ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240475

Adapt PointFormer: 3D Point Cloud Analysis via Adapting 2D Visual Transformers

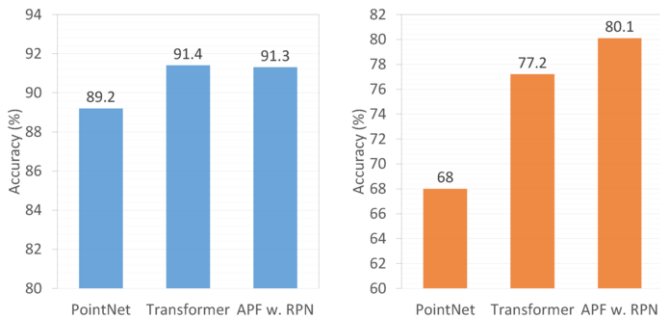
Mengke Li[a,b], Da Li[a,b], Guoqing Yang[a,b], Yiu-ming Cheung[c] and Hui Huang[b,*]

[a] Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, China
[b]VCC, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
[c]Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, China

Abstract. Pre-trained large-scale models have exhibited remarkable efficacy in computer vision, particularly for 2D image analysis. However, when it comes to 3D point clouds, the constrained accessibility of data, in contrast to the vast repositories of images, poses a challenge for the development of 3D pre-trained models. This paper therefore attempts to directly leverage pre-trained models with 2D prior knowledge to accomplish the tasks for 3D point cloud analysis. Accordingly, we propose the Adaptive PointFormer (APF), which fine-tunes pre-trained 2D models with only a modest number of parameters to directly process point clouds, obviating the need for mapping to images. Specifically, we convert raw point clouds into point embeddings for aligning dimensions with image tokens. Given the inherent disorder in point clouds, in contrast to the structured nature of images, we then sequence the point embeddings to optimize the utilization of 2D attention priors. To calibrate attention across 3D and 2D domains and reduce computational overhead, a trainable PointFormer with a limited number of parameters is subsequently concatenated to a frozen pre-trained image model. Extensive experiments on various benchmarks demonstrate the effectiveness of the proposed APF. The source code and more details are available at *https://vcc.tech/research/2024/PointFormer*.

1 Introduction

Compared with the traditional paradigm of training neural networks from scratch [23, 31], pre-trained self-attention-based models [43], represented by BERT [8] and visual transformer (ViT) [11], have shown significant improvement in natural language processing (NLP), image recognition, and related domains. Transformer-based architecture has also been introduced for point cloud analysis in several studies [55, 13, 7] and has shown remarkable progress. Subsequently, the novel parameter-efficient fine-tuning paradigm (PEFT) [51], 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. Recently, multiple 3D pre-trained models have been developed, such as OcCo [44], point-BERT [52], and point-MAE [32], to name a few. However, despite these advancements, a significant scarcity persists in the availability of extensive 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



(a) ModelNet40 (b) ScanObjectNN

Figure 1: Performance comparison. APF w. RPN denotes our proposed APF architecture employing random lightweight PointNet.

3D data, in contrast to the relative abundance of labeled data available in the domains of images and language. For example, OcCo is pre-trained on ModelNet40 [48], a dataset comprising 12,311 synthesized CAD objects from 40 categories. In the realm of images, for instance, there are numerous well-trained transformer-based models, such as ViT-Base [11], comprising 86 million parameters trained on a dataset of 14 million images, and CLIP [36], trained with 400 million image and text pairs. Given this scenario, a question arises: *Can we directly leverage 2D prior knowledge for the analysis of 3D point clouds?* If feasible, the wealth of inexpensive and readily accessible 2D data, coupled with pre-trained models, holds the potential to substantially enhance the methods for point cloud analysis.

The affirmative response is encapsulated in the work of Wang et al. [47], who proposed Point-to-Pixel Prompting (P2P). This approach stands as the first attempt to transfer pre-trained knowledge from the 2D domain to the 3D domain. Nevertheless, P2P requires mapping the point cloud into images, a process characterized by pathological mapping that inevitably leads to the loss of inherent information within point clouds. Joint-MAE [15] investigates the geometric correlation between 2D and 3D representations. Employing a joint encoder and decoder architecture with modal-shared and model-specific decoders facilitates cross-modal interaction. Essentially, both P2P and joint-MAE leverage the point cloud and its corresponding projection to explore image knowledge.

We first devise an empirical study to further investigate the viability of directly applying image priors in point cloud analysis. A randomly initialized lightweight PointNet (RPN) is utilized for aligning

* Corresponding author.

the dimensions of point clouds with those of image tokens and then obtain the random point embedding. The dimension alignment network is fixed during training. Subsequently, a pre-trained 2D transformer undergoes fine-tuning with the sequenced random point embedding as input. The results are shown in Figure 1. It can be observed that, compared to the model trained on 3D data from scratch, the fine-tuned 2D model attains higher accuracy. Therefore, the attention derived from pre-trained models on 2D images exhibits efficacy in analyzing 3D point clouds.

To this end, this paper proposes a novel approach named Adapt PointFormer (APF), which utilizes pre-trained image models for the direct processing of point clouds, thereby adapting 2D image prior knowledge to 3D point clouds.To further effectively leverage 2D self-attention, APF renders the dimension alignment network to be trainable and incorporates point embedding sequencing. To better calibrate the attention of point clouds and image priors, a fine-tuning technique based on AdaptFormer [5], referred to as PointFormer is introduced. Extensive experiments are conducted on various downstream tasks to demonstrate the effectiveness of APF.

In summary, our main contributions are:

- We investigate the potential of the pre-trained image model in 3D point cloud analysis and reveal that directly leveraging 2D priors with minimal fine-tuning can outperform models trained on 3D data from scratch.
- We propose APF, a framework that fine-tunes 2D pre-trained models for direct application to 3D point cloud analysis. It consists of a point embedding module and a point sequencer for feature alignment, followed by a PointFormer module with a minimal number of trainable parameters for attention calibration.
- We conduct extensive experiments on diverse 3D downstream tasks, which demonstrates the superior performance of APF compared to existing methods.

2 Related work

2.1 3D Point Cloud Analysis

CNN-based Methods. Since the introduction of PointNet [33], 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 [26, 39], projection-based [37, 24], and point-based [14, 35]. 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 [14]. 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 problems of point cloud analysis. 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 [33, 34, 45, 40]. In recent years, hybrid methods such as PVCNN [26] and PV-RCNN [39], which combine the strengths of diverse techniques, have achieved notable advancements.

Self-Attention-based Methods. Self-attention operations [43] have been adopted for point cloud processing in several studies [55, 13, 7]. For example, the point transformer [55] and point cloud transformer (PCT) [13] have introduced self-attention networks [43] to improve the capture of local context within the point clouds. Afterward, a plethora of methods based on the self-attention architecture have been proposed. PointMixer [7] enhances self-attention layers through inter-set and hierarchical-set mixing. TokenFusion [46] initially fuses tokens from heterogeneous modalities with point clouds and images, subsequently forwarding the fused tokens to a shared transformer, allowing learning of correlations among multimodal features. AShapeFormer [25] 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 [47] employs a lightweight DGCNN [45] 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 [52] constructs point cloud tokens that represent various geometric patterns, resembling word tokens. Subsequently, pre-trained language models, represented by BERT [8], can be applied to 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 [36] and ViT [11]. Parameter-efficient fine-tuning (PEFT) [51], similar to model reprogramming or adversarial reprogramming [12] 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 in diverse downstream tasks that are different from the pre-trained models [4]. 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. LoRA [19] applies parameter tuning within the multi-head self-attention module in transformers. He *et al.* [16] introduced adapters into the field of computer vision. Bahng *et al.* [3] 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 [5, 30], and visual LoRA [38], to name a few, exhibit outstanding performance with minimal training parameters, reduced epochs, and substantial performance enhancements.

3 Methodology

3.1 Motivation

Self-attention-based architecture, commonly referred to as transformer, has demonstrated noteworthy advancements in the analysis of point clouds [13, 55]. Nonetheless, the transformer architecture [43, 8, 11] exhibits inferior performance in comparison to CNNs when trained from scratch on a midsize dataset like imageNet-1K [53]. In contrast to more readily accessible 2D image data, the acquisition and annotation of 3D point cloud data imposes considerable financial and temporal burdens [27] due to its irregular and inhomogeneous character. This disparity poses a challenge in training transformer-based networks for point cloud analysis. We, therefore, propose to utilize PEFT technology, which demands less training data. However, existing pre-training models predominantly rely on 2D image data. To effectively utilize these pre-trained models, the

calibration of dimensions and attention for 3D point clouds with 2D image prior becomes crucial.

3.2 Preliminaries

2D Visual Transformer. A visual transformer (ViT) 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 the superscript (l) to denote the index of the block. 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) [2] 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 softmax-weighted interactions among the input query, key, and value tokens obtained by three different learnable linear projection weights. 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 $\mathcal{P}_g = \left[\left\{x_{1,j}^P\right\}_{j=1}^k, \left\{x_{2,j}^P\right\}_{j=1}^k, \cdots, \left\{x_{N_s,j}^P\right\}_{j=1}^k\right] \in \mathbb{R}^{N_s\times k\times(d'+C)}$ 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:

$$z_i^{P,(0)} = \texttt{Point\_Embed}\left(\mathcal{X}_i^P\right), \tag{4}$$

where Point_Embed can take various forms such as pointNet [33], pointNeXt [35] and pointMLP [28], 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 [33], 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 [29] to sequence the point embedding:

$$\mathcal{O} = \texttt{Morton\_Order}\left(\mathcal{P}_s\right), \tag{5}$$

where $\mathcal{O} \in \mathbb{R}^{N_s\times 1}$ is the order of input point sets. Morton_Order is achieved by: 1) Representing the coordinates of a point in binary. 2) Interleaving the bits of these binary numbers. 3) Convert the interleaved binary number back to a decimal value, referred to as Morton value (or Z value). The schematic of the Morton-order curve is shown in Figure 4. We sequence the point embedding obtained by Eq (4) according to Morton-order:

$$\mathcal{Z}_s^P = \mathcal{Z}^P\left[\mathcal{O}\right]. \tag{6}$$

For simplicity, we omit the superscript indicating which block the input belongs to. Subsequently, 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 [53]. In comparison to image 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 models. PEFT involves the freezing of the pre-trained backbone that is 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 [20, 3], few-shot [21], or long-tailed [9], as PEFT equips the model with knowledgeable priors. AdapterFormer [5] 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 point tokens alongside the 2D image attention, namely introducing a trainable bottleneck module. Formally, the calibration of point embeddings is calculated as follows:

$$\begin{cases} \tilde{z}_i^{P,(l)} = \texttt{MSA}^l\left(z_i^{P,(l-1)}\right) + z_i^{P,(l-1)} \\ \hat{z}^{P,(l)} = \texttt{ReLU}\left(\texttt{LN}(\tilde{z}_i^{P,(l)}) \cdot \mathbf{W}_{\text{enc}}\right) \cdot \mathbf{W}_{\text{dec}} \\ z_i^{P,(l)} = \texttt{MLP}\left(\texttt{LN}(\tilde{z}_i^{P,(l)})\right) + s\cdot\hat{z}^{P,(l)} + z_i^{P,(l-1)} \end{cases}, \tag{7}$$

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 PointFormer is shown in Figure 3.

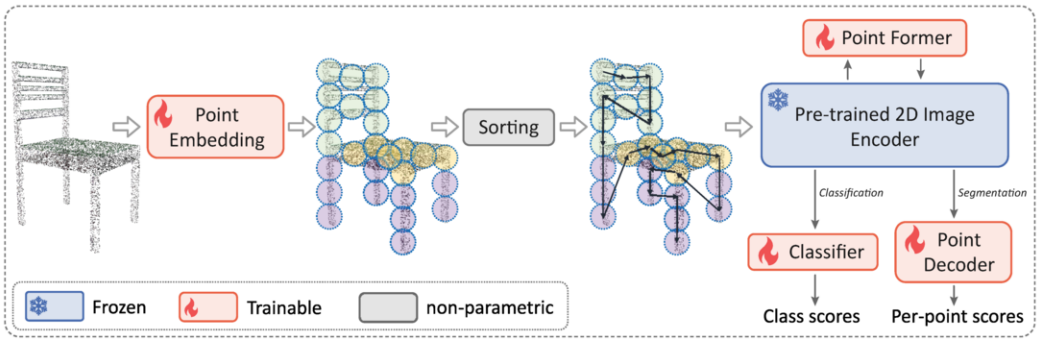


Figure 2: The pipeline of our proposed APF.

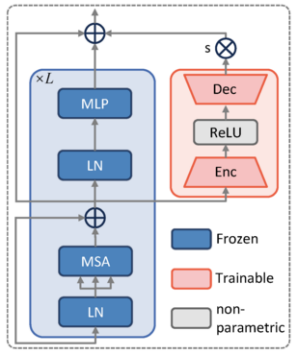


Figure 3: The framework of PointFormer.

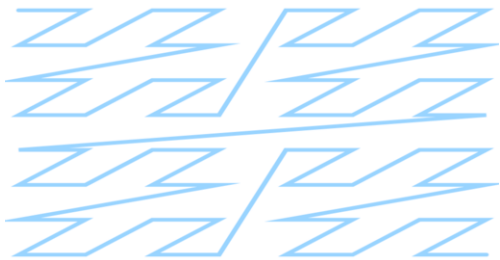


Figure 4: Schematic of Morton-order curve.

In this way, leveraging spatial relationships by 2D attention mechanisms is achieved by: 1) Since the image tokens fed into the 2D model are arranged in order, we sort the point embeddings according to their corresponding point patch centroids utilizing Morton code. This process facilitates better attention adaption. 2) PointFormer is utilized to refine attention discrepancies arising from variations in datasets and data structures.

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. For clarity, we use z_{cls} as a shorthand notation for $z_{\text{cls}}^{P,(l)}$. The predicted logit of each class is given by the softmax of the final linear layer:

$$p_i = \frac{e^{w_i \cdot z_{\text{cls}}}}{\sum_{j=1}^{C} e^{w_j \cdot z_{\text{cls}}}}, \tag{8}$$

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

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. Similarly to classification, the softmax cross-entropy is employed as the loss function.

The overall pipeline of APF is shown in Figure 2.

3.6 Comparison with Existing methods.

The principal disparity between APF and existing methods lies in that APF demonstrates the viability of adapting 2D priors to 3D feature space with minimal training parameters, rather than relying on specific network architectures. APF essentially executes *"2D alignment to 3D"*. The Morton order in the sequencing step aims to mimic the ordered image tokens fed in ViT. The PointFormer module facilitates the adaptation of prior attention in pre-trained ViT. Most existing methods perform *"3D alignment to 2D"*. For example, I2P-MAE [54] leverages 2D knowledge by projecting 3D point clouds onto multiple corresponding 2D images. P2P [47] transforms the 3D point cloud of an object into a single RGB image.

In addition, APF employ 2D pre-trained models from divergent perspectives compared to existing methods: input and model, respectively. For example, P2P focuses on the transformation of 3D point clouds into 2D images at the input level. In contrast, APF incorporates PointFormer at the model level, aiming to adapt the self-attention mechanisms (or feature information) embedded within the 2D priors to accommodate features extracted from point clouds.

Moreover, existing methods, such as ACT [10], point-MAE [32], and I2P-MAE [54], to name a few, necessitate retraining an additional transformer-based network, resulting in additional computational overhead. Conversely, the training parameters in APF consist of the point embedding and PointFormer modules, which have relatively smaller parameter sizes.

4 Experiment

4.1 Datasets and Basic Settings

Datasets. We perform object classification tasks on ModelNet40 [48] and ScanObjectNN [41]. For part segmentation, we utilize ShapeNet-Part [50]. ModelNet40 is an extensively employed 3D dataset comprising 12,311 CAD models distributed across 40 object categories. We follow the official split with 9,843 objects for training and 2,468 for evaluation for a fair comparison. ScanObjectNN is a challenging dataset with inherent scan noise and occlusion, which is sampled from the real world with a comprehensive collection of 15,000 scanned objects spanning 15 distinct classes. Following previous works, we perform experiments on three variants: OBJ-BG, OBJ-ONLY, and PB-T50-RS. ShapeNetPart is a meticulously annotated 3D dataset covering 16 shape categories selected from the ShapeNet dataset. This dataset is annotated with part-level labels from 50 classes and each category is characterized by 2 to 6 distinct parts.

Implementation Details. We follow the settings in [47] 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}. For point embedding, we explore a lightweight PointNet. The ViT-Base (ViT-B) version [11] is utilized as the pre-trained 2D model in the experiments.

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	Acc.(%)
DNN-based model		
PointNet [33]	N/A	89.2
PointNet-OcCo [44]	3D	90.1
PointNet++ [34]	N/A	90.5
DGCNN [45]	N/A	92.9
DGCNN-OcCo [44]	3D	93.0
KPConv [40]	N/A	92.9
PAConv [49]	N/A	93.9
PointMLP [28]	N/A	94.1
Transformer-based model		
Transformer [43]	N/A	91.4
Transformer-OcCo [44]	3D	92.1
Point Transformer [55]	N/A	93.7
PCT [13]	N/A	93.2
Point-BERT [52]	3D	93.2
Point-MAE [32]	3D	93.8
P2P* [47]	2D	92.4
Joint-MAE [15]	3D	94.0
APF (ours) †	2D	**94.2**

Table 2: Object classification results on ScanObjectNN. "Trans." abbreviates Transformer. *: same with Table 1. †: *w. PointNet* means that using a lightweight PointNet for point embedding. ‡: *w. PointMLP* means that using PointMLP for point embedding.

Methods	Pre-trained modality	OBJ-BG	OBJ-ONLY	PB-T50-RS
DNN-based model				
PointNet [33]	N/A	73.8	79.2	68.0
PointNet-OcCo [44]	3D	-	-	80.0
PointNet++ [34]	N/A	82.3	84.3	77.9
DGCNN [45]	N/A	82.8	86.2	78.1
DGCNN-OcCo [44]	3D	-	-	83.9
PRA-Net [6]	N/A	-	-	82.1
PointMLP	N/A	-	-	85.2
Transformer-based model				
Trans. [43]	N/A	79.9	80.6	77.2
Trans.-OcCo [44]	3D	84.9	85.5	78.8
Point-BERT [52]	3D	87.4	88.1	83.1
Point-MAE [32]	3D	90.0	88.3	85.2
P2P* [47]	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**

4.2 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 can be observed that the 3D pre-training OcCo enhances the performance of PointNet and Transformer by 0.9% and 0.7%, respectively. In contrast, APF exhibits superior performance, outperforming PointNet and Transformer by 5.0% and 2.8%, respectively. Furthermore, APF 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. APF consistently outperforms the 3D pre-trained model by a large margin. For example, on the most challenging split, namely PB-T50-RS, the pre-training OcCo improves PointNet by 12.0%. In comparison, APF, employing a PointNet embedding, achieves a remarkable gain of 15.1% over PointNet. Furthermore, APF, when using PointMLP embedding, exhibits superior performance, surpassing PointMLP by 2.6% and outperforming other previous arts. Although 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, on PB-T50-RS, APF with PointMLP outperforms Joint-MAE and Point-MAE by 1.7% and 2.6%, respectively.

Few-shot Classification. To demonstrate the generalization capability of the proposed APF, we conduct experiments under few-shot settings, following the common routine [52, 15]. The "N-way, K-shot" is a conventional configuration, wherein N classes are randomly selected, and each selected class has K training samples and 20 testing samples. We repeat each setting 10 times and report the average performance along with the standard deviation. The results are presented in Table 3. In comparison to both 2D and 3D pre-trained models, APF exhibits superior generalization ability in few-shot learning. For example, APF achieves noteworthy improvements of 2.9%, 2.2%, 3.2%, 3.3% over Transformer-OcCo in four settings. Even in comparison to recently proposed SOTA methods, APF consistently shows superior performance.

Part Segmentation. Following prior works [33, 32, 15], we sample 2,048 points from each input instance and adopt the same segmentation head as Point-MAE [32] and Joint-MAE [15]. The results are shown in Table 4. While APF may not have outperformed SOTA methods across both metrics, it achieves commendable overall performance. In comparison to P2P, which also leverages image priors, APF exhibits superior performance. Compared to JointMAT, APF exhibits slightly lower performance in terms of mIoU_C and mIoU_I. However, it is worth noting that Joint-MAE requires training from scratch, underscoring the comparatively lower computational overhead of APF.

4.3 Further Analysis

Ablation Study. We execute a series of controlled experiments to show the impact of each component of APF. The results are shown in Table 5. We can observe that each module in APF can improve the baseline method, namely PointNet. The integration of Point Sequencer and PointFormer yields the most significant performance enhancement. It is noteworthy that a random PointNet (RPN) serves merely to align dimensions, lacking the ability to extract meaningful features. Nonetheless, APF with RPN still outperforms the vanilla PointNet (92.2% over 89.2%), which shows the potential of image prior in 3D domain.

The Impact of 2D Image Prior. We design an experiment to evaluate the impact of 2D priors on the 3D point cloud analysis. We employ ViT-B, pre-trained on ImageNet-21k, to furnish the 2D prior. A PointNet for input embedding is randomly initialized and, subsequently, its parameters are frozen, leaving only the PointFormer in APF as learnable. Under this setting (APF *w.* RPN in Table 6), the object classification accuracies are 92.2% and 80.1% on ModelNet40 and ScanObjectNN, respectively. APF exhibits considerable performance gains over PointNet by only utilizing randomly initialized embedding projection. Furthermore, it even outperforms the Transformer trained from scratch while incurring significantly lower training costs compared to training from scratch. This observation shows that the prior knowledge embedded in the 2D pre-trained model can significantly aid 3D point cloud analysis, even in scenarios where their training sets and data modalities differ. Moreover, the Point-

Table 3: Few-shot classification results on ModelNet40.

Methods	Pre-trained modality	5-way		10-way	
		10-shot	20-shot	10-shot	20-shot
DNN-based model					
PointNet [33]	N/A	52.0 ± 3.8	57.8 ± 4.9	46.6 ± 4.3	35.2 ± 4.8
PointNet-OcCo [44]	3D	89.7 ± 1.9	$92.4 \pm$ **1.6**	83.9 ± 1.8	$89.7 \pm$ **1.5**
PointNet-CrossPoint [1]	2D	90.9 ± 4.8	93.5 ± 4.4	84.6 ± 4.7	90.2 ± 2.2
DGCNN [45]	N/A	31.6 ± 2.8	40.8 ± 4.6	19.9 ± 2.1	16.9 ± 1.5
DGCNN-OcCo [44]	3D	90.6 ± 2.8	92.5 ± 1.9	$82.9 \pm$ **1.3**	86.5 ± 2.2
DGCNN-CrossPoint [1]	2D	92.5 ± 3.0	94.9 ± 2.1	83.6 ± 5.3	87.9 ± 4.2
Transformer-based model					
Transformer [43]	N/A	87.8 ± 5.2	93.3 ± 4.3	84.6 ± 5.5	89.4 ± 6.3
Transformer-OcCo [44]	3D	94.0 ± 3.6	95.9 ± 2.3	89.4 ± 5.1	92.4 ± 4.6
Point-BERT [52]	3D	94.6 ± 3.1	96.3 ± 2.7	91.0 ± 5.4	92.7 ± 5.1
Point-MAE [32]	3D	96.3 ± 2.5	97.8 ± 1.8	**92.6** ± 4.1	95.0 ± 3.0
Joint-MAE [15]	3D	96.7 ± 2.2	97.9 ± 1.8	**92.6** ± 3.7	95.1 ± 2.6
APF (ours)	2D	**96.9** $\pm$ **1.8**	**98.1** ± 1.8	**92.6** ± 2.4	**95.7** ± 1.6

Table 4: Part segmentation results on ShapeNetPart. mIoU_C (%) is the mean of class IoU. mIoU_I (%) is the mean of instance IoU. "Trans." abbreviates for Transformer.

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 [33]	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++ [34]	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 [45]	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 [40]	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 [49]	84.6	86.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PointMLP [28]	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. [43]	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. [55]	83.7	**86.6**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
PCT [13]	-	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 [44]	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 [52]	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 [32]	-	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* [47]	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	86.1	83.6	84.8	85.4	79.8	91.3	77.0	91.4	88.4	84.4	95.5	76.3	95.3	82.5	59.5	76.1	83.5

Table 5: Impact of each component. The results are obtained on ModelNet40 dataset. RPN: random PointNet, means that the PointNet is frozen with the randomly initialized parameters.

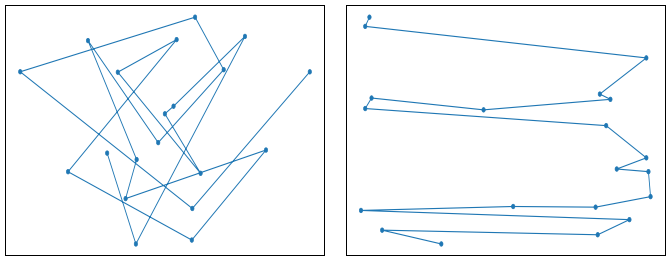
Point Embedding	Point Sequencer	PointFormer	Acc. (%)
PointNet	✗	✗	89.2 (base)
PointNet	✓	✗	93.2 (↑ 4.0)
PointNet	✗	✓	93.5 (↑ 4.3)
PointNet	✓	✓	**94.2** (↑ 5.0)
RPN	✓	✓	92.2 (↑ 3.0)

Former proves beneficial in calibrating 2D-3D attention. Involvement of the point cloud projection in training, as indicated by APF with TPN in Table 6, further improves the performance of APF. This demonstrates the essential role of a representative point embedding in fully leveraging the potential offered by 2D prior knowledge.

Technical details for Morton-order. Sequencing is achieved by Morton code [29]. In detail, the process begins by selecting one point embedding as the initial point. Next, the coordinates of the center points corresponding to point embeddings are encoded into one-dimensional space using Morton code, and subsequently sorted to determine their order. This Morton order, also called Z-order ensures that point embeddings from the closest coordinates are adjacent. Figure 5 shows an example of the comparison between ordered and disordered point clouds. We randomly select 20 points for clear visualization. Figure 6 shows the Z-order sorting in 3D space.

Feature Distributions Visualization. We use t-SNE [42] to visualize feature distributions, which is shown in Figure 7. When the dimensions of point clouds are aligned using a random PointNet, features from different classes overlap, as shown in Figure 7a. In comparison, the aligned embedding obtained by random PointNet can be evidently separated through PointFormer (PF), as shown in Figure 7b. This underscores the efficacy of image priors for point cloud analysis. Similarly, APF further improves the separation of features obtained by the trained PointNet, as shown in Figures 7c and 7d.

Quantity of Trainable Parameters. Table 7 compares the number of trainable parameters with SOTA methods. In contrast to P2P, our method introduces more parameters during point embedding, yet yields a performance improvement. In contrast, APF significantly decreases the number of parameters compared to Point-MAE and Point-BERT. Joint-MAE that achieves SOTA results in part segmen-



(a) Unordered Points (b) Ordered Points

Figure 5: Comparison between ordered and unordered. (20 points are selected for clear visualization.)

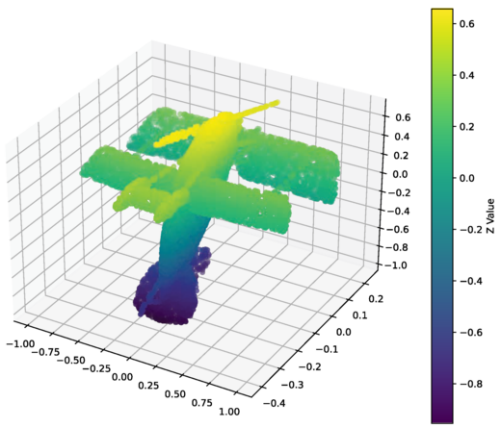
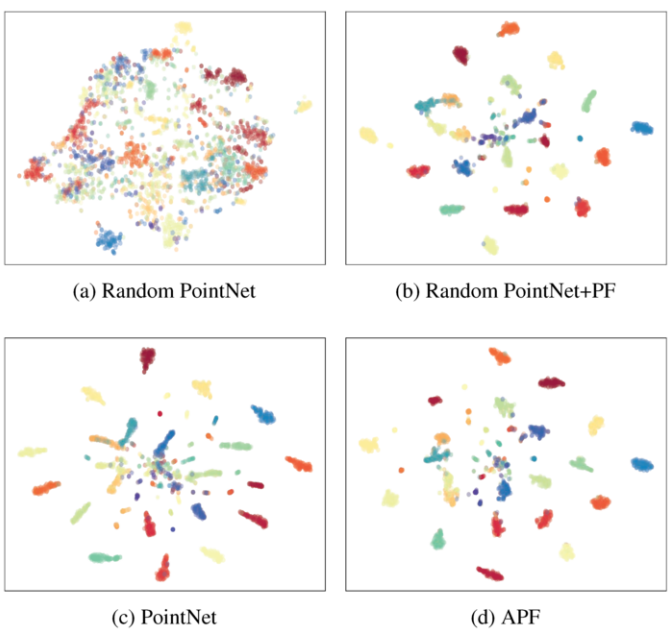


Figure 6: Z-order sorting in 3D Space



(a) Random PointNet (b) Random PointNet+PF

(c) PointNet (d) APF

Figure 7: T-SNE visualization of feature distributions. We show the results on the test set of ModelNet40.

tation needs to train a transformer-based network with two branches from scratch, followed by fine-tuning for downstream tasks. In contrast, our method requires only direct fine-tuning for downstream tasks, leading to fewer trainable parameters and reduced training costs. This reduction in parameters, however, is accompanied by a marginal decrease in performance on specific datasets such as ShapeNetPart, which will be the focus of our future research efforts.

Table 6: Comparison w.r.t. different training strategies. RPN, short for random PointNet. TPN, short for trained PointNet, means that the parameters of the lightweight PointNet are also updated during training. ‡: PB-T50-RS is utilised.

Method	Dataset	Acc. (%)
PointNet	ModelNet40	89.2
Transformer	ModelNet40	91.4
APF *w. RPN*	ModelNet40	92.2 (↑ 3.0)
APF *w. TPN*	ModelNet40	94.2 (↑ 5.0)
PointNet	ScanObjectNN‡	68.0
Transformer	ScanObjectNN‡	77.2
APF *w. RPN*	ScanObjectNN‡	80.1 (↑ 12.1)
APF *w. TPN*	ScanObjectNN‡	83.1 (↑ 15.1)

Table 7: Comparison with existing methods w.r.t. trainable parameters number. The results are on ModelNet40. “Pre-tr. Mod." and “# Tr. param." are short for “pre-training modality" and “parameters number", respectively. *: Reduced-parameter version of APF.

Method	Pre-tr. Mod.	# Tr. param.	Acc. (%)
PointNet++	N/A	1.4M	90.5
PointMLP	N/A	12.6M	94.1
DGCNN-OcCo	3D	1.8M	93.0
Point-BERT	3D	21.1M	93.2
Point-MAE	3D	21.1M	93.8
P2P	2D	0.25M	92.4
APF (ours)	2D	5.8M	**94.2**
APF* (ours)	2D	2.4M	93.7

5 Concluding Remarkings

This paper has initially validated the efficacy of 2D image priors on 3D data using a randomly initialized network for dimension alignment. The finding demonstrates that pre-trained 2D models can contribute to the analysis of point clouds. Then, we have proposed the APF framework for fine-tuning the 2D pre-trained visual model on 3D point cloud datasets. APF consists of a point embedding network for aligning 3D and 2D dimensions, a point sequencer for sorting 3D embedding and the PointFormer for calibrating 2D prior self-attention to 3D embedding space. APF facilitates the fine-tuning of 2D pre-trained models for 3D point cloud analysis without the need for projecting the 3D point cloud onto a 2D image.

Although APF has demonstrated effectiveness, the performance improvement, in comparison with the existing fine-tuning 2D pre-trained model, is accompanied by a modest increase in the number of training parameters. This will be a research focus for improvement in our future work.

Acknowledgments

This work was supported in parts by NSFC (62306181, U21B2023, U2001206), Guangdong Basic and Applied Basic Research Foundation (2023B1515120026, 2023A1515110090, 2024A1515010163), DEGP Innovation Team (2022KCXTD025), Shenzhen Science and Technology Program (RCBS20231211090659101), NSFC/RGC (N_HKBU214/21), RGC GRF (12201321, 12202622, 12201323), RGC SRFS (SRFS2324-2S02).

References

[1] M. Afham, I. Dissanayake, D. Dissanayake, A. Dharmasiri, K. Thilakarathna, and R. Rodrigo. Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In *CVPR*, pages 9902–9912, June 2022.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] H. Bahng, A. Jahanian, S. Sankaranarayanan, and P. Isola. Exploring visual prompts for adapting large-scale models. *arXiv preprint arXiv:2203.17274*, 2022.

[4] A. Chen, Y. Yao, P.-Y. Chen, Y. Zhang, and S. Liu. Understanding and improving visual prompting: A label-mapping perspective. In *CVPR*, pages 19133–19143, 2023.

[5] S. Chen, C. Ge, Z. Tong, J. Wang, Y. Song, J. Wang, and P. Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *NeurIPS*, 35:16664–16678, 2022.

[6] S. Cheng, X. Chen, X. He, Z. Liu, and X. Bai. Pra-net: Point relation-aware network for 3d point cloud analysis. *IEEE TIP*, 30:4436–4448, 2021. doi: 10.1109/TIP.2021.3072214.

[7] J. Choe, C. Park, F. Rameau, J. Park, and I. S. Kweon. Pointmixer: Mlp-mixer for point cloud understanding. In *ECCV*, pages 620–640, 2022.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] B. Dong, P. Zhou, S. Yan, and W. Zuo. LPT: Long-tailed prompt tuning for image classification. In *ICLR*, 2022.

[10] R. Dong, Z. Qi, L. Zhang, J. Zhang, J. Sun, Z. Ge, L. Yi, and K. Ma. Autoencoders as cross-modal teachers: Can pretrained 2d image transformers help 3d representation learning? In *ICLR*, 2023.

[11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[12] G. F. Elsayed, I. Goodfellow, and J. Sohl-Dickstein. Adversarial reprogramming of neural networks. In *ICLR*, 2019.

[13] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021.

[14] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE TPAMI*, 43(12):4338–4364, 2020.

[15] Z. Guo, R. Zhang, L. Qiu, X. Li, and P. Heng. Joint-mae: 2d-3d joint masked autoencoders for 3d point cloud pre-training. In *IJCAI*, pages 791–799, 2023.

[16] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig. Towards a unified view of parameter-efficient transfer learning. In *ICLR*, 2022.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[18] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, pages 2790–2799, 2019.

[19] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

[20] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim. Visual prompt tuning. In *ECCV*, pages 709–727, 2022.

[21] D. Lee, S. Song, J. Suh, J. Choi, S. Lee, and H. J. Kim. Read-only prompt optimization for vision-language few-shot learning. In *CVPR*, pages 1401–1411, 2023.

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

[23] J. Li, M. Pang, Y. Dong, J. Jia, and B. Wang. Graph neural network explanations are fragile. In *ICML*, 2024.

[24] Y. Li, Z. Ge, G. Yu, J. Yang, Z. Wang, Y. Shi, J. Sun, and Z. Li. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. In *AAAI*, volume 37, pages 1477–1485, 2023.

[25] Z. Li, H. Yu, Z. Yang, T. Chen, and N. Akhtar. Ashapeformer: Semantics-guided object-level active shape encoding for 3d object detection via transformers. In *CVPR*, pages 1012–1021, 2023.

[26] Z. Liu, H. Tang, Y. Lin, and S. Han. Point-voxel cnn for efficient 3d deep learning. *NeurIPS*, 32, 2019.

[27] F. Long, T. Yao, Z. Qiu, L. Li, and T. Mei. Pointclustering: Unsupervised point cloud pre-training using transformation invariance in clustering. In *CVPR*, pages 21824–21834, 2023.

[28] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. In *ICLR*, 2022.

[29] G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. *Technical Report*, 1966.

[30] X. Nie, B. Ni, J. Chang, G. Meng, C. Huo, S. Xiang, and Q. Tian. Pro-tuning: Unified prompt tuning for vision tasks. *IEEE TCSVT*, 2023.

[31] M. Pang, B. Wang, M. Ye, Y.-M. Cheung, Y. Zhou, W. Huang, and B. Wen. Heterogeneous prototype learning from contaminated faces across domains via disentangling latent factors. *IEEE TNNLS*, 2024.

[32] Y. Pang, W. Wang, F. E. Tay, W. Liu, Y. Tian, and L. Yuan. Masked autoencoders for point cloud self-supervised learning. In *ECCV*, pages 604–621, 2022.

[33] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017.

[34] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 30, 2017.

[35] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, and B. Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *NeurIPS*, 35:23192–23204, 2022.

[36] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICLR*, pages 8748–8763, 2021.

[37] H. Ran, J. Liu, and C. Wang. Surface representation for point clouds. In *CVPR*, pages 18942–18952, 2022.

[38] J.-X. Shi, T. Wei, Z. Zhou, X.-Y. Han, J.-J. Shao, and Y.-F. Li. Parameter-efficient long-tailed recognition. *arXiv preprint arXiv:2309.10019*, 2023.

[39] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, pages 10529–10538, 2020.

[40] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, pages 6411–6420, 2019.

[41] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, pages 1588–1597, 2019.

[42] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(11), 2008.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[44] H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner. Unsupervised point cloud pre-training via occlusion completion. In *ICCV*, pages 9782–9792, 2021.

[45] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 38(5):1–12, 2019.

[46] Y. Wang, X. Chen, L. Cao, W. Huang, F. Sun, and Y. Wang. Multimodal token fusion for vision transformers. In *CVPR*, pages 12186–12195, 2022.

[47] Z. Wang, X. Yu, Y. Rao, J. Zhou, and J. Lu. P2P: tuning pre-trained image models for point cloud analysis with point-to-pixel prompting. *NeurIPS*, 2022.

[48] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015.

[49] M. Xu, R. Ding, H. Zhao, and X. Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *CVPR*, pages 3173–3182, 2021.

[50] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM TOG*, 35(6):1–12, 2016.

[51] B. X. Yu, J. Chang, H. Wang, L. Liu, S. Wang, Z. Wang, J. Lin, L. Xie, H. Li, Z. Lin, et al. Visual tuning. *arXiv preprint arXiv:2305.06061*, 2023.

[52] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *CVPR*, pages 19313–19322, 2022.

[53] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, pages 558–567, October 2021.

[54] R. Zhang, L. Wang, Y. Qiao, P. Gao, and H. Li. Learning 3d representations from 2d pre-trained models via image-to-point masked autoencoders. In *CVPR*, pages 21769–21780, June 2023.

[55] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *ICCV*, pages 16259–16268, 2021.