3DIT:Project Point Cloud to 3D Image's Token to Make Full Use of Image Model for Point Cloud Analysis

Abstract

This paper investigates how pre-trained models with prior 2D knowledge can be leveraged to tackle challenges in 3D point cloud analysis. We introduce **Adapt Point Former** (APF), an approach that fine-tunes existing image models using a limited number of parameters to efficiently handle point cloud data without converting them to images. In our method, raw points are transformed into point embeddings, aligning their dimensions with those of image tokens. This is crucial due to the intrinsic disorganization in point cloud data, which differs from the orderly nature of 2D data. By sequencing the point embeddings to match the attention patterns derived from image priors, we optimize the processing of point cloud data while minimizing the total training parameters. The image model, with its parameters set, is fine-tuned in conjunction with a point former, enhancing the handling of point cloud data. Our comprehensive experiments across multiple benchmarks validate the effectiveness of APF.

Keywords: APF, pre-trained.

1 Introduction

Pre-trained self-attention-based models [38] represented by BERT [8] and visual transformer (ViT) [10] have exhibited remarkable effectiveness in natural language processing, image recognition, and related domains. Recently, the newly proposed paradigm, parameter-efficient fine-tuning (PEFT) [44], has been introduced to harness the rich prior knowledge and powerful representational capabilities inherent in pre-trained models for a wide array of downstream tasks. However, a deficiency exists in the availability of large well-trained 3D models. Despite the development of various 3D pre-trained models, including OcCo [39], point-BERT [45], and point-MAE [29], to name a few, there remains a notable scarcity of expansive pre-trained models tailored for 3D point cloud analysis. This scarcity is attributed to the considerably higher costs and labor-intensive efforts associated with the acquisition of accurately labeled 3D data, in stark contrast to the relative abundance of labeled data available in the domains of images and language. In the realm of images, for instance, there are numerous well-trained transformer-based models, such as ViT-Base [10], comprising 86 million parameters trained on a dataset of 14 million images, and CLIP [33], trained with 400 million (image, text) pairs. Given this scenario, a question arises: *can we directly leverage 2D pre-trained models for the analysis of 3D point clouds?* If indeed feasible, the wealth of inexpensive and readily accessible 2D data, coupled with pre-trained models, holds the potential to substantially enhance point cloud analysis.

Therefore,we introduces a groundbreaking approach called Adapt Point Former (APF) to utilize pre-trained image models for processing 3D point clouds. This method bypasses the conventional 3D-to-2D projection by initially using a lightweight PointNet, named Random PointNet (RPN), to align the point cloud dimensions with image tokens, generating a Random Point Embedding (RPE). Notably, the dimension alignment network remains unchanged during training. APF then employs a pre-trained 2D transformer, fine-tuned through an AdaptFormer-based method (termed PointFormer) using RPE. Our findings show that APF with RPN outperforms models built from scratch on 3D data in accuracy, highlighting the effectiveness of leveraging 2D model attention for 3D point cloud analysis. APF further enhances this by making the PointNet trainable and adding point embedding sequencing for better feature space calibration between point clouds and image priors.

The main highlights of our work are:

- Demonstrating the potential of using pre-trained image models for 3D point cloud analysis, showing that minimal fine-tuning of 2D priors can surpass traditional 3D models.
- Introducing APF, a framework for adapting 2D pre-trained models for 3D point cloud analysis, featuring a point embedding module and point sequencer for feature alignment, followed by a PointFormer module with minimal trainable parameters for fine-tuning attention.

we conduct comprehensive experiments on various 3D tasks, proving APF's superior performance compared to existing methods.

2 Related works

2.1 3D Point Cloud Analysis

CNN-based Point Cloud Analysis. Since the introduction of PointNet [30], there has been a flourishing development of deep learning-based approaches in the realm of point cloud processing over the past few years. These methods can be categorized into three groups based on the representations of point clouds: voxel-based [7, 25, 36], projection-based [23, 34], and point-based [13, 32]. Voxel-based methods entail the voxelization of input points into regular voxels, utilizing CNNs for subsequent processing. However, these methods tend to incur substantial memory consumption and slower runtime, particularly when a finer-grained representation is required [13]. Projection-based methods encompass the initial conversion of a point cloud into a dense 2D grid, treated thereafter as a regular image, facilitating the application of classical methods to address the problem. However, these methods heavily rely on projection and back-projection processes, presenting challenges, particularly in urban scenes with diverse scales in different directions. In contrast, point-based methods, directly applied to 3D point clouds, are the most widely adopted. Such methods commonly employ shared multi-layer perceptrons or incorporate sophisticated convolution operators [30, 31, 37, 41]. In recent years, hybrid methods such as PVCNN [25] and PV-RCNN [36], which combine the strengths of diverse techniques, have achieved notable advancements.

Self-Attention-based Point Cloud Analysis. Recently, attention operations [38] have been adopted for point cloud processing in several studies [6, 12, 47]. For example, point transformer [47] and point cloud transformer (PCT) [12] have introduced self-attention networks [38] to improve the capture of local context within

point clouds. Afterward, a plethora of methods based on the self-attention architecture have been proposed. PointMixer [6] enhances self-attention layers through introducing inter-set and hierarchical-set mixing. Token-Fusion [40] initially fuses tokens from heterogeneous modalities with point clouds and images, subsequently forwarding the fused tokens to a shared transformer, enabling the learning of correlations among multimodal features. AShapeFormer, as introduced by Li et al. [24], utilizes multi-head attention to effectively encode information pertaining to object shapes. This encoding capability can be seamlessly integrated with established 3D object detection methodologies. Exploiting pre-trained transformer models is also a promising way. P2P [42] employs a lightweight DGCNN [41] for the conversion of point clouds into visually rich and informative images, which serves to facilitate the utilization of pre-trained 2D knowledge. Point-BERT [45] constructs point cloud tokens that represent various geometric patterns, resembling word tokens. Following this, pre-trained language models, represented by BERT [8], can be deployed for downstream tasks such as object classification, part segmentation, and related applications.

2.2 Parameter-Efficient Fine-tuning

Recent advancements for 2D visual recognition incorporate the pre-trained Transformer models, exemplified by models like CLIP [33] and ViT [10]. Parameter-efficient fine-tuning (PEFT) [44], similar to model reprogramming or adversarial reprogramming [11] in the field of adversarial learning is designed to capitalize on the representational capabilities inherited from pre-trained models. PEFT strategically fine-tunes only a few parameters to achieve better performance on diverse downstream tasks that are different from the pre-trained models [3]. Representative methods, including prompt tuning (PT) [22], adapters [18] and Low-rank adapter (LoRA) [19] were initially designed for the purpose of incorporating language instructions into the input text for language models. He et al. [16] introduced adapters into the field of computer vision. Bahng et al. [2] first defined the "visual prompt", mirroring the "prompt" in NLP. Subsequently, PEFT in the realm of computer vision, such as visual prompt tuning (VPT) [20], visual adapters [4, 28], and visual LoRA [35], to name a few, exhibit outstanding performance with minimal training parameters, reduced epochs, and substantial performance enhancements.

3 Method

3.1 Inspiration Behind the Study

The use of Transformers in point cloud analysis, as seen in works like [14] and [47], has shown promising advancements. However, when compared to CNNs, Transformers tend to underperform if trained from scratch on medium-sized datasets like ImageNet-1K, as indicated by [38] and [8]. The challenge becomes more pronounced in the context of 3D point cloud data, which is not only harder to acquire and annotate but also irregular and diverse in nature, leading to significant time and cost implications.

Given these challenges, our approach leans towards the use of PEFT technology, which is more data-efficient. However, a critical aspect to address is the adaptation of existing pre-trained models, which are primarily based on 2D image data, to suit the requirements of 3D point cloud data. This involves the careful calibration of dimensions and attention mechanisms to bridge the gap between 3D point clouds and 2D images.

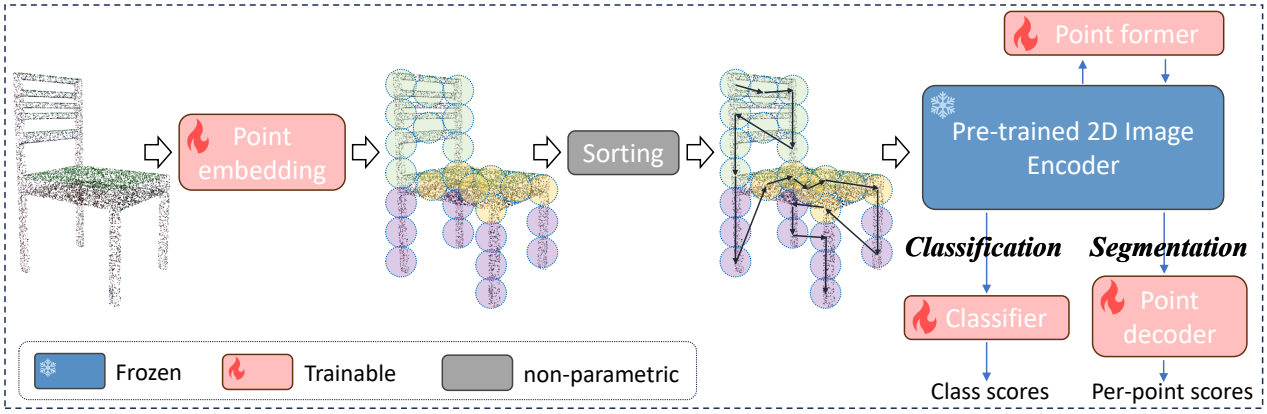

Figure 1. **The pipeline of our proposed APF.**

3.2 Preliminaries

2D Visual Transformer. A visual transformer model comprises an embedding layer and multiple Transformer blocks. For an input image x^I, the model first partitions x^I into m patches, forming a set $\{x_i^I\}_{i=1}^m$. These patches are then embedded into sequences of d^I-dimensional vectors, denoted as $E_0^I = \texttt{Embed}\left(\left[x_1^I, x_2^I, \cdots, x_m^I\right]\right)$, where $E_0^I \in \mathbb{R}^{m\times d}$. E_0^I is subsequently fed into L blocks $\{\phi^{(l)}\}_{i=1}^L$ within the transformer model. We use superscript (l) to denote the index of the block in the transformer model. Formally, this procedural description can be mathematically expressed as:

$$z_i^{I,(0)} = \texttt{Embed}\left(x_i^I\right) + e_i, \tag{1}$$

$$\left[z_{\text{cls}}^{I,(l)}, \mathcal{Z}^{I,(l)}\right] = \phi^{(l)}\left(\left[z_{\text{cls}}^{I,(l-1)}, \mathcal{Z}^{I,(l-1)}\right]\right) \tag{2}$$

where $z_i^{I,(0)} \in \mathcal{R}^d$ and $e_i \in \mathcal{R}^d$ denote the image path embedding and positional embedding, respectively. $\mathcal{Z}^{I,(l)} = [z_1^{I,(l)}, z_2^{I,(l)}, \cdots, z_m^{I,(l)}]$. $z_{\text{cls}}^{I,(l)}$ is an additional learnable token for classification. $\phi^{(l)}$ is composed of multi-head self-attention (MSA) and a MLP layer (MLP) with layer normalization (LN) [1] and residual connection [17]. Specifically, $\phi^{(l)}$ is composed by:

$$\begin{cases} \tilde{z}_i^{I,(l)} = \texttt{MSA}^l\left(z_i^{I,(l-1)}\right) + z_i^{I,(l-1)} \\ z_i^{I,(l)} = \texttt{MLP}^l\left(\texttt{LN}\left(\tilde{z}_i^{I,(l)}\right)\right) + \tilde{z}_i^{I,(l)} \end{cases}. \tag{3}$$

A singular self-attention within $\texttt{MSA}^l$ is calculated by:

$$\texttt{Att}\left(Q^{(l)}, K^{(l)}, V^{(l)}\right) = \texttt{SoftMax}\left(\frac{Q^{(l)}K^{(l)^T}}{\sqrt{d}}\right)V^{(l)}, \tag{4}$$

where $Q^{(l)}$, $K^{(l)}$ and $V^{(l)}$ are the input query, key, and value tokens obtained by three different learnable linear projection weights denoted by $\mathbf{W}_Q$, $\mathbf{W}_K$ and $\mathbf{W}_V$, respectively. Finally, the class prediction is achieved by a linear classification head.

Raw Point Grouping. Given an input point cloud $\mathcal{P} \in \mathbb{R}^{N\times(d'+C)}$, where N represents the number of unordered points, denoted as $\mathcal{P} = \left[x_1^P, x_2^P, \cdots, x_N^P\right]$ and $x_i^P \in \mathbb{R}^{d'+C}$ with d'-dim coordinates and C-dim point feature, we first employ iterative farthest point sampling (FPS) to sample a subset of points $\mathcal{P}_s = \left[x_1^P, x_2^P, \cdots, x_{N_s}^P\right] \in \mathbb{R}^{N_s\times(d'+C)}$. Subsequently, the k-nearest neighbors $\mathcal{P}_g = \left[\left\{x_{1,j}^P\right\}_{j=1}^k, \left\{x_{2,j}^P\right\}_{j=1}^k, \cdots, \left\{x_{N_s,j}^P\right\}_{j=1}^k\right.$

$\mathbb{R}^{N_s \times k \times (d'+C)}$ for each point are identified, wherein each group $\left\{x_{i,j}^P\right\}_{j=1}^k$ within $\mathcal{P}_g$ corresponds to a local region around the centroid point x_i^P, and k represents the number of points adjacent to the N_s centroid points. Following this, embedding $\mathcal{P}_g$ becomes necessary to leverage the pre-trained 2D ViT structure.

3.3 Point Embedding & Sequencing

Point embedding converts the grouped raw points into a structured and representative embedding for enhancing the utilization and calibration with the 2D image tokens. We implement a lightweight network (`Point_Embed`) to obtain the point embedding:

$$z_i^{P,(0)} = \texttt{Point\_Embed}\left(\mathcal{X}_i^P\right), \tag{5}$$

where `Point_Embed` can take various forms such as pointNet [30], pointNeXt [32] and pointMLP [26], to name a few. The input point x_i^P is from $\mathcal{P}_g$. We use $\mathcal{X}_i^P$ to represent the set of k neighboring points $\left\{x_{i,j}^P\right\}_{j=1}^k$ around x_i^P for simplicity. To seamlessly integrate with the 2D pre-trained ViT, the dimension of point embedding should align with image embedding in Eq. (1). Specifically, $z_i^{P,(0)} \in \mathbb{R}^d$. Eventually, the embedding representation of an input point cloud $\mathcal{P}$ for feeding into pre-trained 2D ViT is $\mathcal{Z}^{P,(0)} = \left[z_1^{P,(0)}, z_2^{P,(0)}, \cdots, z_{N_s}^{P,(0)}\right]$.

The inherent unordered nature is one of the most significant properties of point clouds [30], making it different from pixel arrays in image data. Merely aligning the dimension of embeddings is insufficient to fully leverage the attention-related priors of a 2D pre-trained model. We introduce a 3D token sequencer that leverages Morton-order [27] to sequence the point embedding:

$$\mathcal{O} = \texttt{Morton\_Order}\left(\mathcal{P}_s\right), \tag{6}$$

where $\mathcal{O} \in \mathbb{R}^{N_s \times 1}$ is the order of input point sets. We use it to sequence the point embedding obtained by Eq (5):

$$\mathcal{Z}_s^P = \mathcal{Z}^P\left[\mathcal{O}\right]. \tag{7}$$

For simplicity, we omit the superscript indicating which block the input belongs to. Then, the transformer-based model is utilized to acquire point tokens.

3.4 PointFormer

The transformer-based architecture is more data-hungry than CNN-based ones [46]. In comparison to two-dimensional data, the quantity of three-dimensional data is relatively limited, which causes overfitting and cannot fully release the ability of the transformer-based model. In contrast to 2D data, the availability of 3D data is relatively constrained, resulting in issues such as overfitting and a limited realization of the transformer-based model potential. This paper investigates parameter-efficient fine-tuning (PEFT) technology to alleviate overfitting and improve model generalization for 3D point models. PEFT involves the freezing of the pre-trained backbone, previously trained on an extensive dataset, while introducing a limited number of learnable parameters to adapt to the new dataset. This new dataset can be data-rich [2, 20], few-shot [21], or long-tailed [9], as PEFT equips the model with knowledgeable priors. AdapterFormer [4] is an effective PEFT method. It appends the MLP layer in Eq (3) with a bottleneck module and has been empirically validated for its efficacy in handling 2D image data. We utilize this architecture for calibrating the 3D point tokens alongside

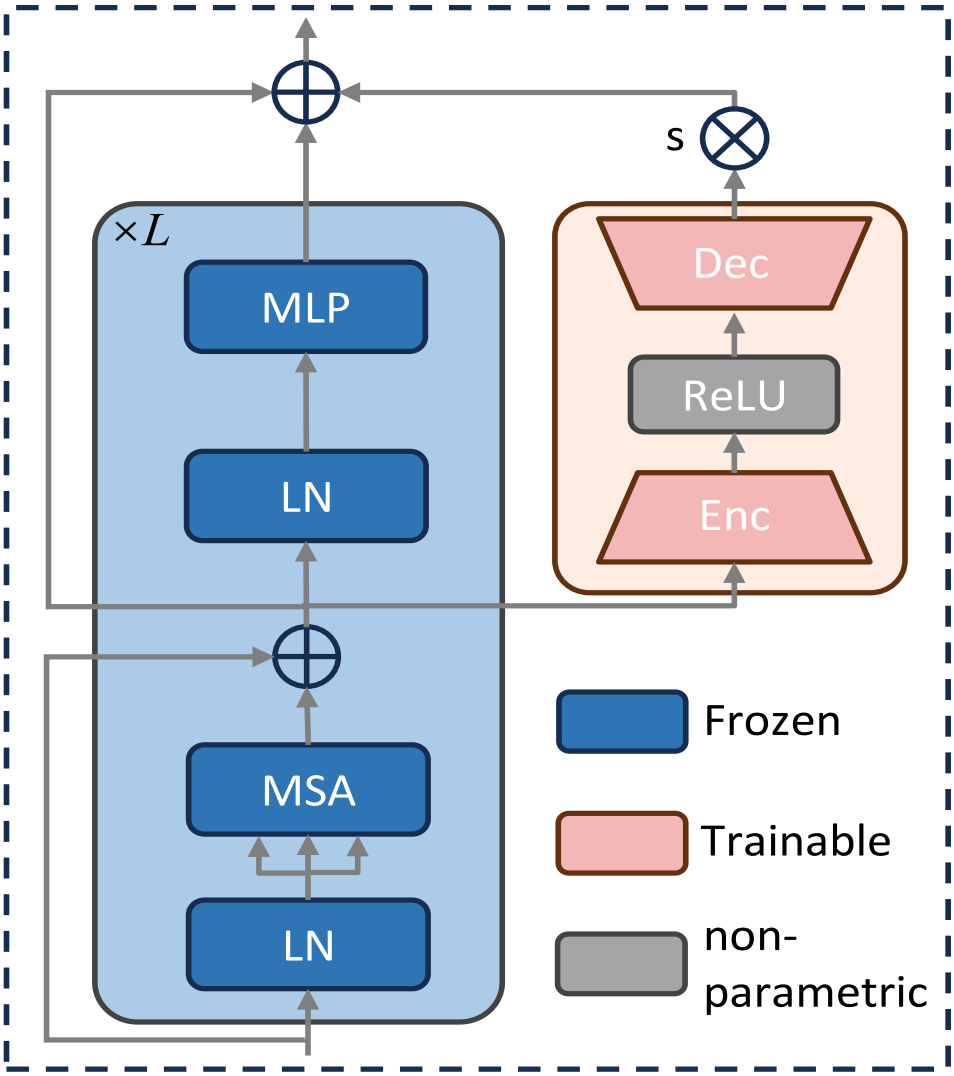

Figure 2. **The framework of PointFormer**.

the 2D image tokens, namely introducing a trainable bottleneck module for token calibration. Formally, the calibration of point embedding is calculated as follows:

$$\begin{cases} \tilde{z}_i^{P,(l)} = \texttt{MSA}^l\left(z_i^{P,(l-1)}\right) + z_i^{P,(l-1)} \\ \hat{z}^{P,(l)} = \texttt{ReLU}\left(\texttt{LN}(\tilde{z}_i^{P,(l)}) \cdot \mathbf{W}_{\text{enc}}\right) \cdot \mathbf{W}_{\text{dec}} \\ z_i^{P,(l)} = \texttt{MLP}\left(\texttt{LN}(\tilde{z}_i^{P,(l)})\right) + s \cdot \hat{z}^{P,(l)} + z_i^{P,(l-1)} \end{cases}, \tag{8}$$

where $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{d\times\hat{d}}$ and $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{\hat{d}\times d}$ are the only learnable parameters for model fine-tuning. The dimensions satisfy $\hat{d} \ll d$. All other parameters within the transformer blocks remain fixed. s is a scale factor. The input of the first multi-head self-attention block $\texttt{MSA}^1$ is from the sorted point embeddings, namely $\left[z_i^{P,(0)}\right] = \mathcal{Z}_s^P$. The framework of the point former is shown in Figure 2.

3.5 Downstream Tasks

Classification. The class token $z_{\text{cls}}^{P,(l)}$ output by the last block for point embedding can be utilized for classification. We use z_{cls} as a shorthand notation for $z_{\text{cls}}^{P,(l)}$ for clarity. The predicted logit of each class is given by the

softmax of the final linear layer, as calculated by:

$$p_i = \frac{e^{w_i \cdot z_{\text{cls}}}}{\sum_{j=1}^{C} e^{w_j \cdot z_{\text{cls}}}}, \tag{9}$$

where w_i is the linear classifier weight and C is the total number of classes. Eventually, the cross-entropy loss can be employed to calculate the loss.

Segmentation. Segmentation needs to predict a label for each point. We employ a U-net style architecture, where the APF serves as the point encoder. The segmentation head concatenates the output features from transformer blocks within the encoder, succeeded by deconvolution interpolation and multiple MLP layers to facilitate dense prediction. Similar to classification, the softmax cross-entropy is employed as the loss function.

The overall pipeline of APF is shown in Figure 1.

4 Implementation details

4.1 Comparing with the released source codes

The code is original

4.2 Experimental environment setup

We mainly follow the settings in [42] and [15], namely the AdamW optimizer in conjunction with the CosineAnnealing scheduler are employed, initializing a learning rate of 5×10^{-4} incorporating a weight decay of 5×10^{-2}. The ViT-Base (ViT-B) version [10] is utilized as the pre-trained 2D model in the experiments.

4.3 Comparison Results

Object Classification. The results are presented in Tables 1 and 2. On ModelNet40, PointNet and Transformer can be seen as the baseline models. It is observed that the 3D pre-training OcCo enhances the performance of PointNet and Transformer by 0.9% and 0.7%, respectively. In contrast, ADP exhibits superior performance, outperforming PointNet and Transformer by 5.0% and 2.8%, respectively. Furthermore, ADP surpasses all other counterparts in performance, including the recently proposed Joint-MAE. On ScanObjectNN, we empirically validate two versions of point embedding methods: PointNet and PointMLP. PointNet, PointMLP and Transformer are considered as the baseline models in this context. ADP consistently outperforms the 3D pre-trained model by a large margin. For example, on the most challenging split, PB-T50-RS, the pre-training OcCo improves PointNet by 12.0%. In comparison, ADP, employing a PointNet embedding, achieves a remarkable gain of 13.1% over PointNet. Furthermore, ADP, when utilizing PointMLP embedding, exhibits superior performance, surpassing PointMLP by 2.6% and outperforming other previous arts. While APF may not exhibit as robust performance on OBJ-BG compared to the most recently proposed Joint-MAE and Point-MAE, it surpasses the majority of existing methods overall. For example, APE with PointMLP outperforms Joint-MAE and Point-MAE by 1.7% and 2.6%, respectively, on PB-T50-RS.

Methods	Pre-trained modality	Acc.(%)
DNN-based model		
PointNet [30]	N/A	89.2
PointNet-OcCo [39]	3D	90.1
PointNet++ [31]	N/A	90.5
DGCNN [41]	N/A	92.9
DGCNN-OcCo [39]	3D	93.0
KPConv [37]	N/A	92.9
PAConv [43]	N/A	93.9
PointMLP [26]	N/A	94.1
Transformer-based model		
Transformer [38]	N/A	91.4
Transformer-OcCo [39]	3D	92.1
Point Transformer [47]	N/A	93.7
PCT [12]	N/A	93.2
Point-BERT [45]	3D	93.2
Point-MAE [29]	3D	93.8
P2P* [42]	2D	92.4
Joint-MAE [15]	3D	94.0
APF (ours) †	2D	**94.2**

Table 1. **Object classification results on ModelNet40**. *: For P2P, we refer to the results obtained based on ViT-B to ensure a fair comparison. †: Using a lightweight PointNet for point embedding.

Methods	Pre-trained modality	OBJ-BG	OBJ-ONLY	PB-T50-RS
DNN-based model				
PointNet [30]	N/A	73.8	79.2	68.0
PointNet-OcCo [39]	3D	-	-	80.0
PointNet++ [31]	N/A	82.3	84.3	77.9
DGCNN [41]	N/A	82.8	86.2	78.1
DGCNN-OcCo [39]	3D	-	-	83.9
PRA-Net [5]	N/A	-	-	82.1
MVTN [5]	N/A	92.6	92.3	82.8
PointMLP	N/A	-	-	85.2
Transformer-based model				
Trans. [38]	N/A	79.9	80.6	77.2
Trans.-OcCo [39]	3D	84.9	85.5	78.8
Point-BERT [45]	3D	87.4	88.1	83.1
Point-MAE [29]	3D	90.0	88.3	85.2
P2P* [42]	2D	-	-	84.1
Joint-MAE [15]	3D	90.9	88.9	86.1
APF *w. PointNet*†	2D	85.5	88.4	83.1
APF *w. PointMLP*†	2D	89.9	**89.0**	**87.8**

Part Segmentation. Following prior works [15, 29, 30], we sample 2,048 points from each input instance and adopt the same segmentation head as Point-MAE [29] and Joint-MAE [29]

Methods	mIoU$_C$	mIoU$_I$	aero-plane	bag	cap	car	chair	ear-phone	guitar	knife	lamp	laptop	motor-bike	mug	pistol	rocket	skate-board	table
DNN-based model																		
PointNet [30]	80.4	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++ [31]	81.9	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
DGCNN [41]	82.3	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
KPConv [37]	85.1	86.4	84.6	86.3	87.2	81.1	91.1	77.8	92.6	88.4	82.7	96.2	78.1	95.8	85.4	69.0	82.0	83.6
PAConv [43]	84.6	86.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PointMLP [26]	84.6	86.1	83.5	83.4	87.5	80.54	90.3	78.2	92.2	88.1	82.6	96.2	77.5	95.8	85.4	64.6	83.3	84.3
Transformer-based model																		
Trans. [38]	83.4	85.1	82.9	85.4	87.7	78.8	90.5	80.8	91.1	87.7	85.3	95.6	73.9	94.9	83.5	61.2	74.9	80.6
Point Trans. [47]	83.7	**86.6**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
PCT [12]	-	86.4	85.0	82.4	89.0	81.2	91.9	71.5	91.3	88.1	86.3	95.8	64.6	95.8	83.6	62.2	77.6	83.7
Trans.-OcCo [39]	83.4	85.1	83.3	85.2	88.3	79.9	90.7	74.1	91.9	87.6	84.7	95.4	75.5	94.4	84.1	63.1	75.7	80.8
Point-BERT [45]	84.1	85.6	84.3	84.8	88.0	79.8	91.0	81.7	91.6	87.9	85.2	95.6	75.6	94.7	84.3	63.4	76.3	81.5
Point-MAE [29]	-	86.1	84.3	85.0	88.3	80.5	91.3	78.5	92.1	87.4	96.1	96.1	75.2	94.6	84.7	63.5	77.1	82.4
P2P* [42]	82.5	85.7	83.2	84.1	85.9	78.0	91.0	80.2	91.7	87.2	85.4	95.4	69.6	93.5	79.4	57.0	73.0	83.6
Joint-MAE [15]	**85.4**	86.3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
APF (ours)	83.4	85.6	82.9	82.5	85.2	80.2	91.0	74.8	91.5	87.8	84.2	96.0	76.0	95.3	83.6	59.0	75.2	82.5

Table 3. **Part segmentation results on ShapeNetPart**. mIoU$_C$ (%) is the mean of class IoU. mIoU$_I$ (%) is the mean of instance IoU. "Trans." abbreviates Transformer.

5 Conclusion and future work

This paper demonstrates the potential of using pre-trained image models for 3D point cloud analysis. It shows that with minimal parameter, 2D priors can outperform traditional models designed for 3D data. And we have introduced a new framework called APF, designed for adapting 2D pre-trained models to 3D point cloud analysis. APF includes a point embedding module and a point sequencer for aligning features, followed by a PointFormer module. This module is notable for its minimal trainable parameters, allowing for fine-tuned attention mechanisms. Building on the current trend of fine-tuning large models, future work could focus on several innovative directions: Efficient Fine-Tuning for Resource-Constrained Environments: Developing methods to fine-tune large models in a more resource-efficient manner, making this approach more accessible for organizations with limited computational resources. Hybrid Models Combining Different Learning Paradigms: Creating hybrid approaches that integrate fine-tuning of large models with other machine learning paradigms, such as unsupervised, semi-supervised, or reinforcement learning, to tackle more complex problems."

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Hyojin Bahng, Ali Jahanian, Swami Sankaranarayanan, and Phillip Isola. Exploring visual prompts for adapting large-scale models. *arXiv preprint arXiv:2203.17274*, 2022.

[3] Aochuan Chen, Yuguang Yao, Pin-Yu Chen, Yihua Zhang, and Sijia Liu. Understanding and improving visual prompting: A label-mapping perspective. In *CVPR*, pages 19133–19143, 2023.

[4] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *NeurIPS*, 35:16664–16678, 2022.

[5] Silin Cheng, Xiwu Chen, Xinwei He, Zhe Liu, and Xiang Bai. Pra-net: Point relation-aware network for 3d point cloud analysis. *IEEE TIP*, 30:4436–4448, 2021.

[6] Jaesung Choe, Chunghyun Park, Francois Rameau, Jaesik Park, and In So Kweon. Pointmixer: Mlp-mixer for point cloud understanding. In *ECCV*, pages 620–640, 2022.

[7] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, pages 3075–3084, 2019.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Bowen Dong, Pan Zhou, Shuicheng Yan, and Wangmeng Zuo. LPT: Long-tailed prompt tuning for image classification. In *ICLR*, 2022.

[10] 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. In *ICLR*, 2021.

[11] Gamaleldin F Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial reprogramming of neural networks. In *ICLR*, 2019.

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

[13] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE TPAMI*, 43(12):4338–4364, 2020.

[14] Ziyu Guo, Xianzhi Li, and Pheng Heng-Ann. Joint-mae: 2d-3d joint masked autoencoders for 3d point cloud pre-training. In *IJCAI*, pages 791–799, 2023.

[15] Ziyu Guo, Renrui Zhang, Longtian Qiu, Xianzhi Li, and Pheng-Ann Heng. Joint-mae: 2d-3d joint masked autoencoders for 3d point cloud pre-training. In *IJCAI*, pages 791–799, 2023.

[16] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *ICLR*, 2022.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[18] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, pages 2790–2799, 2019.

[19] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

[20] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, pages 709–727, 2022.

[21] Dongjun Lee, Seokwon Song, Jihee Suh, Joonmyeong Choi, Sanghyeok Lee, and Hyunwoo J Kim. Read-only prompt optimization for vision-language few-shot learning. In *CVPR*, pages 1401–1411, 2023.

[22] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, 2021.

[23] Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jianjian Sun, and Zeming Li. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. In *AAAI*, volume 37, pages 1477–1485, 2023.

[24] Zechuan Li, Hongshan Yu, Zhengeng Yang, Tongjia Chen, and Naveed Akhtar. Ashapeformer: Semantics-guided object-level active shape encoding for 3d object detection via transformers. In *CVPR*, pages 1012–1021, 2023.

[25] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *NeurIPS*, 32, 2019.

[26] 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. In *ICLR*, 2022.

[27] Guy M Morton. A computer oriented geodetic data base and a new technique in file sequencing. 1966.

[28] Xing Nie, Bolin Ni, Jianlong Chang, Gaofeng Meng, Chunlei Huo, Shiming Xiang, and Qi Tian. Pro-tuning: Unified prompt tuning for vision tasks. *IEEE TCSVT*, 2023.

[29] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *ECCV*, pages 604–621, 2022.

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017.

[31] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 30, 2017.

[32] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *NeurIPS*, 35:23192–23204, 2022.

[33] 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 *ICLR*, pages 8748–8763, 2021.

[34] Haoxi Ran, Jun Liu, and Chengjie Wang. Surface representation for point clouds. In *CVPR*, pages 18942–18952, 2022.

[35] Jiang-Xin Shi, Tong Wei, Zhi Zhou, Xin-Yan Han, Jie-Jing Shao, and Yu-Feng Li. Parameter-efficient long-tailed recognition. *arXiv preprint arXiv:2309.10019*, 2023.

[36] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, pages 10529–10538, 2020.

[37] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, pages 6411–6420, 2019.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[39] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. Unsupervised point cloud pre-training via occlusion completion. In *ICCV*, pages 9782–9792, 2021.

[40] Yikai Wang, Xinghao Chen, Lele Cao, Wenbing Huang, Fuchun Sun, and Yunhe Wang. Multimodal token fusion for vision transformers. In *CVPR*, pages 12186–12195, 2022.

[41] 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 TOG*, 38(5):1–12, 2019.

[42] 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. *NeurIPS*, 2022.

[43] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *CVPR*, pages 3173–3182, 2021.

[44] Bruce XB Yu, Jianlong Chang, Haixin Wang, Lingbo Liu, Shijie Wang, Zhiyu Wang, Junfan Lin, Lingxi Xie, Haojie Li, Zhouchen Lin, et al. Visual tuning. *arXiv preprint arXiv:2305.06061*, 2023.

[45] 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 *CVPR*, pages 19313–19322, 2022.

[46] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, pages 558–567, October 2021.

[47] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, pages 16259–16268, 2021.