the depth of Φ_{RI} and Φ_{RB} has a negative effect on the learning process.

MLP head	ModelNet40
Linear Φ_{RI} / Φ_{RB}	93.64%
Two layer Φ_{RI} / Φ_{RB}	93.32%

Table 7: Using deeper classification heads for Φ_{RI} / Φ_{RB}.

D.3. Feature fusion

In the paper, we used the concatenation of two vectors as a feature fusion method. Here, we explore whether summation of feature vectors and cross-attention of feature vectors are as efficient as simple concatenation. Results are given in Table 8.

Feature fusion	$\emptyset$	$SO(3)$
concatenate	93.64%	50.90%
sum	93.24%	56.43%
cross-attention	92.31%	54.57%

Table 8: Feature fusion strategies: how to combine rotation invariant and rotation dependent features?

Appendix E. Additional experiments: robustness to rotations

Table 9 extends the experiments from Section 4.2.3 to PB-T50-RS dataset from ScanObjectNN (Uy et al., 2019).

Train augmentation: Test augmentation:	$\emptyset$ $\emptyset/SO(3)$	$SO(3)$ $\emptyset/SO(3)$
PointBERT (pretrained)	81.32%/24.51%	66.71%/65.52%
Composite (w PointBERT)	83.87%/57.52% (+2.55%/+33.01%)	78.97%/78.70% (+12.26%/+13.18%)

Table 9: Additional experiments on rotation robustness. Evaluation is done on PB-T50-RS test set under two conditions: no rotation augmentation ($\emptyset$ in Test) and random $SO(3)$ rotation ($SO(3)$ in Test).