# PU-Transformer: Point Cloud Upsampling Transformer

Shi Qiu[1,2], Saeed Anwar[1,2], and Nick Barnes[1]

[1] Australian National University
[2] Data61-CSIRO, Australia
{shi.qiu, saeed.anwar, nick.barnes}@anu.edu.au

**Abstract.** Given the rapid development of 3D scanners, point clouds are becoming popular in AI-driven machines. However, point cloud data is inherently sparse and irregular, causing significant difficulties for machine perception. In this work, we focus on the point cloud upsampling task that intends to generate dense high-fidelity point clouds from sparse input data. Specifically, to activate the transformer's strong capability in representing features, we develop a new variant of a multi-head self-attention structure to enhance both point-wise and channel-wise relations of the feature map. In addition, we leverage a positional fusion block to comprehensively capture the local context of point cloud data, providing more position-related information about the scattered points. As the first transformer model introduced for point cloud upsampling, we demonstrate the outstanding performance of our approach by comparing with the state-of-the-art CNN-based methods on different benchmarks quantitatively and qualitatively.

## 1 Introduction

3D computer vision has been attracting a wide range of interest from academia and industry since it shows great potential in many fast-developing AI-related applications such as robotics, autonomous driving, augmented reality, *etc.* As a basic representation of 3D data, point clouds can be easily captured by 3D sensors [1,2], incorporating the rich context of real-world surroundings.

Unlike well-structured 2D images, point cloud data has inherent properties of *irregularity* and *sparsity*, posing enormous challenges for high-level vision tasks such as point cloud classification [3,4,5], segmentation [6,7,8], and object detection [9,10,11]. For instance, Uy *et al.* [12] fail to classify the real-world point clouds while they apply a pre-trained model of synthetic data; and recent 3D segmentation and detection networks [8,13,14] achieve *worse* results on the distant/smaller objects (*e.g.*, bicycles, traffic-signs) than the closer/larger objects (*e.g.*, vehicles, buildings). If we mitigate point cloud data's *irregularity* and *sparsity*, further improvements in visual analysis can be obtained (as verified in [15]). Thus, point cloud upsampling deserves a deeper investigation.

As a basic 3D low-level vision task, point cloud upsampling aims to generate dense point clouds from sparse input, where the generated data should recover the fine-grained structures at a higher resolution. Moreover, the upsampled

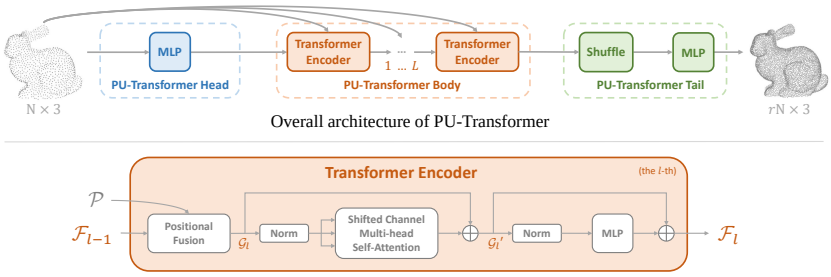


Fig. 1: The details of PU-Transformer. The upper chart shows the overall architecture of the PU-Transformer model containing three main parts: the PU-Transformer head (Sec. 4.1), body (Sec. 4.2), and tail (Sec. 4.3). The PU-Transformer body includes a cascaded set of Transformer Encoders (*e.g.*, $L$ in total), serving as the core component of the whole model. Particularly, the detailed structure of each Transformer Encoder is shown in the lower chart, where all annotations are consistent with Line 3-5 in Alg. 1.

points are expected to lie on the underlying surfaces in a uniform distribution, benefiting downstream tasks for both 3D visual analysis [16,17] and graphic modeling [18,19]. Following the success of Convolution Neural Networks (CNNs) in image super-resolution [20,21,22] and Multi-Layer-Perceptrons (MLPs) in point cloud analysis [3,6], previous methods tended to upsample point clouds via complex network designs (*e.g.*, Graph Convolutional Network [23], Generative Adversarial Network [24]) and dedicated upsampling strategies (*e.g.*, progressive training [25], coarse-to-fine reconstruction [26], disentangled refinement [27]). As far as we are concerned, these methods share a key to point cloud upsampling: learning the representative features of given points to estimate the distribution of new points. Considering that regular MLPs have limited-expression and generalization capability, we need a more powerful tool to extract fine-grained point feature representations for high-fidelity upsampling. To this end, we introduce a succinct transformer model, PU-Transformer, to effectively upsample point clouds following a simple pipeline as illustrated in Fig. 1. The main reasons for adopting transformers to point cloud upsampling are as follows:

*Plausibility in theory.* As the core operation of transformers, self-attention [28] is a set operator [29] calculating long-range dependencies between elements regardless of data order. On this front, self-attention can easily estimate the point-wise dependencies without any concern for the inherent *unorderedness*. However, to comprehensively represent point cloud features, channel-wise information is also shown to be a crucial factor in attention mechanisms [5,11]. Moreover, such channel-wise information enables an efficient upsampling via a simple periodic shuffling [30] operated on the channels of point features, saving complex designs [26,24,27,25] for upsampling strategy. Given these facts, we propose a Shifted Channel Multi-head Self-Attention (SC-MSA) block, which strengthens the point-wise relations in a multi-head form and enhances the channel-wise connections by introducing the overlapping channels between consecutive heads.

*Feasibility in practice.* Since the transformer model was originally invented for natural language processing; its usage has been widely recognized in high-level visual applications for 2D images [31,32,33]. More recently, Chen *et al.* [34] introduced a pre-trained transformer model achieving excellent performance on image super-resolution and denoising. Inspired by the transformer's effectiveness for image-related low-level vision tasks, we attempt to create a transformer-based model for point cloud upsampling. Given the mentioned differences between 2D images and 3D point clouds, we introduce the Positional Fusion block as a replacement for positional encoding in conventional transformers: on the one hand, local information is aggregated from both the *geometric* and *feature* context of the points, implying their 3D positional relations; on the other hand, such *local* information can serve as complementary to subsequent self-attention operations, where the point-wise dependencies are calculated from a *global* perspective.

*Adaptability in various applications.* Transformer-based models are considered as a luxury tool in computer vision due to the huge consumption of data, hardware, and computational resources. However, our PU-Transformer can be easily trained with a *single* GPU in a few hours, retaining a similar model complexity to regular CNN-based point cloud upsampling networks [35,25,27]. Moreover, following a patch-based pipeline [25], the trained PU-Transformer model can effectively and flexibly upsample different types of point cloud data, including but not limited to regular object instances or large-scale LiDAR scenes (as shown in Fig. 3, 4 and 6). Starting with the upsampling task in low-level vision, we expect our approach to transformers will be affordable in terms of resource consumption for more point cloud applications. Our main contributions are:

- To the best of our knowledge, we are the first to introduce a transformer-based model[3] for point cloud upsampling.
- We quantitatively validate the effectiveness of the PU-Transformer by significantly outperforming the results of state-of-the-art point cloud upsampling networks on two benchmarks using three metrics.
- The upsampled visualizations demonstrate the superiority of PU-Transformer for diverse point clouds.

## 2 Related Work

**Point Cloud Networks:** In early research, the projection-based methods [36,37] used to project 3D point clouds into multi-view 2D images, apply regular 2D convolutions and fuse the extracted information for 3D analysis. Alternatively, discretization-based approaches [38] tended to convert the point clouds to voxels [39] or lattices [40], and then process them using 3D convolutions or sparse tensor convolutions [41]. To avoid context loss and complex steps during data conversion, the point-based networks [3,6,4] directly process point cloud data via MLP-based operations. Although current mainstream approaches in point cloud upsampling prefer utilizing MLP-related modules, in this paper, we focus on an

[3]The project page is: https://github.com/ShiQiu0419/PU-Transformer.

advanced transformer structure [28] in order to further enhance the point-wise dependencies between known points and benefit the generation of new points.

**Point Cloud Upsampling:** Despite the fact that current point cloud research in low-level vision [35,42] is less active than that in high-level analysis [3,8,9], there exists many outstanding works that have contributed significant developments to the point cloud upsampling task. To be specific, PU-Net [35] is a pioneering work that introduced CNNs to point cloud upsampling based on a PointNet++ [6] backbone. Later, MPU [25] proposed a patch-based upsampling pipeline, which can flexibly upsample the point cloud patches with rich local details. In addition, PU-GAN [24] adopted the architecture of Generative Adversarial Networks [43] for the generation problem of high-resolution point clouds, while PUGeo-Net [44] indicated a promising combination of discrete differential geometry and deep learning. More recently, Dis-PU [27] applies disentangled refinement units to gradually generate the high-quality point clouds from coarse ones, and PU-GCN [23] achieves good upsampling performance by using graph-based network constructions [4]. Moreover, there are some papers exploring *flexible-scale* point cloud upsampling via meta-learning [15], self-supervised learning [45], decoupling ratio with network architecture [46], or interpolation [47], *etc.* As the first work leveraging transformers for point cloud upsampling, we focus on the effectiveness of PU-Transformer in performing the fundamental *fixed-scale* upsampling task, and expect to inspire more future work in relevant topics.

**Transformers in Vision:** With the capacity in parallel processing as well as the scalability to deep networks and large datasets [48], more visual transformers have achieved excellent performance on image-related tasks including either low-level [49,34] or high-level analysis [32,33,31,50]. Due to the inherent gaps between 3D and 2D data, researchers introduce the variants of transformer for point cloud analysis [51,52,53,54,55], using vector-attention [29], offset-attention [56], and grid-rasterization [57], *etc.* However, since these transformers still operate on an overall classical PointNet [3] or PointNet++ architecture [6], the improvement is relatively limited while the computational cost is too expensive for most researchers to re-implement. To simplify the model's complexity and boost its adaptability in point cloud upsampling research, we only utilize the general structure of transformer encoder [32] to form the body of our PU-Transformer.

## 3 Methodology

### 3.1 Overview

As shown in Fig. 1, given a sparse point cloud $\mathcal{P} \in \mathbb{R}^{N\times 3}$, our proposed PU-Transformer can generate a dense point cloud $\mathcal{S} \in \mathbb{R}^{rN\times 3}$, where $r$ denotes the upsampling scale. Firstly, the PU-Transformer head extracts a preliminary feature map from the input. Then, based on the extracted feature map and the inherent 3D coordinates, the PU-Transformer body gradually encodes a more comprehensive feature map via the cascaded Transformer Encoders. Finally, in the PU-Transformer tail, we use the shuffle operation [30] to form a dense feature map and reconstruct the 3D coordinates of $\mathcal{S}$ via an MLP.

**Algorithm 1:** PU-Transformer Pipeline

**input:** a sparse point cloud $\mathcal{P} \in \mathbb{R}^{N\times 3}$
**output:** a dense point cloud $\mathcal{S} \in \mathbb{R}^{rN\times 3}$
# PU-Transformer Head
$\mathcal{F}_0 = \mathrm{MLP}(\mathcal{P})$
# PU-Transformer Body
**for** *each Transformer Encoder* **do**
  # $l = 1 \dots L$
  # the $l$-th Transformer Encoder
  $\mathcal{G}_l = \textbf{PosFus}(\mathcal{P}, \mathcal{F}_{l-1})$;
  $\mathcal{G}_l{}' = \textbf{SC-MSA}\big(\mathrm{Norm}(\mathcal{G}_l)\big) + \mathcal{G}_l$;
  $\mathcal{F}_l = \mathrm{MLP}\big(\mathrm{Norm}(\mathcal{G}_l{}')\big) + \mathcal{G}_l{}'$;
**end for**
# PU-Transformer Tail
$\mathcal{S} = \mathrm{MLP}\big(\mathrm{Shuffle}(\mathcal{F}_L)\big)$

In Alg. 1, we present the basic operations that are employed to build our PU-Transformer. As well as the operations ("MLP" [3], "Norm" [58], "Shuffle" [30]) that have been widely used in image and point cloud analysis, we propose two novel blocks targeting a transformer-based point cloud upsampling model *i.e.*, the Positional Fusion block ("**PosFus**" in Alg. 1), and the Shifted-Channel Multi-head Self-Attention block ("**SC-MSA**" in Alg. 1). In the rest of this section, we introduce these two blocks in detail. Moreover, for a compact description, we only consider the case of an *arbitrary* Transformer Encoder; thus, in the following, we discard the subscripts that are annotated in Alg. 1 denoting a Transformer Encoder's specific index in the PU-Transformer body.

### 3.2 Positional Fusion

Usually, a point cloud consisting of $N$ points has two main types of context: the 3D coordinates $\mathcal{P} \in \mathbb{R}^{N\times 3}$ that are explicitly sampled from synthetic meshes or captured by real-world scanners, showing the original geometric distribution of the points in 3D space; and the feature context, $\mathcal{F} \in \mathbb{R}^{N\times C}$, that is implicitly encoded by convolutional operations in $C$-dimensional embedding space, yielding rich latent clues for visual analysis. Older approaches [35,25,24] to point cloud upsampling generate a dense point set by heavily exploiting the encoded features $\mathcal{F}$, while recent methods [44,23] attempt to incorporate more geometric information. As the core module of the PU-Transformer, the proposed Transformer Encoder leverages a Positional Fusion block to encode and combine both the given $\mathcal{P}$ and $\mathcal{F}$[4] of a point cloud, following the local geometric relations between the scattered points.

Based on the metric of *3D-Euclidean distance*, we can search for neighbors $\forall p_j \in Ni(p_i)$ for each point $p_i \in \mathbb{R}^3$ in the given point cloud $\mathcal{P}$, using the k-nearest-neighbors (knn) algorithm [4]. Coupled with a grouping operation, we thus obtain a matrix $\mathcal{P}_j \in \mathbb{R}^{N\times k\times 3}$, denoting the 3D coordinates of the neighbors for all points. Accordingly, the relative positions between each point

[4] equivalent to "$\mathcal{F}_{l-1}$" in Alg. 1

and its neighbors can be formulated as:

$$\Delta\mathcal{P} = \mathcal{P}_j - \mathcal{P}, \quad \Delta\mathcal{P} \in \mathbb{R}^{N\times k\times 3}; \tag{1}$$

where $k$ is the number of neighbors. In addition to the neighbors' relative positions showing each point's local detail, we also append the centroids' positions in 3D space, indicating the global distribution for all points. By duplicating $\mathcal{P}$ in a dimension expanded $k$ times, we concatenate the local *geometric* context:

$$\mathcal{G}_{geo} = \text{concat}\big[\underset{k}{\text{dup}}(\mathcal{P}); \Delta\mathcal{P}\big] \in \mathbb{R}^{N\times k\times 6}. \tag{2}$$

Further, for the feature matrix $\mathcal{F}_j \in \mathbb{R}^{N\times k\times C}$ of all searched neighbors, we conduct similar operations (Eq. 1 and 2) as on the counterpart $\mathcal{P}_j$, computing the relative features as:

$$\Delta\mathcal{F} = \mathcal{F}_j - \mathcal{F}, \quad \Delta\mathcal{F} \in \mathbb{R}^{N\times k\times C}; \tag{3}$$

and representing the local *feature* context as:

$$\mathcal{G}_{feat} = \text{concat}\big[\underset{k}{\text{dup}}(\mathcal{F}); \Delta\mathcal{F}\big] \in \mathbb{R}^{N\times k\times 2C}. \tag{4}$$

After the local *geometric* context $\mathcal{G}_{geo}$ and local *feature* context $\mathcal{G}_{feat}$ are constructed, we then fuse them for a comprehensive point feature representation. Specifically, $\mathcal{G}_{geo}$ and $\mathcal{G}_{feat}$ are encoded via two MLPs, $\boldsymbol{\mathcal{M}}_{\Phi}$ and $\boldsymbol{\mathcal{M}}_{\Theta}$, respectively; further, we comprehensively aggregate the local information, $\mathcal{G} \in \mathbb{R}^{N\times C'}$[5], using a concatenation between the encoded two types of local context, followed by a max-pooling function operating over the neighborhoods. The above operations can be summarized as:

$$\mathcal{G} = \max_{k}\Big(\text{concat}\big[\boldsymbol{\mathcal{M}}_{\Phi}(\mathcal{G}_{geo}); \boldsymbol{\mathcal{M}}_{\Theta}(\mathcal{G}_{feat})\big]\Big). \tag{5}$$

Unlike the local graphs in DGCNN [4] that need to be updated in every encoder based on the *dynamic* relations in embedding space, both of our $\mathcal{G}_{geo}$ and $\mathcal{G}_{feat}$ are constructed (*i.e.*, Eq. 2 and 4) and encoded (*i.e.*, $\boldsymbol{\mathcal{M}}_{\Phi}$ and $\boldsymbol{\mathcal{M}}_{\Theta}$ in Eq. 5) in the same way, following *fixed* 3D geometric relations (*i.e.*, $\forall p_j \in Ni(p_i)$ defined upon *3D-Euclidean distance*). The main benefits of our approach can be concluded from two aspects: (i) it is practically efficient since the expensive knn algorithm just needs to be conducted once, while the searching results can be utilized in all Positional Fusion blocks of the PU-Transformer body; and (ii) the local *geometric* and *feature* context are represented in a similar manner following the same metric, contributing to *fairly fusing* the two types of context. A detailed behavior analysis of this block is provided in the supplementary material.

Overall, the Positional Fusion block can not only encode the positional information about a set of unordered points for the transformer's processing, but also aggregate comprehensive local details for accurate point cloud upsampling.

[5] equivalent to "$\mathcal{G}_l$" in Alg. 1

**Algorithm 2:** Shifted Channel Multi-head Self-Attention (**SC-MSA**)

**input:** a point cloud feature map: $\mathcal{I} \in \mathbb{R}^{N \times C'}$
**output:** the refined feature map: $\mathcal{O} \in \mathbb{R}^{N \times C'}$
**others:** channel-wise split width: $w$
channel-wise shift interval: $d$, $d < w$
the number of heads: $M$

1 $\mathcal{Q} = \text{Linear}(\mathcal{I})$ # Query Mat $\mathcal{Q} \in \mathbb{R}^{N \times C'}$
2 $\mathcal{K} = \text{Linear}(\mathcal{I})$ # Key Mat $\mathcal{K} \in \mathbb{R}^{N \times C'}$
3 $\mathcal{V} = \text{Linear}(\mathcal{I})$ # Value Mat $\mathcal{V} \in \mathbb{R}^{N \times C'}$
4 **for** $m \in \{1, 2, ..., M\}$ **do**
5 $\quad \mathcal{Q}_m = \mathcal{Q}[:, (m-1)d : (m-1)d + w]$;
6 $\quad \mathcal{K}_m = \mathcal{K}[:, (m-1)d : (m-1)d + w]$;
7 $\quad \mathcal{V}_m = \mathcal{V}[:, (m-1)d : (m-1)d + w]$;
8 $\quad \mathcal{A}_m = \text{softmax}(\mathcal{Q}_m {\mathcal{K}_m}^T)$;
9 $\quad \mathcal{O}_m = \mathcal{A}_m \mathcal{V}_m$;
10 **end for**
11 **obtain:** $\{\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_M\}$
12 $\mathcal{O} = \text{Linear}\Big(\text{concat}\big[\{\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_M\}\big]\Big)$

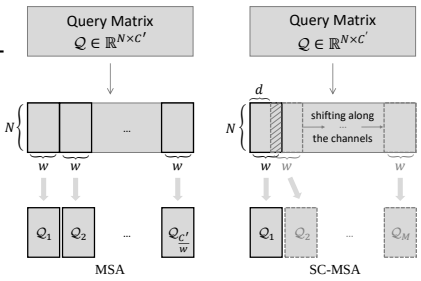


Fig. 2: Examples of how regular MSA [28] and our SC-MSA generate the low-dimensional splits of query matrix $\mathcal{Q}$ for multi-head processing (the same procedure applies to $\mathcal{K}$ and $\mathcal{V}$).

### 3.3 Shifted Channel Multi-head Self-Attention

Different from previous works that applied complex upsampling strategies (*e.g.*, GAN [24], coarse-to-fine [26], task-disentangling [27]) to estimate new points, we prefer generating dense points in a simple way. Particularly, PixelShuffle [30] is a periodic shuffling operation that efficiently reforms the *channels* of each point feature to represent new points without introducing additional parameters. However, with regular multi-head self-attention (MSA) [28] serving as the main calculation unit in transformers, only *point-wise* dependencies are calculated in each independent head of MSA, lacking integration of *channel-related* information for shuffling-based upsampling. To tackle this issue, we introduce a Shifted Channel Multi-head Self-Attention (SC-MSA) block for the PU-Transformer.

As Alg. 2 states, at first, we apply linear layers (denoted as "Linear", and implement as a $1 \times 1$ convolution) to encode the query matrix $\mathcal{Q}$, key matrix $\mathcal{K}$, and value matrix $\mathcal{V}$. Then, we generate low-dimensional splits of $\mathcal{Q}_m, \mathcal{K}_m, \mathcal{V}_m$ for each head. Particularly, as shown in Fig. 2, regular MSA generates the *independent* splits for the self-attention calculation in corresponding heads. In contrast, our SC-MSA applies a window (dashed square) shift along the channels to ensure that any two consecutive splits have an overlap of $(w-d)$ channels (slashed area), where $w$ is the channel dimension of each split and $d$ represents the channel-wise shift interval each time. After generating the $\mathcal{Q}_m, \mathcal{K}_m, \mathcal{V}_m$ for each head in the mentioned manner, we employ self-attention (Alg. 2 steps 8-9) to estimate the point-wise dependencies as the output $\mathcal{O}_m$ of each head. Considering the fact that any two consecutive heads have part of the input in common (*i.e.*, the overlap channels), thus the connections between the outputs $\{\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_M\}$ (Alg. 2 step 11) of multiple heads are established. There are two major benefits of such connections: (i) it is easier to integrate the information between the *connected* multi-head outputs (Alg. 2 step 12), compared to using the *independent* multi-

head results of regular MSA; and (ii) as the overlapping context is captured from the channel dimension, our SC-MSA can further enhance the channel-wise relations in the final output $\mathcal{O}$, better fulfilling an efficient and effective shuffling-based upsampling strategy than only using regular MSA's point-wise information. These benefits contribute to a faster training convergence and a better upsampling performance, especially when we deploy fewer Transformer Encoders. More practical evidence is provided in the supplementary material.

It is worth noting that SC-MSA requires the shift interval to be smaller than the channel-wise width of each split (*i.e.*, $d < w$ as in Alg. 2) for a shared area between any two consecutive splits. Accordingly, the number of heads in our SC-MSA is higher than regular MSA (*i.e.*, $M > C'/w$ in Fig. 2). More implementation detail and the choices of parameters are provided in Sec. 4.2.

## 4 Implementation

### 4.1 PU-Transformer Head

As illustrated in Fig. 1, our PU-Transformer model begins with the head to encode a preliminary feature map for the following operations. In practice, we only use a single layer MLP (*i.e.*, a single $1 \times 1$ convolution, followed by a batch normalization layer [59] and a ReLU activation [60]) as the PU-Transformer head, where the generated feature map size is $N \times 16$.

### 4.2 PU-Transformer Body

To balance the model complexity and effectiveness, empirically, we leverage *five* cascaded Transformer Encoders (*i.e.*, $L = 5$ in Alg. 1 and Fig. 1) to form the PU-Transformer body, where the channel dimension of each output follows: $32 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 256$. Particularly, in each Transformer Encoder, we only use the Positional Fusion block to encode the corresponding channel dimension (*i.e.*, $C'$ in Eq. 5), which remains the same in the subsequent operations. For all Positional Fusion blocks, the number of neighbors is empirically set to $k = 20$ as used in previous works [4,23].

In terms of the SC-MSA block, the primary way of choosing the shift-related parameters is inspired by the Non-local Network [61] and ECA-Net [62]. Specifically, a reduction ratio $\psi$ [61] is introduced to generate the low-dimensional matrices in self-attention; following a similar method, the channel-wise width (*i.e.*, channel dimension) of each split in SC-MSA is set as $w = C'/\psi$. Moreover, since the channel dimension is usually set to a power of 2 [62], we simply set the channel-wise shift interval $d = w/2$. Therefore, the number of heads in SC-MSA becomes $M = 2\psi - 1$. In our implementation, $\psi = 4$ is adopted in all SC-MSA blocks of PU-Transformer.

### 4.3 PU-Transformer Tail

Based on the practical settings above, the input to the PU-Transformer tail (*i.e.*, the output of the last Transformer Encoder) has a size of $N \times 256$. Then, the

periodic shuffling operation [30] reforms the channels and constructs a dense feature map of $rN \times 256/r$, where $r$ is the upsampling scale. Finally, another MLP is applied to estimate the upsampled point cloud's 3D coordinates ($rN \times 3$).

## 5 Experiments

### 5.1 Settings

**Training Details:** In general, our PU-Transformer is implemented using Tensorflow [63] with a single GeForce 2080 Ti GPU running on the Linux OS. In terms of the hyperparameters for training, we heavily adopt the settings from PU-GCN [23] and Dis-PU [27] for the experiments in Tab. 1 and Tab. 2, respectively. For example, we have a batch size of 64 for 100 training epochs, an initial learning rate of $1 \times 10^{-3}$ with a 0.7 decay rate, *etc.* Moreover, we only use the modified Chamfer Distance loss [25] to train the PU-Transformer, minimizing the average closest point distance between the input set $\mathcal{P} \in \mathbb{R}^{N \times 3}$ and the output set $\mathcal{S} \in \mathbb{R}^{rN \times 3}$ for efficient and effective convergence.

**Datasets:** Basically, we apply two 3D benchmarks for our experiments:

- **PU1K:** This is a new point cloud upsampling dataset introduced in PU-GCN [23]. In general, the PU1K dataset incorporates 1,020 3D meshes for training and 127 3D meshes for testing, where most 3D meshes are collected from ShapeNetCore [64] covering 50 object categories. To fit in with the patch-based upsampling pipeline [25], the training data is generated from patches of 3D meshes via Poisson disk sampling. Specifically, the training data includes 69,000 samples, where each sample has 256 input points (low resolution) and a ground-truth of 1,024 points (4× high resolution).
- **PU-GAN Dataset:** This is an earlier dataset that was first used in PU-GAN [24] and generated in a similar way as PU1K but on a smaller scale. To be concrete, the training data comprises 24,000 samples (patches) collected from 120 3D meshes, while the testing data only contains 27 meshes. In addition to the PU1K dataset consisting of a large volume of data targeting the basic 4× upsampling experiment, we conduct both 4× and 16× upsampling experiments based on the compact data of the PU-GAN dataset.

**Evaluation Metrics:** As for the testing process, we follow common practice that has been utilized in previous point cloud upsampling works [25,24,27,23]. To be specific, at first, we cut the input point cloud into multiple seed patches covering all the $N$ points. Then, we apply the trained PU-Transformer model to upsample the seed patches with a scale of $r$. Finally, the farthest point sampling algorithm [3] is used to combine all the upsampled patches as a dense output point cloud with $rN$ points. For the 4× upsampling experiments in this paper, each testing sample has a low-resolution point cloud with 2,048 points, as well as a high-resolution one with 8,196 points. Coupled with the original 3D meshes, we quantitatively evaluate the upsampling performance of our PU-Transformer based on three widely used metrics: (i) Chamfer Distance (CD), (ii) Hausdorff Distance [65] (HD), and (iii) Point-to-Surface Distance (P2F). A lower value under these metrics denotes better upsampling performance.

Table 1: Quantitative comparisons (4× Upsampling) to state-of-the-art methods on the *PU1K* dataset [23]. ("**CD**": Chamfer Distance; "**HD**": Hausdorff Distance; "**P2F**": Point-to-Surface Distance. "**Model**": model size; "**Time**": average inference time per sample; "**Param.**": number of parameters. *: self-reproduced results, –: unknown data.)

| Methods | **Model** (MB) | **Time** ($\times 10^{-3}$s) | **Param.** ($\times 10^{3}$) | Results ($\times 10^{-3}$) **CD** ↓ | **HD** ↓ | **P2F** ↓ |
|---|---|---|---|---|---|---|
| PU-Net [35] | 10.1 | 8.4 | 812.0 | 1.155 | 15.170 | 4.834 |
| MPU [25] | 6.2 | 8.3 | 76.2 | 0.935 | 13.327 | 3.551 |
| PU-GACNet [66] | – | – | **50.7** | 0.665 | 9.053 | 2.429 |
| PU-GCN [23] | **1.8** | **8.0** | 76.0 | 0.585 | 7.577 | 2.499 |
| Dis-PU* [27] | 13.2 | 10.8 | 1047.0 | 0.485 | 6.145 | 1.802 |
| **Ours** | 18.4 | 9.9 | 969.9 | **0.451** | **3.843** | **1.277** |

Table 2: Quantitative comparisons to state-of-the-art methods on the *PU-GAN* dataset [24]. (All metric units are $10^{-3}$. The best results are denoted in **bold**.)

| Methods | 4× Upsampling **CD** ↓ | **HD** ↓ | **P2F** ↓ | 16× Upsampling **CD** ↓ | **HD** ↓ | **P2F** ↓ |
|---|---|---|---|---|---|---|
| PU-Net [35] | 0.844 | 7.061 | 9.431 | 0.699 | 8.594 | 11.619 |
| MPU [25] | 0.632 | 6.998 | 6.199 | 0.348 | 7.187 | 6.822 |
| PU-GAN [24] | 0.483 | 5.323 | 5.053 | 0.269 | 7.127 | 6.306 |
| PU-GCN* [23] | 0.357 | 5.229 | 3.628 | 0.256 | 5.938 | 3.945 |
| Dis-PU [27] | 0.315 | 4.201 | 4.149 | **0.199** | 4.716 | 4.249 |
| **Ours** | **0.273** | **2.605** | **1.836** | 0.241 | **2.310** | **1.687** |

### 5.2 Point Cloud Upsampling Results

**PU1K:** Table 1 shows the quantitative results of our PU-Transformer on the PU1K dataset. It can be seen that our approach outperforms other state-of-the-art methods on all three metrics. In terms of the Chamfer Distance metric, we achieve the best performance among all the tested networks, since the reported values of others are all higher than ours of 0.451. Under the other two metrics, the improvements of PU-Transformer are particularly significant: compared to the performance of the recent PU-GCN [23], our approach can almost *halve* the values assessed under both the Hausdorff Distance (HD: 7.577→3.843) and the Point-to-Surface Distance (P2F: 2.499→1.277).

**PU-GAN Dataset:** We also conduct point cloud upsampling experiments using the dataset introduced in PU-GAN [24]. under more upsampling scales. As shown in Table 2, we achieve best performance under all three evaluation metrics for the 4× upsampling experiment. However, in the 16× upsampling test, we (CD: 0.241) are slightly behind the latest Dis-PU network [27] (CD: 0.199) evaluated under the Chamfer Distance metric: the Dis-PU applies two CD-related items as its loss function, hence getting an edge for CD metric only. As for the results under Hausdorff Distance and Point-to-Surface Distance metrics, our PU-Transformer shows significant improvements again, where some values (*e.g.*, P2F in 4×, HD and P2F in 16×) are even lower than *half* of Dis-PU's results.

Table 3: Ablation study of the PU-Transformer's components tested on the *PU1K* dataset [23]. Specifically, models $A_1$-$A_3$ investigate the effects of the Positional Fusion block, models $B_1$-$B_3$ compare the results of different self-attention approaches, and models $C_1$-$C_3$ test the upsampling methods in the tail.

| models | PU-Transformer Body | | PU-Transformer Tail | Results ($\times 10^{-3}$) | | |
|---|---|---|---|---|---|---|
| | Positional Fusion | Attention Type | | **CD** ↓ | **HD** ↓ | **P2F** ↓ |
| $A_1$ | None | SC-MSA | Shuffle | 0.605 | 6.477 | 2.038 |
| $A_2$ | $\mathcal{G}_{geo}$ | SC-MSA | Shuffle | 0.558 | 5.713 | 1.751 |
| $A_3$ | $\mathcal{G}_{feat}$ | SC-MSA | Shuffle | 0.497 | 4.164 | 1.511 |
| $B_1$ | $\mathcal{G}_{geo}$ & $\mathcal{G}_{feat}$ | SA [61] | Shuffle | 0.526 | 4.689 | 1.492 |
| $B_2$ | $\mathcal{G}_{geo}$ & $\mathcal{G}_{feat}$ | OSA [56] | Shuffle | 0.509 | 4.823 | 1.586 |
| $B_3$ | $\mathcal{G}_{geo}$ & $\mathcal{G}_{feat}$ | MSA [28] | Shuffle | 0.498 | 4.218 | 1.427 |
| $C_1$ | $\mathcal{G}_{geo}$ & $\mathcal{G}_{feat}$ | SC-MSA | MLPs [35] | 1.070 | 8.732 | 2.467 |
| $C_2$ | $\mathcal{G}_{geo}$ & $\mathcal{G}_{feat}$ | SC-MSA | DupGrid [25] | 0.485 | 3.966 | 1.380 |
| $C_3$ | $\mathcal{G}_{geo}$ & $\mathcal{G}_{feat}$ | SC-MSA | NodeShuffle [23] | 0.505 | 4.157 | 1.404 |
| **Full** | $\mathcal{G}_{geo}$ & $\mathcal{G}_{feat}$ | SC-MSA | Shuffle | **0.451** | **3.843** | **1.277** |

**Overall Comparison:** The experimental results in Table 1 and 2 indicate the great effectiveness of our PU-Transformer. Moreover, given quantitative comparisons to CNN-based (*e.g.*, GCN [67], GAN [43]) methods under different metrics, we demonstrate the superiority of transformers for point cloud upsampling by only exploiting the fine-grained feature representations of point cloud data.

### 5.3 Ablation Studies

**Effects of Components:** Table 3 shows the experiments that replace PU-Transformer's major components with different options. Specifically, we test three simplified models ($A_1$-$A_3$) regarding the Positional Encoding block output (Eq. 5), where employing both local *geometric* $\mathcal{G}_{geo}$ and *feature* $\mathcal{G}_{feat}$ context (model "Full") provides better performance compared to the others. As for models $B_1$-$B_3$, we apply different self-attention approaches to the Transformer Encoder, where our proposed SC-MSA (Sec. 3.3) block shows higher effectiveness on point cloud upsampling. In terms of the upsampling method used in the PU-Transformer tail, some learning-based methods are evaluated as in models $C_1$-$C_3$. Particularly, with the help of our SC-MSA design, the simple yet efficient periodic shuffling operation (*i.e.*, PixelShuffle [30]) indicates good effectiveness in obtaining a high-resolution feature map.

**Robustness to Noise:** As the PU-Transformer can upsample different types of point clouds, including real scanned data, it is necessary to verify our model's robustness to noise. Concretely, we test the pre-trained models by adding some random noise to the sparse input data, where the noise is generated from a standard normal distribution $\mathcal{N}(0, 1)$ and multiplied with a factor $\beta$. In practice, we conduct the experiments under three noise levels: $\beta = 0.5\%$, $1\%$ and $2\%$. Table 4 quantitatively compares the testing results of state-of-the-art methods. In most tested noise cases, our proposed PU-Transformer achieves the best performance, while Dis-PU [27] shows robustness under the CD metric as explained in Sec. 5.2.

Table 4: The model's robustness to random noise tested on the *PU1K* dataset [23], where the noise follows a normal distribution of $\mathcal{N}(0, 1)$ and $\beta$ is the noise level.

| Methods | $\beta = 0.5\%$ | | | $\beta = 1\%$ | | | $\beta = 2\%$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | **CD** ↓ | **HD** ↓ | **P2F** ↓ | **CD** ↓ | **HD** ↓ | **P2F** ↓ | **CD** ↓ | **HD** ↓ | **P2F** ↓ |
| PU-Net [35] | 1.006 | 14.640 | 5.253 | 1.017 | 14.998 | 6.851 | 1.333 | 19.964 | 10.378 |
| MPU [25] | 0.869 | 12.524 | 4.069 | 0.907 | 13.019 | 5.625 | 1.130 | 16.252 | 9.291 |
| PU-GCN [23] | 0.621 | 8.011 | 3.524 | 0.762 | 9.553 | 5.585 | 1.107 | 13.130 | 9.378 |
| Dis-PU [27] | 0.496 | 6.268 | 2.604 | **0.591** | 7.944 | 4.417 | **0.858** | 10.960 | 7.759 |
| **Ours** | **0.453** | **4.052** | **2.127** | 0.610 | **5.787** | **3.965** | 1.058 | **9.948** | **7.551** |

Table 5: Model Complexity of PU-Transformer using different numbers of Transformer Encoders. (Tested on the *PU1K* dataset [23] with a single GeForce 2080 Ti GPU.)

| # Transformer Encoders | # Parameters | Model Size | Training Speed (per batch) | Inference Speed (per sample) | Results ($\times 10^{-3}$) | | |
|---|---|---|---|---|---|---|---|
| | | | | | **CD** ↓ | **HD** ↓ | **P2F** ↓ |
| $L = 3$ | 438.3k | 8.5M | 12.2s | 6.9ms | 0.487 | 4.081 | 1.362 |
| $L = 4$ | 547.3k | 11.5M | 15.9s | 8.2ms | 0.472 | 4.010 | 1.284 |
| $\boldsymbol{L = 5}$ | 969.9k | 18.4M | 23.5s | 9.9ms | 0.451 | **3.843** | 1.277 |
| $L = 6$ | 2634.4k | 39.8M | 40.3s | 11.0ms | **0.434** | 3.996 | **1.210** |

**Model Complexity:** Generally, our PU-Transformer is a light (<1M parameters) transformer model compared to image transformers [48,33,32] that usually have more than 50M parameters. In particular, we investigate the complexity of our PU-Transformer by utilizing different numbers of the Transformer Encoders. As shown in Table 5, with more Transformer Encoders being applied, the model complexity increases rapidly, while the quantitative performance improves slowly. For a better balance between effectiveness and efficiency, we adopt the model with *five* Transformer Encoders ($L = 5$) in this work. Overall speaking, the PU-Transformer is a powerful and affordable transformer model for the point cloud upsampling task.

### 5.4 Visualization

**Qualitative Comparisons:** The qualitative results of different point cloud upsampling models are presented in Fig. 3 and 4. Since we utilize the self-attention based structure to capture the point-wise dependencies from a global perspective, the PU-Transformer's output can better illustrate the overall contours of input point clouds producing fewer outliers (as shown in the zoom-in views of Fig. 3). Particularly, based on the rich local context encoded by our Positional Fusion block, the PU-Transformer precisely upsamples the real point clouds (compared in Fig. 4), retaining a uniform distribution and much structural detail.

**Upsampling Different Input Sizes:** Fig. 5 shows the results of upsampling different sizes of point cloud data using PU-Transformer. Given a relatively low-resolution point cloud (*e.g.*, 256 or 512 input points), our proposed model is still able to generate dense output with high-fidelity context (*e.g.*, the head/foot of

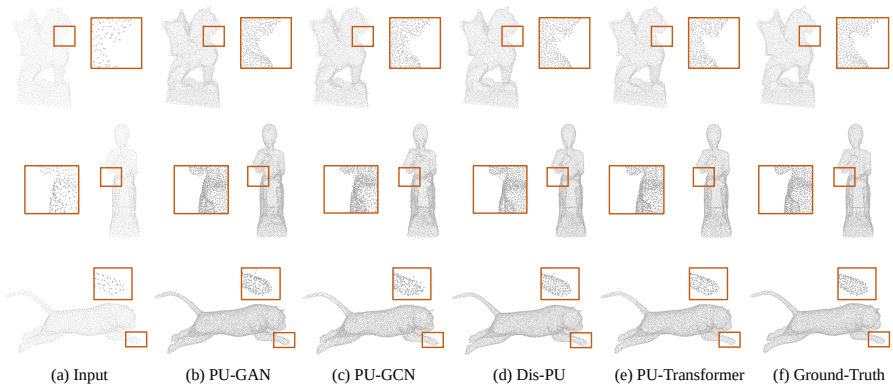


Fig. 3: Comparisons to state-of-the-art methods (PU-GAN [24], PU-GCN [23], Dis-PU [27]) in (4×) upsampling *synthetic* point cloud data using 2048 input points.

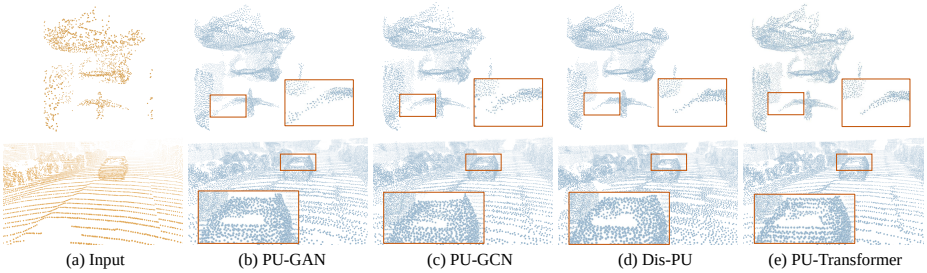


Fig. 4: Comparisons to state-of-the-art methods (PU-GAN [24], PU-GCN [23], Dis-PU [27]) in (4×) upsampling *real* point cloud data from ScanObjectNN [12] dataset and SemanticKITTI [68] dataset.

"Panda"). As the input size increases, the new points are uniformly distributed, covering the main flat areas (*e.g.*, the body of "Panda").

**Upsampling Real Point Clouds:** In addition to Fig. 4, we provide more upsampling results (4× and 16×) on real point cloud samples (*i.e.*, "chair", "office", "room", "street") from *ScanObjectNN* [12], *S3DIS* [69], *ScanNet* [70], and *SemanticKITTI* [68], respectively. As Fig. 6 clearly illustrates, by addressing the sparsity and non-uniformity of raw inputs, not only is the overall quality of point clouds significantly improved, but also the representative features of object instances are enhanced. Particularly, the contours of upsampled object instances (*e.g.*, *tables* in "office/room", *cars* in "street") are clearly distinct from the complex surroundings, obtaining high-fidelity details for visual analysis. More examples for visualization are included in the supplementary material.

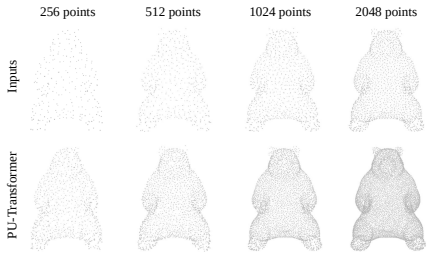


Fig. 5: PU-Transformer's 4× upsampling results, given different sizes of input point cloud data.

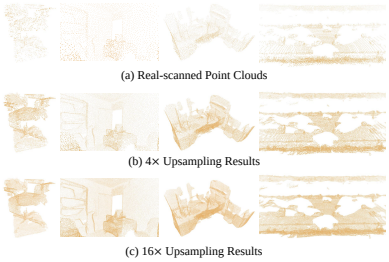


Fig. 6: PU-Transformer's 4× and 16× upsampling results, given different real point clouds.

## 6 Limitations and Future Work

**Upsampling Efficiency:** Compared to the recent works such as Point Transformer [29] (∼7.76M parameters) or PoinTr [55] (∼22.7M), PU-Transformer (∼0.97M) is an efficient transformer for point clouds. However, it still consumes more parameters than some CNN-based counterparts [35,9,23,27] shown in Table 1. As for inference speed, our approach is very close to others due to the succinct pipeline design, while methods that exploit complex network [24], upsampling strategy [27] or geometric calculations [44] will be a bit slower.
**Upsampling Flexibility:** To generate different resolutions of output, our PU-Transformer may require some post-processing such as multiple inference iterations and farthest point sampling [3]. For flexible point cloud upsampling, in future work, we will improve the adaptability of the PU-Transformer's body.
**Future Work:** As a light-weight transformer targeting point clouds, our PU-Transformer has great potential in practice. For example, we could design a *multi-functional* tail to solve different low-level vision problems such as upsampling, completion, and denoising. Moreover, we could further optimize the efficiency of the PU-Transformer in learning fine-grained point feature representations, benefiting the high-level visual analysis of large-scale point clouds.

## 7 Conclusions

This paper focuses on low-level vision for point cloud data in order to tackle its inherent *sparsity* and *irregularity*. Specifically, we propose a novel transformer-based model, PU-Transformer, targeting the fundamental point cloud upsampling task. Our PU-Transformer shows significant quantitative and qualitative improvements on different point cloud datasets compared to state-of-the-art CNN-based methods. By conducting related ablation studies and visualizations, we also analyze the effects and robustness of our approach. In the future, we expect to further optimize its efficiency for real-time applications and extend its adaptability in high-level 3D visual tasks.

## References


1. Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W.: 3-d mapping with an rgb-d camera. IEEE transactions on robotics **30** (2013) 177–187
2. Jaboyedoff, M., Oppikofer, T., Abellán, A., Derron, M.H., Loye, A., Metzger, R., Pedrazzini, A.: Use of lidar in landslide investigations: a review. Natural hazards **61** (2012) 5–28
3. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 652–660
4. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) **38** (2019) 146
5. Qiu, S., Anwar, S., Barnes, N.: Geometric back-projection network for point cloud classification. IEEE Transactions on Multimedia (2021)
6. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. (2017) 5099–5108
7. Qiu, S., Anwar, S., Barnes, N.: Dense-resolution network for point cloud classification and segmentation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). (2021) 3813–3822
8. Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: Randla-net: Efficient semantic segmentation of large-scale point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 11108–11117
9. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2019) 9277–9286
10. Qi, C.R., Chen, X., Litany, O., Guibas, L.J.: Imvotenet: Boosting 3d object detection in point clouds with image votes. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. (2020) 4404–4413
11. Qiu, S., Wu, Y., Anwar, S., Li, C.: Investigating attention mechanism in 3d point cloud object detection. In: International Conference on 3D Vision (3DV), IEEE (2021)
12. Uy, M.A., Pham, Q.H., Hua, B.S., Nguyen, T., Yeung, S.K.: Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2019) 1588–1597
13. Qiu, S., Anwar, S., Barnes, N.: Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (2021) 1757–1767
14. Park, D., Ambrus, R., Guizilini, V., Li, J., Gaidon, A.: Is pseudo-lidar needed for monocular 3d object detection? In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). (2021) 3142–3152
15. Ye, S., Chen, D., Han, S., Wan, Z., Liao, J.: Meta-pu: An arbitrary-scale upsampling network for point cloud. IEEE Transactions on Visualization and Computer Graphics (2021)
16. Liu, Y., Fan, B., Xiang, S., Pan, C.: Relation-shape convolutional neural network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 8895–8904

17. Qiu, S., Anwar, S., Barnes, N.: Pnp-3d: A plug-and-play for 3d point clouds. arXiv preprint arXiv:2108.07378 (2021)
18. Mitra, N.J., Nguyen, A.: Estimating surface normals in noisy point cloud data. In: Proceedings of the nineteenth annual symposium on Computational geometry, ACM (2003) 322–328
19. Mitra, N.J., Gelfand, N., Pottmann, H., Guibas, L.: Registration of point cloud data from a geometric optimization perspective. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM (2004) 22–31
20. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE transactions on pattern analysis and machine intelligence **38** (2015) 295–307
21. Kim, J., Lee, J.K., Lee, K.M.: Accurate image super-resolution using very deep convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 1646–1654
22. Anwar, S., Khan, S., Barnes, N.: A deep journey into super-resolution: A survey. ACM Computing Surveys (CSUR) **53** (2020) 1–34
23. Qian, G., Abualshour, A., Li, G., Thabet, A., Ghanem, B.: Pu-gcn: Point cloud upsampling using graph convolutional networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 11683–11692
24. Li, R., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Pu-gan: a point cloud upsampling adversarial network. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 7203–7212
25. Yifan, W., Wu, S., Huang, H., Cohen-Or, D., Sorkine-Hornung, O.: Patch-based progressive 3d point set upsampling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 5958–5967
26. Liu, X., Liu, X., Han, Z., Liu, Y.S.: Spu-net: Self-supervised point cloud upsampling by coarse-to-fine reconstruction with self-projection optimization. arXiv preprint arXiv:2012.04439 (2020)
27. Li, R., Li, X., Heng, P.A., Fu, C.W.: Point cloud upsampling via disentangled refinement. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 344–353
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. (2017) 5998–6008
29. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2021) 16259–16268
30. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 1874–1883
31. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision, Springer (2020) 213–229
32. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)

33. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). (2021) 10012–10022
34. Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., Ma, S., Xu, C., Xu, C., Gao, W.: Pre-trained image processing transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 12299–12310
35. Yu, L., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Pu-net: Point cloud upsampling network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 2790–2799
36. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE international conference on computer vision. (2015) 945–953
37. Lawin, F.J., Danelljan, M., Tosteberg, P., Bhat, G., Khan, F.S., Felsberg, M.: Deep projective 3d semantic segmentation. In: International Conference on Computer Analysis of Images and Patterns, Springer (2017) 95–107
38. Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M.: Deep learning for 3d point clouds: A survey. IEEE transactions on pattern analysis and machine intelligence (2020)
39. Huang, J., You, S.: Point cloud labeling using 3d convolutional neural network. In: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE (2016) 2670–2675
40. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 2530–2539
41. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 3075–3084
42. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: Pcn: Point completion network. In: 2018 International Conference on 3D Vision (3DV), IEEE (2018) 728–737
43. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
44. Qian, Y., Hou, J., Kwong, S., He, Y.: Pugeo-net: A geometry-centric network for 3d point cloud upsampling. In: European Conference on Computer Vision, Springer (2020) 752–769
45. Zhao, Y., Hui, L., Xie, J.: Sspu-net: Self-supervised point cloud upsampling via differentiable rendering. In: Proceedings of the 29th ACM International Conference on Multimedia. (2021) 2214–2223
46. Luo, L., Tang, L., Zhou, W., Wang, S., Yang, Z.X.: Pu-eva: An edge-vector based approximation solution for flexible-scale point cloud upsampling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2021) 16208–16217
47. Qian, Y., Hou, J., Kwong, S., He, Y.: Deep magnification-flexible upsampling over 3d point clouds. IEEE Transactions on Image Processing **30** (2021) 8354–8367
48. Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in vision: A survey. arXiv preprint arXiv:2101.01169 (2021)
49. Yang, F., Yang, H., Fu, J., Lu, H., Guo, B.: Learning texture transformer network for image super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 5791–5800

50. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)
51. Yew, Z.J., Lee, G.H.: Regtr: End-to-end point cloud correspondences with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2022) 6677–6686
52. Fan, H., Yang, Y., Kankanhalli, M.: Point 4d transformer networks for spatio-temporal modeling in point cloud videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 14204–14213
53. Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., Lu, J.: Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2022) 19313–19322
54. Fan, H., Yang, Y., Kankanhalli, M.: Point spatio-temporal transformer networks for point cloud video modeling. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022)
55. Yu, X., Rao, Y., Wang, Z., Liu, Z., Lu, J., Zhou, J.: Pointr: Diverse point cloud completion with geometry-aware transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2021) 12498–12507
56. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: Pct: Point cloud transformer. Computational Visual Media **7** (2021) 187–199
57. Mazur, K., Lempitsky, V.: Cloud transformers: A universal approach to point cloud processing tasks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2021) 10715–10724
58. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
59. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
60. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Icml. (2010)
61. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 7794–7803
62. Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W., Hu, Q.: Eca-net: Efficient channel attention for deep convolutional neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (2020)
63. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16). (2016) 265–283
64. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
65. Berger, M., Levine, J.A., Nonato, L.G., Taubin, G., Silva, C.T.: A benchmark for surface reconstruction. ACM Transactions on Graphics (TOG) **32** (2013) 1–17
66. Han, B., Zhang, X., Ren, S.: Pu-gacnet: Graph attention convolution network for point cloud upsampling. Image and Vision Computing (2022) 104371
67. Li, G., Muller, M., Thabet, A., Ghanem, B.: Deepgcns: Can gcns go as deep as cnns? In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2019) 9267–9276

68. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of lidar sequences. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 9297–9307
69. Armeni, I., Sax, S., Zamir, A.R., Savarese, S.: Joint 2d-3d-semantic data for indoor scene understanding. arXiv preprint arXiv:1702.01105 (2017)
70. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 5828–5839