IMAGE2POINT: 3D POINT-CLOUD UNDERSTANDING WITH 2D IMAGE PRETRAINED MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

3D point-clouds and 2D images are different visual representations of the physical world. While human vision can understand both representations, computer vision models designed for 2D image and 3D point-cloud understanding are quite different. Our paper explores the potential for transferring between these two representations by empirically investigating the feasibility of the transfer, the benefits of the transfer, and shedding light on why the transfer works. We discovered that we can indeed use the same architecture and pretrained weights of a neural net model to understand both images and point-clouds. Specifically, we can transfer the pretrained image model to a point-cloud model by *inflating* 2D convolutional filters to 3D and then **f**inetuning the **i**mage-**p**retrained models (FIP). We discover that, surprisingly, models with minimal finetuning efforts — only on input, output, and optionally batch normalization layers, can achieve competitive performance on 3D point-cloud classification, beating a wide range of point-cloud models that adopt task-specific architectures and use a variety of tricks. When finetuning the whole model, the performance further improves significantly. Meanwhile, we also find that FIP improves data efficiency, achieving up to 10.0 points top-1 accuracy gain on few-shot classification. It also speeds up training of point-cloud models by up to 11.1x to reach a target accuracy.

1 INTRODUCTION

Point-cloud is an important visual representation for 3D computer vision. It is widely used in applications such as autonomous driving (Behley et al., 2019; Caesar et al., 2020; Yue et al., 2018), robotics (Armeni et al., 2017; Pomerleau et al., 2015; Xu et al., 2021), augmented and virtual reality (Sketchup, 2021; Wu et al., 2015; Shi et al., 2015), *etc.* A point-cloud represents visual information in a highly different way from a 2D image. A point-cloud consists of a set of unordered points lying on object surface, with each point encoding its spatial x, y, z coordinates and potentially other features. In contrast, a 2D image organizes visual features as a dense 2D pixel array. Due to the representation differences, 2D image and 3D point-cloud understanding are treated as separate problems. Image models and point-cloud models are designed to have different architectures and are trained on different types of data. Few research efforts have tried to directly transfer models from images to point-clouds or vice versa.

Intuitively, both 3D point-clouds and 2D images are visual representations of the physical world. Their low-level representations are drastically different, but they can represent the same underlying visual concept. Furthermore, human vision has no problem understanding both representations. However, can computer vision models be trained on one modality understand the other?

Somewhat surprisingly, the answer to the question above is: Yes, 2D image models trained on image datasets can be transferred to understand 3D point-clouds with minimal efforts. As illustrated in Figure 1, we transfer a 2D ConvNet to a 3D ConvNet whose input is a 3D voxel representation converted from a point-cloud. Based on a pretrained 2D ConvNet, we *inflate* its 2D convolutional filters to 3D by copying the filter weights along a third dimension. We add linear input and output layers to the network; and on a target point-cloud dataset, we only finetune the input/output layers and optionally the normalization layers, while keeping the original model weights untouched. We term such partially-finetuned-image-pretrained models as *FIP-IO* (finetuning only input and output layer) or *FIP-IO+BN* (finetuning input, output, and BN layers). FIP-IO+BN can achieve competitive

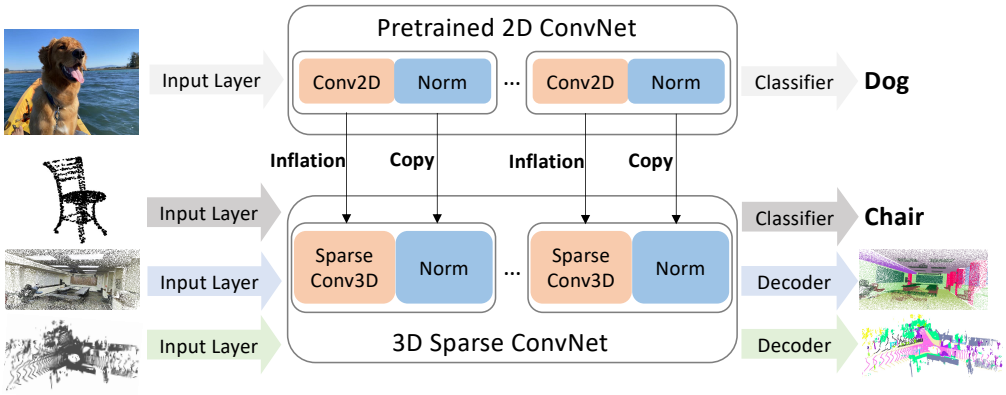

Figure 1: We investigate the feasibility of pretrained 2D ConvNets transferring to 3D sparse ConvNets. With filter inflation and finetuning only the input, output layer (classfier for classfication task and decoder for semantic segmentation task), and optionally normalization layers, 3D Sparse ConvNets are capable of dealing with point-cloud classfication, indoor, and driving scene segmentation.

performance up to 90.8% top-1 accuracy on the ModelNet 3D Wharehouse dataset, on top a ResNet50, outperforming many previous point-cloud models that adopt task-specific model architectures and tricks.

Most point-cloud models except projection-based models are only trained from scratch. Based on our surprising discovery, we further investigate fully-finetuned-image-pretrained models (termed as *FIP-ALL*). We observe that FIP-ALL brings significant improvement on top of ResNet series. Besides applying FIP-ALL to voxel-based method, we also find it generalizes to other popular methods, such as point-based method (PointNet++ (Qi et al., 2017)) and projection-based method (SimpleView (Goyal et al., 2021a)), as well as current popular vision transformers (ViT (Dosovitskiy et al., 2020)). Specifically, FIP-ALL largely outperforms the training-from-scratch by 0.88, 0.50, 3.50, 4.18 points top-1 accuracy on top of PointNet++, SimpleView, ViT-B-16, and ViT-L-16, respectively. In addition to the performance gain, FIP-ALL exhibits superior data efficiency with up to 10.0 points improvement in few-shot classification on the ModelNet 3D Wharehouse dataset. We also find that comparing with training-from-scratch, FIP-ALL dramatically speeds up the training by using 11.1 times fewer epochs to reach a target validation accuracy.

In order to understand why the image pretraining can be utilized and benefit point-cloud understanding, we conduct experiment to shed light on this by studying the network dissection (Bau et al., 2017), text-shape representation transferring (Geirhos et al., 2018), and distribution distance, hoping these can inspire the research community to explore further.

2 Related Work

2.1 Point-cloud processing model

3D convolution-based method is one of the mainstreams in point-cloud processing approaches which efficiently process point-clouds based on voxelization. In particular, in this approach, voxelization is used to rasterize point-clouds into regular grids (called voxels), thus conventional 3D convolutions can be applied. Sparse convolution is proposed to apply on the non-empty voxels (Liu et al., 2015; Choy et al., 2019; Tang et al., 2020; Zhou et al., 2020; Yan et al., 2018; Feng et al., 2021), largely improving the efficiency of 3D convolutions.

Projection-based method attempts to project a 3D point-cloud to a 2D plane and uses 2D convolution to extract features (Wang et al., 2018; Wu et al., 2018; 2019; Xu et al., 2020; Su et al., 2015; Lawin et al., 2017; Boulch et al., 2017). Specifically, the bird-eye-view projection (Yang et al., 2018; Lang et al., 2019) and the spherical projection (Wu et al., 2018; 2019; Xu et al., 2020; Milioto et al., 2019) make great progress in outdoor point-cloud tasks.

Point-based method directly processes the point-cloud data. The most classic methods, PointNet (Qi et al., 2016) and PointNet++ (Qi et al., 2017), consume points by customized feature aggregation. Many works further develop advanced local-feature aggregation operators that mimic the convolution to structure data (Xu et al., 2021; Li et al., 2018b; Hua et al., 2018; Liu et al., 2019; 2020; Wang et al., 2017; Li et al., 2018a; Komarichev et al., 2019).

2.2 Pretraining in 2D and 3D vision

Pretraining in 2D vision has shown effectiveness under supervised (Dosovitskiy et al., 2020; Girshick et al., 2014), self-supervised (Jing & Tian, 2020; Goyal et al., 2021b), and unsupervised contrastive approach (He et al., 2020; Bachman et al., 2019; Chen et al., 2020a; Caron et al., 2020; Chen et al., 2020c; Hjelm et al., 2018). After pretraining on a large amount of data, a 2D model requires much fewer computational resources and data for finetuning to reach competitive performance on downstream tasks (Kataoka et al., 2020; Caron et al., 2019; Chen et al., 2020b; Henaff, 2020).

Pretraining in 3D vision has been studied similarly as pretraining in 2D vision: both self-supervised and contrastive pretraining (Xie et al., 2020) show promising results. Due to the lack of large, annotated point-cloud datasets, pretraining in 3D vision is motivated for data efficiency (Xu & Lee, 2020). Recent works (Hou et al., 2020; Zhang et al., 2021) consider pretraining methods, for example, Contrastive Scene Contents which making use of both point-level correspondences and spatial contexts, with data efficiency in mind.

2.3 Cross-modal transfer learning

Cross-modal transfer learning attempts to take advantage of data from different modalities (Dai & Nießner, 2018; Liu et al., 2021b). For example, Liu et al. (2021a) proposed pixel-to-point knowledge transfer (PPKT) from 2D to 3D which uses aligned RGB and RGB-D images during pretraining. Our work does not rely on joint image-point-cloud pretraining. Instead, we directly transfer an image-pretrained model to point-cloud with the simplest pretraining-finetuning scheme.

Some of the previous works for video and medical images (Carreira & Zisserman, 2017; Shan et al., 2018) have adopted the method of simply extending a pretrained 2D convolutional filter along time or depth direction for transferring to 3D models. Between language and image modality, transfer learning with minimal finetuning also shows a competitive performance (Lu et al., 2021).

3 Converting a 2D ConvNet to a 3D ConvNet

In this paper, we primarily focus on the 3D sparse-convolution based method to process point-clouds because it is flexible to all point-cloud tasks. As discussed in 2.1, we consider a set of points where each point is represented by its 3D coordinates and optionally additional features such as intensity and RGB. We then voxelize/quantize these points into voxels according to their 3D space coordinates, following Choy et al. (2019). A voxel's feature is inherited from the point that lies in the voxel. If there are multiple points in a voxel, then we average all points' feature and assign the mean to the voxel. If there is no point in the voxel, then we simply set the voxel's feature to 0. When using sparse convolution, we skip the computation on empty voxels.

Given a pretrained 2D ConvNet, we convert it to a 3D ConvNet that takes 3D voxels as input. The key element of this procedure is to convert 2D convolution filters to 3D, *i.e.* constructing 3D filters with the weights directly inherited from 2D filters. A 2D convolutional filter can be represented as a 4D tensor of shape $[M, N, K, K]$, representing output dimension, input dimension, and two spatial kernel sizes, respectively. A 3D convolutional filter has an extra dimension, and its shape is $[M, N, K, K, K]$. To better illustrate, we ignore the output and input dimensions and only consider a spatial slice of the 2D filter with shape $[K, K]$. The simplest way to convert this 2D filter to 3D is to copy the 2D filter and repeat it by K times along a third dimension. This operation is the same as the *inflation* technique used by (Carreira & Zisserman, 2017) to initialize a video model with a pretrained 2D ConvNet.

Besides convolution, other operations such as downsampling, BN, nonlinear activation can be easily migrated to 3D. Our 3D model inherits the architecture of the original 2D ConvNet, but we also add a linear layer as the input layer and an output layer depending on the target task. For classification, we

use a global average pooling layer followed by one fully connected layer to get the final prediction. For semantic segmentation, the output layer is a U-Net style decoder (Ronneberger et al., 2015). The architecture of the input/output layers is described in more detail in supplementary A.5.

4 EMPIRICAL EVALUATION

To explore the image to point-cloud transfer, we study three settings: 1) partially-finetuned-image-pretrained model, only finetuning input and output layers (FIP-IO), 2) finetuning input, output, and batch normalization layers (FIP-IO+BN), and 3) finetuning the whole pretrained network (FIP-ALL). Under the three settings, we extensively explore the feasibility of transferring the image-pretrained model for point-cloud understanding and the benefits of this. The entire empirical evaluation is organized as four questions: 1) Can we transfer pretrained-image models to recognize point-clouds? (Section 4.1) 2) Can image-pretraining benefit the performance of point-cloud recognition? (Section 4.2) 3) Can image-pretrained model improve the data efficiency on point-cloud recognition? (Section 4.3) 4) Can image-pretrained model accelerate training point-cloud models? (Section 4.4)

Datasets. We benchmark the transferred models on ModelNet 3D Wharehouse classification (Wu et al., 2015), S3DIS indoor segmentation (Armeni et al., 2017), and SemanticKITTI outdoor segmentation (Behley et al., 2019) tasks. ModelNet 3D Wharehouse is a CAD model classification dataset that consists of point-clouds with 40 categories. CAD models in this benchmark come from 3D Warehouse (Sketchup, 2021). In this benchmduiark, we only utilize x, y, z coordinates as features. S3DIS is a dataset collected from real-world indoor scenes and includes 3D scans of Matterport Scanners from 6 areas. It provides point-wise annotations for indoor objects like chair, table, and bookshelf, *etc.* SemanticKITTI dataset from KITTI Vision Odometry (Geiger et al., 2012) is a driving scene dataset. It provides dense point-wise annotations for the complete 360 degrees field-of-view of the deployed automotive lidar, which is currently one of the most challenging datasets.

ResNet (He et al., 2016a) series is used mostly throughout our experiments. Depending on the experiments, ResNets are pretrained on Tiny-ImageNet, ImageNet-1K, ImageNet-21K (Deng et al., 2009), and Fractal database (FractalDB) (Kataoka et al., 2020). Our pretrained models are directly downloaded from various sources, with detailed links provided in Section A.1. To study the benefits of using pretrained image models, we also utilize PointNet++ (Qi et al., 2017), ViT (Dosovitskiy et al., 2020), and SimpleView (Goyal et al., 2021a) as our baselines.

4.1 CAN WE TRANSFER PRETRAINED-IMAGE MODELS TO RECOGNIZE POINT-CLOUDS?

To evaluate the feasibility of transferring pretrained 2D image models to 3D point-cloud tasks, we conduct experiments on top of the ResNet series since there are abundant open-source pretrained ResNet available. In particular, we convert 2D ConvNets into 3D ConvNets using the procedure described in Section 3. We hypothesize that, if a pretrained 2D image model is capable of understanding point-clouds directly, we can see an nontrivial performance by only finetuning input and output layers of the transferred model. Further, as we gradually relax the frozen parameters—finetuning BN parameters as well, the transferred model can achieve better performance, even surpassing training-from-scratch performance.

We conduct two groups of experiments with FIP-IO and FIP-IO+BN, with the results shown in Figure 2. The first is to inflate ResNet50 pretrained from different image datasets, including Tiny-ImageNet, ImageNet1K, ImageNet21K, FractalDB1K and FractalDB10K, and evaluate on the ModelNet 3D Wharehouse.

We surprisingly discovered that, even if we only finetune the input and output layers while keeping the image-pretrained weights freezed, the FIP-IO pretrained from ImageNet1K, FractalDB1K and FractalDB10K achieve competitive performance. Specifically, ResNet50 FIP-IO performance with ImageNet1K pretraining outperforms 3D ShapeNet (Wu et al., 2015) and DeepPano (Shi et al., 2015), which were the state-of-the-arts in 2015, by 4.2 and 3.6 points respectively in top-1 accuracy on ModelNet 3D Wharehouse. More importantly, with ImageNet21K pretrained model, ResNet50 FIP-IO+BN surpass training-from-scratch by 0.48 points, even beating a variety of well-known methods including PointNet (Qi et al., 2016), MVCNN (Su et al., 2015), DGCNN (Wang et al., 2019), KDNet (Klokov & Lempitsky, 2017), *etc.*

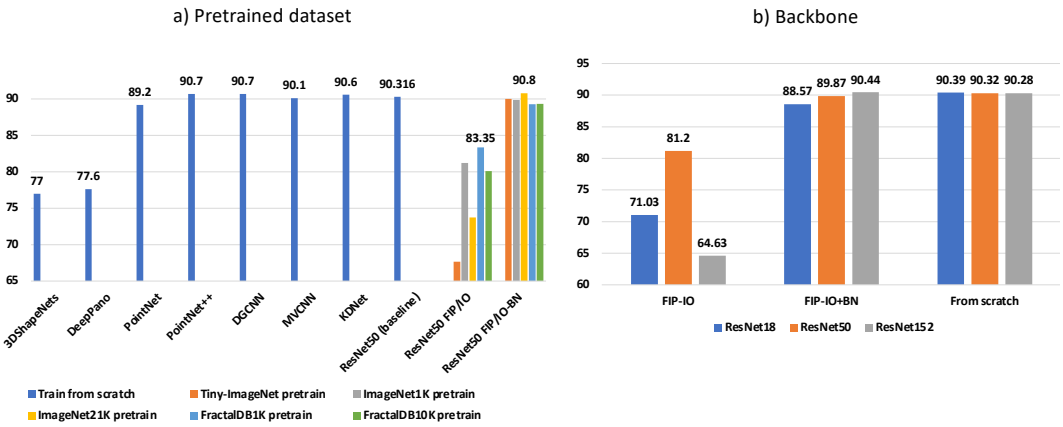

Figure 2: **a)** the left figure shows the performance of FIP-IO and FIP-IO+BN on top of **ResNet50** pretrained on different datasets. **b)** the right figure shows the performance of FIP-IO and FIP-IO+BN on top of different ResNet backbones. All the ResNet models are pretrained on ImageNet1K.

Table 1: ModelNet 3D Wharehouse classification results (top-1 accuracy %) of fully-finetuned-image-pretrained models (FIP-ALL) based on different pretrained models.

Method	**ResNet18**	**ResNet50**	**ResNet152**	**ResNet101×2**
From Scratch	90.39	90.32	90.28	90.03
FIP-ALL on ImageNet1K	90.52 (**+0.13**)	90.92 (**+0.60**)	91.09 (**+0.81**)	90.52 (**+0.49**)
FIP-ALL on ImageNet21K	-	91.05 (**+0.73**)	-	-
Method	**PointNet++(SSG)**	**ViT-B-16**	**ViT-L-16**	**SimpleView**
From Scratch	90.34	84.27	83.48	93.3
FIP-ALL on ImageNet1K	91.22 (**+0.88**)	-	-	93.8 (**+0.50**)
FIP-ALL on ImageNet21K	-	87.77 (**+3.50**)	87.66 (**+4.18**)	-

Table 2: Indoor scene and outdoor scene segmentation results (mIoU %) of fully-finetuned-image-pretrained Model (FIP-ALL). In this table, all image-pretrained models are pretrained on ImageNet1K.

Method	**S3DIS (mIoU %)**		**SemanticKITTI (mIoU %)**	
	PointNet++(SSG)	**ResNet18**	**HRNetV2-W48**	**ResNet18**
From Scratch	52.45	55.09	44.12	64.75
FIP-ALL on ImageNet1K	55.01 (**+2.56**)	56.62 (**+1.53**)	47.53 (**+3.41**)	65.57 (**+0.82**)

The second group of experiments are based on different ResNets, as shown in the right figure of Figure 2. All the ResNet models are pretrained on ImageNet1K. We observe that FIP-IO+BN on top of different ResNet models is highly competitive to training-from-scratch, and FIP-IO+BN on top of ResNet152 is even better than training-from-scratch by 0.16 points top-1 accuracy.

Therefore, we surprisingly found out the answer of "Can we transfer pretrained-image models to recognize point-clouds?": Yes, the pretrained 2D image models can be directly used for recognizing point-cloud. It is also noteworthy that the pretraining dataset is not restricted to natural but also synthetic images like those in FractalDB1K/10K.

4.2 Can image-pretraining benefit point-cloud recognition?

From the previous subsection, we find surprising that image-pretrained model can be directly used for point-cloud understanding. In this subsection, we investigate whether image-pretrained model is helpful to improve the performance on point-cloud tasks. We use different baselines, including voxelization-based method (simply ResNet), point-based method (PointNet++ (Qi et al., 2017)), projection-based method (SimpleView (Goyal et al., 2021a)), and current popular transformer-based

Table 3: Few-shot experiments on top of different ResNets on ModelNet 3D Wharehouse dataset.

Few-shot	**ResNet18**	**ResNet50 (from scratch/FIP-ALL)**	**ResNet152**
10-shot	72.2±0.8/73.2±0.6 **(+1.0)**	71.7±0.7/74.1±0.8 **(+2.4)**	69.8±1.1/73.9±0.4 **(+4.1)**
5-shot	63.7±1.6/66.6±0.8 **(+2.9)**	62.4±1.1/66.0±2.2 **(+3.6)**	59.4±0.8/66.5±0.9 **(+7.1)**
1-shot	26.8±4.4/36.8±0.6 **(+10.0)**	28.1±0.4/34.1±0.2 **(+6.0)**	23.3±4.3/33.2±1.3 **(+9.9)**

method (ViT-B-16 and ViT-L-16 (Dosovitskiy et al., 2020)). We fully finetune them on three point-cloud datasets: classification on ModelNet 3D Wharehouse, indoor scene segmentation on S3DIS, and outdoor scene segmentation on SemanticKITTI, as shown in Table 1 and Table 2.

For PointNet++, we use ImageNet1K to pretrain: we break each image into pixels and regard it as a point-cloud. For ViT, we directly use the open-source pretrained model and finetune it on ModelNet 3D Wharehouse. All the implementation details are illustrated in supplementary A. 1.

Table 1 presents performance on ModelNet 3D Wharehouse dataset. We observe that FIP-ALL steadily and significantly improves all the baselines. With pretraining, deeper models improve more. For example, ResNet18 can be only improved by 0.13% top-1 accuracy, but pretraining on ImageNet1K leads to 0.81 points top-1 accuracy improvement on top of ResNet152. Moreover, larger pretrained datasets also bring larger performance gain. Specifically, ResNet50 FIP-ALL from ImageNet21K can reach 91.05% top-1 acc, with 0.73 points improvement over training-from-scratch. Such FIP-ALL significantly outperforms a series of well-known methods such as (Qi et al., 2016; 2017; Klokov & Lempitsky, 2017; Wang et al., 2019; Su et al., 2015; Li et al., 2018a).

We also explore FIP-ALL on different architectures, as shown in the second group of Table 1. In particular, FIP-ALL on top of PointNet++, ViT-B-16, ViT-L-16 and SimpleView with image dataset pretraining improve the training-from-scratch by 0.88, 3.50, 4.18, 0.50 points, respectively. Especially for the current superior baseline in image recognition, ViT-B-16 and ViT-L-16, the improved performance is quite significant, revealing the huge potential of using image-pretrained models for point cloud recognition.

For the challenging indoor and outdoor scene segmentation, using ImageNet1K pretrained models (FIP-ALL on ImageNet1K) also consistently improve the training-from-scratch, as shown in Table 2. PointNet++ (resp. ResNet18) pretrained on ImageNet1K outperforms the training-from-scratch by 2.56 points (resp. 1.53 points) mIoU on S3DIS dataset. For SemanticKITTI, we utilize the commonly used projection-based method with 2D ConvNet HRNet. With ImageNet1K pretraining, we observe 3.41 points mIoU improvement, a large margin in such a difficult task. Since HRNetV2-W48 has rich pretrained models, we finetune Cityscapes pretrained HRNetV2-W48 and observe this enhances more (5.25% mIoU improvement over training from scratch). Even for the ResNet18 with a high from-scratch performance of 64.75% mIoU, the ImageNet1K pretraining can also bring 0.82 points mIoU improvement.

Therefore, the answer to "Can image-pretraining benefit point-cloud recognition" is Yes. Image-pretraining can indeed improve point-cloud recognition, which can generalize to a wide range of backbones as well as benefit more challenging tasks.

4.3 Can image-pretrained model improve the data efficiency on point-cloud recognition?

Data efficiency is extremely important in point-cloud understanding due to the huge labor of collecting and annotating point-cloud data. In this subsection, we investigate whether the image-pretrained model can help to improve the data efficiency by conducting few-shot setting experiments, including 1-shot, 5-shot, and 10-shot. We conduct 3 trials for each setting and report the results as mean $\pm$ std.

In detail, for each class (ModelNet 3D Wharehouse involves 40 classes), we randomly choose a few point-clouds as training data, and still evaluate on the whole test set. We compare the results between training-from-scratch and FIP-ALL pretrained on the ImageNet1K dataset. The experimental results are shown in Table 3. We observe that FIP-ALL dramatically surpasses training-from-scratch on the low data regime (1-shot): pretraining on ImageNet1K brings 10.0, 6.0, and 9.9 points top-1 accuracy

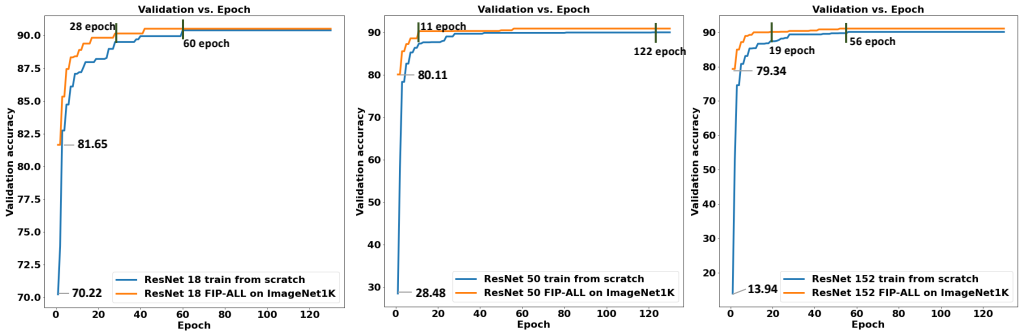

Figure 3: The curves of validation accuracy w.r.t training epoch. We compare the results between the training-from-scratch and the FIP-ALL on the ImageNet1K, on top of ResNet18, ResNet50, and ResNet152, respectively.

improvement for ResNet18, ResNet50, and ResNet152, respectively. For 5-shot and 10-shot settings, using ImageNet1K pretraining can still consistently improve the performance. However, we also observe that as the amount of training data increases, the performance gain becomes saturated.

Therefore, our answer to "Can image-pretrained model improve the data efficiency on point-cloud recognition?" is: Yes. Image-pretrained model can improve the data efficiency on point-cloud recognition, especially on the low data regime. When the training data increases, it can still improve the performance, but the gain becomes marginal.

4.4 Can image-pretrained model accelerate point-cloud training?

We also investigate whether image-pretrained model can help point-cloud task train faster. The results are shown in Figure 3.

We find that surprisingly, after training only one epoch on ModelNet 3D Wharehouse dataset, FIP-ALL on ImageNet1K achieves very impressive performance yet the performance of training-from-scratch is still at a low level. For example, after the first epoch, ResNet50 (resp. ResNet152) with training from scratch can only achieve 28.48% (resp. 13.94%) top-1 accuracy while ResNet50 (resp. ResNet152) with ImageNet1K pretraining reaches 80.11% (resp. 79.34%) top-1 accuracy. Moreover, to reach 90% top-1 accuracy, a non-trivial performance, FIP-ALL significantly accelerates the training by 2.14x (28 vs. 60 epoch), 11.1x (11 vs. 122 epoch), 2.95x (19 vs. 56 epoch) over training-from-scratch, on top of ResNet18, ResNet50, and ResNet152, respectively.

Therefore, our answer to "Can image-pretrained model accelerate point-cloud training?" is still: Yes. The image-pretrained model can significantly accelerate the training speed of point-cloud tasks.

5 Discussion

In this section, we attempt to shed light on why transferring image-pretrained models for point-cloud understanding works. Inspired by recent related works (Geirhos et al., 2018; Brendel & Bethge, 2019; He et al., 2015; 2016b; Bau et al., 2017), we explore this from the aspects of the network dissection, texture-shape representation transferring, the distance between feature distributions.

5.1 What does the pretrained-image model transfer to point-cloud model?

Does the image-pretrained model transfer the visual concepts? Inspired by the network dissection for 2D Broden dataset (Bau et al., 2017), we also attempt to explore what the transferred units are looking at in the ModelNet 3D Wharehouse dataset. We present the visualization of FIP-IO+BN pretrained on ImageNet1K. The visualization is to show the most activated cases when the whole dataset passes through each unit of the last model stage, as displayed in Figure 4. More visualization

Table 4: Texture-shape representation transferring experiment. The results are evaluated on ModelNet 3D Wharehouse.

Method	Pretrained dataset	top-1 accuracy
ResNet50 FIP-IO	ImageNet1K	81.20
ResNet50 FIP-IO	Stylized-ImageNet	83.52
BagNet17 FIP-IO	ImageNet1K	57.53
BagNet33 FIP-IO	ImageNet1K	68.40

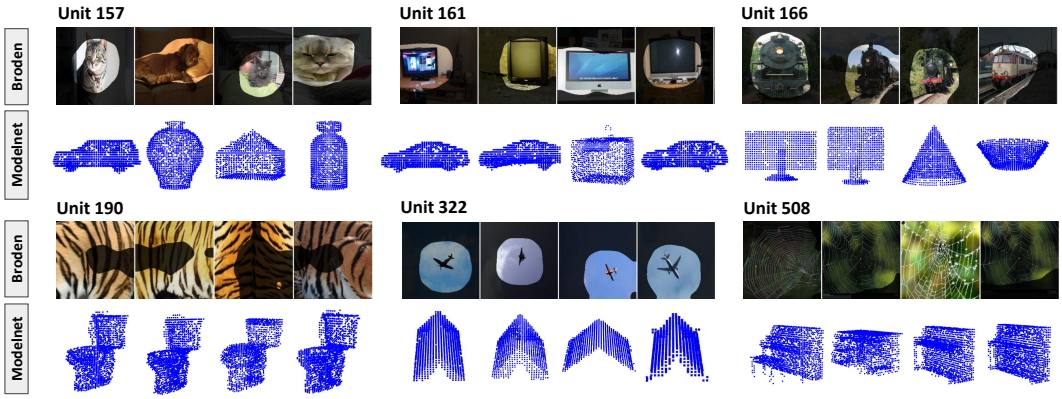

Figure 4: Network Dissection of FIP-IO+BN. The visualization displays what the units are looking at on both image dataset and ModelNet 3D Wharehouse dataset.

can be found in supplementary 5. From the visualization alone, we do not get obvious cues of what visual concepts are transferred between the two modalities. For example, unit 161 strongly activating to computer screens in the Broden, yet it is looking at cars and shelves in the ModelNet 3D Wharehouse. However, we unexpectedly find that the pretrained units are prone to cluster similar objects. In fact, such clustering ability is an important cue of performing well on classification tasks (Hartigan & Wong, 1979; Caron et al., 2021).

Does pretrained-image model transfer shape or texture representation? Recent work (Geirhos et al., 2018) proposed that models learn texture and shape from ImageNet. We follow this direction to futher explore what pretrained-image mode transfers. For the experiment, we take two image-pretrained models with either more of the shape or texture representation and compare the FIP-IO performance.

To force the model to acquire more shape representation, (Geirhos et al., 2018) stylizes the images in ImageNet into artwork style, such that the models trained on that are confused by variant textures, hence having stronger shape representation. We directly take the pretrained ResNet50 on stylized ImageNet as the stronger shape representation model.

To get a stronger texture representation model, we are inspired by BagNet (Brendel & Bethge, 2019). By controlling the receptive field, BagNet breaks the shape in a image and focuses more on the texture information. Our experimental result is shown in Table 4. BagNet17 means the size of attended patches is 17×17, and BagNet33 means the size of attended patches is 33×33. Note that after inflating the BagNet17 and BagNet33, both of the architectures are totally as same as the inflated ResNet50. Besides, both BagNet17/33 and ResNet pretrained on stylized-ImageNet1K perform worse than the original ResNet50 on ImageNet classification (Geirhos et al., 2018; Brendel & Bethge, 2019).

We can observe that the ResNet50 FIP-IO on Stylized-ImageNet (with stronger shape representation) outperforms the baseline ResNet50 FIP-IO on ImageNet1K over 2.32 points top-1 accuracy, while both inflated BagNets performs dramatically worse than the baseline. This shows that the **shape representations** are better transferred from image to point-cloud modality.

Table 5: Examples of first-wasserstein distance between the distribution of image features and point-cloud features on top of pretrained ResNet18. The first row shows the average of the results from all the 16 layers in ResNet18.

Image Model	**FIP-IO**	**FIP-IO+BN**	**FIP-ALL**
Average of 16 layers	2.1×10^2	0.27	0.093
Layer 1	1.9	0.2	0.094
Layer 4	2.6	0.14	0.051
Layer 8	13	0.26	0.051
Layer 16	1.2×10^3	0.89	0.59

5.2 WHY DOES FINETUNING BATCH NORMALIZATION HELP THE TRANSFERRING?

Our experiment in Figure 2 shows that finetuning BN, in addition to the input and output layer, can greatly improve the transfer performance compared with only finetuning input/output. It is interesting why such a small part of the network, in terms of parameter size and FLOPs, can have a big impact.

We hypothesize that batch normalization layers shift and scale the feature distribution. To measure the effect, we leverage the first-wasserstein distance (Rubner et al., 2000), a way to calculate the distance between distributions. For estimating the feature distribution of the pretrained-image model, we pass the whole ImageNet1K dataset into the pretrained-image models, collect the pre-activation features after each convolution layer, then sample 15,000 data points from the element-wise distribution of collected features (we assume the pre-activation features present Gaussian distribution). For the ModelNet 3D Wharehouse dataset, we use FIP-IO, FIP-IO+BN, and FIP-ALL to conduct the same operation on the ModelNet 3D Wharehouse dataset and also collect 15,000 data points. We then calculate the FWD between the element-wise feature distribution of image-pretrained model's and each of the FIP-IO, FIP-IO+BN, FIP-ALL model's. The example results are shown in Table. 5. We observe that, the FWD between FIP-IO (point-cloud features) and image model (image features) is very large, yet after finetuning batch normalization layers, the distance is dramatically reduced. This suggests that batch normalization plays a critical role of transferring the point-cloud representation to be closer to the image representation.

6 CONCLUSION

In this work, we use finetuned-image-pretrained models (FIP) to explore the feasibility of transferring the image-pretrained model for point-cloud understanding, and the benefits of using image-pretrained models on point-cloud tasks. We surprisingly discover that, with simply inflating a 2D pretrained ConvNet and minimal finetuning — input, output, and optionally batch normalization layer (FIP-IO or FIP-IO+BN), the image pretrained models can achieve very competitive performance on 3D point-cloud classification, even beating a wide range of point-cloud models that adopt a variety of tricks. Moreover, we find that when finetuning all the parameters of the pretrained models (FIP-ALL), the performance can be significantly improved on point-cloud classification, indoor and outdoor scene segmentation. Full finetuning generalizes to most of the popular point-cloud methods. Besides, we also find that FIP-ALL can improve the data efficiency on few-shot learning, and accelerate the training speed by a large margin. After this, we shed light on why the image-pretrained model can be used for point-cloud understanding from three aspects, network dissection, texture-shape representation transferring, feature distribution distance. Compared with previous works that seek improvements from designing architectures and pretraining only on the point-cloud modality, our work is not limited by the architecture design and small scale point-cloud dataset. We believe that image pretraining is one of the solutions to the bottleneck of point-cloud understanding and do hope this direction can inspire the research community in the future.

REFERENCES

Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6541–6549, 2017.

J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. Unstructured point cloud semantic labeling using deep segmentation networks. *3DOR*, 2:7, 2017.

Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *International Conference on Learning Representations*, 2019. URL `https://openreview.net/pdf?id=SkfMWhAqYQ`.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2959–2968, 2019.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

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

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020b.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020c.

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3075–3084, 2019.

Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 452–468, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

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.

Di Feng, Yiyang Zhou, Chenfeng Xu, Masayoshi Tomizuka, and Wei Zhan. A simple and efficient multi-task network for 3d object detection and road understanding. *arXiv preprint arXiv:2103.04056*, 2021.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361, 2012.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. *arXiv preprint arXiv:2106.05304*, 2021a.

Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021b.

John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pp. 4182–4192. PMLR, 2020.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Ji Hou, Benjamin Graham, Matthias Nießner, and Saining Xie. Exploring data-efficient 3d scene understanding with contrastive scene contexts. *arXiv preprint arXiv:2012.09165*, 2020.

Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 984–993, 2018.

Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, Eisuke Yamagata, Ryosuke Yamada, Nakamasa Inoue, Akio Nakamura, and Yutaka Satoh. Pre-training without natural images. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 863–872, 2017.

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*, pp. 7421–7430, 2019.

Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12697–12705, 2019.

Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pp. 95–107. Springer, 2017.

Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9397–9406, 2018a.

Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on χ-transformed points. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 828–838, 2018b.

Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 806–814, 2015.

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*, pp. 5239–5248, 2019.

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: Pixel-to-point knowledge transfer for 3d pretraining. *arXiv preprint arXiv:2104.04687*, 2021a.

Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. A closer look at local aggregation operators in point cloud analysis. In *European Conference on Computer Vision*, pp. 326–342. Springer, 2020.

Zhengzhe Liu, Xiaojuan Qi, and Chi-Wing Fu. 3d-to-2d distillation for indoor scene parsing. *arXiv preprint arXiv:2104.02243*, 2021b.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4213–4220. IEEE, 2019.

François Pomerleau, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104, 2015.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2016. URL `http://arxiv.org/abs/1612.00593`. cite arxiv:1612.00593.

Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses, 2021.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

Hongming Shan, Yi Zhang, Qingsong Yang, Uwe Kruger, Mannudeep K Kalra, Ling Sun, Wenxiang Cong, and Ge Wang. 3-d convolutional encoder-decoder network for low-dose ct via transfer learning from a 2-d trained network. *IEEE transactions on medical imaging*, 37(6):1522–1534, 2018.

Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters*, 22(12):2339–2343, 2015. doi: 10.1109/LSP.2015.2480802.

Sketchup. 3d modeling online free|3d warehouse models. `https://3dwarehouse.sketchup.com`, 2021.

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*, pp. 945–953, 2015.

Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019.

Haotian* Tang, Zhijian* Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European Conference on Computer Vision*, 2020.

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.

Zining Wang, Wei Zhan, and Masayoshi Tomizuka. Fusing bird's eye view lidar point cloud and front view camera image for 3d object detection. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1–6. IEEE, 2018.

Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *ICRA*, 2018.

Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *ICRA*, 2019.

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 (CVPR)*, June 2015.

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*, pp. 574–591. Springer, 2020.

Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *European Conference on Computer Vision*, pp. 1–19. Springer, 2020.

Chenfeng Xu, Bohan Zhai, Bichen Wu, Tian Li, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. You only group once: Efficient point-cloud processing with token representation and relation inference module. *arXiv preprint arXiv:2103.09975*, 2021.

Xun Xu and Gim Hee Lee. Weakly supervised semantic point cloud segmentation: Towards 10x fewer labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13706–13715, 2020.

Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18 (10):3337, 2018.

Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7652–7660, 2018.

Xiangyu Yue, Bichen Wu, Sanjit A Seshia, Kurt Keutzer, and Alberto L Sangiovanni-Vincentelli. A lidar point cloud generator: from a virtual world to autonomous driving. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pp. 458–464, 2018.

Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. *arXiv preprint arXiv:2101.02691*, 2021.

Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. *arXiv preprint arXiv:2008.01550*, 2020.

A APPENDIX

A.1 IMPLEMENTATION DETAIL

Our experiments are conducted on ModelNet 3D Wharehouse, S3DIS, and SemanticKITTI dataset. For the ModelNet 3D Wharehouse dataset, we train all models on the train set and evaluate on the validation set. For the S3DIS, we train all models on area 1, 2, 3, 4, 6 and evaluate on area 5. For the SemanticKITTI dataset, we train all models on splits 00-10 except 08 which is used for evaluation. For each of the dataset, all ResNet-series models use the same training scheme, and all experiments are implemented with Pytorch.

When training on the *ModelNet 3D Wharehouse dataset*, coordinates of point-cloud are randomly scaled, translated, and jittered. We use SGD optimizer with momentum 0.9, weight-decay 10^{-4}, and initial learning rate 0.1 with cosine learning rate scheduler. Each mini batch is set to 32, and models are trained for 300 epochs. For both training and inference phase, we only utilize x, y, z coordinates without other features and set voxel size to 0.05. The experiments for ModelNet 3D Wharehouse are all conducted on a Titan RTX GPU.

When training on the *S3DIS dataset*, we concatenate all subparts of an indoor scene to train and validate on. Along x, y directions, scenes are randomly applied horizontal flip. RGB features are randomly jittered, translated, and auto contrasted. Finally, we normalize and clip point-clouds. We set voxel size to 0.05. We use SGD optimizer with momentum 0.9, weight-decay 10^{-4}, and initial learning rate 0.1 with polynomial learning rate scheduler. Each mini batch is set to 3, and models are trained for 400 epochs on 2 Titan RTX GPUs.

When training on the *SemanticKITTI dataset*, coordinates of each point-cloud are randomly scaled and rotated. We use SGD optimizer with momentum 0.9, weight-decay 10^{-4}, and initial learning rate 0.24 with cosine warmup learning rate scheduler. Each mini batch is set to 2, and models are trained for 15 epochs on 4 Titan RTX GPUs. For both training and inference phase, we utilize x, y, z coordinates as well as intensity feature and set voxel size to 0.05.

Most of our pretrained models come from open-sources, [1] [2] [3] [4] [5] [6] , so we do not need to take time and computational resources for pretraining. We use torchsparse [7] to produce sparse 3D convolutions.

Details of Section 4.1. In this section, we take the ResNet architecture, inflate the pretrained models of different image dataset, and add linear input and output layers as shown in Section A.5. The ResNet50 pretrained on ImageNet1K is directly taken from Pytorch. We use the same training recipe provided by Pytorch to train the ResNet50 on Tiny-ImageNet. The pretrained ResNet50 on ImageNet21K comes from (Ridnik et al., 2021).

Details of Section 4.2, Section 4.3 and 4.4. In this section, the ResNet pretrained models are taken from the same sources as illustrated above.

For PointNet++ pretraining on ImageNet1K, we utilize the PointNet++ SSG version (Qi et al., 2017). We break the images into pixels, and regard the pixels as a point-cloud, coordinates of which are the x, y position in original image, with appending $z = 1$ to all pixels. Then we set center sampling number to 1024 and 256 for first and second stage, the radius are set into 8 and 64 for them respectively. For each center point, we will query 64 neighbouring points. The training recipe is also provided by Pytorch.

For ViT models, we directly take the pretrained weights from Dosovitskiy et al. (2020). To apply it on ModelNet 3D Wharehouse, we sample 256 centers and group 64 nearby points, regarding these as "point-cloud patches". Then we use a linear embedding to project the point-cloud patches into a sequence, and the ViT process them as same as the images. Except the linear embedding and the final

[1] https://pytorch.org/vision/stable/models.html
[2] https://github.com/Alibaba-MIIL/ImageNet21K
[3] https://github.com/hirokatsukataoka16/FractalDB-Pretrained-ResNet-PyTorch
[4] https://github.com/HRNet/HRNet-Semantic-Segmentation/tree/pytorch-v1.1
[5] https://github.com/rgeirhos/Stylized-ImageNet
[6] https://github.com/wielandbrendel/bag-of-local-features-models
[7] https://github.com/mit-han-lab/torchsparse

Table 6: ResNet50 results (evaluated on ModelNet 3D Wharehouse) of fintuning the mean and variance in batch normalization layers on different datasets. IO indicates input and output layer, IOms indicate input, output, mean and variance, IOmsWb indicates input, output and the whole BN.

Layers	**Tiny-ImageNet**	**ImageNet1K**	**ImageNet21K**	**FractalDB1K**	**FractalDB10K**
IO	67.666	81.199	73.744	83.347	80.105
IOms	83.793	82.942	84.076	72.326	79.66
IOmsWb	89.992	89.87	90.721	89.263	89.344
From scratch	90.316	90.316	90.316	90.316	90.316

Table 7: ResNet18, 50, 152 results (evaluated on ModelNet 3D Wharehouse) of finetuing the mean and variance in batch normalization layers.

Layers	**ResNet18**	**ResNet50**	**ResNet152**
IO	71.029	81.199	64.627
IOmv	81.888	82.942	82.658
IOmvWb	88.574	89.87	90.438
From Scratch	90.397	90.316	90.276

output classifier, all the models are kept same as the origin version. For the experiments on S3DIS and SemanticKITTI, the architecture detail of ResNet18 is shown in A.5 listing 2.

For SimpleView model, all the experiment settings are as same as Goyal et al. (2021a). The only difference is to use or not use the pretrained ResNet18. For HRNetV2-W48, we directly use the ImageNet1K pretrained model and Cityscape pretrained model from (Sun et al., 2019).

We conduct three trials on the few-shot experiments. For each trial, we change the random seed but keep all the others are same. To plot the training speed curve, we directly use the training log without any other change like smoothing.

A.2 Finetuning the mean and variance of batch normalization.

For the first group of experiments, ResNet50 FIP either has IO or IO+BN finetuned. In addition to these two experimental setting, we also investigate fintuning input, output layers, and mean, variance of normalization layers, while fixing the convolution layer weights, normalization layer weights and bias. The full experiments result with this extra setting is reported in Table 6 and 7. We can observe that compared with only finetuning input and output layers, updating mean and variance can also largely improve the performance of point-cloud recognition. As suggested in Section 5.2 in the main paper, updating mean and variance is also to push the finetuned pretrained image model to generate point-cloud representation as similar as image representation.

A.3 More visualization of Network dissection.

We present more visualization results based on the technique of network dissection, as shown in Figure 5. We can observe that for most cases, although there is no obvious visual concept transferring, the pretrained filters are prone to cluster similar objects on ModelNet 3D Wharehouse dataset.

A.4 Details of First-Wasserstein Distance on ResNet18.

We list all the results in each layer of ResNet18, as shown in Table 8. We can observe that for each layer, the first-wasserstein distance is largely reduced when finetuning more parameters. Interestingly, we also find that the distance between training-from-scratch on ModelNet 3D Wharehouse and image models are not too far, which may indicate the image and point-cloud could be represented similar even no transferring.

Table 8: First-wasserstein distance between distance between different features of all 16 layers in ResNet18

Image Model	FIP/IO	FIP/IO-BN	FIP/ALL	Training from scratch
Average	2.1×10^2	0.27	0.093	0.15
Layer 1	1.9	0.2	0.094	0.12
Layer 2	1.1	0.064	0.11	0.048
Layer 3	1.3	0.24	0.068	0.095
Layer 4	2.6	0.14	0.051	0.077
Layer 5	1.1	0.23	0.039	0.082
Layer 6	3	0.21	0.025	0.063
Layer 7	5.4	0.29	0.05	0.15
Layer 8	13	0.26	0.051	0.097
Layer 9	12	0.28	0.04	0.095
Layer 10	26	0.2	0.077	0.13
Layer 11	54	0.27	0.032	0.13
Layer 12	2.2×10^2	0.25	0.076	0.094
Layer 13	1.9×10^2	0.29	0.028	0.11
Layer 14	7.7×10^2	0.16	0.075	0.041
Layer 15	9.5×10^2	0.29	0.081	0.088
Layer 16	1.2×10^3	0.89	0.59	1

A.5 Details of used architectures

```
Class 3DRes_cls(nn.Module):
    def __init__(self, res_block):
        super().__init__() # res_block means the residual block as same as the conventional ResNet.
        self.input_layer = nn.Sequential(
            sparse_conv3d(input_dim, layer1_Idim, k=3, s=1),
            sparse_bn(layer1_Idim))

        self.layer1 = inflated_resnet_layer1(res_block, layer1_Idim, layer1_Odim)
        self.layer2 = inflated_resnet_layer2(res_block, layer2_Idim, layer2_Odim)
        self.layer3 = inflated_resnet_layer3(res_block, layer3_Idim, layer3_Odim)
        self.layer4 = inflated_resnet_layer4(res_block, layer4_Idim, layer4_Odim)

        self.output_layer = nn.Sequential(
            global_average_pooling,
            nn.Linear(layer4_Odim, class_num),
            nn.bn(class_num))

    def forward(self, x):
        x = self.input_layer(x)
        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)
        x = self.layer4(x)
```

```
        return self.output_layer(x)
```

Listing 1: Pseudo code of inflated ResNet with linear input and output for classification

```
Class 3DRes_seg(nn.Module):
    def __init__(self, res_block):
        super().__init__() # res_block means the residual block as same as the conventional ResNet.
        self.input_layer = nn.Sequential(
            sparse_conv3d(input_dim, layer1_Idim, k=3, s=1),
            sparse_bn(layer1_Idim),
            sparse_ReLU(True),
            sparse_conv3d(layer1_Idim, layer1_Idim, k=3, s=1),
            sparse_bn(layer1_Idim),
            sparse_ReLU(True),
            sparse_conv3d(layer1_Idim, layer1_Idim, k=3, s=2),
            sparse_bn(layer1_Idim),
            sparse_ReLU(True))

        self.layer1 = inflated_resnet_layer1(res_block, layer1_Idim, layer1_Odim)
        self.layer2 = inflated_resnet_layer2(res_block, layer2_Idim, layer2_Odim)
        self.layer3 = inflated_resnet_layer3(res_block, layer3_Idim, layer3_Odim)
        self.layer4 = inflated_resnet_layer4(res_block, layer4_Idim, layer4_Odim)

        self.up1 = sparse_deconv(layer4_Odim, layer4_Odim, k=2, s=2),
        self.decode1 = self.Sequential(
            res_block(layer4_Odim+layer3_Odim, layer3_Odim),
            res_block(layer3_Odim, layer3_Odim))

        self.up2 = sparse_deconv(layer3_Odim, layer3_Odim, k=2, s=2)
        self.decode2 = self.Sequential(
            res_block(layer3_Odim+layer2_Odim, layer2_Odim),
            res_block(layer2_Odim, layer2_Odim))

        self.up3 = sparse_deconv(layer2_Odim, layer2_Odim, k=2, s=2)
        self.decode3 = self.Sequential(
            res_block(layer2_Odim+layer1_Odim, layer1_Odim),
            res_block(layer1_Odim, layer1_Odim))

        self.up4 = sparse_deconv(layer1_Odim, layer1_Odim, k=2, s=2)
        self.decode4 = self.Sequential(
            res_block(layer1_Odim+layer1_Odim, layer1_Odim),
            res_block(layer1_Odim, layer1_Odim))

        self.output_layer = nn.Sequential(
            nn.Linear(layer1_Odim, class_num))

    def forward(self, x):
        x_i = self.input_layer(x)
        x1 = self.layer1(x_i)
        x2 = self.layer2(x1)
        x3 = self.layer3(x2)
        x4 = self.layer4(x3)

        x3_ = self.decoder1(cat(x3, self.up1(x4)))
        x2_ = self.decoder2(cat(x2, self.up2(x3_)))
        x1_ = self.decoder3(cat(x1, self.up3(x2_)))
        xi_ = self.decoder4(cat(x_i, self.up4(x1_)))
        return self.output_layer(xi_)
```

Listing 2: Pseudo code of inflated ResNet for segmentation

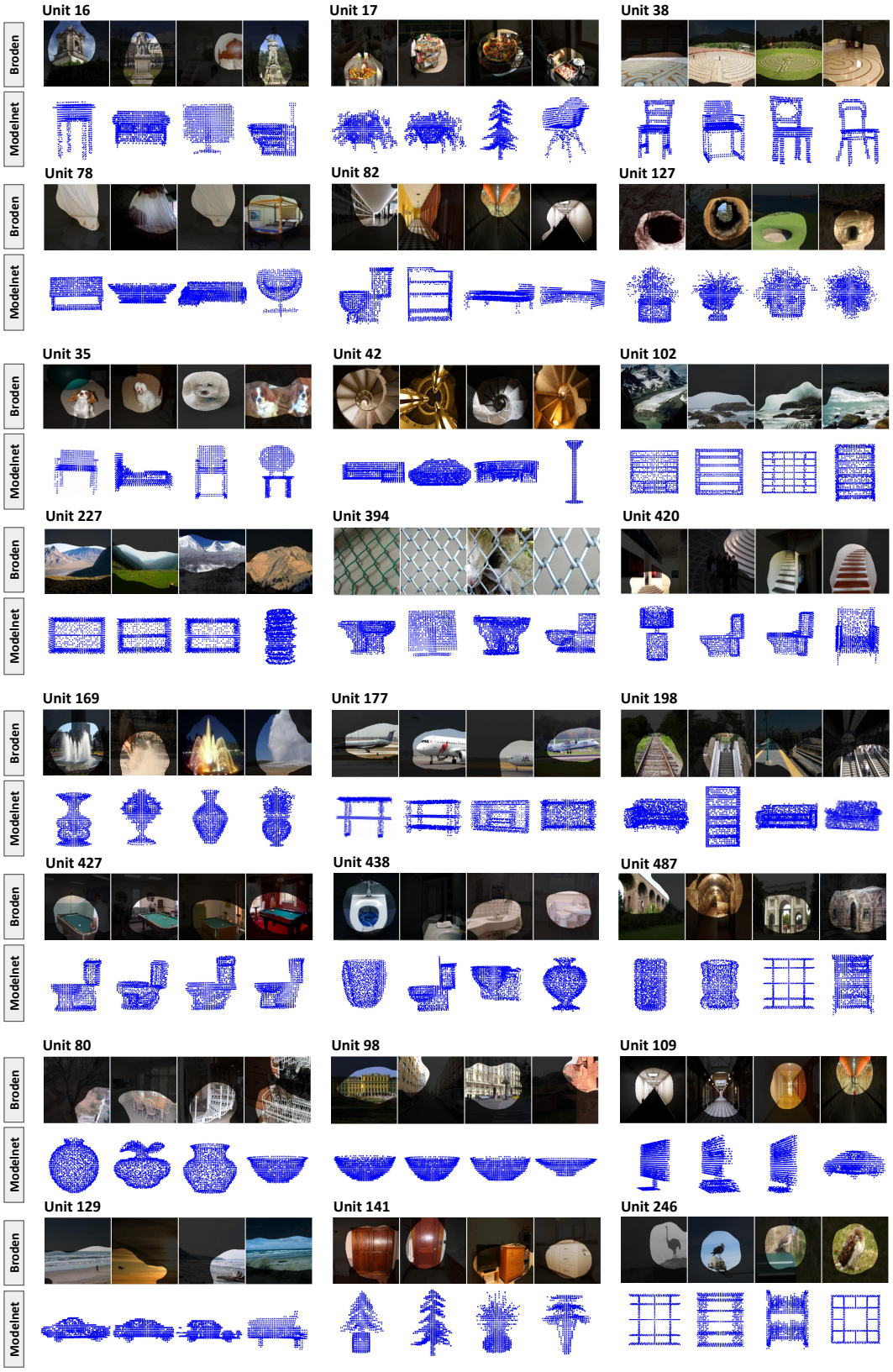

Figure 5: More network dissection results.