Arbitrary Point Cloud Upsampling with Spherical Mixture of Gaussians

Anthony Dell'Eva*
University of Bologna
Vislab Srl - Ambarella Inc
anthony.delleva2@unibo.it

Marco Orsingher*
University of Parma
Vislab Srl - Ambarella Inc
marco.orsingher@unipr.it

Massimo Bertozzi
University of Parma
bertozzi@ce.unipr.it

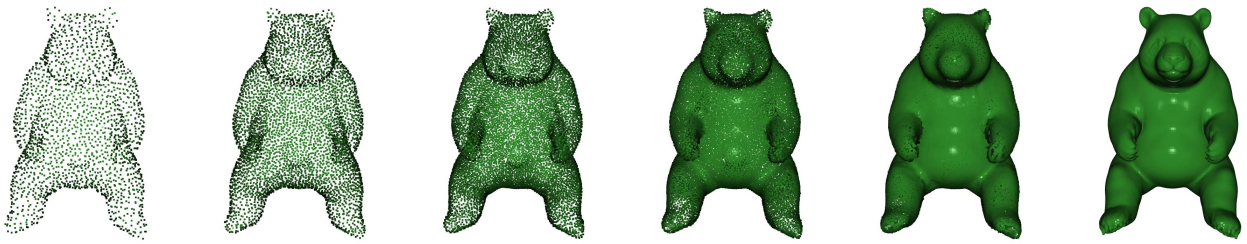

Figure 1. The proposed approach can perform point cloud upsampling from a sparse input with N points to a high-resolution output with $r \times N$ points. The upsampling ratio $r \in \mathbb{R}$ can be specified arbitrarily at test time, even if the model is trained a single time with $r = 4$.

Abstract

Generating dense point clouds from sparse raw data benefits downstream 3D understanding tasks, but existing models are limited to a fixed upsampling ratio or to a short range of integer values. In this paper, we present APU-SMOG, a Transformer-based model for Arbitrary Point cloud Upsampling (APU). The sparse input is firstly mapped to a Spherical Mixture of Gaussians (SMOG) distribution, from which an arbitrary number of points can be sampled. Then, these samples are fed as queries to the Transformer decoder, which maps them back to the target surface. Extensive qualitative and quantitative evaluations show that APU-SMOG outperforms state-of-the-art fixed-ratio methods, while effectively enabling upsampling with any scaling factor, including non-integer values, with a single trained model. The code will be made available.

1. Introduction

Point clouds are a common way to represent 3D data as an unordered list of points, which can be thought of as discrete samples from the underlying surface of the object. In recent years, the wide availability of low-cost scanning sensors has driven the research attention towards 3D point clouds analysis for several applications such as augmented reality, robotics and autonomous driving [42, 2, 18, 33]. However, the sparsity and the noise level in raw data from such sensors pose key challenges in point cloud processing for downstream tasks, e.g. classification, segmentation and surface reconstruction. To this end, we focus on point cloud upsampling, which consists in generating a dense and uniform set of points from a sparse and noisy input.

Building on pioneering works for neural point processing [24, 25], learning-based methods achieve state-of-the-art results in point cloud upsampling [39, 12, 26, 13] and significantly outperform classical optimization-based techniques [10, 15, 11]. One of the main limitations of current approaches is the tight coupling between the upsampling ratio and the network architecture. Typically, this value must be specified in advance and different models must be re-trained from scratch for different rates. Despite recent efforts on designing flexible networks with a user-defined upsampling factor at test time, existing works still limit its value to be integer and lower than a given bound [27, 19, 36], or reconstruct the whole surface with ground truth normals as an intermediate step [6].

Our main goal is to remove these limitations and to enable arbitrary upsampling from raw point clouds with a single trained model, as shown in Fig. 1. The key intuition of

*These authors contributed equally to the work.

2022 International Conference on 3D Vision (3DV) | 978-1-6654-5670-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/3DV57658.2022.00058

978-1-6654-5670-8/22/$31.00 ©2022 IEEE
DOI 10.1109/3DV57658.2022.00058

the presented method is to split the upsampling procedure in two steps: (i) firstly, the input point cloud is mapped to a probability distribution on a canonical domain; (ii) then, an arbitrary number of points can be sampled from such distribution and mapped back to the target surface.

Inspired by recent works on point cloud autoencoders [8, 14, 3], we propose to use the unit sphere as intermediate representation and we define a Spherical Mixture of Gaussians (SMOG) distribution on such domain. In this way, each input point is associated with a mixture weight and a bivariate Gaussian in spherical coordinates, whose parameters are estimated by a neural network.

The inverse mapping from the samples on the unit sphere to the desired shape is implemented by querying the decoder of a Transformer model [31] with an arbitrary number of points, which effectively decouples the network architecture and the value of the upsampling ratio. Moreover, we design both a feature extraction backbone and a residual refinement block based on local self-attention, which has proved to learn powerful representations on point cloud data [21, 40, 41]. Therefore, our main contributions can be summarized as follows:

- We present a novel approach for point cloud upsampling with arbitrary scaling factors, including non-integer values, with a single trained model.
- We propose to learn a mapping from the low-resolution input point cloud to a probability distribution on the unit sphere. This distribution can then be sampled arbitrarily and the inverse mapping is learned to generate the high-resolution output.
- We design a fully attention-based network architecture with state-of-the-art performances on multiple benchmarks and strong generalization capabilities.

2. Related Work

Canonical Primitives In the context of point cloud autoencoders, a common procedure to generate the output is to feed the decoder with a global latent vector encoding the input and a canonical primitive (e.g. a 2D grid [35]), which is deformed to match the target surface. Following the insights in [8], several works use the unit sphere with *uniform* sampling as an intermediate representation [14, 3]. Recently, TearingNet [22] proposed to learn topology-friendly representations by additionally estimating pointwise offsets on the 2D domain. We take a step further by directly learning a mean vector on the unit sphere and 2D variances in spherical coordinates for each point in the input shape. This allows to define a distribution from which an arbitrary number of points are sampled and mapped back to the surface.

Learning-based Upsampling Point cloud upsampling is an inherently ill-posed problem, since a finite number of samples correspond to many underlying surfaces and viceversa. For this reason, PU-Net [39] pioneered the idea of learning geometric priors from data and outperformed previous classical methods [11, 15, 10]. Building on this seminal idea of learning and expanding multi-scale point features, MPU [37] proposes a patch-based progressive strategy for upsampling at different levels of detail, while PU-GAN [12] casts the upsampling procedure in a generative adversarial framework. PU-GCN [26] introduces several modules built upon graph convolutional network that can be integrated into other architectures, whereas Dis-PU [13] disentangles the upsampling task into two cascaded subnetworks for dense point generation and spatial refinement. Despite showing promising results, all these works require a fixed upsampling ratio r and train different models for varying values of r. This strategy does not adapt to real-world point clouds with different quality and it increases the training time significantly.

Arbitrary Upsampling Recently, a few works [19, 27, 36] emerged with the goal of decoupling the upsampling ratio and the network architecture, thus achieving flexible upsampling. Meta-PU [36] employs meta-learning to predict the weights of residual graph convolution blocks dynamically for different values of r. However, the model first generates a set of $r_{max} \times N$ points, which are then downsampled to the desired ratio using farthest point sampling (FPS). MAFU [27] and PU-EVA [19] exploit the local geometry of the tangent plane at each point to sample a variable number of candidate points in its neighborhood, but they are limited to *integer* upsampling factors within a predefined range $[r_{min}, r_{max}]$. In addition, the former requires normal vectors information to be trained. On the other hand, our approach is designed to support any value of $r \in \mathbb{R}$, without restrictions. Neural Points [6] is a concurrent work that performs upsampling with unconstrained ratios by first encoding the continuous underlying surface with neural fields and then sampling an arbitrary number of points from it. Their main limitation is the requirement of ground truth normals for training, which are difficult to estimate with high accuracy, especially for noisy and sparse inputs. Conversely, our model operates in a discrete-to-discrete way on raw point clouds without normals.

Transformers for Point Clouds The Transformer model has revolutionized both natural language processing [31] and computer vision [4], thanks to the attention mechanism at its core. Since the Transformer architecture is permutation invariant by design and thus naturally suited for 3D data, early works in the field focused on adapting its modules to point cloud processing [41, 5, 9, 20].

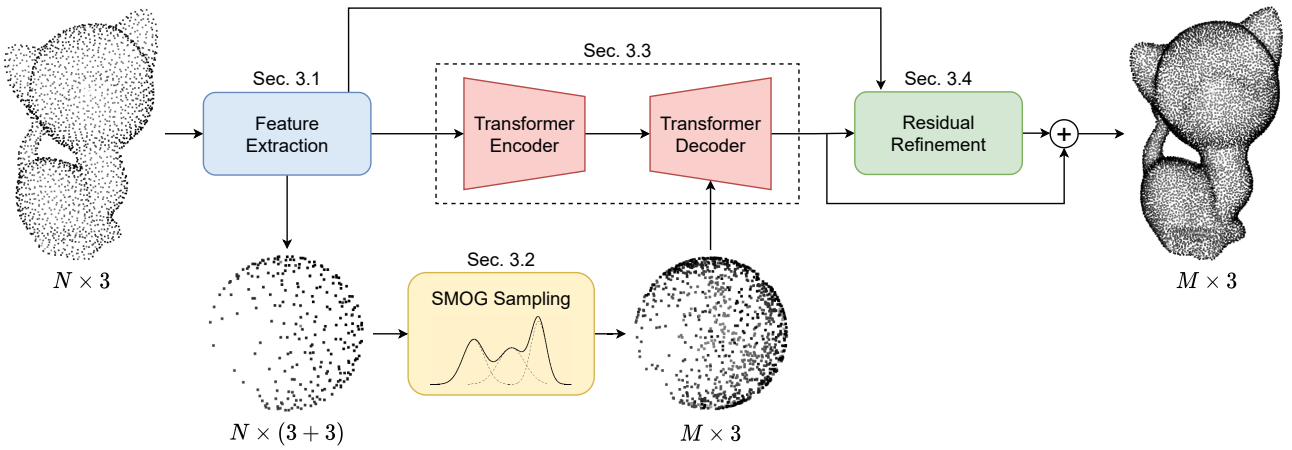

Figure 2. A high-level overview of the method. The low-resolution input point cloud with size $N \times 3$ is firstly mapped to a probability distribution on the unit sphere, with a mean vector and covariance matrix associated to each point. Then, this distribution can be sampled to produce the high-resolution output with size $M \times 3$, being $M = r \times N$ and r the arbitrary upsampling ratio.

Attention-based methods have achieved state-of-the-art results in many 3D tasks, such as point cloud completion [40], object detection [21] and classification [34]. In this work, we propose to combine the Transformer architecture with an attention-based refinement module for point cloud upsampling. To the best of our knowledge, there is only one concurrent work on this aspect [28]. However, they solely leverage the Transformer encoder, while we also exploit the possibility of querying the Transformer decoder for arbitrary upsampling.

3. Method

Denoting by $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{N}$ the unordered sparse input point cloud of N 3D points, our objective is to generate an arbitrarily denser point set $\mathcal{Q}_r = \{\mathbf{q}_i \in \mathbb{R}^3\}_{i=1}^{M}$ with $M = r \times N$ points, where $r \in \mathbb{R}$ is the upsampling factor. Note that point cloud upsampling is an ill-posed task, since there is not a single feasible correct output. This means that $\mathcal{Q}$ should represent the same underlying surface $\mathcal{S}$, while not being necessarily a superset of $\mathcal{P}$. To this end, we design a fully attention-based end-to-end network for arbitrary point cloud upsampling, taking advantage of Gaussian mixture sampling and Transformer queries to enable flexible ratios. An overview of our framework is shown in Fig. 2.

The details are explained in the following: in Sec. 3.1 we present our lightweight feature extractor, while Sec. 3.2 describes the input mapping to a distribution on the unit sphere and the relative sampling. In Sec. 3.3 we show how to leverage the Transformer model to obtain a flexible number of output points. Lastly, in Sec. 3.4 we illustrate the refinement of the 3-dimensional coordinates to generate the final upsampled point cloud.

3.1. Feature Extraction

The first step of our pipeline is to extract point-wise features from the $N \times 3$ input point cloud to obtain a $N \times D$ feature map. Differently from previous methods [26, 39, 12] that employ either PointNet [24, 25] or DGCNN [32] as backbone, we design a lightweight network based on Point Transformer [41]. For each input point $\mathbf{p}_i \in \mathcal{P}$, its associated feature is computed by the Point Transformer Layer (PTL) as follows:

$$\mathbf{f}_i = \sum_{\mathbf{p}_j \in \mathcal{N}(\mathbf{p}_i)} \rho(\gamma(\beta(\mathbf{p}_i) - \psi(\mathbf{p}_j) + \delta)) \odot (\alpha(\mathbf{p}_j) + \delta) \quad (1)$$

where $\mathcal{N}(\mathbf{p}_i)$ is the set of k nearest neighbors of $\mathbf{p}_i$, $\delta = \eta(\mathbf{p}_i - \mathbf{p}_j)$ is the positional encoding and the symbol $\odot$ denotes the element-wise product. The mappings α, β and ψ are simple linear layers, γ and η are two-layers MLP, while ρ is the softmax function. Note that in the context of the classical interpretation of the Transformer model [31], α generates the *values*, β the *queries* and ψ the *keys*.

3.2. Spherical Mixture of Gaussians Sampling

The core intuition of our approach is to map the input point cloud into a probability distribution on a canonical domain which can be conveniently sampled. To this end, we estimate the parameters of a Gaussian Mixture Model (GMM) on the unit sphere from the deep features extracted by the PTL. The choice of this domain over a 2D square is motivated by the fact that the unit sphere is a manifold without boundaries, which avoids the truncation of the Gaussian distributions in the GMM.

More specifically, we define a K components GMM $\Gamma = \{w_i, \mu_i, \Sigma_i\}_{i=1}^{K}$, where w_i, μ_i and Σ_i are the mixture weight, mean and covariance of the i-th Gaussian and

$K = N$. Therefore, the likelihood of a point $\mathbf{z}$ on the unit sphere is given by a weighted combination of individual components $u_i(\mathbf{z})$:

$$u_\Gamma(\mathbf{z}) = \sum_{i=1}^{N} w_i u_i(\mathbf{z}) \tag{2}$$

Each component is weighted equally (i.e. $w_i = \frac{1}{N}$) and its parameters (μ_i, Σ_i) are estimated by a MLP ξ:

$$(\mu_i, \Sigma_i) = \xi(\mathbf{f}_i) \tag{3}$$

The MLP predicts the mean vector μ_i in Cartesian coordinates and the covariance matrix Σ_i in spherical coordinates. In particular, the covariance matrix is built by exploiting the constraint of symmetry:

$$\mathbf{\Sigma}_i = \begin{bmatrix} \sigma_\theta^2 & \sigma_{\theta\phi} \\ \sigma_{\theta\phi} & \sigma_\phi^2 \end{bmatrix} \tag{4}$$

where σ_θ^2, σ_ϕ^2 and $\sigma_{\theta\phi}$ are the outputs of the covariance head of the MLP corresponding to the azimuth angle $\theta \in [0, 2\pi]$ and the elevation angle $\phi \in [0, \pi]$, respectively. In order to generate a dense output point cloud, we sample an arbitrary number of points from the distribution and learn the inverse mapping from the spherical domain to the target surface. This process is illustrated in Fig. 3 and additional details are provided in the supplementary material.

3.3. Transformer Model

The second core intuition which underpins our work is the possibility of querying the Transformer model [31] with an arbitrary number of points. Inspired by the 3DETR architecture [21], the set of $N \times D$ features produced by our backbone is fed to an encoder to produce a new feature map of dimension $N \times D$. The Transformer encoder has a single layer with a four-headed attention and an MLP with two layers. These features, along with a set of $r \times N$ queries, are processed by the two-layers Transformer decoder to generate $r \times N$ 3-dimensional Euclidean coordinates, which represent the coarse upsampled point cloud. The queries are obtained by embedding with Fourier positional encoding [29] the points sampled from the unit sphere manifold.

3.4. Residual Refinement

The coarse output from the Transformer decoder might be noisy, non-uniform distributed and have several outliers. Motivated by other works [13, 27], we design an attention-based residual refinement with a similar architecture as the feature extractor described in Sec. 3.1. In particular, we employ a single Point Transformer Block (PTB) [41] composed by a PTL, two linear layers, a ReLU activation function and a residual connection. Differently from Eq. 1, in this case the PTL takes as input the local features extracted from the backbone, while the positional encodings are computed with the 3D coarse coordinates. The D-dimensional vector produced by the PTB is finally projected to a 3-dimensional space with a two-layers MLP that computes the residual for each point, which is added to the coarse prediction to obtain the refined result.

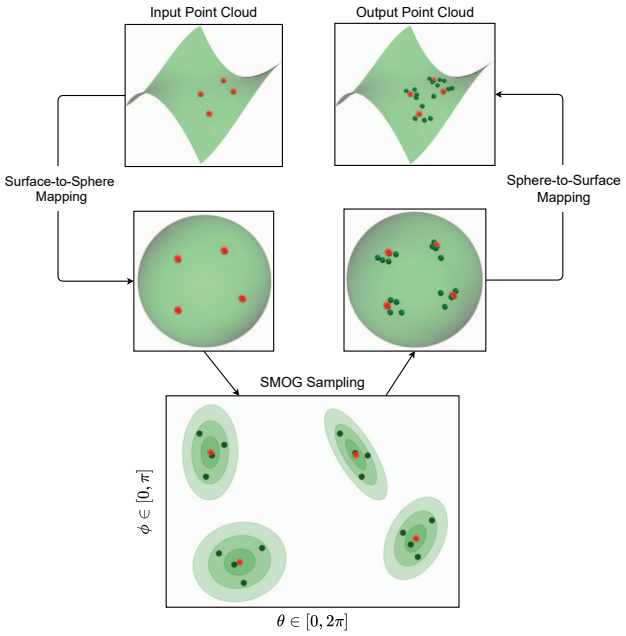

Figure 3. The sampling process shown in detail. Red points are the Gaussian means on the unit sphere associated to each input point, while green points are the arbitrary samples from the SMOG. Red points on the right branch are maintained for visualization.

3.5. Loss Function

Let $\widetilde{\mathcal{Q}}_r = \{\widetilde{\mathbf{q}}_i \in \mathbb{R}^3\}_{i=1}^{rN}$ be the coarse prediction generated by the Transformer with upsampling rate r, $\mathcal{Q}_r = \{\mathbf{q}_i \in \mathbb{R}^3\}_{i=1}^{rN}$ the refined prediction and $\mathcal{Y}_r = \{\mathbf{y}_i \in \mathbb{R}^3\}_{i=1}^{rN}$ the ground-truth upsampled point cloud. For training, we adopt a similar strategy as in [6] to define a distance between two generic point clouds $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^3\}_{i=1}^{X}$ and $\mathcal{Z} = \{\mathbf{z}_i \in \mathbb{R}^3\}_{i=1}^{Z}$ as:

$$d_\Pi(\mathcal{X}, \mathcal{Z}) = \frac{1}{X} \sum_{i=1}^{X} \|\mathbf{x}_i - \Pi(\mathbf{x}_i, \mathcal{Z})\|_2^2 \tag{5}$$

where $\Pi(\cdot, \cdot)$ is the projection of a 3D point to a point cloud. This term is computed as the weighted combination of the nearest points to $\mathbf{x}$ in $\mathcal{Z}$, with indices $\mathcal{N}(\mathbf{x}, \mathcal{Z})$:

$$\Pi(\mathbf{x}, \mathcal{Z}) = \frac{\sum_{k \in \mathcal{N}(\mathbf{x}, \mathcal{Z})} w_k \mathbf{z}_k}{\sum_{k \in \mathcal{N}(\mathbf{x}, \mathcal{Z})} w_k} \tag{6}$$

The weights w_k are given by:

$$w_k = e^{-\alpha \|\mathbf{x} - \mathbf{z}_k\|_2^2}, \; k \in \mathcal{N}(\mathbf{x}, \mathcal{Z}) \tag{7}$$

with $\alpha = 10^3$. The corresponding loss is then defined by the following bidirectional sum:

$$\mathcal{L}_{\Pi}(\mathcal{X}, \mathcal{Z}) = d_{\Pi}(\mathcal{X}, \mathcal{Z}) + d_{\Pi}(\mathcal{Z}, \mathcal{X}) \tag{8}$$

Our arbitrary upsampling network is trained with the sum of three losses:

$$\mathcal{L} = \mathcal{L}_{\Pi}(\widetilde{\mathcal{Q}}_4, \mathcal{Y}_4) + \mathcal{L}_{\Pi}(\mathcal{Q}_4, \mathcal{Y}_4) + \mathcal{L}_{ACD}(\widetilde{\mathcal{Q}}_1, \mathcal{P}) \tag{9}$$

where the last term is the Augmented Chamfer Distance (ACD) between the reconstructed point cloud and the input, defined for two generic point clouds as follows:

$$\mathcal{L}_{ACD}(\mathcal{X}, \mathcal{Z}) = \max \left\{ \frac{1}{X} \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{z} \in \mathcal{Z}} \|\mathbf{x} - \mathbf{z}\|_2^2, \frac{1}{Z} \sum_{\mathbf{z} \in \mathcal{Z}} \min_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - \mathbf{z}\|_2^2 \right\} \tag{10}$$

In practice, each training iteration consists of two forward passes and a single backward pass. During the upsampling forward pass, the Transformer decoder is queried with $4 \times N$ points sampled from the SMOG and the projection loss components for both the coarse and refined upsampled outputs are computed. On the other hand, in the reconstruction phase, the estimated means of the SMOG components are fed to the decoder and the ACD with respect to the input point cloud is evaluated. This additional term is required to learn the proper positions of the Gaussian means on the unit sphere, which acts effectively as an intermediate probabilistic representation of the input shape. The ablation studies in Sec. 4.5 prove this insight with numerical evidence.

4. Experiments

4.1. Settings

Datasets In order to compare the proposed APU-SMOG with state-of-the-art *fixed-ratio* methods, we employ the challenging PU1K dataset [26], which contains 1147 synthetic 3D models of various shapes, split into 1020 training samples and 127 testing samples. Consistently with existing literature [26, 13, 12, 37], 50 patches are extracted from each training shape with 256 points as input to the model and 1024 points as ground truth ($r = 4$). During testing, 2048 points are sampled from the original mesh with Poisson disk sampling. We employ a similar strategy as in [38] to extract overlapping patches with 256 geodesically close points. The network predicts the upsampled outputs at patch level and the final result is given by combining the overlapping patches with FPS to obtain 8192 points.

Moreover, we evaluate our approach against *flexible-ratio* methods on the widely used PU-GAN dataset [12], which is composed by 147 shapes collected from the PU-Net [39], MPU [37] and Visionair [1] repositories. The same patch-based training and testing procedure is followed with 2048 points as input, but ground truth point clouds with different sizes are generated to adapt to the values of the scaling factor r. Finally, we test the generalization capabilities of our approach on real-world LiDAR point clouds from the KITTI dataset [7].

Method	CD↓	HD↓	P2F μ↓	P2F σ↓
PU-Net [39]	1.155	11.626	4.834	6.799
MPU [37]	0.935	10.298	3.551	5.971
PU-GAN [12]	0.885	16.539	3.717	5.746
PU-GCN [26]	0.584	5.822	2.499	4.441
Dis-PU [13]	**0.511**	<u>4.104</u>	<u>2.013</u>	<u>2.926</u>
Ours	<u>0.528</u>	**2.549**	**1.667**	**2.075**

Table 1. Quantitative comparison with state-of-the-art methods on the PU1K dataset and $r = 4$. The units are all 10^{-3} and lower is better. Best and second results are **bold** and <u>underlined</u>.

Evaluation Metrics Following previous works, the results are quantitatively evaluated with the Chamfer Distance (CD), Hausdorff Distance (HD) and Point-to-Surface (P2F) distance metrics. The CD is the sum of squared distances between nearest neighbor correspondences of the input and ground truth point clouds, while the HD effectively measures the influence of outliers in the predicted results. On the other hand, the P2F distance is computed against the underlying surface, thus estimating the quality of the upsampled point cloud as an approximation of the real shape. All the metrics are averaged on the whole test set and a lower value indicates better upsampling performance.

Comparison Methods For the task of upsampling with a fixed ratio $r = 4$ on the PU1K dataset, we provide quantitative comparison against the state-of-the-art models PU-Net [39], MPU [37], PU-GAN [12], PU-GCN [26] and Dis-PU [13]. On the other hand, the flexible methods MAFU [27], PU-EVA [19] and Neural Points [6] are used as baselines for arbitrary upsampling on the PU-GAN dataset. For a fair comparison, we used the official pre-trained models when available and re-trained the other ones with the official published code.

Implementation Details Our framework is implemented in PyTorch [23]. The features dimension is set to $D = 128$. The MLPs in the Transformer model has hidden dimension equal to 64 in the encoder and 128 in the decoder. For the local feature extractor and the refinement module, the number of neighboring points are set to 32 and $4r$, respectively. The projection components in the loss function are weighted with $\lambda_1 = \lambda_2 = 0.01$, while the reconstruction compo-

	Method	$r=4$			$r=8$			$r=12$			$r=16$		
		CD↓	**HD↓**	**P2F↓**	**CD↓**	**HD↓**	**P2F↓**	**CD↓**	**HD↓**	**P2F↓**	**CD↓**	**HD↓**	**P2F↓**
Fixed	PU-GAN [12]	0.274	4.694	1.943	0.489	6.985	2.621	0.233	6.093	2.548	0.209	6.055	2.556
	PU-GCN [26]	0.304	2.656	2.541	0.256	4.175	2.825	0.204	4.157	2.737	0.195	4.176	2.716
	Dis-PU [13]	0.360	5.133	2.868	0.352	7.028	3.338	0.291	6.694	3.258	0.271	6.645	3.240
Flexible	MAFU [27]	<u>0.322</u>	<u>2.116</u>	**1.721**	<u>0.195</u>	<u>2.389</u>	<u>2.037</u>	<u>0.164</u>	<u>2.392</u>	<u>2.034</u>	<u>0.158</u>	<u>2.367</u>	<u>1.971</u>
	NePs [6]	0.368	4.556	<u>1.875</u>	0.254	10.146	**1.928**	0.203	10.018	**1.922**	0.159	9.263	**1.957**
	PU-EVA [19]	0.394	7.676	2.915	0.322	7.951	3.148	0.290	8.191	3.234	0.286	8.390	3.269
	Ours	**0.276**	**1.909**	2.634	**0.194**	**1.628**	2.613	**0.162**	**1.626**	2.635	**0.149**	**1.948**	2.666

Table 2. Quantitative comparison with state-of-the-art methods on the PU-GAN dataset and flexible upsampling ratios. The units are all 10^{-3} and lower is better. Best and second results *among flexible methods* are **bold** and <u>underlined</u>. Fixed methods are included as reference.

nent weight is set to $\lambda_3 = 1$. In the $\Pi(\cdot,\cdot)$ computation, k is equal to 4. The model is trained with a batch size of 64 for 100K iterations on a single V100 GPU with 32GB of RAM, using random rotation and perturbation as data augmentation techniques to avoid overfitting. The AdamW [17] optimizer is used with a learning rate decay following a cosine schedule [16] from $5\text{e}{-4}$ to $1\text{e}{-6}$, a weight decay of 0.1 and a gradient L_2 norm clipping of 0.1.

4.2. Quantitative Results

The quantitative evaluation of the method on the PU1K test set and a fixed upsampling ratio $r = 4$ for point clouds with $N = 2048$ points is presented in Tab. 1. It can be noticed that we achieve the lowest HD value, indicating that APU-SMOG performs upsampling with fewer outliers with respect to state-of-the-art models, as well as the lowest P2F distance, showing a better approximation of the underlying surface. Our approach falls behind Dis-PU [13] in terms of the CD metric by a slight margin. Nevertheless, this quantity measures the consistency of the result with the ground truth *point cloud*, rather than with the *target shape*: two different sets of points sampled from the same mesh have a non-zero CD, despite belonging to the same surface.

Furthermore, the numerical performances as a function of the ratio r on the PU-GAN test set are provided in Tab. 2. In order to be able to compare against state-of-the-art flexible methods [27, 19], we constrain r to be an integer value smaller than 16, even if we can generate predictions with any $r \in \mathbb{R}$. Our approach achieves the best CD value, as well as the best HD value by a significant margin, along the whole spectrum of ratios. Note that both MAFU [27] and Neural Points [6] have a lower P2F metric since they require ground truth normals at training time, which helps in finding the correct surface. On the other hand, our approach consistently outperforms PU-EVA [19], which is trained from raw point clouds. For completeness, we include the comparison against *fixed-ratio* models as reference. To generate predictions for $r \in \{8, 12, 16\}$, each network is queried iteratively with $r = 4$ and the desired number of points is obtained with FPS.

4.3. Qualitative Results

Fig. 4 shows qualitative upsampling results in comparison with state-of-the-art methods. These results have been obtained under the same settings as Tab. 1, namely with all the models trained on the PU1K dataset with fixed upsampling ratio $r = 4$ for input point clouds having size $N = 2048$. Close-up views show that APU-SMOG is particularly effective in preserving fine-grained structures, such as the piano's pole and the motorcycle mirror, and disambiguating complex shape (see the bag's handle). It can be noticed that other models fail to distinguish between different details of the surface and tend to merge them together, thus producing noisy point clouds. Moreover, the proposed attention-based residual refinement block generates refined outputs with fewer outliers (e.g. the plane motors). Additional qualitative results, including more comparisons and reconstructed surfaces from predicted point clouds, can be found in the supplementary material.

4.4. Generalization and Robustness

We conduct further experiments to demonstrate the robustness and generalization capabilities of APU-SMOG. Firstly, we provide qualitative results on real-world point clouds from the KITTI dataset [7], without any fine-tuning. This task is particularly challenging, since street-level LiDAR data with noise and occlusions are very different from synthetic training samples. Fig. 5 shows the generalization power of our approach on different urban elements such as cars, trucks and pedestrians. See the supplementaries for more examples on real-scanned point clouds [30].

Since the previous qualitative results in Fig. 4 have been generated with point clouds having $N = 2048$ points, we show in Fig. 6 that the predictions of our method closely follow the underlying surfaces for a wide variety of input sizes. Even for the sparsest input with 256 points, thin structures

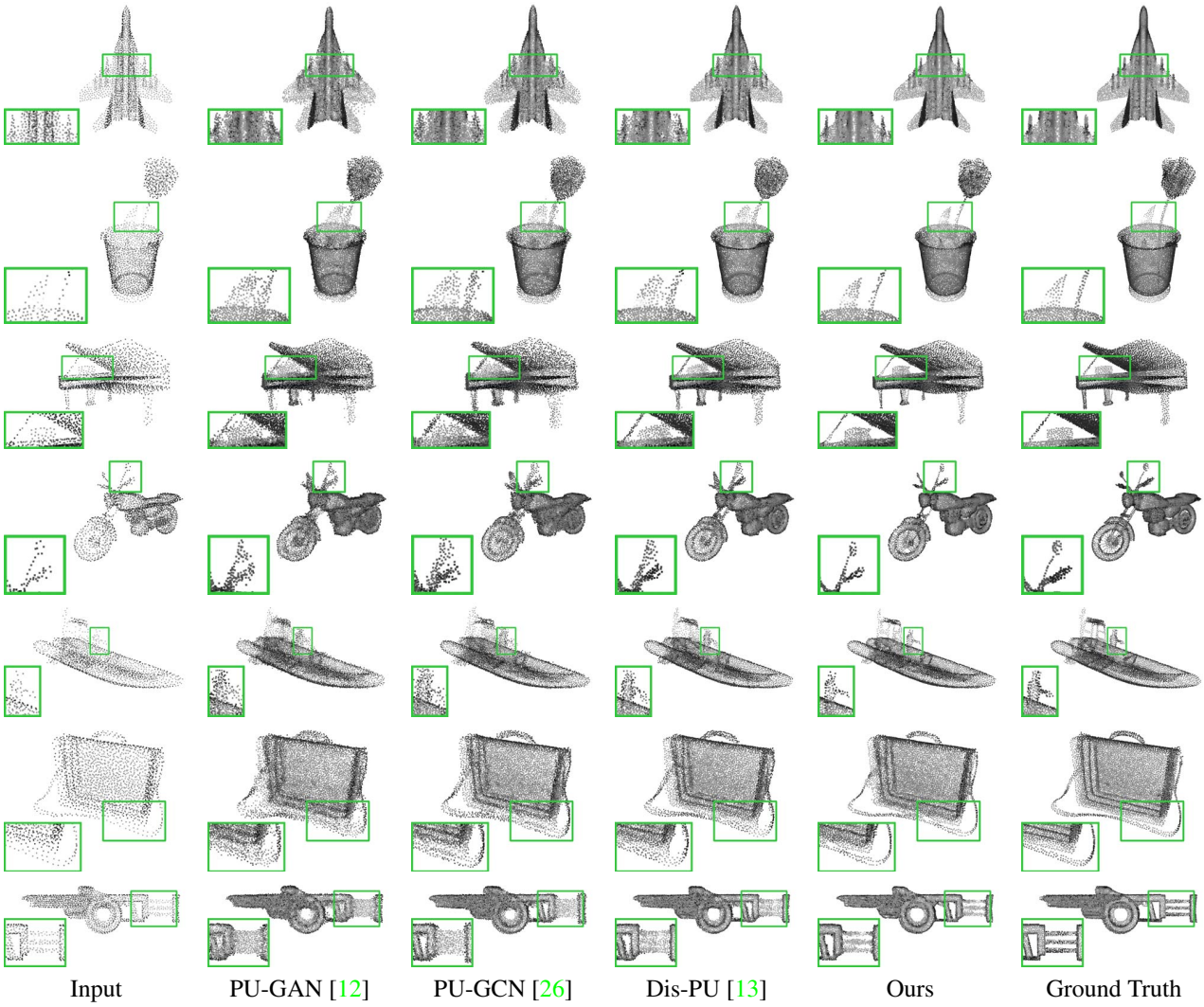

Figure 4. Qualitative comparison with state-of-the-art methods on the PU1K dataset. Inputs with 2048 points (left) are upsampled to 8192 points (right), with upsampling ratio $r = 4$. Details are best viewed when zoomed in.

such as the horse's legs are upsampled correctly. In order to simulate real noisy point clouds from scanning sensors, Fig. 7 shows the upsampling results for three different levels of additive Gaussian noise. It can be noticed that the duck shape is successfully maintained for both the clean and the corrupted inputs.

Finally, the key novelty of our approach is the possibility to specify an arbitrary upsampling factor at test time, thus producing consistent results with the desired output resolution. This flexibility is shown in Fig. 1 and additional examples with more details can be found in the supplementaries.

4.5. Ablation Studies

We perform a set of ablation studies to evaluate the contribution of each module to the proposed pipeline. The presented results in Tab. 3 have been obtained with different versions of our model trained on the PU1K dataset with fixed upsampling ratio $r = 4$.

Ablation	**CD↓**	**HD↓**	**P2F↓**
FoldingNet-like [35]	0.558	2.761	1.700
$N/4$ SMOG components	0.564	2.803	1.686
w/o refinement	0.572	8.536	2.182
w/o rec. loss, $\mathcal{L}_{\Pi}$ ups.	0.561	2.843	1.726
w/o rec. loss, $\mathcal{L}_{ACD}$ ups.	0.816	6.459	2.694
Ours	**0.528**	**2.549**	**1.667**

Table 3. Ablation studies on different versions of our model. The units are all 10^{-3} and **bold** denotes the best performance.

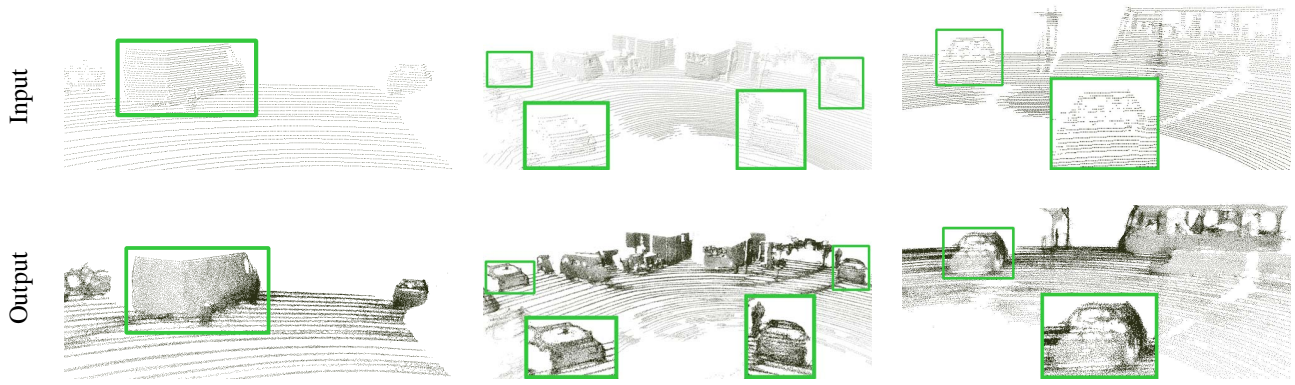

Figure 5. Generalization to real-world point clouds from the KITTI dataset [7]. APU-SMOG can upsample successfully various urban instances, including fine-grained structures such as the car window. Details are best viewed when zoomed in.

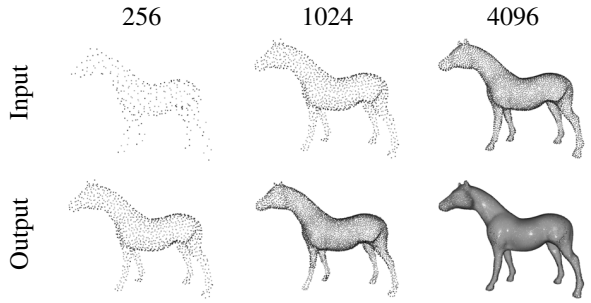

Figure 6. Effect of input point cloud size on the upsampling results.

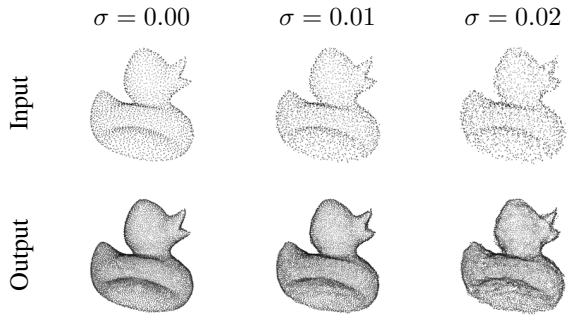

Figure 7. Effect of additive noise on the upsampling results.

In order to measure the influence of the SMOG representation, we replace it with a FoldingNet-like [35] strategy, i.e. sampling the unit sphere uniformly. The quantitative results confirm that our approach is able to generate better predictions, as it adaptively learns a local probability distribution around each point.

Furthermore, we investigate the influence of the number of GMM components on the output point clouds, reducing it to $K = N/4$. The numerical evaluation exhibits decent performances, suggesting that the model associates each Gaussian distribution to a local neighborhood of points. However, having a single component for each input point leads to the best results overall.

The third row in Tab. 3 proves the effectiveness of the attention-based residual refinement step. The significantly higher HD value indicates that the raw output from the Transformer decoder contains several outliers, which are correctly positioned closer to the surface by this module.

Moreover, we train our model without the reconstruction loss and notice a performance drop. This implies that $\mathcal{L}_{ACD}$ is a strong bias towards learning the correct positions of the Gaussian means on the unit sphere. Finally, the advantage of the projection loss for upsampling over ACD is shown in the last row. The remarkable difference in all the metrics justifies the choice of $\mathcal{L}_{\Pi}$ in the final design.

5. Conclusion

In this paper, we present a novel approach for point cloud upsampling with arbitrary scaling factors. A Transformer-based architecture is designed to decouple the upsampling process in two key steps: (i) firstly, the sparse input is mapped to an intermediate representation as a Spherical Mixture of Gaussians; (ii) then, such distribution is sampled arbitrarily and the Transformer decoder learns to map each sample back to the surface. The predictions are further improved by an attention-based residual refinement module, which allows to achieve state-of-the-art results on different benchmarks. This strategy enables arbitrary upsampling, since the model is trained a single time with a fixed ratio and it can be queried at test time with any desired value.

The main limitation of our work is the patch-based training and testing procedure, which require a long inference time when upsampling large point clouds. As future work, we plan to tackle this issue and to learn the weights of the GMM jointly with the Gaussian parameters.

References

[1] Visionair. http://www.infra-visionair.eu/. Accessed: 2022-05-20. 5

[2] Daniele Cattaneo, Matteo Vaghi, and Abhinav Valada. Lcdnet: Deep loop closure detection and point cloud registration for lidar slam. *IEEE Transactions on Robotics*, 2022. 1

[3] An-Chieh Cheng, Xueting Li, Min Sun, Ming-Hsuan Yang, and Sifei Liu. Learning 3d dense correspondence via canonical point autoencoder. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 2

[5] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. Point transformer. *IEEE Access*, 9:134826–134840, 2021. 2

[6] Wanquan Feng, Jin Li, Hongrui Cai, Xiaonan Luo, and Juyong Zhang. Neural points: Point cloud representation with neural fields. *arXiv preprint arXiv:2112.04148*, 2021. 1, 2, 4, 5, 6

[7] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 5, 6, 8

[8] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018. 2

[9] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. 2

[10] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM transactions on graphics (TOG)*, 28(5):1–7, 2009. 1, 2

[11] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. Edge-aware point set resampling. *ACM transactions on graphics (TOG)*, 32(1):1–12, 2013. 1, 2

[12] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: a point cloud upsampling adversarial network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7203–7212, 2019. 1, 2, 3, 5, 6, 7

[13] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Point cloud upsampling via disentangled refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 344–353, 2021. 1, 2, 4, 5, 6, 7

[14] Ruihui Li, Xianzhi Li, Ke-Hei Hui, and Chi-Wing Fu. SP-GAN:sphere-guided 3d shape generation and manipulation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 40(4), 2021. 2

[15] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Transactions on Graphics (TOG)*, 26(3):22–es, 2007. 1, 2

[16] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. 6

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 6

[18] Chenxu Luo, Xiaodong Yang, and Alan Yuille. Self-supervised pillar motion learning for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3183–3192, 2021. 1

[19] Luqing Luo, Lulu Tang, Wanyi Zhou, Shizheng Wang, and Zhi-Xin Yang. 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*, pages 16208–16217, 2021. 1, 2, 5, 6

[20] Kirill Mazur and Victor Lempitsky. Cloud transformers: A universal approach to point cloud processing tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10715–10724, 2021. 2

[21] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2906–2917, 2021. 2, 3, 4

[22] Jiahao Pang, Duanshun Li, and Dong Tian. Tearingnet: Point cloud autoencoder to learn topology-friendly representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5

[24] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, pages 652–660, 2017. 1, 3

[25] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 1, 3

[26] Guocheng Qian, Abdulellah Abualshour, Guohao Li, Ali Thabet, and Bernard Ghanem. Pu-gcn: Point cloud upsampling using graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11683–11692, 2021. 1, 2, 3, 5, 6, 7

[27] Yue Qian, Junhui Hou, Sam Kwong, and Ying He. Deep magnification-flexible upsampling over 3d point clouds. *IEEE Transactions on Image Processing*, 30:8354–8367, 2021. 1, 2, 4, 5, 6

[28] Shi Qiu, Saeed Anwar, and Nick Barnes. Pu-transformer: Point cloud upsampling transformer. *arXiv preprint arXiv:2111.12242*, 2021. 3

[29] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 4

[30] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019. 6

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 3, 4

[32] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019. 3

[33] Kento Yabuuchi, David Robert Wong, Takeshi Ishita, Yuki Kitsukawa, and Shinpei Kato. Visual localization for autonomous driving using pre-built point cloud maps. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 913–919. IEEE, 2021. 1

[34] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3323–3332, 2019. 3

[35] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 206–215, 2018. 2, 7, 8

[36] Shuquan Ye, Dongdong Chen, Songfang Han, Ziyu Wan, and Jing Liao. Meta-pu: An arbitrary-scale upsampling network for point cloud. *IEEE Transactions on Visualization and Computer Graphics*, 2021. 1, 2

[37] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based progressive 3d point set upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5958–5967, 2019. 2, 5

[38] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *Proceedings of the European conference on computer vision (ECCV)*, pages 386–402, 2018. 5

[39] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2790–2799, 2018. 1, 2, 3, 5

[40] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12498–12507, 2021. 2, 3

[41] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 2, 3, 4

[42] Yiqin Zhao and Tian Guo. Pointar: Efficient lighting estimation for mobile augmented reality. In *European Conference on Computer Vision*, pages 678–693. Springer, 2020. 1