APSIPA Transactions on Signal and Information Processing, 2023, 12, e30

Original Paper

An Arbitrary-Rate Upsampling Network based on Light-Transformer for 3D Point Cloud

Yakun Yang, Anhong Wang* and Songyu Zhai

School of Electronic and Information Engineering, Taiyuan University of Science and Technology, Taiyuan, China

ABSTRACT

Point cloud upsampling can provide a dense and uniform representation, which is crucial for improving the quality of 3D reconstruction. While many traditional and deep learning methods of point cloud upsampling have been proposed, their performance need to be further improved. Additionally, most of the existing methods are networks with a single sampling rate, which is inefficient and inconvenient in practical applications. To address these limitations, we propose an arbitrary-rate upsampling network based on a lightweight Transformer for 3D point cloud in this paper. First, a Light-Transformer with skip-attention is designed to extract point cloud features, this method not only has a strong learning ability, but also saves computing and storage resources. Then, after expanding features using 2D grid mechanism and shuffle operation, a coordinate regression module with residual refinement unit is designed to rectify and obtain the precise upsampled point cloud. Next, through the proposed upsampling network based on Light-Transformer, the upsampled point cloud with the maximum sampling rate can be getted, followed by Farthest Point Sampling, we can obtain the point cloud with arbitrary sampling rate. Extensive experiments demonstrate

*Corresponding author: Anhong Wang, ahwang@tyust.edu.cn. This work was supported by the National Natural Science Foundation of China (No.62072325), The University Science and Technology Innovation Project of Shanxi (No.2022L326), The Postgraduate Innovation Project of Shanxi (No.2021Y669).

Received 31 March 2023; Revised 16 May 2023

ISSN 2048-7703; DOI 10.1561/116.00000219

that our proposed method achieves superior upsampling performance, and achieves arbitrary-rate point cloud upsampling.

Keywords: Point cloud upsampling, deep learning, light-transformer, residual refinement unit, arbitrary-rate upsampling.

1 Introduction

With the rapid development of 3D sensors, point cloud has emerged as an important data format. Because point clouds can provide rich information such as shape, scale, and color for 3D geometry, they have been widely used in autonomous driving, virtual reality, robotics, and other fields. However, point clouds obtained by 3D sensors are often sparse, non-uniform, and noisy, which will affect the subsequent processing and practical applications. Therefore, point cloud upsampling is necessary to make it dense, uniform, and evenly distributed on the underlying surface, which can be beneficial for downstream tasks such as 3D visual analysis and graphic modeling [10].

Traditional methods [2, 3, 6, 9, 21] for point cloud upsampling are optimization-based, they heavily rely on local geometric prior knowledge such as normal vectors and curvatures. With the success of deep learning, deep learning-based methods [7, 8, 11, 17, 19, 26–28] have achieved better performance than traditional methods. These deep learning upsampling networks typically include feature extraction, feature expansion, and coordinate regression module to estimate the distribution of generated dense points after learning representative features of sparse point clouds. For example, PU-Net [28] uses PointNet++ [16] to extract features, PU-GCN [17] uses graph convolutional network [14] to extract features. These CNNs and MLPs have limited expression and generalization capabilities, therefore a more powerful model is needed to extract fine-grained point features for high-fidelity upsampling. PCT [4] is the first to use the advanced Transformer [20] for feature extraction, which has more powerful learning abilities than other feature extraction methods. However, Transformer also has the problem of high overhead of computing resources and storage resources. Considering these issues, this paper proposes a lightweight Transformer for feature extraction, which can retain powerful learning capabilities and save resources. Furthermore, relative coordinates can highlight small changes better than absolute coordinates, therefore, a residual refinement unit is added in the coordinate regression module to obtain more accurate 3D coordinates of the point cloud. So far, a Light-Transformer upsampling network with residual refinement unit is proposed to improve the performance of the upsampling method.

In addition, most upsampling networks only train a single network at a time to generate a point cloud with a defined sampling rate. To obtain point clouds with different upsampling rates, multiple networks need to be trained, which consumes too much computational and storage resources, and is not convenient for practical applications. To this end, a network that can achieve arbitrary-rate upsampling is needed. Existing methods [18, 24, 30] have achieved good results, but their structures are relatively complex. Therefore, we propose a simple arbitrary-rate upsampling method for point cloud. Through the former proposed Light-Transformer-based upsampling network, we can obtain a point cloud with a maximum sampling rate, and then use FPS [13] to select any number of points to achieve point cloud upsampling at any rate.

In summary, the contributions of this paper are as follows:

1. We propose a Light-Transformer-based upsampling network for point cloud. This Light-Transformer with skip-attention can better extracts point cloud features and saves computing and memory resources, while the newly added residual refinement unit can obtain more accurate point cloud coordinates.

2. We propose a simple arbitrary-rate upsampling network for point cloud. This network combined with the FPS algorithm, can upsampling point cloud at any rate with only one network and one training.

3. We evaluate our network on public datasets and demonstrate that the proposed upsampling method can achieve good performance and can achieve arbitrary-rate upsampling.

2 Related Work

2.1 Traditional Upsampling Method for Point Cloud

The traditional point cloud upsampling methods are optimization-based, including the following: Alexa *et al.* [2] insert new points at the vertices of the Voronoi diagram of local tangent planes on surfaces to achieve point cloud upsampling. To get rid of the local planar fitting, Lipman *et al.* [9] proposed a local optimization projection operator (LOP) to approximate the surface, where we can resample the point cloud to obtain dense point cloud. Similarly, Huang *et al.* [6] developed an edge-aware point set resampling method, called EAR, where LOP is used to push points away from the edges. Recently, Wu *et al.* [21] proposed a pooling method based on deep point representations to fill in points in missing regions. To achieve point cloud upsampling, Borges *et al.* [3] proposed a super-resolution method via self-correlation. Although these methods can achieve good upsampling results, they heavily rely on

hand-crafted geometric priors of the point cloud. For the complex geometric structures of 3D objects, the performance of these methods is limited by poor geometric priors.

2.2 Deep Learning Upsampling Method for Point Cloud

PU-Net [28] was the first end-to-end deep network for 3D point cloud upsampling. It learned multi-level features for each point, then performed feature expansion, and finally utilized the expanded point features to generate 3D coordinates of dense point cloud. Later, EC-Net [27] learned edge-aware features to better capture the local geometric structure of the point cloud and preserve the sharp features. MPU [26] was a patch-based progressive upsampling network that constructs cascaded upsampling structures on point cloud at different levels to capture detailed information. Point cloud generation [1, 12, 22], which aimed at generating the 3D point cloud directly from latent prior, laid the foundation for various tasks such as shape completion, upsampling, synthesis, etc. PU-GAN [7] was a Generative Adversarial Network (GAN) designed to learn upsampled points distributions of point cloud from latent space. PU-GCN [17], which used graph convolutional networks as feature extractors, also achieved good performance. Dis-PU [8] consisted of two sub-networks to perform generation and optimization tasks respectively. PC2-PU [11] has two new modules named Patch Correlation and Position refinement for point cloud upsampling, which combine the information of adjacent patches and the relative position relationships between points to improve performance and mitigate the influence of noise points and outliers. Most of the above methods use CNNs and MLPs when extracting point cloud features. Because Transformer has shown strong learning capabilities in point cloud, as the first network to combine Transformer with point cloud upsampling, PU-Transformer [19] showed significant quantitative and qualitative improvements over state-of-the-art CNN-based methods on different point cloud datasets, but its structure is relatively complex. In addition to the above supervised learning methods, there are also some point cloud upsampling networks based on self-supervised learning [31] that have become a new research direction.

2.3 Arbitrary Rate Upsampling Method for Point Cloud

For arbitrary-rate upsampling, inspired by the Meta-SR [5] in image processing, Ye et al. proposed Meta-PU [24] for flexible point cloud upsampling, where a meta sub-network is learned to dynamically adjust the weights of the upsampling blocks. Qian et al. designed a neural network Flexible-PU [18], which adaptively learns unified and sorted interpolation weights as well as high-order refinements by analyzing the local geometric structure of input point clouds. In addition to supervised methods, a self-supervised method called SSASPU [30]

based on implicit neural representation for point cloud upsampling can achieve both self-supervised and flexible point cloud upsampling. However, these methods have relatively complex structures and require more computational and storage resources.

3 Proposed Method

3.1 Overview

Based on the research and bottleneck analysis of the above upsampling methods, in this paper, we propose a point cloud upsampling network based on Light-Transformer. An overview of our network is shown in Figure 1. Given a sparse, non-uniform point cloud $P \in \mathrm{R}^{N\times 3}$, our network extracts the point features through a novel Light-Transformer to obtain the features $F \in \mathrm{R}^{N\times d}$, which is then extended to $F_1 \in \mathrm{R}^{rN\times d_1}$ by feature expansion module, where r is the upsampling rate, and finally uses the coordinate regression module with a residual refinement unit to get a dense, uniform upsampled point cloud $Q \in \mathrm{R}^{rN\times 3}$. This upsampling network is trained by a joint loss function including the sampling loss and the regularization loss.

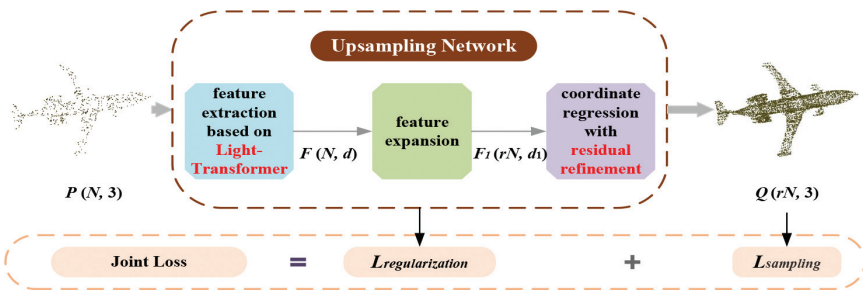


Figure 1: The overview of the point cloud upsampling network based on Light-Transforme.

3.2 Feature Extraction Module based on Light-Transformer

Relying on attention mechanism and parallel processing, Transformer achieved excellent performance in vision tasks. And it has permutation invariance to input, which is very suitable for point cloud with irregular structure. So, in our upsampling network, we will use Transformer to extract the features of point cloud.

Among the existing works, the earliest combination of point cloud and Transformer are PCT [4] and PT [29]. Due to the strong learning ability, their classification accuracy is greatly improved compared with PointNet [15], but

the computational cost and model size are also greatly increased. In order to capture the accurate global context information with limited computational cost and storage space, we propose a lightweight Transformer as the feature extraction module. The structure of the Light-Transformer is shown in Figure 2.

A general Transformer usually consists of Input Embedding, Positional Encoding, Multi-Head Attention, Add&Norm, Feed Forward, etc. In the Multi-Head Attention Layer, every head has performed self-attention calculations, which requires expensive computational costs. But these attention information are highly relevant and have a lot of redundancy, so calculating attention values multiple times causes excessive unnecessary resource consumption. In our network, we address this problem by adding a skip-attention strategy to the Multi-Head Attention layer.

Considering that the 3D coordinates of the point cloud can represent its position information, we first replace the Input Embedding and Positional Encoding in Transformer with a simple Linear layer to obtain the initial features, transform the original point cloud $P \in \mathrm{R}^{N\times 3}$ into features $F_i \in \mathrm{R}^{N\times d}$.

Secondly, in Multi-Head Attention layer, as shown in the lower part of Figure 2, we only perform self-attention calculations in the first attention head. Send the point cloud feature $F_i \in \mathrm{R}^{N\times d}$ obtained by the simple Input Embedding layer into the first attention head, after transformation via Linear layer, we can get the Value $v \in \mathrm{R}^{N\times d}$, the Query $q \in \mathrm{R}^{N\times d}$, and after the transpose operation, the Key $k \in \mathrm{R}^{d\times N}$ is also obtained. Multiply k and q to get the $N \times N$ Attention Map, then multiply v and the Attention Map to get the Attention Value $AV \in \mathrm{R}^{N\times d}$. After getting the Attention Value, make the residual with the input features $F_i \in \mathrm{R}^{N\times d}$, and then through Linear transformation, we can get the features F_1 of the first attention head. In the second attention head, we will no longer perform attention value calculations,

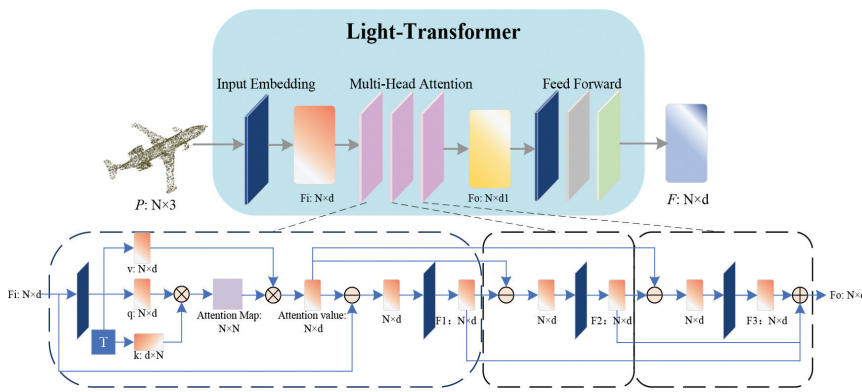


Figure 2: The structure of the Light-Transformer for feature extraction.

but use the Attention Value in the first head. We then make the residuals of F_1 and AV , which are fed into the Linear layer to obtain the output feature F_2 of the second attention head. By analogy, the output features $F_k \in \mathrm{R}^{N\times d}$ of the k-th head is obtained, k is the number of the multi-head. Next, these above features are concat together to get the point cloud features F_o. This process can be expressed by the following Equation (1).

$$\begin{aligned} AR &= \mathrm{softmax}(q \cdot k^T) \cdot v \\ F_1 &= \mathrm{Liner}(F_i - AR) \\ F_2 &= \mathrm{Liner}(F_1 - AR) \\ F_k &= \mathrm{Liner}(F_{k-1} - AR) \\ F_o &= \mathrm{concat}(F_1, F_2, \ldots F_k) \end{aligned} \tag{1}$$

Finally, we input the feature F_o obtained by the Multi-Head Attention layer to the Feed Forward network that composed of Linear layer, Normalization layer and Rectified Linear layer to get the point cloud features $F \in \mathrm{R}^{N\times d}$.

Such a Light-Transformer not only satisfies the powerful learning capabilities of the Transformer, but also saves memory and computing resources. And the attention mechanism can help the network select more important features of point cloud for subsequent tasks.

3.3 Feature Expansion Module

Points and features are interchangeable, we can regard N points with rd-dimensional features as rN points with d-dimensional features, so the feature expansion module can realize the increase of points number.

As shown in Figure 3, we first use the MLPs to get the d_1-dimensional features of N points, and then copy r times to obtain the d_1-dimensional features of rN points. However, such simple convolution and replication will make the generated points too similar, which is not conducive to generate the dense and uniform upsampled point cloud. Therefore, we learn from the 2D grid mechanism in FoldingNet [23] to form a unique 2D vector for each feature, which is then added to the features of each point to obtain different point features $F \in \mathrm{R}^{rN\times(d_1+2)}$. It increases the diversity of the point cloud features and thus makes the individual points subtly different. Finally, we use the Shuffle operation to get the extended features $F_1 \in \mathrm{R}^{rN\times d_1}$.

3.4 Coordinate Regression Module with a Residual Refinement Unit

The coordinate regression module regresses the d_1-dimensional feature F_1 into three-dimensional coordinates and obtains the dense point cloud Q. Considering that the absolute coordinates vary more than the relative offsets in

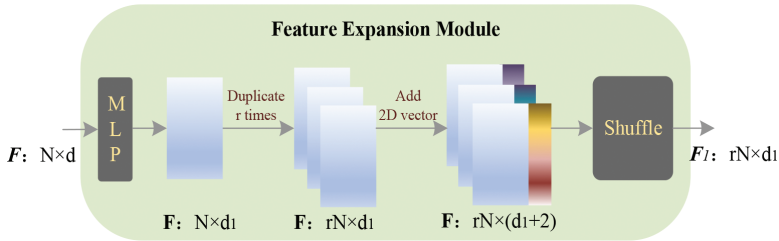


Figure 3: The structure of the feature expansion module.

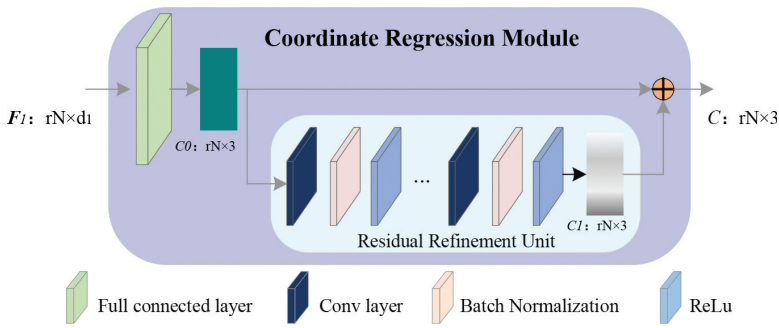


Figure 4: The structure of the coordinate regression module with a residual refinement unit.

3D space, and the residuals can highlight slight variations, in this paper, we propose to add a residual refinement unit to obtain more accurate point cloud coordinates.

As shown in Figure 4, first, through the fully connected layer, the feature $F_1 \in \mathrm{R}^{rN \times d_1}$ is transformed into the 3D coordinates of rN points. At this time, the 3D coordinates C_0 are coarse. To reduce the effect of noise, and generate a better dense point cloud, we add a residual refinement unit to refine the coarse coordinates. The residual refinement unit consists of Convolution layer, Batch Normalization layer, and Rectified Linear Units, which regresses the residual offsets C_1 of each point's coordinates. Finally, add the offsets C_1 obtained by refinement to the coarse coordinates C_0 to get the fine 3D coordinates $C \in R^{rN \times 3}$.

3.5 Loss Function

We propose a joint loss function consisting of the sampling loss and the regularization loss for end-to-end training of upsampling network. As shown in Equation (2), the sampling loss constrains the generated dense point cloud, which includes the reconstruction loss and the repulsion loss. The regularization

loss constrains the parameters in the upsampling network.

$$\begin{aligned} L_{up} &= \lambda_{sampling} L_{sampling} + \lambda_{regularization} L_{regularization} \\ &= \lambda_{rec} L_{rec} + \lambda_{rep} L_{rep} + \lambda_{reg} L_{reg} \end{aligned} \tag{2}$$

where λ is the parameter for balancing each item.

Specifically, as shown in Equation (3), the reconstruction loss is calculated by CD (Chamfer distance) to evaluate the difference between the generated dense point cloud and the ground truth:

$$L_{rec}{=}L_{CD}(Q, \overline{Q}) = \frac{1}{Q} \sum_{q \in Q} \min_{\overline{q} \in \overline{Q}} \|q - \overline{q}\|_2^2 + \frac{1}{\overline{Q}} \sum_{\overline{q} \in \overline{Q}} \min_{q \in Q} \|\overline{q} - q\|_2^2 \tag{3}$$

where, Q is the dense upsampled point cloud, $\overline{Q}$ is the ground truth, $\|\|_2^2$ is a 2-norm. The former of L_{rec} ensures that the upsampled points Q are as close as possible to the ground truth $\overline{Q}$, and the latter ensures that Q are evenly distributed in the ground truth $\overline{Q}$.

The repulsion loss as shown in Equation (4) ensure that the generated points are evenly distributed on the underlying surface:

$$L_{rep} = \sum_{i=o}^{N} \sum_{i' \in K(i)} \eta\left(\|q_{i'} - q_i\|_2\right) w\left(\|q_{i'} - q_i\|_2\right) \tag{4}$$

where, N is the number of generated points, $K(i)$ is the k nearest neighbor set of the point q_i, $\eta(r)$ and $w(r)$ are the decay function. If q_i is too close to other neighbor points, this loss will be big.

The regularization loss is shown in Equation (5):

$$L_{reg} = \|\theta\|_2 \tag{5}$$

where θ refers to the parameters in the upsampling network.

3.6 Arbitrary-Rate Upsampling Network

Now we have proposed a new Light-Transformer-based upsampling network that can get dense point cloud, but the problem is that each training can only get point cloud at a fixed upsampling rate. In practical applications, we may need different point clouds at the same time, which requires to train different upsampling networks with multiple times. Such a solution not only consumes computing and memory resources, but also is inconvenient to apply.

To this end, we propose a simple arbitrary-rate upsampling network. As shown in Figure 5, with the proposed upsampling network based on Light-Transformer, we can get a point cloud $Q_0 \in R^{r_{\max} N \times 3}$ with a maximum

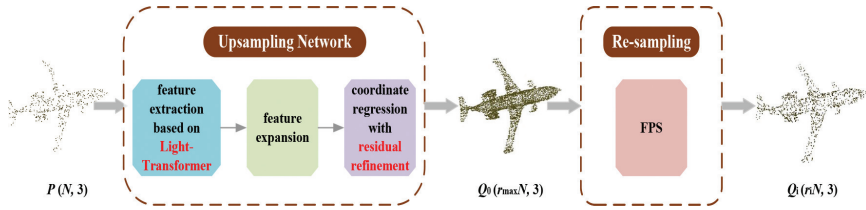


Figure 5: The overview of the arbitrary-rate point cloud upsampling network.

upsampling rate $r_{\max}$, e.g. $r_{\max}$ =16. The point cloud at this time has lots of points, is dense and uniform enough to meet the requirements of upsampling. Afterwards, we use a simple re-sampling module with FPS algorithm to downsample the points from Q_0 to get the arbitrary rate point cloud $Q_i \in R^{r_i N \times 3}$. This simple design enables arbitrary-rate upsampling while consuming less computational and memory resources.

4 Experiments and Discussions

4.1 Experimental Setup

4.1.1 Dataset

We use the PU1K dataset [17] in PU-GCN for experiments, which includes 1147 3D models, 1020 models are used for training and the rest 127 are used for testing. The training set contains 120 models from the dataset of PU-GAN and 900 models from ShapeNet [25] . The testing set contains 27 models from PU-GAN and 100 models from ShapeNet. They cover a wide range of 3D objects, including simple and complex shapes.

4.1.2 Evaluation Metrics

In order to evaluate different upsampling methods objectively and fairly, we use common evaluation metrics including Chamfer Distance(CD), Hausdorff Distance(HD), Point-to-surFace(P2F) Distance. The smaller these distances, the higher the quality of the upsampling point cloud.

4.1.3 Experimental Setup

We use a computer with NVIDIA Quadro RTX 8000 GPU for our experiments, and train the proposed network on the TensorFlow platform for 100 epochs with the batch size of 64. We then choose the Adam optimizer and set the initial learning rate to 0.001. The Linear layer dimension d is 256, the head of the

Table 1: The quantitative results of different upsampling methods on $r = 4$.

Method	CD	HD	P2F	Model Size	Param
PU-Net	1.118×10^{-3}	15.797×10^{-3}	4.961×10^{-3}	10.1M	812×10^3
PU-GAN	0.667×10^{-3}	9.676×10^{-3}	2.485×10^{-3}	9.6M	684×10^3
PU-GCN	0.676×10^{-3}	10.023×10^{-3}	2.675×10^{-3}	1.8M	76×10^3
Dis-PU	0.485×10^{-3}	6.145×10^{-3}	1.802×10^{-3}	13.2M	1047×10^3
PU-Transformer	0.451×10^{-3}	3.834×10^{-3}	1.277×10^{-3}	18.4M	970×10^3
Ours	0.520×10^{-3}	6.450×10^{-3}	2.166×10^{-3}	8.5M	429×10^3

Attention layer k is 4, d_1 is 1024. We set the corresponding hyperparameters of the loss function λ_{rec} , λ_{rep}, λ_{reg} to 100, 10, 1, respectively. Other settings remain the same as PU-GCN.

4.2 Comparsion with Other Different Upsampling Method

We compared our network with state-of-the-art learning-based methods PU-Net, PU-GAN, PU-GCN, Dis-PU and PU-Transformer. For a fair comparison, we trained these networks on the same dataset with the sparse input of 256 points and the upsampling rate r set to 4. We get the quantitative results shown in Table 1.

We can see that our network performs better than PU-Net, PU-GAN, PU-GCN in terms of all metrics CD, HD, P2F. This is due to the fact that our network uses Light-Transformer to extract features, which has a strong learning ability and can select more representative features of point cloud. Secondly, the residual refinement unit makes the upsampled point cloud more accurate and close to the ground truth. But unfortunately, the performance of our method is not as good as Dis-PU and PU-Transformer. We analyzed the reasons may be that Dis-PU decomposes the point cloud upsampling into two sub-networks, a dense generator and a spatial refiner, the dense generator infers a coarse but dense output while the spatial refiner further fine-tunes the coarse output, this is equivalent to performing two upsampling operations. In addition, PU-Transformer uses a new Transformer to extract the features, this Transformer has a more powerful feature learning ability than our proposed Ligh-Transformer due to its Positional Fusion block and Shifted-Channel Multi-head Self-Attention block. However, although the above two methods achieve better performance, they also increase parameters and model size as shown in Table 1. From the experimental results, it can be seen that compared with PU-Net, PU-GAN, Dis-PU, and PU-Transformer, our method has a smaller model size and fewer parameters, which has a great advantage in resource consumption. It is only larger than PU-GCN, but outperforms PU-GCN. Therefore, our method is more practical due to

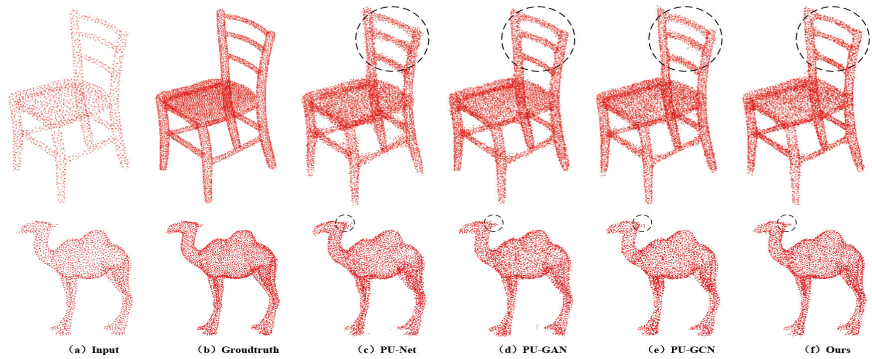


Figure 6: The visualization of the point clouds with different upsampling methods.

trading off upsampling performance and storage resources, and can guarantee comparable performance while reducing model size.

To demonstrate the performance of our network more intuitively, Figure 6 shows the subjective comparison results of different methods.

The first row is the visualization of the chair and the second row is the visualization of the camel. Each row from left to right is the input point cloud, the ground truth, the upsampling results of the PU-Net, PU-GAN, PU-GCN, and ours. It can be clearly seen that compared with other methods, our result is closest to the ground truth, and it can achieve better reconstruction effect in the local details, such as the back of the chair and the ears of the camel. This is because that our network uses Light-Transformer to extract point cloud features, the attention mechanism in Transformer helps the network learn to select more representative and important features and points, and the residual refinement unit helps to obtain more accurate points.

4.3 Comparsion with Arbitrary-Rate Upsampling Method

To demonstrate the effectiveness and simplicity of our proposed arbitrary-rate upsampling method, we compared the single-rate and arbitrary-rate networks. We compare the results of directly upsampling to 4× point cloud using a single-rate network versus first upsampling to 16x and then FPS to 4x using an arbitrary-rate network. As shown in Table 2, there is little difference between the two methods, and even due to upsampling to a high rate, the result is more accurate than upsampling to a relatively low rate. But the single rate network needs to be trained one by one according to the sampling rate, while the arbitrary-rate network only needs to be trained once, the total training time is greatly shortened. This arbitrary-rate network has great significance for practical applications, which can save computing and memory

resources. Regrettably, our arbitrary-rate network is not flexible in arbitrary-rate upsampling, but it only needs to add a re-sampling module based on FPS to the previous upsampling network, so its advantage is simple and convenient.

Table 2: Comparison of the arbitrary rate upsampling network with the single rate network at $r = 4$.

Method	CD	HD	P2F
single-rate network	0.520×10^{-3}	6.450×10^{-3}	2.166×10^{-3}
arbitrary-rate network	0.507×10^{-3}	6.180×10^{-3}	2.009×10^{-3}

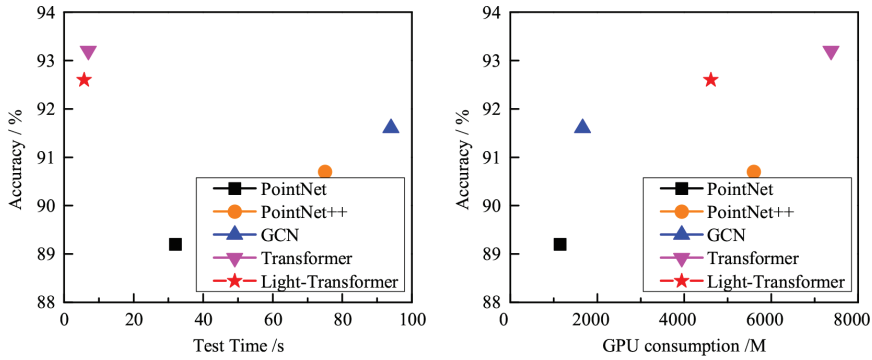
4.4 Ablation Study

In this paper, we proposed Light-Transformer to extract the point cloud features. To illustrate the efficiency of our method, we compared five feature extraction methods, namely PointNet, PointNet++, GCN, Transformer and our Light-Transformer. In order to verify their performance and easy implementation, we apply them to the simplest point cloud task, that is, point cloud classification, to see their overall accuracy. The higher the accuracy, the better the performance of the feature extraction methods. In the point cloud classification task, we use the ModelNet40 as dataset, which contains 12311 3D objects in 40 categories, of which 9843 are used for training and 2468 are used for testing. We also recorded the test time and GPU memory consumption for the entire test dataset to reflect on the efficiency of the method. The experiment results are shown in Table 3.

As shown in Table 3, the Light-Transformer proposed in this paper achieved an overall accuracy of 92.6%, which is higher than PointNet, PointNet++ and GCN. In the case of obtaining an accuracy comparable to that of Transformer, the test time was shorter, and the GPU occupancy was smaller than that of Transformer.

Table 3: Comparison of different feature extraction methods in point cloud classification task.

Method	Accuracy	Test Time	GPU Memory
PointNet	89.2%	32s	1147M
PointNet++	90.7%	75s	5603M
GCN	91.6%	94s	1663M
Transformer	93.2%	7s	7379M
Light-Transformer	92.6%	5.8s	4613M



(a) accuracy vs. test time. (b) accuracy vs. GPU memory.

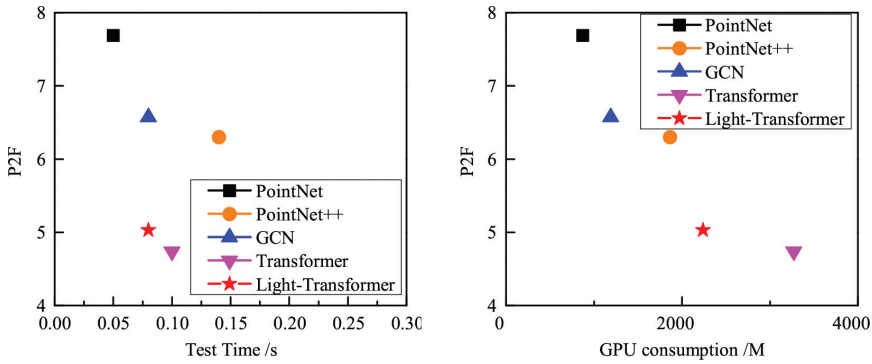
Figure 7: The trade-off accuracy vs. test time and GPU memory consumption.

Table 4: Comparison of different feature extraction methods in point cloud upsampling task.

Method	P2F	Test Time	GPU Memory
PointNet	7.69×10^{-3}	0.05s	871M
PointNet++	6.30×10^{-3}	0.14s	1865M
GCN	6.57×10^{-3}	0.08s	1189M
Transformer	4.74×10^{-3}	0.10s	3271M
Light-Transformer	5.03×10^{-3}	0.08s	2239M

The trade-off of accuracy vs. test time and GPU memory consumption are shown in Figure 7. The closer this method is to the upper left in the figure, the better the performance. We can see our method achieved a significantly better accuracy vs. the test time trade-off and the GPU memory consumption trade-off compared with other methods. With similar accuracy, our method was the fastest. Also, with similar accuracy, our method saved on GPU consumption compared with Transformer. However, the fly in the ointment is, its GPU memory consumption was more than that of PointNet and GCN.

In addition, we apply the above five feature extraction methods to the upsampling task to see the performance, and keep the same feature expansion module and coordinate regression module as PU-Net. We use P2F to evaluate the upsampling performance, the smaller the P2F, the better the upsampling performance. We also use the test time and the GPU memory for the entire test dataset to evaluate the resource consume. In point cloud upsampling, we use the PU-Net dataset, which has 60 different models from the Visionair repository, 40 for training, and the rest 20 for testing. The experimental results are shown in Table 4 and Figure 8.



(a) P2F vs. test time.

(b) P2F vs. GPU memory.

Figure 8: The trade-off P2F vs. test time and GPU memory consumption.

It can be seen from the Table 4 that the P2F value of the upsampling method based on Light-Transformer is lower than that of PointNet, PointNet++, GCN, and it is not much different from Transformer. At the same time, the test time and GPU consumption of the method based on Light-Transformer are also smaller than those of Transformer. The experiment results indicate that when the Light-Transformer proposed in this paper is applied to the point cloud upsampling task, it can obtain relatively high task performance and achieve a better trade-off with time and GPU consumption.

The trade-off of P2F vs. test time and GPU memory consumption are shown in Figure 8. The closer this method is to the lower left in the figure, the better the performance. We can see our method achieved a better upsampling performance vs. the test time trade-off and the GPU memory consumption trade-off compared with other methods.

To further verify our Light-Transformer, we added the ablation experiments to see the effectiveness of different parts of the Transformer. We compared four kinds of networks, first is the complete Transformer, second is the Positional Encoding module and Input Embedding module were replaced by Linear layers while the Multi-Head Attention layer and Feed Forward network remained unchanged with the complete Transformer, third is the Positional Encoding module and Input Embedding module were replaced by Linear layers while the Multi-Head Attention layer was changed to Single-Head Attention layer, and fourth is the Light-Transformer where Positional Encoding module and Input Embedding module were replaced by Linear layers while the Multi-Head Attention layer uses the four heads based on skip-attention mechanism. We then use the simplest point cloud classification task to evaluate the performance of these Transformer, the higher the classification accuracy, the better the Transformer model. The experimental results are shown in Table 5.

Table 5: Ablation study for different parts of the Transformer.

Method	Accuracy	Test Time	Model Size
compelte Transformer	93.2%	7.0s	11.0M
Transformer with four attention layer	92.9%	6.4s	10.4M
Transformer with one attention layer	92.0%	4.5s	5.7M
Light-Transformer with skip attention layer	92.6%	5.8s	7.4M

Table 6: Ablation study for residual refinement unit.

Method	CD	HD	P2F
Ours w/o residual refinement unit	0.588×10^{-3}	7.032×10^{-3}	2.415×10^{-3}
Ours	0.520×10^{-3}	6.450×10^{-3}	2.166×10^{-3}

As shown in Table 5, the complete Transformer can obtain the highest classification accuracy, but its model size is also the largest and the test time is the longest correspondingly. After replacing the Positional Encoding module and Input Embedding module with the Linear layer, the accuracy, model size, and test time all drop slightly. When the Multi-Head Attention layer was changed to Single-Head Attention layer, the accuracy, model size, and test time drop significantly. The Light-Transformer we proposed can obtain a relatively high classification accuracy while the model size is smaller and the test time is shorter than the complete Transformer, achieving a better trade-off between performance and efficiency.

Next, We conducted an ablation study to quantitatively evaluate the performance of the residual refinement unit. The result is shown in Table 6. We can see that our network with all the proposed components performs best. When the residual refinement unit was removed, the performance of the network decreased.

To demonstrate the effectiveness of the residual refinement unit more intuitively, we performed surface reconstruction on the point cloud using MeshLab. The visualization result is shown in Figure 9. The first column is the reconstruction result of the ground truth, the second column is the reconstruction result of the point cloud obtained by our upsampling network without residual refinement unit, and the third column is the reconstruction result of the point cloud obtained by our complete upsampling network.

It can be seen from the Figure 9 that the reconstruction result of the ground truth is uniform and smooth, with few holes. The reconstruction result generated by the upsampling network without residual refinement unit has many holes, especially the joints of curves and surfaces, such as the hair of the face and the bend of the finger. After adding the residual refinement unit, the number of holes is greatly reduced, and the reconstruction result at the

Figure 9: The reconstruction results of the network with or without residual refinement unit.

bends is significantly improved, which shows that the residual refinement unit is necessary, making the generated upsampling point cloud are closer to the ground truth and evenly distributed on the surface of the object.

5 Conclusion

In this paper, we present a novel and simple arbitrary-rate point cloud upsampling network based on Light-Transformer. Firstly, a Light-Transformer is proposed to extract features of point cloud, which not only retains the powerful learning ability of Transformer, but also reduces computing and storage resources. Then a coordinate regression module with residual refinement unit is

proposed to refine the coarse 3D coordinates of the point cloud. Next, in order to facilitate applications and save resources, an arbitrary-rate upsampling network combined with the FPS algorithm is proposed, which can obtain point cloud at arbitrary upsampling rate simply. Extensive experiments demonstrate that our upsampling network achieves better performance and resource trade off than other similar methods.

Unfortunately, there are still some problems with our method, such as the inability to deal with the occluded parts of point cloud. In the future, we will try to use the generation model to solve these problems. In addition, we will also explore other methods for point cloud upsampling, such as semi-unsupervised learning and continual learning. The former can reduce the dependence on the ground truth, and the latter can greatly apply the knowledge learned before to the learning of future tasks, both of which will improve learning efficiency and thus improve the performance of point cloud upsampling. Finally, our arbitrary-rate upsampling network is just a simple and easy-to-implement idea, we will improve and design a method that can truly achieve flexible arbitrary-rate upsampling.

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International conference on machine learning*, PMLR, 2018, 40–9.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on visualization and computer graphics*, 9(1), 2003, 3–15.

[3] T. M. Borges, D. C. Garcia, and R. L. De Queiroz, "Fractional super-resolution of voxelized point clouds," *IEEE Transactions on Image Processing*, 31, 2022, 1380–90.

[4] 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, 2021, 187–99.

[5] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, "Meta-SR: A magnification-arbitrary network for super-resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, 1575–84.

[6] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang, "Edge-aware point set resampling," *ACM transactions on graphics (TOG)*, 32(1), 2013, 1–12.

[7] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-gan: a point cloud upsampling adversarial network," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, 7203–12.

[8] R. Li, X. Li, P.-A. Heng, and C.-W. Fu, "Point cloud upsampling via disentangled refinement," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, 344–53.

[9] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," *ACM Transactions on Graphics (TOG)*, 26(3), 2007, 22–es.

[10] H. Liu, H. Yuan, J. Hou, R. Hamzaoui, and W. Gao, "PUFA-GAN: A Frequency-Aware Generative Adversarial Network for 3D Point Cloud Upsampling," *IEEE Transactions on Image Processing*, 31, 2022, 7389–402.

[11] C. Long, W. Zhang, R. Li, H. Wang, Z. Dong, and B. Yang, "PC2-PU: Patch Correlation and Point Correlation for Effective Point Cloud Upsampling," *arXiv preprint arXiv:2109.09337*, 2021.

[12] S. Luo and W. Hu, "Diffusion probabilistic models for 3d point cloud generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, 2837–45.

[13] C. Moenning and N. A. Dodgson, "Fast marching farthest point sampling," *tech. rep.*, University of Cambridge, Computer Laboratory, 2003.

[14] A. V. Phan, M. Le Nguyen, Y. L. H. Nguyen, and L. T. Bui, "Dgcnn: A convolutional neural network over large-scale labeled graphs," *Neural Networks*, 108, 2018, 533–43.

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, 652–60.

[16] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, 30, 2017.

[17] G. Qian, A. Abualshour, G. Li, A. Thabet, and B. Ghanem, "Pu-gcn: Point cloud upsampling using graph convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, 11683–92.

[18] Y. Qian, J. Hou, S. Kwong, and Y. He, "Deep magnification-flexible upsampling over 3d point clouds," *IEEE Transactions on Image Processing*, 30, 2021, 8354–67.

[19] S. Qiu, S. Anwar, and N. Barnes, "Pu-transformer: Point cloud upsampling transformer," in *Proceedings of the Asian Conference on Computer Vision*, 2022, 2475–93.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, 30, 2017.

[21] S. Wu, H. Huang, M. Gong, M. Zwicker, and D. Cohen-Or, "Deep points consolidation," *ACM Transactions on Graphics (ToG)*, 34(6), 2015, 1–13.

[22] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, 4541–50.

[23] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, 206–15.

[24] S. Ye, D. Chen, S. Han, Z. Wan, and J. Liao, "Meta-pu: An arbitrary-scale upsampling network for point cloud," *IEEE transactions on visualization and computer graphics*, 28(9), 2021, 3206–18.

[25] L. Yi, L. Shao, M. Savva, H. Huang, Y. Zhou, Q. Wang, B. Graham, M. Engelcke, R. Klokov, V. Lempitsky, *et al.*, "Large-scale 3d shape reconstruction and segmentation from shapenet core55," *arXiv preprint arXiv:1710.06104*, 2017.

[26] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-based progressive 3d point set upsampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, 5958–67.

[27] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Ec-net: an edge-aware point set consolidation network," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, 386–402.

[28] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-net: Point cloud upsampling network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, 2790–9.

[29] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, 16259–68.

[30] W. Zhao, X. Liu, Z. Zhong, J. Jiang, W. Gao, G. Li, and X. Ji, "Self-supervised arbitrary-scale point clouds upsampling via implicit neural representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, 1999–2007.

[31] Y. Zhao, L. Hui, and J. Xie, "Sspu-net: Self-supervised point cloud upsampling via differentiable rendering," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, 2214–23.