Invariant Training 2D-3D Joint Hard Samples for Few-Shot Point Cloud Recognition

Anonymous ICCV submission

Paper ID 5891

Abstract

We tackle the data scarcity challenge in few-shot point cloud recognition of 3D objects by using a joint prediction from a conventional 3D model and a well-trained 2D model. Surprisingly, such an ensemble, though seems trivial, has hardly been shown effective in recent 2D-3D models. We find out the crux is the less effective training for the "joint hard samples", which have high confidence prediction on different wrong labels, implying that the 2D and 3D models do not collaborate well. To this end, our proposed invariant training strategy, called INVJOINT*, does not only emphasize the training more on the hard samples, but also seeks the invariance between the conflicting 2D and 3D ambiguous predictions.* INVJOINT *can learn more collaborative 2D and 3D representations for better ensemble. Extensive experiments on 3D shape classification with widely adopted ModelNet10/40, ScanObjectNN and Toys4K, and shape retrieval with ShapeNet-Core validate the superiority of our* INVJOINT*. Codes are in Appendix.*

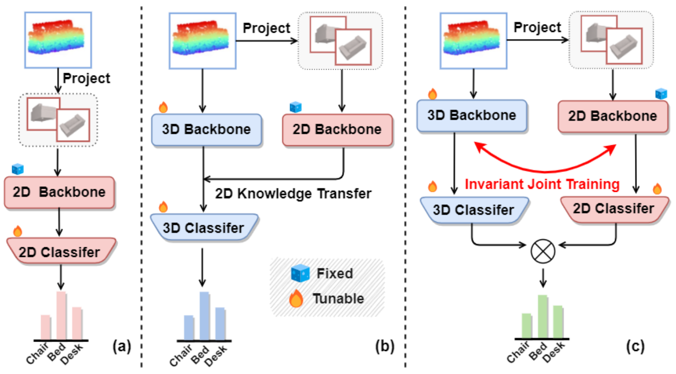

Figure 1. Comparisons of our framework with existing 2D-3D methods, which can be categorized into (a) Directly projecting point cloud into multi-view images as inputs, and then fine-tuning the 2D models with a frozen backbone. (b) Indirectly leveraging the 2D pretrained knowledge as a constraint or supervision, transferring them via knowledge distillation or contrastive learning, and then only using the optimized 3D pathway for prediction. (c) In contrast, our INVJOINT, based on ensemble paradigm, makes the best of the 2D and 3D worlds by joint prediction in inference.

1. Introduction

As the point cloud representation of a 3D object is sparse, irregularly distributed, and unstructured, a deep recognition model requires much more training data than the 2D counterpart [15, 19]. Not surprisingly, this makes few-shot learning even more challenging, such as recognizing a few newly-collected objects in AR/VR display [20] and robotic navigation [1]. Thanks to the recent progress of large-scale pre-trained multi-modal foundation models [37, 24, 31], the field of 2D few-shot or zero-shot recognition has experienced significant improvements. Therefore, as shown in Figure 1(a), a straightforward solution for 3D few-shot recognition is to project a point cloud into a set of multi-view 2D images [10], through rendering and polishing [49], and then directly fed the images into a well-trained 2D model [61].

Although effective, the projected images are inevitably subject to incomplete geometric information and rendering artifacts. To this end, as shown in Figure 1(b), another popular solution attempts to take the advantage of both 2D and 3D by transferring the 2D backbone to the 3D counterpart via knowledge distillation [58], and then use the 3D pathway for final recognition. So far, you may ask: as the data in few-shot learning is already scarce, during inference time, why do we still have to choose one domain or the other? Isn't it common sense to combine them for better prediction? In fact, perhaps for the same reason, the community avoids answering the question—our experiments (Section 4) show that a naive ensemble, no matter with early or late fusion, is far from being effective as it only brings marginal improvement.

To find out the crux, let's think about in what cases, the ensemble can correct the individually wrong predictions by joint prediction, *e.g.*, if the ground-truth is "Bench" and

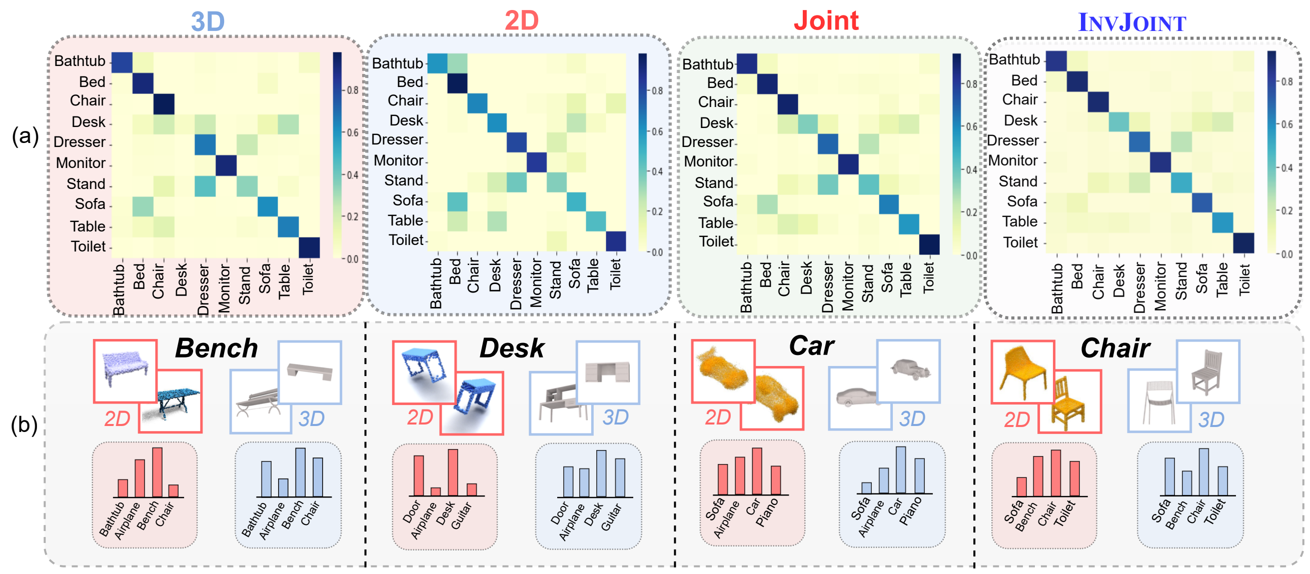

Figure 2. (a) 3D and 2D models are confused by different classes, thus a simple late fusion cannot turn the joint confusion matrix more diagonal. (b) Qualitative examples of **joint hard samples** with their logits distribution.

neither 2D nor 3D considers "Bench" as the highest confidence, however, their joint prediction peaks at "Bench". The cases are: 1) the ground-truth confidence of the two models cannot be too low, and 2) that of the other classes cannot be too high. In one word, 2D and 3D are collaborative. However, as shown by the class confusion matrices of training samples in Figure 2(a), since 2D and 3D are confused by different classes, their ensemble can never turn the matrix into a more diagonal one. This implies that their joint prediction in inference may be still wrong.

Therefore, the key is to make the joint confusion matrix more diagonal than each one. To this end, we focus on the **joint hard samples**, which have high confidence prediction on different wrong labels respectively. See Figure 2(b) for some qualitative examples, exhibiting a stark difference in logits distribution among modalities. However, simply re-training them like the conventional hard negative mining [44, 41] is less effective because the joint training is easily biased towards the "shortcut" hard samples in one domain. For example, if the 3D model has a larger training loss than 2D, probably due to a larger sample variance [64], which is particularly often in few-shot learning, the joint training will only take care of 3D, leaving 2D still or even more confused. Due to limited space, the probability theory perspective of **joint hard samples** with qualitative analysis is given in *Appendix Part B*.

By consolidating the idea of making use of joint hard examples for improving few-shot point cloud recognition, we propose an **invariant training** strategy. As illustrated in Figure 3, if a sample ground-truth is "Bench" and 2D prediction is confused between "Bench" and "Chair", while the 3D counterpart is uncertain about "Bench" and "Airplane", the pursuit of invariance will remove the variant "Chair" and "Airplane', and eventually keep the common "Bench" in each model. Specifically, we implement the strategy as INVJOINT, which has two steps to learn more collaborative 2D and 3D representations (Section 3.2). **Step 1**: it selects those joint hard samples by firstly fitting a Gaussian Mixture Model of sample-wise loss, and then picking them according to the fused logit distribution. **Step 2**: A joint learning module focusing on the selected joint hard samples effectively capture the collaborative representation across domains through an invariant feature selector. After the INVJOINT training strategy, a simple late-fusion technique can be directly deployed for joint prediction in inference (Section 3.4). Figure 2(a) shows that the joint confusion matrix of training data is significantly improved after INVJOINT.

We conduct extensive few-shot experiments on several synthetic [53, 42, 4] and real-world [46] point cloud 3D classification datasets. INVJOINT gains substantial improvements over existing SOTAs. Specifically, on ModelNet40, it achieves an absolute improvements of *5.85%* on average and *15.89%* on 1-shot setting compared with the current SOTA: PointCLIP [61]. In addition, the ablation studies demonstrate the component-wise contributions of INVJOINT. In summary, we make three-fold contributions:

- We propose INVJOINT that aims to make the best of the 2D and 3D worlds. To the best of our knowledge, it is the first work that makes 2D-3D ensemble work in point cloud 3D few-shot recognition.
- We attribute the ineffective 2D-3D ensemble to the *"joint hard samples"*. INVJOINT exploits their 2D-3D conflicts to remove the ambiguous predictions.

- InvJoint is a plug-and-play training module whose potential could be further unleashed with the evolving backbone networks.

2. Related Work

Point cloud is the prevailing representation of 3D world. The community has proposed various deep neural networks for point clouds, including convolution-based [56, 23, 25, 7], graph-based [48, 63, 26], MLP-based [34, 35, 32, 36], and the recently introduced Transformer-based [62, 60, 11]. Despite the fast progress, the performance of these models is limited due to the lack of a properly pre-trained backbone for effective feature representation. To this end, three main directions are explored: 1) intra-modality unsupervised representation learning, 2) project-and-play by 2D networks, 3) 2D-to-3D knowledge transfer.

Point Cloud Unsupervised Feature Learning: The early work PointContrast [54] establishes the correspondence between points from different camera views and performs point-to-point contrastive learning in the pre-training stage. Besides contrastive learning, data reconstruction is also explored. OcCo [47] learns point cloud representation by developing an autoencoder to reconstruct the scene from occluded inputs. However, they generalize poorly to downstream tasks due to the relatively small pre-training datasets.

Project-and-Play by 2D Networks: The most straightforward way to make use of 2D networks for 3D point cloud understanding is to transfer point clouds into 2D images. Pioneered by MVCNN [43] that uses multi-view images rendered from pre-defined camera poses and produces global shape signature by performing cross-view max-pooling, the follow-up works are mainly devoted to more sophisticated view aggregation techniques [13]. However, the 2D projection inevitably loses 3D structure and thus leads to sub-optimal 3D recognition.

2D-to-3D Knowledge Transfer: It transfers knowledge from a well-pretrained 2D image network to improve the quality of 3D representation via cross-modality learning. Given point clouds and images captured in the same scene, PPKT [28] first projects 3D points into images to establish the correspondence, and then performs cross-modality contrastive learning in a pixel-to-point manner. CrossPoint [2] proposes a self-supervised joint learning framework that boosts feature learning of point clouds by enforcing both intra- and inter-modality correspondences. The most related work to ours is PointCLIP [61], which exploits an off-the-shelf image visual encoder pretrained by CLIP [37] to address the problem of few-shot point cloud classification. Different from PointCLIP which directly fine-tunes 2D models for inference, our proposed InvJoint significantly improves 2D-3D joint prediction by invariant training on joint hard samples.

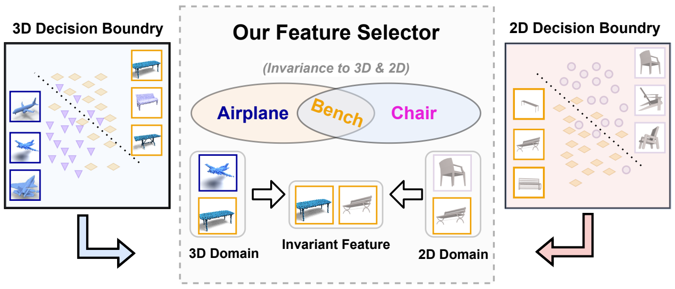

Figure 3. Illustration of the invariant training idea. Given the predictions of a "Bench" sample in both domains, the invariance selector removes the conflict confusion ("Chair" and "Airplane") and keeps the common "Bench".

3. InvJoint

InvJoint is an invariant training strategy that selects and then trains 2D-3D joint hard samples for few-shot point cloud recognition by using 2D-3D joint prediction. The overview of InvJoint is illustrated in Figure 4. Given 3D point clouds, we first perform image rendering to produce a set of 3D-projected multi-view images as the corresponding 2D input. Then, the point clouds and multi-view images are respectively fed into the 3D and 2D branches for modality-specific feature encoding (Section 3.1). Next, we select joint hard samples and feed them into the invariant learning module for better collaborative 2D-3D features (Section 3.2). At the inference stage, we introduce a simple fusion strategy for joint prediction (Section 3.4).

3.1. Multi-modality Feature Encoding

Point Cloud Feature Encoder: In our 3D branch, we extract the geometric features from point clouds input with the widely adopted DGCNN [48]. Then a trainable projection layer is applied for feature dimension alignment with the 2D feature introduced later. We denote the encoder and its trainable parameters as $E_{3\text{D}}$, and its output feature as $\mathbf{x}_3$.

Image Feature Encoder: Since the paired point clouds and images are not always available, to improve the applicability of our method, we adopt the differentiable rendering technique to generate photo-realistic images for the 2D views. Specifically, an alpha compositing renderer [52] is deployed with trainable parameters of cameras for optimized recognition. After obtaining the rendered multi-view images, we feed them into the frozen CLIP [37] visual encoder (*i.e.*, the pre-trained ViT-B model) with an additional trainable linear adapter [9] to narrow the gap between the rendered images and the original CLIP images. See *Appendix* for more implementation details. We denote the image encoder as $E_{2\text{D}}$ and its output feature as $\mathbf{x}_2$.

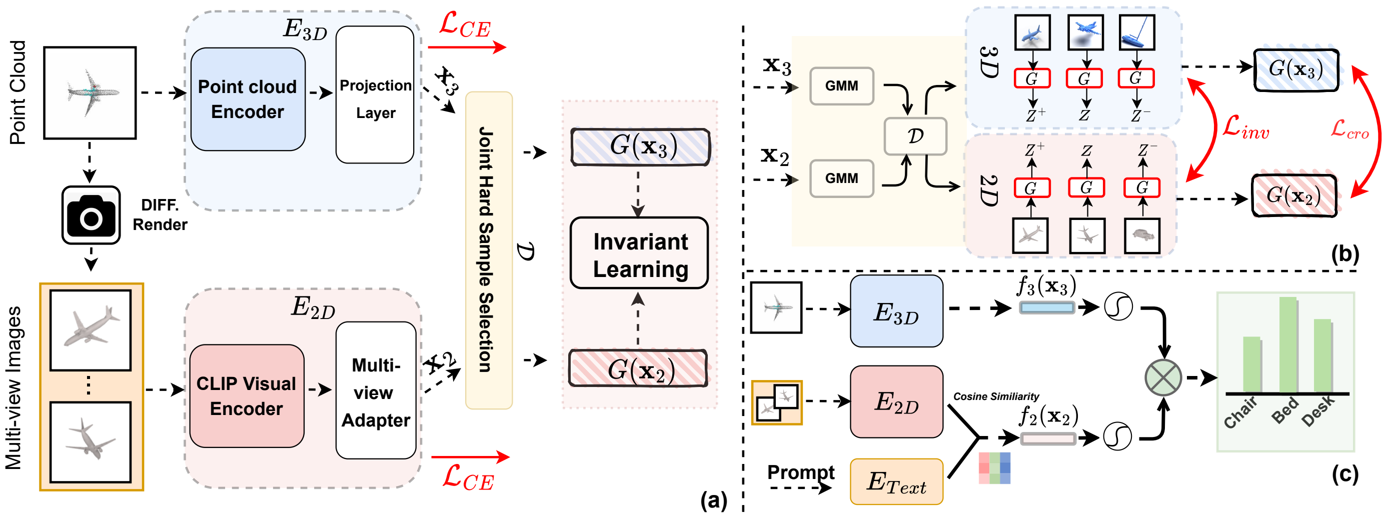

Figure 4. (a) The training pipeline of InvJoint. E_{3D}, E_{2D} (including the renderer), and G are trainable parameters. (b) The zoom-in diagram of Invariant Learning. Note that $\mathcal{L}_{\text{inv}}$ only trains G. (c) The inference pipeline, where ⟲ denotes the Softmax layer.

3.2. Invariant Joint Training

As we discussed in Section 1, due to the feature gap between $\mathbf{x}_2$ and $\mathbf{x}_3$, there are joint hard samples that prevent the model from learning collaborative 2D-3D features. As illustrated in Figure 4(b), our invariant joint training contains the following two steps:

Step 1: Joint Hard Sample Selection. We first conduct Hard Example Mining (**HEM**) in each modality, then combining them subject to a joint prediction threshold.

In a certain modality, one is considered as hard sample if its training loss is larger [16] than a pre-defined threshold. In particular, we adopt a two-component Gaussian Mixture Model (GMM) [57] $P(g \mid \mathcal{L}_{CE})$ to normalize the per-sample cross-entropy (CE) loss distribution for each modality respectively: $\mathcal{D}_{i\in\{2,3\}} = \{\mathbf{x}_i \mid P\left(g \mid \mathcal{L}_{CE}(\mathbf{x}_i)\right) < p_i\}$, where g is the Gaussian component with smaller mean, p_i is a probability threshold. Please refer to *Appendix* for further ablation of other HEM algorithms. Subsequently, the joint hard samples are chosen based on the criterion that the sum of 2D and 3D logits in most likely non-ground-truth class is large: $\mathcal{D} = \{(\mathbf{x}_2 \in \mathcal{D}_2, \mathbf{x}_3 \in \mathcal{D}_3) \mid \max_{i\neq gt} f_2^i(\mathbf{x}_2) + f_3^i(\mathbf{x}_3) > r\}$, where $f^i(\boldsymbol{x})$ denotes the logits output of the i-th class and r is a threshold parameter.

Step 2: Cross-modal Invariance Learning. The goal of this step is to acquire a non-conflicting feature space by reconciling the 2D-3D features of samples in $\mathcal{D}$. Meanwhile, we also don't want the purist of invariance—seeking common features—to negatively affect the complementary nature of the 2D-3D representation. Therefore, we devise a gate function G which works as a soft mask that applies element-wise multiplication to select the non-conflicting features for each modality, *e.g.*, $G(\mathbf{x}_2)$ for 2D and $G(\mathbf{x}_3)$ for 3D, and the following invariance training only tunes G while freezing the feature extractors E_{3D} and E_{2D}.

Inspired by invariant risk minimization (IRM) [3], we consider the point cloud feature $\boldsymbol{x}_3$ and the image features $\boldsymbol{x}_2$ as two environments, and propose the modality-wise IRM. Due to IRM essentially regularizes the model to be equally optimal across environments, *i.e.*, modalities, we can learn a unified gate function G to select non-conflicting features for each modality by:

$$\begin{aligned}&\min_{G} \sum_{\mathbf{x_e}\in\{\mathbf{x}_2,\mathbf{x}_3\}} R_e(\mathbf{x}_e, y; G) \\ &\text{s.t. } G \in \arg\min_{G} R_e(\mathbf{x}_e, y; G), \forall \mathbf{x}_e \in \{\mathbf{x}_2, \mathbf{x}_3\},\end{aligned} \tag{1}$$

where $R_e(\mathbf{x}_e, y; G)$ is the empirical risk under e, G denotes a learnable mask layer multiplied on $\mathbf{x}_e$.

Particularly, we implement $R_e(\cdot)$ as our modality-wise IRM by a contrastive loss:

$$\mathcal{L}_e(\boldsymbol{z}_e, \theta) = -\log \frac{\exp\left(\boldsymbol{z}_e^{\mathrm{T}} \boldsymbol{z}_e^{+} \cdot \theta\right)}{\exp\left(\boldsymbol{z}_e^{\mathrm{T}} \boldsymbol{z}_e^{+} \cdot \theta\right) + \sum_{\boldsymbol{z}_e^-} \exp\left(\boldsymbol{z}_e^{\mathrm{T}} \boldsymbol{z}_e^{-} \cdot \theta\right)}, \tag{2}$$

where $z_e = G(\boldsymbol{x}_e)$, which is a element-wise product; we take the augmented $\mathbf{x}_e$ in the same class as positive $\boldsymbol{z}_e^+$, while the representation of other classes as negative $\boldsymbol{z}_e^-$ in both modalities respectively. Note that we follow the common practice to utilize regular spatial transformations, *e.g.*, rotation, scaling and jittering as the augmented point cloud in $\mathbf{x}_3$; we consider the different rendering views as the augmented images in $\mathbf{x}_2$. In this way, G is optimized through the proposed modality-wise IRM loss:

$$\mathcal{L}_{\text{inv}} = \sum_{\mathbf{x}_e\in\{\mathbf{x}_2,\mathbf{x}_3\}} \mathcal{L}_e\left(G(\mathbf{x}_e), \theta\right) + \lambda \left\| \nabla_{\theta=1} \mathcal{L}_e\left(G(\mathbf{x}_e), \theta\right) \right\|_2^2, \tag{3}$$

where λ is a trade-off hyper-parameter; θ is a dummy classifier to calculate the gradient penalty across modality, which encourages G to select the non-conflicting features.

3.3. Overall Loss

During the training stage, we formulate the overall training objective as:

$$\min_{G,E_{\text{2D}},E_{\text{3D}}} \mathcal{L}_{CE}(E_{\text{2D}},E_{\text{3D}}) + \mathcal{L}_{\text{inv}}(G) + \mathcal{L}_{\text{align}}(E_{\text{2D}},E_{\text{3D}}), \tag{4}$$

where $\mathcal{L}_{CE}$ is the standard cross-entropy loss, $\mathcal{L}_{\text{inv}}$ is the modality-wise IRM loss to optimize gate function G, and $\mathcal{L}_{\text{align}}$ is defined as follow:

Cross-modality Alignment Loss. After the gate function G filters the non-conflicting features, the multi-modality encoders E_{3D} & E_{2D} are eventually regularized in collaborative feature space by the cross-modality NT-Xent loss [5] for further alignment:

$$\mathcal{L}_{align} = -\log \frac{\exp\left(\boldsymbol{z}^{\mathrm{T}}\boldsymbol{z}^{+}\cdot\tau\right)}{\exp\left(\boldsymbol{z}^{\mathrm{T}}\boldsymbol{z}^{+}\cdot\tau\right) + \sum_{\boldsymbol{z}^{-}}\exp\left(\boldsymbol{z}^{\mathrm{T}}\boldsymbol{z}^{-}\cdot\tau\right)}, \tag{5}$$

where we use $\boldsymbol{z} = G(\mathbf{x}_2)$ and $\boldsymbol{z}^{+} = G(\mathbf{x}_3)$ for brevity; τ is a temperature parameter. The objective is to maximize the cosine similarity of $G(\mathbf{x}_2)$ and $G(\mathbf{x}_3)$, which are the 3D/2D non-conflicting feature of the same sample, while minimizing the similarity with all the others in the feature space for modality alignment. Note that each loss optimizes different set of parameters — the feature encoder E_{2D} and E_{3D} is frozen when IRM penalty updates; the gate function G is only optimized by the modality-wise IRM loss.

3.4. Joint Inference

We devise a simple multi-modality knowledge fusion strategy for joint prediction in inference. In Figure 4(c), the 3D branch E_{3D} takes point clouds as input to predict classification logits $f_3(\boldsymbol{x}_3)$, and the 2D branch E_{2D} takes multi-view images as input to produce a visual feature embedding $\mathbf{x}_2$ for each of them. To make the best of our 2D branch that initialized with the CLIP model, we follow pretext tasks of CLIP pretraining to use the cosine similarity of image-text pairs for logits computation. Specifically, we get the textual embedding $\mathbf{x}_{\text{text}}$ by placing category names into a pre-defined text template, *e.g.*, *"point cloud of a big [CLASS]"* and feeding the filled template to the textual encoder of CLIP model. The image-text similarity for the i-th rendered image is computed as $\frac{\mathbf{x}_{\text{text}}^{\top}\cdot\mathbf{x}_2^i}{\|\mathbf{x}_{\text{text}}\|\left\|\mathbf{x}_2^i\right\|}$. Once we obtain the cosine similarity of each rendered image, we average them to get the classification logits $f_2(\boldsymbol{x}_2)$ from 2D branches. After that, the fused prediction is computed as

$$f_{ens} = \text{Softmax}(f_2(\boldsymbol{x}_2)) \cdot \text{Softmax}(\varphi(f_3(\boldsymbol{x}_3)), \tag{6}$$

where the softmax function is leveraged to normalize the weight; $\varphi(\cdot)$ is served as a non-linear modulator to control the sharpness of 3D logits distribution, denoted as $\varphi(x) = \exp(-\beta\cdot(1-x))$, where β is the post logit-adjustment described in *Appendix*. Through such simple logits fusion, P_{ens} can effectively fuse the prior multi-modal knowledge and ameliorate few-shot point cloud classification.

4. Experiments

4.1. Implementation Details

Image Rendering. We exploited a differentiable point cloud renderer (*i.e.*, the alpha compositing renderer [52]) in our pipeline. It uses learnable parameters $\boldsymbol{r} = \{\rho,\theta,\phi\}$ to indicate the camera's pose and position, where ρ is the distance to the rendered object, θ is the azimuth, and ϕ is the elevation. Other than the parameter $\boldsymbol{r}$, the light pointing is fixed towards the object center and the background is set as pure color. We further resized the rendered images to 224×224, and colored them by the values of their normal vectors or kept them white if the information of normal is not available.

Network Architectures. We adopted ViT-B/16 [8] and textual transformers pretrained with CLIP [37] as our 2D visual encoder and textual encoder, respectively. Their parameters were frozen throughout our training stage. Following the practice in [61], we set the handcraft language expression template as " 3D rendered photo of a [CLASS]" for textual encoding. As for 3D backbones, for fair comparison with other methods, we exploited the widely-adopted DGCNN [48] point cloud feature encoder as E_{3D}.

Training Setup. InvJoint was end-to-end optimized at the training stage. For each point cloud input, we sampled 1,024 points via Farthest Point Sampling [33], and applied standard data augmentations, including rotation, scaling, jittering, and color auto-contrast. For rendered images, we only applied center-crop as the data augmentation, since the background is purely white. InvJoint was trained for 50 epochs with a batch size of 32. We adopted SGD as the optimizer [29], and set the weight decay to 10^{-4} and the learning rate to 0.01. Cosine annealing [30] was employed as the learning rate scheduler. All of our experiments were conducted on 4 NVIDIA Tesla A100 GPUs.

4.2. Few-shot Object Classification

Dataset and Settings. We compared our InvJoint with other state-of-the-art models on four datasets: ModelNet10 [53], ModelNet40 [53], ScanObjectNN [46] and Toys4K [42]. ModelNet40 is a synthetic CAD dataset, containing 12,331 objects from 40 categories, where the point clouds are obtained by sampling the 3D CAD models. ModelNet10 is the core indoor subset of ModelNet40, containing approximately 5k different shapes of 10 categories. ScanObjectNN is a more challenging real-world dataset, composed of 2,902 objects selected from ScanNet [6] and

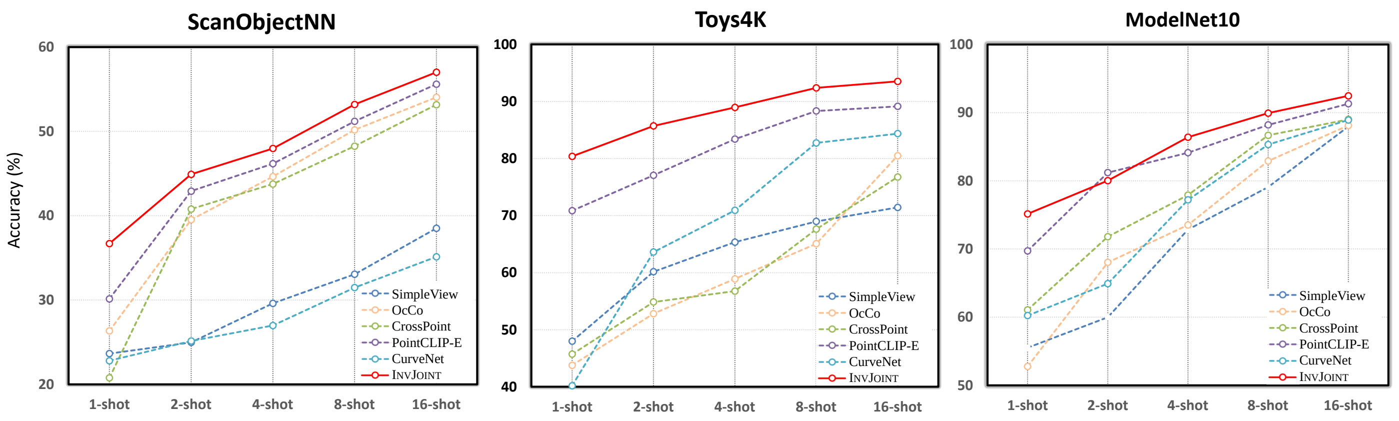

Figure 5. Few-shot performance comparisons between InvJoint and other methods, including the state-of-the-art PointCLIP-E (denotes PointCLIP with simple late fusion), on ModelNet10, ScanObjectNN and Toys4K. Our InvJoint shows consistent superiority to other models under 1, 2, 4, 8, and 16-shot settings.

Table 1. Few-shot performance on ModelNet40 with several 2D/3D state-of-the-art methods. PointCLIP-E denotes the ensemble of PointCLIP and DGCNN.

Category	Method	1-shot	2-shot	4-shot	8-shot	16-shot
2D	PointCLIP [61]	52.96	66.73	74.47	80.96	85.45
	SimpleView [10]	26.42	35.14	58.53	69.20	78.55
3D	OcCo [47]	46.92	54.08	60.15	72.98	75.08
	cTree [40]	15.13	24.98	27.90	34.12	50.59
	Jigsaw [38]	11.24	20.98	25.76	31.89	46.85
Joint	Crosspoint [2]	48.24	59.95	64.25	75.75	79.70
	Shape-FEAT [42]	37.78	49.92	54.10	61.98	70.75
	PointCLIP-E	53.70	67.14	76.32	80.82	85.90
Ours	InvJoint	**68.85**	**70.24**	**78.95**	**82.85**	**88.94**

SceneNN [18], where the point cloud shapes are categorized into 15 classes. Toys4K [42] is a recently collected dataset specially designed for low-shot learning with 4,179 objects across 105 categories.

We followed the settings in [61] to conduct few-shot classification experiments: for K-shot settings, we randomly sampled K point clouds from each category in the training dataset. Our point cloud encoder E_{3D}, as well as the other point-based methods in few-shot settings were all pre-trained on ShapeNet [4], which originally consists of more than 50,000 CAD models from 55 categories.

Performance Comparison. Table 1 reports the few-shot classification performance on ModelNet40 dataset. Several state-of-the art methods, including image-based, point-based and multi-modality-based ones, are compared. Note that post-search in PointCLIP [61] is not leveraged for a fair comparison. Our InvJoint achieves inspiring performance, outperforming all other methods by a large margin. Remarkably, InvJoint achieves an absolute improvements of *5.85 %* on average against PointCLIP [61]. The superiority of our method becomes more obvious when it comes to harder conditions with fewer samples. For example, in 1-shot settings, InvJoint outperforms PointCLIP [61] and Crosspoint [2] by *15.89 %* and *20.61 %* respectively. Besides, we can observe from Table 1 that simply ensembling PointCLIP [61] and DGCNN [48] couldn't provide enough enhancement, which demonstrates that the conventional ensemble strategy cannot work well without tackling the **"joint hard samples"**.

Figure 5 depicts the results in the other three datasets. Not surprisingly, InvJoint consistently outperforms other methods across datasets and in most settings, further demonstrating the robustness of our method. Particularly, in the recently collected benchmark Toys4K with the largest number of object categories, InvJoint shows an overwhelming performance, *i.e.*, *93.52 %* accuracy with 16-shots, while most 3D models achieve really low performance due to their poor generalization ability.

4.3. Other Downstream Tasks

Besides few-shot object classification, we also deployed InvJoint in the following downstream tasks to show its more collaborative 2D-3D features.

Dataset and Settings. We followed the settings in [13] to provide the empirical results of 3D shape retrievals task on ModelNet40 [53] and ShapeNet Core55 [39]. Furthermore, we also experienced InvJoint on ModelNet40 and ModelNet40-C for many-shot object classification. ModelNet40-C [45] is a comprehensive dataset with 15 corruption types and 5 severity levels to benchmark the corruption robustness of 3D point cloud recognition. Note that in all the three following downstream tasks, our point cloud encoder E_{3D} is trained *from scratch* for a fair comparison.

(i) Shape Retrieval. Table 3 presents the performance comparison with some recently introduced image-based and point-based methods in terms of mean average precision

Table 2. Object classification results on ModelNet40 and Modelnet40-C. "Corr Err" and "Clean Err" denote the error rate on ModelNet40-C and ModelNet40, respectively.

Methods	Augmentation	Corr Err	Clean Err
PCT [11]	PointCutMix-K	16.5	**6.9**
	PointCutMix-R	16.3	7.2
DGCNN [48]	RSMix	18.1	7.1
	PointCutMix-R	17.3	7.4
PointNet++ [35]	PointCutMix-R	19.1	7.1
	PointMixup	19.3	7.1
SimpleView [10]	PointCutMix-R	19.7	7.9
RSCNN [17]	PointCutMix-R	17.9	7.6
INVJOINT (DGCNN)	RSMix	16.8 (1.3 ↓)	**6.9** (0.2 ↓)
INVJOINT (PointNet++)	PointCutMix-R	17.6 (1.5 ↓)	7.0 (0.1 ↓)
INVJOINT (PCT)	PointCutMix-K	**15.9** (0.6 ↓)	**6.9** (0.3 ↓)

Table 3. 3D Shape Retrieval. We compare the performance (mAP) of INVJOINT on ModelNet40 and ShapeNet Core55. INVJOINT achieves the best retrieval performance among recent state-of-the-art methods on both datasets.

Methods	Data Type	ModelNet40	ShapeNet Core
PVNet [59]	Points	89.5	-
Densepoint [27]	Points	88.5	-
RotNet [22]	20 Views	-	77.2
MLVCNN [21]	24 Views	92.9	
MVCNN [13]	12 Views	80.2	73.5
MVTN [13]	12 Views	92.2	82.9
ViewGCN [51]	20 Views	-	78.4
VointNet [14]	12 Views	-	83.3
INVJOINT	10 Views	**93.7**	**84.1**

(mAP) for the shape retrieval task. Note that some methods in Table 3 are designed specifically for retrieval, *e.g.*, MLVCNN [21]. Surprisingly, INVJOINT improves the retrieval performance by a large margin in ShapeNet core with only 10 Views of rendered images. INVJOINT also demonstrates state-of-the-art results (*93.7 %* mAP) on ModelNet40. Figure 6 shows some qualitative retrieval examples.

(ii) Many-shot Object Classification. Although our proposed INVJOINT is mainly designed under few-shot settings, it can also achieve comparable performance with state-of-the-art methods on sufficient data. As depicted in Table 2, the performance of 3D baselines are significantly improved with lower error rate by INVJOINT. Specifically, with PCT as the encoder E_{3d} in 3D branch, we followed [45] to conduct PointCutMix-K as point cloud augmentation strategy, our INVJOINT achieve *6.9 %* and *15.9 %* error rate on ModelNet40 / ModelNet40-C respectively.

4.4. Ablation Analysis and Discussion

Q1: ***How does* INVJOINT *make the best of 2D and 3D world?*** To better diagnose the effectiveness of INVJOINT,

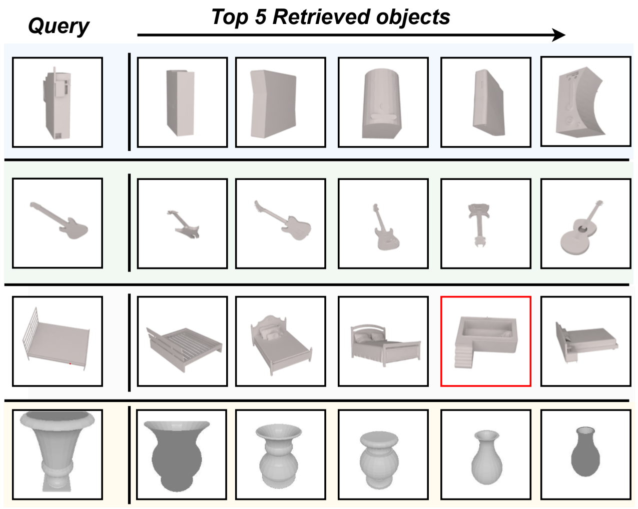

Figure 6. Qualitative Examples for 3D Shape Retrieval on ModelNet40: (*left*): Query objects from the test set. (*right*): Top 5 matches +for each query, with mistakes highlighted in red.

we first illustrate the improvements of our joint inference, comparing with each branch performance as well as the simple late fusion in Figure 7(b). Then we give the decent definition of *Conflict Ratio* C_{err} to reflect the degree of modality conflict: Given the set of test sample index as $\mathbf{T}$, we define the index of samples with correct 2D, 3D and Joint predictions as $\mathbf{T}_{\text{2D}}$, $\mathbf{T}_{\text{3D}}$ and $\mathbf{T}_{\text{Joint}}$. C_{err} is given by $\frac{||(\mathbf{T}_{\text{2D}} \setminus \mathbf{T}_{\text{Joint}}) \cup (\mathbf{T}_{\text{3D}} \setminus \mathbf{T}_{\text{Joint}})||}{||T||}$, which calculates the ratio of those can be recognized by one modality but failed in joint prediction . Under such definition, we further analyze the variation curve of C_{err} at the training stage.

A1: Specifically in Figure 7(b), the proposed INVJOINT outperforms the late fusion by *4.7 %* on average in different settings of ModelNet40, which concretely demonstrates the superiority of multi-modality collaboration through INVJOINT. It is clear from Figure 7(a) that our method gradually mitigates the modality conflict while separate training of each branch and then ensembling remains high *Conflict Ratio* C_{err}. From these two aspects, we could give a conclusion: the higher performance of INVJOINT indeed attributes to the removal of conflict and ambiguous predictions.

Q2: ***What impact performance of* INVJOINT *considering component-wise contributions?*** we removed each individual step of Invariant Joint Training and replaced the late fusion strategy to examine the component-wise and loss-wise contributions. The results are shown in Table 6.

A2: In a multi-step framework, we observed that the exclusion of any component from INVJOINT resulted in a significant decrease in performance. Specifically, upon removal of either step 1 or step 2, the Top-1 Accuracy exhibited an average degradation of *4.06 %* ↓ and *3.02 %* ↓. Similarly considering the loss functions, the Top-1 Accuracy will av-

Table 4. Performances with different visual encoders on ModelNet40. RN50 /101 denotes ResNet-50 /101, and ViT-B/32 represents vision transformer with 32 × 32 / 16 × 16 patch embeddings. Accuracy of 2D branch (*left in each cell*) and InvJoint (*right in each cell*) are reported.

Model	1-shot	2-shot	4-shot	8-shot	16-shot
RN50	59.18 | 66.05	64.12 | 68.90	68.23 | 76.41	71.10 | 81.25	77.23 | 85.93
RN101	60.19 | 66.42	**66.98** | 69.10	70.45 | 78.90	72.74 | 82.60	78.16 | 87.10
ViT/16	**63.62** | **68.85**	66.65 | **70.24**	**72.35** | **78.95**	**75.68** | **82.85**	**81.20** | **88.94**
ViT/32	61.29 | 67.34	66.08 | 69.70	70.14 | 77.62	73.14 | 81.90	80.12 | 88.32

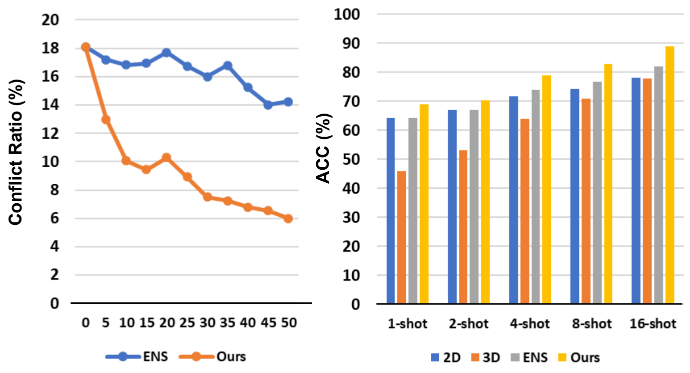

Figure 7. (a) The variation curve of the *Conflict Ratio* C_{err} on 16-shots ModelNet40, which degrades significantly with InvJoint. (b) Evaluations (Top-1 Accuracy) on ModelNet40 with different few-shot settings. Joint inference with InvJoint outperforms late fusion baseline by a large margin.

eragely degrade by *5.45 %* ↓ and *3.02 %* ↓ respectively if $\mathcal{L}_{CE}$ is adopted alone (w/o Step1 & 2) and if $\mathcal{L}_{\text{inv}}$ is not adopted (w/o Step 2). Further hyper-parameter sensitivity (*e.g.*, λ in Eq (3)) analyses are included in *Appendix*.

Q3: ***How about the robustness of* InvJoint*?*** As shown in Table 4, we compared the effect of different prompts designs for few-shot InvJoint. Moreover, we also implemented different CLIP visual backbones from ResNet [15] to ViT [8], reporting the results of individual 2D branch as well as the joint prediction of InvJoint.

A3: From Table 4 and 5, we could find out that the performance of 2D branch is directly impacted by the prompt and backbone choices to some extent. However, with the cooperative 3D-2D joint prediction, our proposed InvJoint shows its relatively strong robustness, *e.g.*, reducing the standard deviation from *10.87 %* to *5.51 %* among the different designs of prompts. More empirical analysis on different point cloud augmentation strategies as well as the choices of 3D backbones is included in *Appendix*.

Q4: ***Why Ensemble?*** One may ask why multi-modality ensembling should be regarded as an interesting contribution, since ensembling itself is a well-studied approach [50, 12] that is often viewed as an "engineering stragety" for improving leader board performance.

Table 5. Performances with different prompt designs on 16-shot Toys4K. [CLASS] denotes the class token, and [Learnable Tokens] denotes learnable prompts with fixed length.

Prompts	E_{2D}	Joint
"a photo of a [CLASS]."	88.18	92.90
"a point cloud photo of a [CLASS]."	89.32	93.18
"point cloud of a big [CLASS]."	89.71	92.95
"3D CAD model of [CLASS]."	90.10	93.33
"3D rendered photo of [CLASS]."	89.14	**93.52**
"3D object of a big [CLASS]."	**90.32**	92.98
"[Learnable Tokens] + [CLASS]"	60.76	78.57

Table 6. Effectiveness for each component on few-shot ScanObjectNN and ModelNet40. Performance of 2-shot (*left in each cell*) and 16-shot (*right in each cell*) are reported.

Step1	Step2	Fusion Type	ScanObjectNN	ModelNet40
✗	✗	$f_{2d} + f_{3d}$	39.13 | 51.90	65.67 | 79.46
✗	✗	$f_{2d} \times f_{3d}$	39.06 | 52.21	66.14 | 81.09
✗	✓	$f_{2d} + f_{3d}$	41.90 | 52.93	66.54 | 81.42
✗	✓	$f_{2d} \times f_{3d}$	39.94 | 53.45	67.29 | 83.91
✓	✗	$f_{2d} + f_{3d}$	41.60 | 52.78	65.90 | 84.71
✓	✗	$f_{2d} \times f_{3d}$	42.72 | 53.62	67.75 | 85.82
✓	✓	$f_{2d} + f_{3d}$	44.16 | 56.19	69.16 | 87.61
✓	✓	$f_{2d} \times f_{3d}$	**44.91** | **57.02**	**70.24** | **88.94**

A4: We would like to justify: **(1)** We illustrate that ensemble without conflict matters, and prior 3D-2D approaches such as knowledge distillation [58], parameter inflation [55] are not as effective as InvJoint, especially under data deficiency. **(2)** As far as we know, InvJoint is the first 3D-2D ensembling framework as a fusion method for few-shot pointcloud recognition. What we propose is neither the added 2D classifier (a necessary engineering implementation) nor the ensemble paradigm in Figure 1(c), but a joint learning algorithm to improve the ineffective 2D-3D ensemble. Though simple, it is remarkably useful and should be considered as a strong baseline for future study.

5. Conclusion

We pointed out the crux to a better 2D-3D ensemble in few-shot point cloud recognition is the effective training on "joint hard samples", which implies the conflict and ambiguous predictions between modalities. To resolve such modality conflict, we presented InvJoint, a plug-and-play training module for "joint hard samples", which seeks the invariance between modalities to learn more collaborative 3D and 2D representation. Extensive experiments on 3D few-shot recognition and shape retrieval datasets verified the effectiveness of our methods. In future, we will focus on exploring the potential of InvJoint for wider 3D applications, *e.g.*, point cloud part segmentation, object detection.

References

[1] Martin Adams, Martin David Adams, and Ebi Jose. *Robotic navigation and mapping with radar*. Artech House, 2012. 1

[2] Mohamed Afham, Isuru Dissanayake, Dinithi Dissanayake, Amaya Dharmasiri, Kanchana Thilakarathna, and Ranga Rodrigo. Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9902–9912, 2022. 3, 6

[3] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. 4

[4] 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. 2, 6

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 5

[6] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 5

[7] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1201–1209, 2021. 3

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 5, 8

[9] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *arXiv preprint arXiv:2110.04544*, 2021. 3

[10] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *International Conference on Machine Learning*, pages 3809–3820. PMLR, 2021. 1, 6, 7

[11] 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(2):187–199, 2021. 3, 7

[12] Neha Gupta, Jamie Smith, Ben Adlam, and Zelda E Mariet. Ensembles of classifiers: a bias-variance perspective. 8

[13] 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. 3, 6, 7

[14] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Voint cloud: Multi-view point cloud representation for 3d understanding. *arXiv preprint arXiv:2111.15363*, 2021. 7

[15] 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*, pages 770–778, 2016. 1, 8

[16] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. 4

[17] Linshu Hu, Mengjiao Qin, Feng Zhang, Zhenhong Du, and Renyi Liu. Rscnn: A cnn-based method to enhance low-light remote-sensing images. *Remote Sensing*, 13(1):62, 2020. 7

[18] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *2016 fourth international conference on 3D vision (3DV)*, pages 92–101. Ieee, 2016. 6

[19] Gao Huang, Zhuang Liu, Geoff Pleiss, Laurens Van Der Maaten, and Kilian Weinberger. Convolutional networks with dense connectivity. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 1

[20] Ho Jin Jang, Jun Yeob Lee, Jeonghun Kwak, Dukho Lee, Jae-Hyeung Park, Byoungho Lee, and Yong Young Noh. Progress of display performances: Ar, vr, qled, oled, and tft. *Journal of Information Display*, 20(1):1–8, 2019. 1

[21] Jianwen Jiang, Di Bao, Ziqiang Chen, Xibin Zhao, and Yue Gao. Mlvcnn: Multi-loop-view convolutional neural network for 3d shape retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8513–8520, 2019. 7

[22] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5010–5019, 2018. 7

[23] Artem Komarichev, Zichun Zhong, and Jing Hua. A-cnn: Annularly convolutional neural networks on point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7421–7430, 2019. 3

[24] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *arXiv preprint arXiv:2201.12086*, 2022. 1

[25] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018. 3

[26] Yawei Li, He Chen, Zhaopeng Cui, Radu Timofte, Marc Pollefeys, Gregory S Chirikjian, and Luc Van Gool. Towards efficient graph convolutional networks for point cloud handling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3752–3762, 2021. 3

[27] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely

contextual representation for efficient point cloud processing. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5239–5248, 2019. 7

[28] Yueh-Cheng Liu, Yu-Kai Huang, Hung-Yueh Chiang, Hung-Ting Su, Zhe-Yu Liu, Chin-Tang Chen, Ching-Yu Tseng, and Winston H Hsu. Learning from 2d: Contrastive pixel-to-point knowledge transfer for 3d pretraining. *arXiv preprint arXiv:2104.04687*, 2021. 3

[29] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 5

[30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[31] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019. 1

[32] 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. 3

[33] Carsten Moenning and Neil A Dodgson. Fast marching farthest point sampling. Technical report, University of Cambridge, Computer Laboratory, 2003. 5

[34] 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, 2017. 3

[35] 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, 2017. 3, 7

[36] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *arXiv preprint arXiv:2206.04670*, 2022. 3

[37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 1, 3, 5

[38] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. *Advances in Neural Information Processing Systems*, 32, 2019. 6

[39] Konstantinos Sfikas, Theoharis Theoharis, and Ioannis Pratikakis. Exploiting the panorama representation for convolutional neural network classification and retrieval. In *3DOR@ Eurographics*, 2017. 6

[40] Charu Sharma and Manohar Kaul. Self-supervised few-shot learning on point clouds. *Advances in Neural Information Processing Systems*, 33:7212–7221, 2020. 6

[41] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016. 2

[42] Stefan Stojanov, Anh Thai, and James M Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1798–1808, 2021. 2, 5, 6

[43] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 3

[44] Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. Stochastic class-based hard example mining for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7251–7259, 2019. 2

[45] Jiachen Sun, Qingzhao Zhang, Bhavya Kailkhura, Zhiding Yu, Chaowei Xiao, and Z Morley Mao. Benchmarking robustness of 3d point cloud recognition against common corruptions. *arXiv preprint arXiv:2201.12296*, 2022. 6, 7

[46] 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. 2, 5

[47] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. Unsupervised point cloud pre-training via occlusion completion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9782–9792, 2021. 3, 6

[48] 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. 3, 5, 6, 7

[49] Ziyi Wang, Xumin Yu, Yongming Rao, Jie Zhou, and Jiwen Lu. P2p: Tuning pre-trained image models for point cloud analysis with point-to-pixel prompting. *arXiv preprint arXiv:2208.02812*, 2022. 1

[50] Andrew Webb, Charles Reynolds, Wenlin Chen, Henry Reeve, Dan Iliescu, Mikel Lujan, and Gavin Brown. To ensemble or not ensemble: When does end-to-end training fail? In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 109–123. Springer, 2020. 8

[51] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1850–1859, 2020. 7

[52] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7467–7477, 2020. 3, 5

[53] 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. 2, 5, 6

[54] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *European conference on computer vision*, pages 574–591. Springer, 2020. 3

[55] Chenfeng Xu, Shijia Yang, Tomer Galanti, Bichen Wu, Xiangyu Yue, Bohan Zhai, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Image2point: 3d point-cloud understanding with 2d image pretrained models. In *European Conference on Computer Vision*, pages 638–656. Springer, 2022. 8

[56] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018. 3

[57] Guorong Xuan, Wei Zhang, and Peiqi Chai. Em algorithms of gaussian mixture model and hidden markov model. In *Proceedings 2001 international conference on image processing (Cat. No. 01CH37205)*, volume 1, pages 145–148. IEEE, 2001. 4

[58] Xu Yan, Heshen Zhan, Chaoda Zheng, Jiantao Gao, Ruimao Zhang, Shuguang Cui, and Zhen Li. Let images give you more: Point cloud cross-modal training for shape analysis. *arXiv preprint arXiv:2210.04208*, 2022. 1, 8

[59] Haoxuan You, Yifan Feng, Rongrong Ji, and Yue Gao. Pvnet: A joint convolutional network of point cloud and multi-view for 3d shape recognition. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1310–1318, 2018. 7

[60] 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. 3

[61] 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, 2022. 1, 2, 3, 5, 6

[62] 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. 3

[63] Haoran Zhou, Yidan Feng, Mingsheng Fang, Mingqiang Wei, Jing Qin, and Tong Lu. Adaptive graph convolution for point cloud analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4965–4974, 2021. 3

[64] Zhi-Hua Zhou. *Machine learning*. Springer Nature, 2021. 2