PU-Dense: Sparse Tensor-Based Point Cloud Geometry Upsampling

Anique Akhtar, *Student Member, IEEE*, Zhu Li, *Senior Member, IEEE*, Geert Van der Auwera, *Senior Member, IEEE*, Li Li, *Member, IEEE*, and Jianle Chen, *Senior Member, IEEE*

***Abstract*— Due to the increased popularity of augmented and virtual reality experiences, the interest in capturing high-resolution real-world point clouds has never been higher. Loss of details and irregularities in point cloud geometry can occur during the capturing, processing, and compression pipeline. It is essential to address these challenges by being able to upsample a low Level-of-Detail (LoD) point cloud into a high LoD point cloud. Current upsampling methods suffer from several weaknesses in handling point cloud upsampling, especially in dense real-world photo-realistic point clouds. In this paper, we present a novel geometry upsampling technique, PU-Dense, which can process a diverse set of point clouds including synthetic mesh-based point clouds, real-world high-resolution point clouds, real-world indoor LiDAR scanned objects, as well as outdoor dynamically acquired LiDAR-based point clouds. PU-Dense employs a 3D multiscale architecture using sparse convolutional networks that hierarchically reconstruct an upsampled point cloud geometry via progressive rescaling and multiscale feature extraction. The framework employs a UNet type architecture that downscales the point cloud to a bottleneck and then upscales it to a higher level-of-detail (LoD) point cloud. PU-Dense introduces a novel Feature Extraction Unit that incorporates multiscale spatial learning by employing filters at multiple sampling rates and receptive fields. The architecture is memory efficient and is driven by a binary voxel occupancy classification loss that allows it to process high-resolution dense point clouds with millions of points during inference time. Qualitative and quantitative experimental results show that our method significantly outperforms the state-of-the-art approaches by a large margin while having much lower inference time complexity. We further test our dataset on high-resolution photo-realistic datasets. In addition, our method can handle noisy data well. We further show that our approach is memory efficient compared to the state-of-the-art methods.**

***Index Terms*— Point cloud, upsampling.**

Manuscript received September 28, 2021; revised February 15, 2022, April 15, 2022, May 12, 2022, and May 20, 2022; accepted May 23, 2022. Date of publication June 13, 2022; date of current version June 20, 2022. This work was supported in part by the NSF under Award 1747751 and Award 2148382. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ioan Tabus. *(Corresponding author: Anique Akhtar.)*

Anique Akhtar and Zhu Li are with the Department of Computer Science and Electrical Engineering, University of Missouri–Kansas City, Kansas City, MO 64110 USA (e-mail: aniqueakhtar@mail.umkc.edu; zhu.li@ieee.org).

Li Li is with the Department of Computer Science and Electrical Engineering, University of Missouri–Kansas City, Kansas City, MO 64110 USA, and also with the CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application System, University of Science and Technology of China, Hefei 230027, China (e-mail: lil1@ustc.edu.cn).

Geert Van der Auwera and Jianle Chen are with Qualcomm Technologies Inc., San Diego, CA 92121 USA (e-mail: geertv@qti.qualcomm.com; cjianle@qti.qualcomm.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIP.2022.3180904, provided by the authors.

Digital Object Identifier 10.1109/TIP.2022.3180904

I. Introduction

THERE has been a surge in the usage of 3D point clouds in augmented/virtual reality (AR/VR), telepresence [1], [2], surveillance and autonomous driving [3], [4] which has been accompanied by significant advances in 3D sensors and capturing techniques [5]. With the rapid advancement of 3D point cloud acquisition technologies, such as LiDAR (Light Detection And Ranging) sensors, high precision point cloud representations have become affordable. Moreover, the recent advances in GPU power capabilities have enabled real-time rendering and visualization of dense 3D point clouds. Recent advances in point cloud compression by groups like MPEG [6] and JPEG Pleno [7] have enabled the efficient transfer of larger point clouds. These developments have now allowed the capture and utilization of very high definition real-world point clouds with millions of points per frame.

Based on their usage, point clouds can be categorized into point cloud scenes and point cloud objects. Point cloud scenes are dynamically acquired and are typically captured by LiDAR sensors. LiDAR sensors mounted on top of a vehicle for mobile mapping and autonomous navigation [8] are examples of a dynamically acquired point cloud scene. Point cloud objects can be further subdivided into static objects and dynamic objects. A static point cloud is a single object, whereas a dynamic point cloud is time-varying, where each instance of a dynamic point cloud is a static point cloud. Dynamic time-varying point clouds are used in AR/VR, volumetric video, and telepresence and can be generated using 3D models, i.e., CGI, or captured from real-world scenes using various methods such as multiple cameras with depth sensors surrounding the object and capturing movement over time. Advancement in sensor technologies has allowed the capture of photo-realistic point clouds with millions of points per object. These real-world high-resolution point clouds tend to be dense and pose complicated challenges in point cloud processing due to their size. In this paper, we introduce a novel point cloud geometry upsampling technique that does not just work on synthetic point clouds but can also process a diverse set of point clouds including dense high-resolution photo-realistic point clouds, real-world LiDAR scanned objects, as well as dynamically acquired outdoor LiDAR point clouds.

Recently, there have been considerable advances in point cloud upsampling along with advances in problems closely resembling point cloud upsampling like point cloud completion [9]–[12] and point cloud denoising [13]–[16].

Point cloud upsampling methods can be broadly categorized as optimization-based methods [17]–[21] and deep learning-based methods [22]–[27]. Optimization-based methods usually fit local geometry by utilizing geometry priors that only work well for low precision smooth surfaces. Optimization-based methods are computationally intensive and are difficult to scale to larger point clouds. Deep learning-based methods like PU-Net [22], 3PU [24], and PU-GAN [25] can effectively learn point cloud structures from data. However, these methods use kNN search-based patch selection for neighborhood feature aggregation. Raw points within the patch can be in any order and are processed directly by fully connected layers without considering the relative point location in the overall 3D representation or the point's distance from its neighbors. Furthermore, these methods are also memory hungry and computationally intensive, which is why they are limited to fixed small input sizes. Therefore, the current state-of-the-art in point cloud upsampling fails to build deeper architectures with large receptive fields that can effectively learn discriminative features, and be able to efficiently work on denser point clouds that have a large number of points.

To resolve these issues, we propose a novel PU-Dense architecture that can upsample synthetic point clouds, real-world scanned sparse point clouds as well as dense photo-realistic point clouds. The proposed framework offers the following contributions:

- PU-Dense employs a UNet [28] type encoder-decoder architecture that hierarchically reconstructs an upsampled point cloud via progressive rescaling and multiscale feature extraction. PU-Dense introduces a novel feature extraction (FE) unit with Inception-Residual Block (IRB) and a 3D Dilated Pyramid Block (3D-DPB) to extract 3D multiscale features with different field-of-views in a computationally efficient manner.
- PU-Dense is a fully convolutional geometry upsampling network that is translation invariant and has a variable input size that takes advantage of the sparse nature of point clouds and employs sparse convolutions [29] that tend to be memory efficient. Rather than a distance-based loss function, PU-Dense employs a memory-efficient binary voxel classification loss and utilizes 3D data representation that enables efficient learning of 3D point features. While the previous state-of-the-arts limit their input to a fixed number of points between a few hundred and a few thousand points, our method can process millions of points per iteration with variable input size.
- Rather than creating our own synthetic dataset like the previous works, we intend to use a standardized dataset. We train our network on ShapeNet while testing on ShapeNet as well as real-world photo-realistic point cloud datasets like MPEG's 8i Voxelized Surface Light Field (8iVSLF), JPEG Pleno's 8i Voxelized Full Bodies (8iVFB), Queen by technicolor, real-world LiDAR scanned objects from ScanObjectNN dataset, and dynamically acquired outdoor LiDAR dataset from KITTI. Experimental results show that our method considerably outperforms the previous works in not just synthetic point cloud upsampling but also dense photo-realistic point clouds as well as sparse LiDAR-based datasets. We show that the proposed method is robust against noise. Moreover, the results show that PU-Dense is faster as well as more memory efficient when compared with the other state-of-the-art point cloud upsampling.

II. Related Work

A. Optimization-Based Upsampling Methods

Earlier works formulated point cloud upsampling as an optimization problem. A pioneering solution proposed by Alexa *et al.* [17] computed a Voronoi diagram on the moving least squares (MLS) surface. Lipman *et al.* [18] developed a parametrization-free method using a locally optimal projection operator (LOP) to resample points and reconstruct surfaces based on an L1 norm. This work was subsequently improved to weighted LOP [19] and continuous LOP [20] methods to consolidate point sets with noise, outliers, and nonuniformities. However, LOP-based methods' performance suffers when upsampling sharp edges and corners because they assume points are sampled from smooth surfaces. Huang *et al.* [21] introduced an edge-aware (EAR) approach that is designed to preserve sharp features by first resampling away from edges and then progressively approaching edges and corners. These optimization-based methods are not data-driven, assume insufficient priors, and often require additional attributes. Furthermore, these methods are computationally intensive so they are not scalable to high-resolution point clouds with a large number of points.

B. Deep Learning in Point Clouds

Recent advances in deep learning have seen a lot of success in point cloud processing using deep learning models [30]. The raw format of point cloud lacks point order and has an irregular structure which brings new challenges in employing deep learning solutions for point cloud processing.

1) Grid-Based Architectures: Pioneer works [31], [32] have tried to extend 2D convolutional neural networks to 3D space by voxelizing the point cloud into uniform voxels and applying 3D convolutions to them. However, due to the sparsity of point clouds, most of the computations are wasted on empty voxels. Projection methods [33]–[36] project 3D data onto 2D planes and then process these 2D planes. However, these projection-based methods are not as effective as processing the 3D point cloud. There is also a recent voxel-based method [37] that employs *Gridding* which is an interpolation to map each point onto the eight vertices of the voxel and then employs 3D convolutions to the voxelized representation.

2) Point-Based Methods: Raw point cloud data acquired from 3D sensors are in the form of unordered points. Point-based methods process raw point cloud data using permutation invariant feature extraction networks. PointNet [38] was among the first deep learning solutions that can directly process raw 3D point cloud data. They employed pointwise fully connected layers and symmetric max-pooling to make the process permutation invariant. This work was subsequently improved to PointNet++ [39] that introduced hierarchical

learning to learn more meaningful and discriminative features. PointCNN [40] was among the first works to use the convolutional layers on raw point cloud data. PointCNN proposed a permutation invariant χ-Conv that employs MLP layers to learn the permutation followed by a convolutional layer. The VoxelNet [41] divides point clouds using voxel partition and then uses PointNets to learn point-wise features. However, point-based methods are computationally expensive, which imposes severe constraints on building larger networks or their applicability on point clouds with a large number of points.

3) New Point Cloud Data Representation: There has been significant work on graph-based models which can operate on unordered 3D data. Graphs can be constructed from 3D point clouds in a variety of ways [42]–[44] which have shown promising results in point cloud processing. Octree-based convolutional neural networks were introduced in [45] that converted the data into an octree data representation and employed convolutions on octants of the octree data structure. SparseConvNet [46] by Facebook was among the first sparse convolutional neural networks that achieved state-of-the-art results on point clouds. SparseConvNet employed submanifold sparse convolutions [47] that exploited the sparse nature of point clouds and ensured that the convolutions would not "dilate" the data. MinkowskiNet [29] is another such implementation that employs sparse convolutions for 3D point cloud learning. Sparse convolutions exploit the inherent sparsity of point cloud data and store point cloud data in sparse tensors where convolutions are only performed on the voxels that are occupied. This makes the sparse convolutions much more computationally efficient allowing for a much deeper architecture to be built and the ability to process hundreds of thousands of points in a single inference time. We employ sparse convolutions using Minkowski Engine in our work.

C. Deep Learning-Based Upsampling Methods

PU-Net [22] was the pioneer deep learning upsampling work on point cloud that uses PointNet++ for feature extraction. PU-Net uses multi-branch MLPs to expand features with a joint reconstruction and repulsion loss to generate uniform point clouds. PU-Net operates on small patch level and does not consider the spatial relations among the points which results in the output lacking fine-grained high-resolution geometry structures. EC-Net [23] intended to improve PU-Net work and introduced a point-to-edge distance loss, which can help preserve the edges. However, EC-Net requires the tedious work of labeling the point cloud data with annotated edge and surface information. Wang *et al.* [24] proposed 3PU-Net which introduced a patch-based progressive upsampling inspired by image super-resolution methods. 3PU learns point-wise features similar to the methods before and employs cascaded $2\times$ upsampling networks. Each subnet in 3PU appends a 1D code to the features which limits each subnet upsampling to a factor of 2. However, multiple subnets can be progressively employed to get a large upsampling factor, say 16 times. 3PU only supports an upsampling factor in powers of 2 and also requires careful step-by-step training. PU-GAN [25] introduced a generative adversarial network for point cloud upsampling. Their performance improvement mainly comes from the introduction of a discriminator. PUGeo-Net [26] introduced a geometric-centric approach where they upsample point clouds by learning the first and second fundamental forms of the local geometry. However, their method needs additional supervision in the form of normals, which many point clouds like those generated by LiDAR sensors do not come with.

However, these deep learning-based upsampling methods use PointNet++ type architecture where the raw unordered points are stored in 2D arrays and kNN search is employed for neighborhood feature aggregation without considering the point location in the overall 3D representation. Point-based approaches perform upsampling by splitting scenes into smaller chunks, effectively restricting the model's ability to learn from global context. These methods employ deep learning operations that are usually used for 2D images to learn 3D features, which tends to be an inefficient approach. Furthermore, all these methods are limited to a small fixed input size and employ patch-based scaling to be able to process larger point clouds. Additionally, even after using smaller patches, these methods suffer from memory issues and are computationally intensive, especially their loss functions. This limits these architectures to a relatively shallower network which decreases their ability to learn discriminative features. Due to these reasons, the previous upsampling methods are often limited to and tested on small mesh-based point clouds and give poor results when applied on dense high-resolution real-world scanned point clouds such as the ones used in AR/VR or the MPEG standards.

Note that recently there has been some point cloud upsampling work that has emerged in parallel to our work. Recently, Qian *et al.* [48] proposed PU-GCN that uses a multi-level feature extraction using an inception-based graph convolutional network. They employ shuffling rather than duplicating features to expand the feature space for upsampling. Dis-PU [49] proposes to disentangle the upsampling task using two cascaded sub-networks, a dense generator, and a spatial refiner, to obtain both distribution uniformity and proximity-to-surface. Inspired by Meta-SR from image super-resolution, Meta-PU [50] was proposed to support point cloud upsampling of arbitrary scale factors with a single model. Meta-PU proposes a meta-subnetwork to dynamically adjust the weights of their residual graph convolution (RGC) upsampling network for different scaling factors. The upsampling network outputs a dense maximum points point cloud which is downsampled using farthest point sampling (FPS) to the desired ratio. However, this whole process is computationally intensive and inefficient when employed to point clouds with a large number of points. More recently, Flexible-PU [51] has been proposed that can also work for an arbitrary upsampling ratio using a lightweight neural network. Flexible-PU explicitly involves the local neighborhood information in the learning process. They generate new samples by an affine combination of neighboring points projected onto the tangent plane which are further refined by a self-attention-based refinement module.

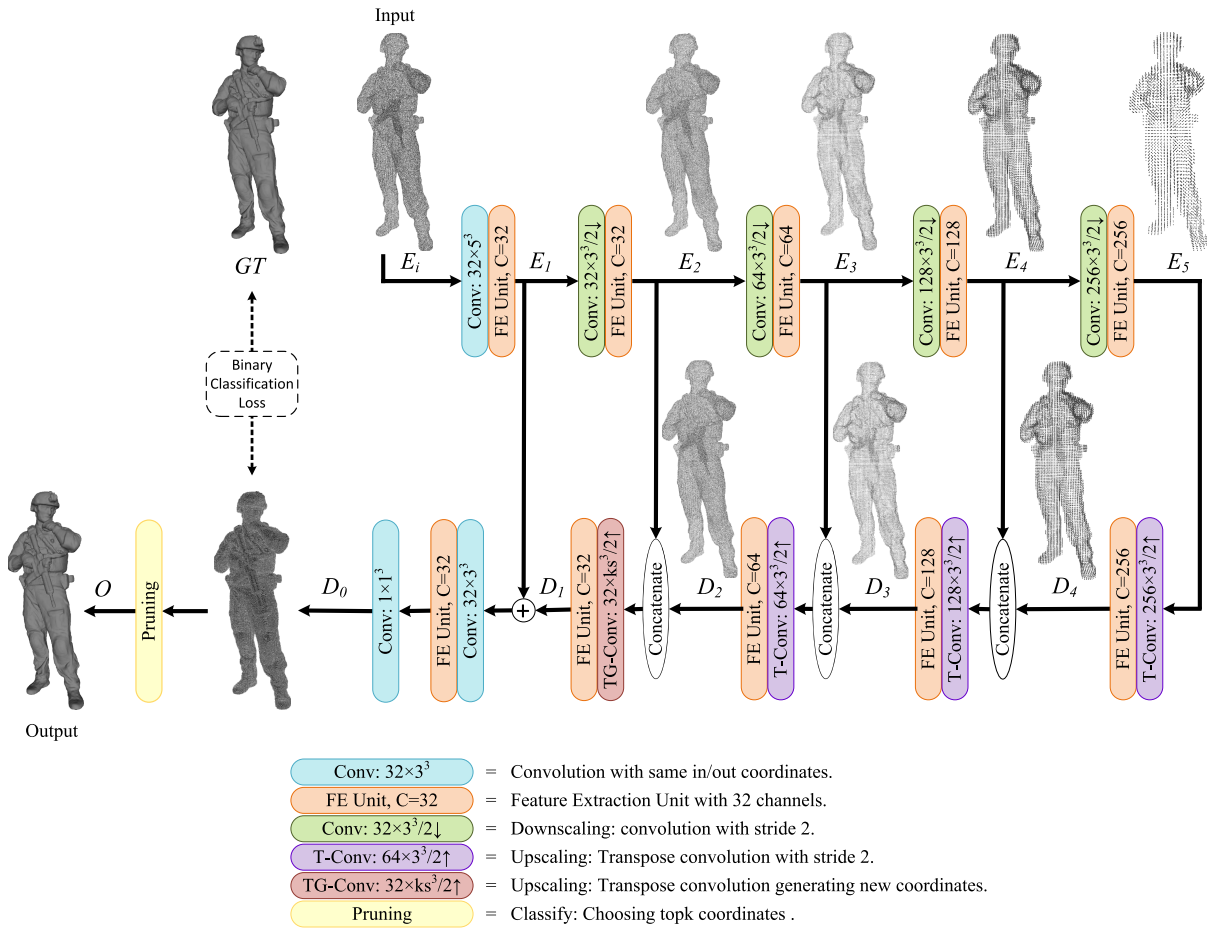

Fig. 1. PU-Dense network architecture consisting of encoder and decoder branches with skip connections in the middle. E_i is the input to the network that gets downscaled in the encoder branch by using convolutions of stride 2. The decoder side upscales the tensor by using transpose convolutions. *TG-Conv*: Transpose convolution generating new coordinates is employed in the last upscaling layer of the decoder side to populate additional coordinates around the existing coordinates. PU-Dense employs a voxel-based binary cross-entropy loss to compare decoder output D_0 with ground truth GT. Finally, pruning is applied to classify D_0 by choosing top k coordinates with the largest features. For convolutions, the terminology 32×3^3 denotes a kernel size of 3^3 with a channel size of 32. An example of tensor sizes throughout the network is shown in Table I.

III. PU-Dense

Given a lower level-of-detail (LoD) point cloud $\mathcal{X} = \{x_i \in \mathbb{R}^3\}_{i=1}^{M}$ with M points and an upsampling ratio R, we seek to generate a denser higher LoD point cloud $\mathcal{X}_R = \{x_i^r \in \mathbb{R}^3\}_{i,r=1}^{M,R}$ that is distributed uniformly over the underlying surface.

A. Preprocessing

Converting a point cloud from its raw format to a 3D volumetric representation is called *voxelization*. PU-Dense employs voxelization that allows it to use 3D convolutions to learn 3D features, which are more consistent and accurate in feature representation compared to the previous works where kNN is employed on unordered points to aggregate neighborhood features. Since we are only working with geometry and not other attributes, we assign feature f to each coordinate where $f(x, y, z) = 1$, if the voxel is occupied, and $f(x, y, z) = 0$ otherwise. We represent each input point cloud using a data tensor with a set of coordinates $C = \{(x_i, y_i, z_i)\}_i$ and their associated features $F = \{f(x_i, y_i, z_i)\}_i$.

B. The Network

PU-Dense is the first fully convolutional geometry upsampling network which makes it translation-invariant with variable input size. PU-Dense employs a multiscale U-Net [28] encoder-decoder type architecture built on Minkowski engine [29] utilizing sparse convolutions. Sparse convolutions exploit the inherent sparsity of point cloud data and are much more memory efficient. Sparse convolution is defined in [29] as:

$$f_u^{out} = \sum_{i \in N_3(u, C^{in})} W_i f_{u+i}^{in} \quad \text{for } u \in C^{out}, \tag{1}$$

where N_3 is a set of offsets that define the shape of a kernel and $N_3(u, C^{in}) = \{i | u + i \in C^{in}, i \in N_3\}$ is the set of offsets from the current center, u, that exists in C^{in}. C^{in} and C^{out} are the input and output coordinates. f_u^{in} and f_u^{out} are the input and output features at location u. W_i denotes the kernel value at offset i.

The proposed network architecture, shown in Fig. 1, has an encoder downscaling network and a decoder upscaling

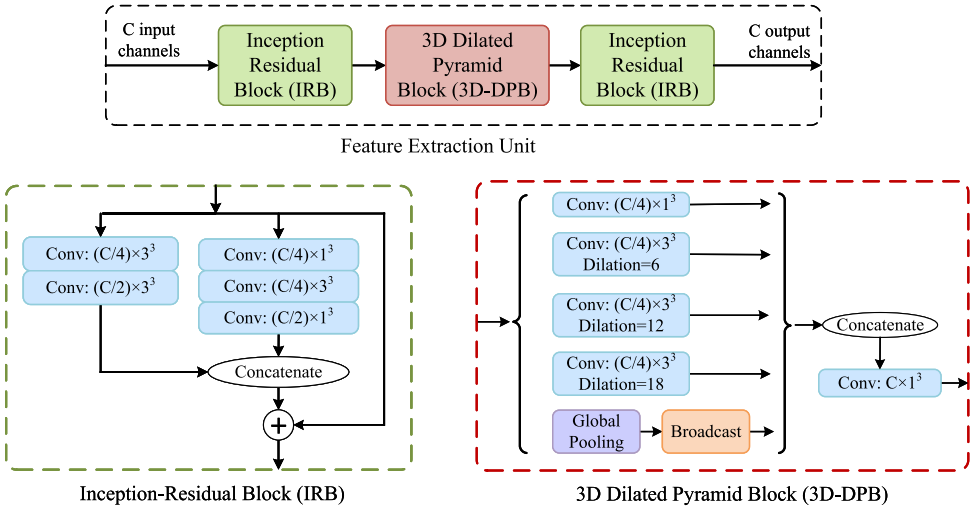

Fig. 2. *FE Unit*: Feature Extraction Unit. Each FE Unit consists of two Inception-Residual Blocks (IRB) and one 3D Dilated Pyramid Block (3D-DPB). FE Unit efficiently learns 3D multiscale spatial features by employing multiple sampling rates and receptive field.

network, with residual connections in between. For convolutions, the terminology 32×3^3 denotes a kernel size of 3^3 with a channel size of 32. The PU-Dense architecture employs a total of four different types of sparse convolutions: (i) Convolution, (ii) Downscaling Convolution, (iii) Transpose Convolution, and (iv) Transpose Convolution generating new coordinates. Employing these four different kinds of sparse convolutions, PU-Dense builds a 3D multiscale spatial architecture employing hierarchical learning using an encoder-decoder architecture.

Convolution (Referred as *Conv*). PU-Dense employs sparse convolutions with the same in/out coordinates ($C^{in} = C^{out}$). The coordinates of the sparse tensor stay the same after passing through these kernels and only the feature changes.

Downscaling convolution (Referred as *Conv*$/2\downarrow$): PU-Dense employs sparse convolution with stride of two to halve the scale of each geometric dimension.

Transpose convolution (Referred as *T-Conv*$/2\uparrow$): Sparse transpose convolution with a stride of two is used to map the tensor coordinates back to their previous upscaled coordinates.

Transpose convolution generating new coordinates (Referred as *TG-Conv*$/2\uparrow$): This is a special case of sparse transpose convolution where the convolution generates new coordinates before aggregating features [52]. The new coordinates are generated on the voxels the kernel covers during convolution around the input coordinates. PU-Dense employs *TG-Conv* in the last upscaling layer of the decoder with a variable kernel size of ks. Variable kernel size is used here to optimize the architecture according to the resolution of the point cloud and the upsampling ratio. A larger ks is suitable for a sparser point cloud and would result in a larger number of output coordinates (C^{out}) generated. In this work, we employ a kernel size of $ks = 5$ for upsampling ratios of 4x and 8x.

Table I shows an example of sparse tensor sizes in the PU-Dense network during $4\times$ upsampling. Since a $ks = 5$ is used in *TG-Conv*, the size of the new generated coordinates in D_1 is 4,379,676. PU-Dense employs **Pruning** to choose the top$k$ features and their corresponding coordinates from D_0 to form the final output tensor O, where k is 830,397 in this particular example. *Pruning* is implemented on D_0 which has a feature length of 1 making it easier to choose the top$k$ features and their corresponding coordinates to create the output tensor O.

TABLE I

AN EXAMPLE OF TENSOR SIZES IN PU-DENSE ARCHITECTURE (FIG. 1) FOR $4\times$ UPSAMPLING USING $ks = 5$ FOR *longdress* PC FROM 8I

Tensor	Size of coordinates (C)	Size of features (F)
GT	830,397	1
E_i	207,599	1
E_1	207,599	32
E_2	139,244	32
E_3	52,612	64
E_4	14,440	128
E_5	3,623	256
D_4	14,440	256
D_3	52,612	128
D_2	139,244	64
D_1	4,379,676	32
D_0	4,379,676	1
O	830,397	1

1) Feature Extraction Unit: We introduce a novel feature extraction unit containing two *Inception-Residual Blocks* and a single *3D Dilated Pyramid Block* as shown in Fig. 2.

Inception-Residual Block (IRB) is inspired by Inception-ResNet architecture [53] which was originally proposed for 2D images. IRB has been implemented with great success using 3D sparse convolutions in earlier works [54]. Similar to 2D Inception-ResNet architecture, IRB employs filters of different sizes on the same feature scale while squeezing the feature domain and adding a residual link to the inception block. The IRB block employs $1 \times 1 \times 1$ convolutions which do not learn any spatial patterns but do learn patterns across the depth (cross channel) of each occupied voxel in the sparse tensor. Combining kernels

of different sizes ($1 \times 1 \times 1$ and $3 \times 3 \times 3$) allows the block to learn patterns across the depth (channel) as well across spatial patterns. This combined with a residual link leads to a computationally efficient layer which much like its 2D counterpart, Inception-ResNet architecture, is more discriminative and converges faster [53].

3D Dilated Pyramid Block (3D-DPB) is inspired by the success of spatial pyramid pooling [55] as well as pyramid blocks in image segmentation tasks [56] which showed that it is effective to resample features at different scales for accurately and efficiently classifying regions of an arbitrary scale. In 3D-DPB, we employ multiple parallel 3D sparse convolutions with different dilation rates to implement a 3D pyramid architecture. Dilation convolution, also called atrous convolution [57], allows us to arbitrarily enlarge the field-of-view of filters at any convolutional layer. A 3D global pooling is also implemented, which is followed by broadcasting the pooled feature to all the occupied coordinates. The features from all 3D-DPB layers are then concatenated followed by a $1 \times 1 \times 1$ convolution to refine the features. 3D-DPB probes the incoming sparse tensors with filters at multiple sampling rates and effective fields-of-views, thus capturing objects as well as geometric context at multiple scales. 3D-DPB with different dilation rates effectively captures multi-scale information, allowing us to extract denser feature maps by arbitrarily enlarging the receptive field.

C. Loss Function

The loss functions used in previous works usually employ distance-based metrics like chamfer distance, reconstruction loss, or repulsion loss, which tend to be memory inefficient and fail to learn accurate reconstruction. This is one of the reasons why the previous works are limited to processing small point clouds or small patches (usually less than 1024 points). PU-Dense implements an efficient voxel-based binary occupancy classification loss [54], [58] that allows us to process millions of points at a time. During training, PU-Dense applies Binary Cross Entropy (BCE) loss on the output of the decoder (D_0) and compares the occupied voxels to the ground truth point cloud (GT) as shown in Fig. 1. BCE loss during training is calculated using the following formula:

$$L_{BCE} = -\frac{1}{N}\sum_{i}(x_i log(p_i) + (1 - x_i) log(1 - p_i)) \quad (2)$$

where x_i is the voxel label that is either occupied (1) or empty (0) in the GT point cloud. p_i is the probability of the voxel being occupied and is calculated using a *sigmoid* function applied to the decoder output D_0.

D. Scalability

To the best of our knowledge, all current deep learning-based point cloud upsampling models are constrained to a fixed number of input points and suffer from memory issues. These models can typically process a fixed number of points at one moment, e.g., PU-Net: 1024 points, EC-Net: 1024 points, PU-GAN: 256 points, PUGeo-Net: 256 points. All these methods employ patch-based point cloud processing methods. However, PU-Net is limited to smaller point clouds. EC-Net, PU-GAN, and 3PU have extended their patch-based approaches to partition larger point clouds into smaller overlapping patches to operate on the individual patch separately. These patches are upsampled independently, then merged together. The merged pointset is then resampled using farthest point sampling to obtain uniform point distribution despite overlapping regions. Because of all the pre-processing and post-processing involved, these methods work well on smaller point clouds but are very inefficient when applied to point clouds with a large number of points.

Our method is the first fully convolutional upsampling method and, therefore, has variable input point cloud size. Moreover, since we employ efficient sparse convolutions and also have a memory-efficient loss function, we can process a much larger number of points. To make the network further scalable to even bigger point clouds, we employ a simple kd-tree partition that divides the point cloud into smaller non-overlapping point clouds that can be processed separately as well as in parallel.

IV. Experimental Results

We performed extensive experiments, quantitatively and qualitatively, compared our methodology with state-of-the-art point cloud upsampling methods, and evaluated various aspects of our model. We perform comprehensive experiments to test PU-Dense on different scenarios: (1) Test the proposed method on real-world scanned objects, (2) Show visual results for mesh surface reconstruction of the upsampled point clouds, (3) Test the robustness of our method against Gaussian noise, (4) Show comparative results for inference time and trainable parameters for different methods, (5) Perform ablation study to show the effectiveness of different components of PU-Dense. Further experimental details, extended results, and additional visual results can be found in the supplemental material.

A. Dataset

We train our network on *ShapeNet* dataset [59] while test it on *ShapeNet*, *8iVFB v2* [60], *8iVSLF* [61], *Technicolor*, *ScanObjectNN* [62], and KITTI [63] datasets. We evaluate the performance of our approach on a diverse set of point clouds in terms of spatial density and content type. *ShapeNet* is a mesh-based dataset, *ScanObjectNN* dataset is a real-world scanned object dataset containing sparse point cloud objects, *KITTI* is a dynamically acquired outdoor LiDAR dataset, whereas *8iVFB v2*, *8iVSLF*, and *Technicolor* are real-world captured dense photo-realistic dynamic point cloud datasets for immersive communication used in MPEG [64] and JPEG [7] point cloud compression standardization.

- **ShapeNet**. We randomly select ≈24000 3D mesh models from the core dataset of ShapeNet. We sampled the mesh model into point clouds by randomly generating points on the surfaces of the mesh, then randomly rotated and quantized the point cloud to 7-bit precision. The size of the point cloud in this dataset is between 5 000 to 50 000 points per point cloud.
- **8iVFB v2**. JPEG Pleno's 8i Voxelized Full Bodies dataset [60]. This is a dynamic voxelized point cloud

dataset where each sequence represents a 10 second long video captured at 30 fps with a total of 300 frames. Each frame has a resolution of 1024 with 10-bit precision. We use four sequences from this dataset: *Longdress*, *Loot*, *Red and Black*, and *Soldier* containing about 770 000, 790 000, 730 000, and 1 060 000 points per frame respectively.

- **8iVSLF**. MPEG's 8i Voxelized Surface Light Field dataset [61] of dynamic point clouds. It has a resolution of 4096 with 12-bit precision. We use two sequences from this dataset *Boxer* and *Thaidancer* containing about 3 130 000 and 3 490 000 points respectively.
- **Technicolor**. We employ the dynamic point cloud sequence *Queen* produced by Technicolor (https://www.technicolor.com/fr) with about 1 million points per frame.
- **ScanObjectNN**. [62] This is a real-world scanned object dataset which contains $\approx 15,000$ objects that are categorized into 15 categories with 2902 unique object instances. Compared with other datasets, this is a low-resolution sparse point cloud dataset with 2048 points per point cloud. We compare visual results for *ScanObjectNN* later in this section.
- **KITTI**. [63] This is a dynamically acquired dataset captured by LiDAR for autonomous driving. This is a sparse dataset that captures outdoor scenes. We perform visual comparisons between different methods when evaluated on KITTI dataset and show the results later in this section.

B. Implementation Details

We use kernel size, $ks = 5$ in our experiments. Unlike previous work where small patches had to be extracted from the training dataset and fed into the network, we trained our network with a batch size of 8 by feeding eight *ShapeNet* point clouds into our network each iteration. We implemented the proposed framework in PyTorch with Minkowski Engine [29].

C. Evaluation Metrics

Previous works have employed different quality assessment metrics; some works have even introduced their novel evaluation metrics. We consider commonly-used evaluation metrics that compare the reconstructed point cloud to the ground truth point cloud to quantitatively evaluate the performance of different methods. These methods are Chamfer distance (CD), point-to-point (D1) based mean squared error peak signal-to-noise ratio (D1 PSNR), point-to-plane (D2) based mean squared error peak signal-to-noise ratio (D2 PSNR), and point-to-point (D1) based Hausdorff PSNR. *MSE D1 PSNR*, *MSE D2 PSNR*, and *Hausdorff PSNR* has been adopted by both MPEG and AVS standards as an evaluation metric for point cloud quality [65]. We obtain the geometry PSNRs using the *pc_error* MPEG tool [65].

We define the Chamfer distance between PC A and PC B as:

$$d_{CD}(A,B) = \sum_{x \in A} \min_{y \in B} ||x-y||_2^2 + \sum_{y \in B} \min_{x \in A} ||x-y||_2^2 \quad (3)$$

Intuitively, the first term measures an approximate distance from each upsampled point to the target surface, and the second term rewards an even coverage of the upsampled surface and penalizes gaps.

The geometry quality metrics point-to-point MSE PSNR is obtained from a normalization factor and the mean squared error (MSE), as defined in (4) which is computed from PC A to PC B as well as in the opposite direction. The PSNRs of the two directions are then combined to obtain a single symmetric PSNR value with the maximum pooling function, as defined in (5).

$$PSNR_{A,B}^{MSE} = 10 \log_{10} \left(\frac{p_s^2}{d_{A,B}^{MSE}} \right) \quad (4)$$

$$PSNR^{MSE} = min(PSNR_{A,B}^{MSE}, PSNR_{B,A}^{MSE}) \quad (5)$$

In (4), p_s is signal peak and $d_{A,B}^{MSE}$ is the average squared error (i.e., MSE) between all points in PC A and their corresponding nearest neighbor point in PC B. The point-to-point MSE is computed from the point-to-point distance or error $\overrightarrow{e}(i,j)$ between each point in PC A and its nearest neighbor in PC B, $d_{A,B}^{Po2Po}$;

$$d_{A,B}^{MSE} = \frac{1}{N_A} \sum_{\forall a_i \in A} d_{A,B}^{Po2Po}(i) \ \text{, with} \quad (6)$$

$$d_{A,B}^{Po2Po} = ||\overrightarrow{e}(i,j)||_2^2 \quad (7)$$

As far as the signal peak p_s in (4) is concerned, the largest diagonal (LD) distance of the PC bounding box is typically used for non-voxelized data. For *ShapeNet*, *8iVFB* and *Queen* datasets we use $p_s = 1024$. For *8iVSLF* dataset we use $p_s = 4096$. In the D2 PSNR, MSE is still based on the distance between each point to its nearest neighbor but this distance is now computed from the projection of the point-to-point error vector $\overrightarrow{e}(i,j)$ along the normal vector of the underlying surface at point j in PC B.

Hausdorff PSNR is derived in a similar symmetrical manner but instead of using point-to-point MSE distance, the Hausdorff distance is employed. The Hausdorff distance is defined as the largest distance between all the points in point cloud A and their nearest neighbor in reference point cloud B, thus defining the Hausdorff distance as:

$$d_{A,B}^{Haus} = \max_{\forall i \in A} d^{A,B}(i) \quad (8)$$

D. Comparison With State-of-the-Arts

We were unable to run optimization-based methods like **EAR** on large point clouds like the 8iVFB dataset. **PU-Net** [22] is an earlier work, the code of which is not scalable to larger point clouds, so we did not include it in our comparisons. We chose four state-of-the-art upsampling methods that were scalable to larger point clouds and have so far shown the best results in point cloud upsampling: **PU-GAN** [25], **3PU** [24], **PU-GCN** [48] and **Dis-PU** [49]. Their models are trained with the author-released code, and all settings are the same as stated in their papers. These networks were originally trained on relatively sparser point clouds generated from small synthetic mesh-based objects. To make the comparison fairer, we retrained these networks

TABLE II
EXTENDED COMPARATIVE RESULTS (CD (10^{-2}) AND PSNR)

Dataset	Upsampling Method	4x		8x	
		CD (10^{-2}) ↓	MSE PSNR (dB) ↑	CD (10^{-2}) ↓	MSE PSNR (dB) ↑
ShapeNet	Downsampled PC	108.18	64.63	199.94	61.96
	3PU	76.36	68.65	149.20	65.37
	PU-GAN	49.41	70.64	174.58	64.88
	PU-GCN	48.15	70.90	65.81	69.59
	Dis-PU	36.23	72.19	55.62	70.23
	PU-Dense (Ours)	**18.82**	**75.24**	**30.52**	**73.11**
8iVFB	Downsampled PC	114.63	64.38	222.91	61.49
	3PU	67.04	69.41	105.43	66.83
	PU-GAN	45.60	70.92	117.66	66.19
	PU-GCN	46.30	70.96	63.71	69.78
	Dis-PU	32.47	72.72	51.68	70.59
	PU-Dense (Ours)	**19.38**	**75.05**	**33.18**	**72.57**
8iVSLF	Downsampled PC	286.67	73.17	600.34	70.00
	3PU	135.41	78.98	202.82	76.78
	PU-GAN	121.52	79.58	231.39	76.34
	PU-GCN	103.84	80.28	138.81	79.17
	Dis-PU	91.99	81.16	129.79	79.52
	PU-Dense (Ours)	**58.38**	**83.93**	**102.82**	**81.79**
Queen	Downsampled PC	106.69	64.69	196.46	62.04
	3PU	57.13	70.19	90.90	67.55
	PU-GAN	41.67	71.43	110.42	66.36
	PU-GCN	42.67	71.24	59.47	70.01
	Dis-PU	30.29	73.08	47.32	70.90
	PU-Dense (Ours)	**15.76**	**75.93**	**25.45**	**73.76**

TABLE III
QUANTITATIVE COMPARISON WITH STATE-OF-THE-ART APPROACHES USING D2 PSNR FOR 4× AND 8× UPSAMPLING

Dataset	Upsampling Method	D2 PSNR (dB)	
		4x	8x
8iVFB	3PU	72.71	69.02
	PU-GAN	74.21	67.96
	PU-GCN	75.66	74.05
	Dis-PU	76.70	74.79
	PU-Dense (Ours)	**79.05**	**76.45**

on the same data as our model, i.e., ShapeNet, and then tested them on our test datasets. We generated about 24000 patches from ShapeNet dataset for training purposes. For 3PU a patch size of 1024 was used for 8× upsampling and 2048 for 4× upsampling, For PU-GAN, PU-GCN, and Dis-PU a patch size of 256 was used for 4× as well as 8× upsampling.

E. Objective Evaluation

For *Chamfer Distance (CD)*, lower values are better whereas for *D1 PSNR*, *D2 PSNR* and *Hausdorff PSNR* higher values are better. We compare our work with other state-of-the-art methods and display the results in Table II, Table III, and Table IV. In Table II we show the *Chamfer Distance (CD)* and *D1 PSNR* comparison results on four different datasets for both 4× as well as 8× upsampling for all five compared methods. The comparative results show that PU-Dense outperforms other state-of-the-arts by substantially decreasing the CD while significantly improving the D1 PSNR. Among the previous methods, PU-GCN and Dis-PU perform decently well on all the datasets. Dis-PU performs the best among the previous works. PU-GCN and Dis-PU perform similarly for 8× upsampling, however, Dis-PU performs much better than PU-GCN at 4× upsampling. Overall, PU-Dense outperforms all of these methods.

We measure the point-to-plane D2 PSNR using the normals of the ground truth point clouds using 8iVFB dataset and show the results in Table III. We can see that PU-Dense outperforms the state-of-the-art on D2 metric also. The Hausdorff PSNR is also calculated on 8iVFB dataset, the results of which are shown in Table IV. Hausdorff distance tends to be sensitive to outliers because the metric distance corresponds to the greatest of all the distances from a point in one point cloud to the closest point in the other point cloud (original to reconstructed, and vice-versa) [66]. The Hausdorff PSNR decreases for both PU-GAN and 3PU whereas PU-GCN, Dis-PU, and PU-Dense improve the Hausdorff PSNR. We see that PU-Dense outperforms the state-of-the-arts in this metric too. While the distance-based distortion loss function tends to add outliers in the upsampling process, PU-Dense generates points closest to the actual point cloud surface with limited outliers. This is because the new points generated by PU-Dense are generated using *TG-Conv*, which generates new points in the neighborhood of the already occupied voxels. This neighborhood is determined by the kernel size ks.

F. Visual Comparisons

To visualize large photo-realistic point clouds like 8i sequences, it is not possible to show point-level visual

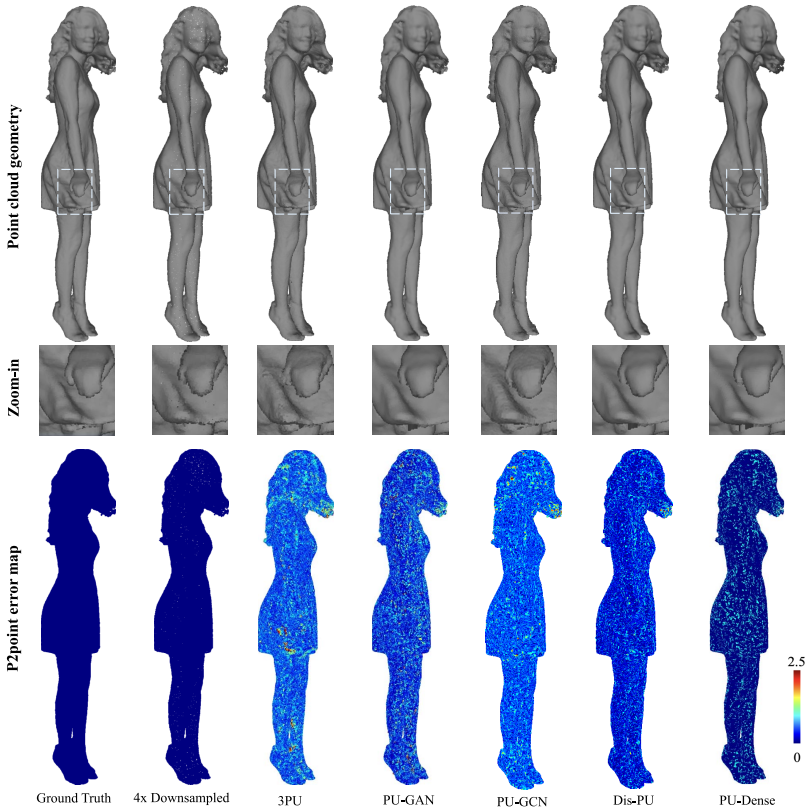

Fig. 3. 4× upsampling visual results on sequence *Red and Black*.

TABLE IV
QUANTITATIVE COMPARISON WITH STATE-OF-THE-ART APPROACHES USING HAUSDORFF PSNR FOR 4× AND 8× UPSAMPLING

Dataset	Upsampling Method	Hausdorff PSNR (dB)	
		4x	8x
8iVFB	Downsampled PC	51.95	49.10
	3PU	45.56	45.51
	PU-GAN	46.97	41.38
	PU-GCN	52.29	49.48
	Dis-PU	52.86	49.63
	PU-Dense (Ours)	**52.97**	**49.65**

upsampling results. Therefore, for these point clouds, we visualize their geometry by calculating their normals and with vertical shading. We also plot the error map based on the point-to-point (P2point) D1 distance between the point cloud and its ground truth to visualize the error distribution. We show the visual results of our experiments for 4× upsampling in Fig. 3 and for 8× upsampling in Fig. 4 using a point cloud frame from sequence *Red and Black* and *Longdress* respectively. We can see that our method generates limited outliers while populating points very close to the surface of the ground truth point cloud. This results in a high-resolution point cloud with considerably better quality than the other state-of-the-arts. Since PU-Dense uses *TG-Conv* to generate new points around the currently occupied voxels, there are limited outliers generated as is evident in Fig. 3 and Fig. 4. We also show the zoomed-in point cloud geometry views. The results of our proposed method are more precise, sharper, and cleaner, especially around key positions, such as corners and edges. We show further visual results on different datasets later in this section. We show further visual results on additional datasets in the next sections. Additional visual results and a description of how these figures are generated are also provided in the supplemental materials.

G. Evaluation on Real-World Scanned Objects

We also examined the performance of the proposed method on real-world scanned object dataset with 2048 points from *ScanObjectNN*. We show the comparative visual results of upsampling a point cloud from *ScanObjectNN* dataset in Fig. 5. *ScanObjectNN* dataset is captured through LiDAR and consists of sparse point clouds. We visualize the upsampled point cloud as well as the reconstructed mesh surface using the ball-pivoting algorithm for the input point cloud and each of the upsampled point clouds. As can be seen in Fig. 5, the proposed method (PU-Dense) works well on even sparse object point cloud datasets. In the upsampled point cloud generated by PU-Dense, the points generated are a lot more structured. This is due to the generation of new points using TG-Conv with binary voxel classification and pruning that generates newer points around the previously occupied voxels. We can notice that the holes in the point cloud get inpainted by the other methods, whereas, PU-Dense keeps the original shape of the point cloud. The reconstructed mesh

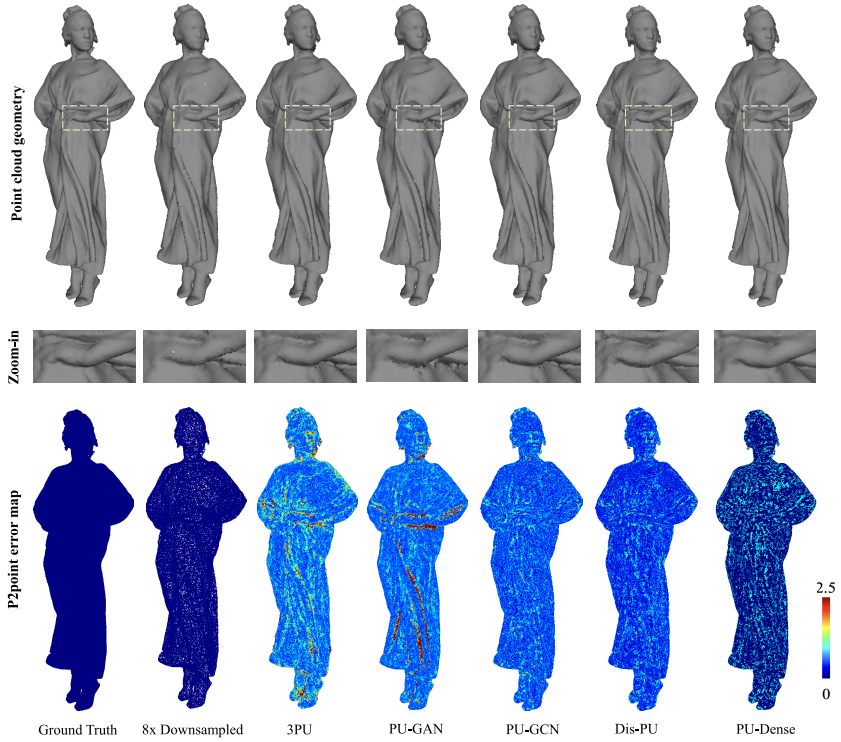

Fig. 4. 8× upsampling visual results on sequence *Longdress*.

surface quality for PU-dense is also better compared with the state-of-the-art. We can see PU-GCN and Dis-PU performs better than 3PU and PU-GAN on this dataset.

H. Dynamically Acquired Outdoor LiDAR Dataset

We examine the performance of PU-Dense, as well as 3PU, PU-GAN, PU-GCN, and Dis-PU on dynamically-acquired outdoor LiDAR dataset from KITTI [63] used for autonomous driving. This dataset is much more sparse compared to 8iVFB, 8iVSLF, and Queen datasets and also has non-uniform point distribution. We show the visual results from four street point clouds in Fig. 6. Even for sparse and non-uniform point clouds from the real-world LiDAR dataset, our proposed method can significantly improve the quality by upsampling these point clouds. In our experiments, all five methods are trained on regularly sampled and relatively dense ShapeNet dataset. Therefore, it is a considerable achievement for PU-Dense to be able to generalize to a sparse non-uniform dataset. Whereas the performance of the other methods suffers when evaluated on the KITTI dataset. This is because PU-Dense is fully convolutional making it translation-invariant that can process point clouds with a different number of points and sparsity levels. Whereas the other methods are patch-based solutions where the sparsity of the patch greatly influences the results. When the networks were trained on ShapeNet, the patch size was much smaller compared to an outdoor LiDAR point cloud. We can see that the patch-based upsampling results in the clustering of the points together within the patch. We believe this might be due to uneven sampling because of the *farthest point sampling* employed in these patch-based methods where not enough patches are sampled closer to the LiDAR resulting in clustering. PU-GCN and Dis-PU perform better than 3PU and PU-GAN on the KITTI dataset when trained on ShapeNet.

I. Mesh Generation

Fig. 7 shows the visualized results of 3D surface reconstruction using the ball-pivoting algorithm for the ground truth, input point cloud, and the upsampled point clouds from 3PU, PU-GAN, PU-GCN, Dis-PU, and PU-Dense on ShapeNet dataset. In the 3D mesh reconstruction task, the result is greatly influenced by the density as well as the quality of the upsampled point cloud. We can see the effectiveness of our proposed upsampling method by surface reconstruction in Fig. 7. For both 4× and 8× upsampling, surfaces reconstructed from PU-Dense upsampled point clouds are a lot more structured and exhibit richer geometry details. PU-Dense can recover more details and better preserve the smoothness of smooth regions as well as preserve the sharp edges without creating outliers. Although we notice that PU-GCN, and Dis-PU surface reconstruction qualities are much better compared to 3PU and PU-GAN. We also notice that PU-GAN produces a lot more outliers when it comes to 8× upsampling as seen in both Fig. 4 and Fig. 7.

J. Evaluation on Noisy Data

In this section, we evaluate the robustness of different methods to noise. Fig. 9 quantitatively compares the PU-Dense, 3PU, and PU-GAN on different levels of noise using ShapeNet dataset for 4× upsampling. We plot D1 MSE PSNR for each method under 7 levels of Gaussian noise. It can be seen that

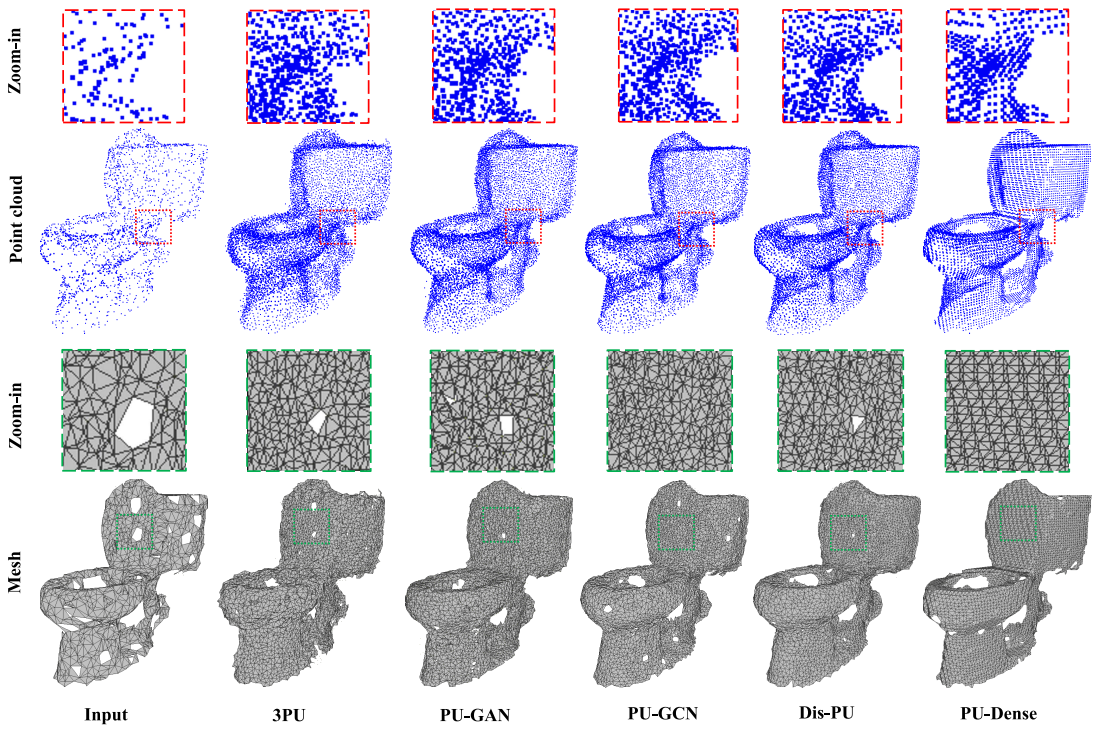

Fig. 5. Visual comparison for 4× upsampling on ScanObjectNN dataset. Input is the real-scanned point cloud object with 2048 points. The rest of the five are the output for each of the methods: 3PU, PU-GAN, PU-GCN, Dis-PU, and PU-Dense (Ours). The top row shows the zoomed-in regions from the point cloud. The third row is the zoomed-in mesh surface from the mesh reconstructed surface from corresponding point clouds using the ball-pivoting algorithm.

TABLE V
QUANTITATIVE COMPARISON: AVERAGE EVALUATION TIME PER POINT CLOUD FOR 4× UPSAMPLING ON 8IVFB DATASET

Upsampling Method	Inference time (sec)	Computation time (min)
3PU	123.76	24.49
PU-GAN	31.55	23.60
PU-GCN	**19.64**	23.20
Dis-PU	34.13	23.77
PU-Dense (Ours)	40.76	**00.79**

the performance of all methods decreases as the noise level increases. Nevertheless, the proposed method consistently achieves the best performance under each noise level. For PU-Dense, we notice the sharpest decrease when the noise is initially added and then the quality consistently decreases with an increase in noise. The robustness against noise can be further increased by augmenting noise during training of PU-Dense, something that was not employed during our training. We further show visual upsampling results for the PU-Dense on various noisy point clouds in Fig. 8. We visualize both 4× and 8× upsampling with 0%, 1%, and 2% Gaussian noise. We observe that even after adding noise, the upsampled point clouds visually look similar to the noise-free upsampled point cloud, demonstrating the robustness of the proposed method against noise.

K. Inference Time and Trainable Parameters

We compare the average computational time as well as average inference time to 4× upsample a single 8iVFB point cloud in Table V. The time in the experiments is calculated as the average time to process a single 8iVFB point cloud on NVIDIA GeForce GTX 1080 Ti GPU. The computation time includes the whole end-to-end pipeline including the pre-processing and post-processing while the inference time includes the time required by the network to infer on all the patches. Note that the implementation of *3PU*, *PU-GAN*, *PU-GCN*, and *Dis-PU* is not optimized for larger point clouds. Since *3PU*, *PU-GAN*, *PU-GCN*, and *Dis-PU* are patch-based solutions, most of the computational time is spent in pre-processing and post-processing. This includes employing farthest point sampling to sample smaller overlapping patches from the surface of the point cloud and then processing these individual patches. After the processing, these patches are merged and then downsampled by again employing farthest point sampling. *PU-Dense* is a lot faster compared to other state-of-the-art in processing point clouds, because it employs efficient sparse convolutions, has limited pre-processing and post-processing, and employs an efficient binary voxel-classification loss all of which enables PU-Dense to process a large point cloud (up to 1 million points) in a single iteration without running out of GPU memory (NVIDIA GeForce GTX 1080 Ti).

We also compare the number of trainable parameters in each of the upsampling methods in Table VI. We can see that *PU-Dense* has many more trainable parameters and is much larger than the other networks. This is partly because the loss functions in other methods often employ very large matrix multiplications (e.g., Chamfer loss) that consume a lot of memory. Another advantage of PU-Dense is that it utilizes sparse convolution which takes advantage of the sparse nature

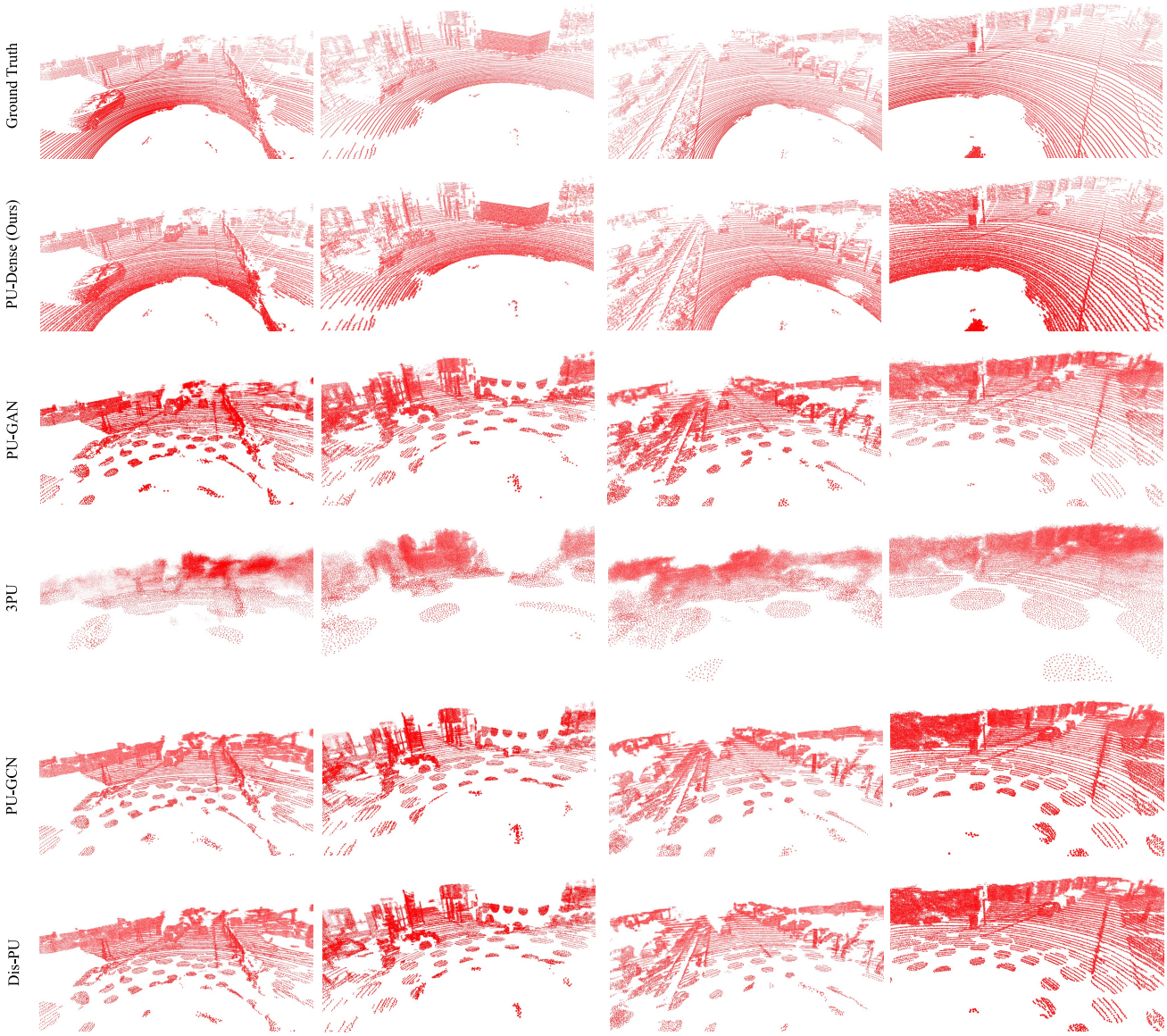

Fig. 6. Visual comparisons for outdoor LiDAR dataset from KITTI for 4× upsampling.

TABLE VI
QUANTITATIVE COMPARISON: NUMBER OF TRAINABLE PARAMETERS

Upsampling Method	Trainable parameters
3PU	152,054
PU-GAN	541,601
PU-GCN	75,971
Dis-PU	1,046,966
PU-Dense (Ours)	**13,172,441**

of the point cloud and is, therefore, memory efficient. This allows PU-Dense to build a much deeper and wider model which was not possible in the previous works. PU-Dense is the first deeper point cloud upsampling network whereas all the previous methods are much shallower. This makes PU-Dense more powerful, with higher discriminative power, as well as a larger receptive field. PU-Dense can process up to a million points per inference due to its memory efficiency.

L. Ablation Study

We perform a quantitative comparison of different versions of PU-Dense by removing specific components from the PU-Dense pipeline and tabulating the results in Table VII. First, we removed the *3D-DPB* from our *FE Unit* and replaced it with another *IRB*. This network is shown as *PU-Dense w/o 3D-DPB* in Table VII and contains three *IRB* units in each *FE Unit*. Then, in the second method, we removed the *IRB* units from the *FE Unit* and replaced them with *3D-DPB*. We call this *PU-Dense w/o IRB*. Then, in the third method, we removed both the *IRB* unit as *3D-DPB* unit from the *FE Unit* and replaced them with multiple *ResNet* blocks with approximately the same number of parameters as in

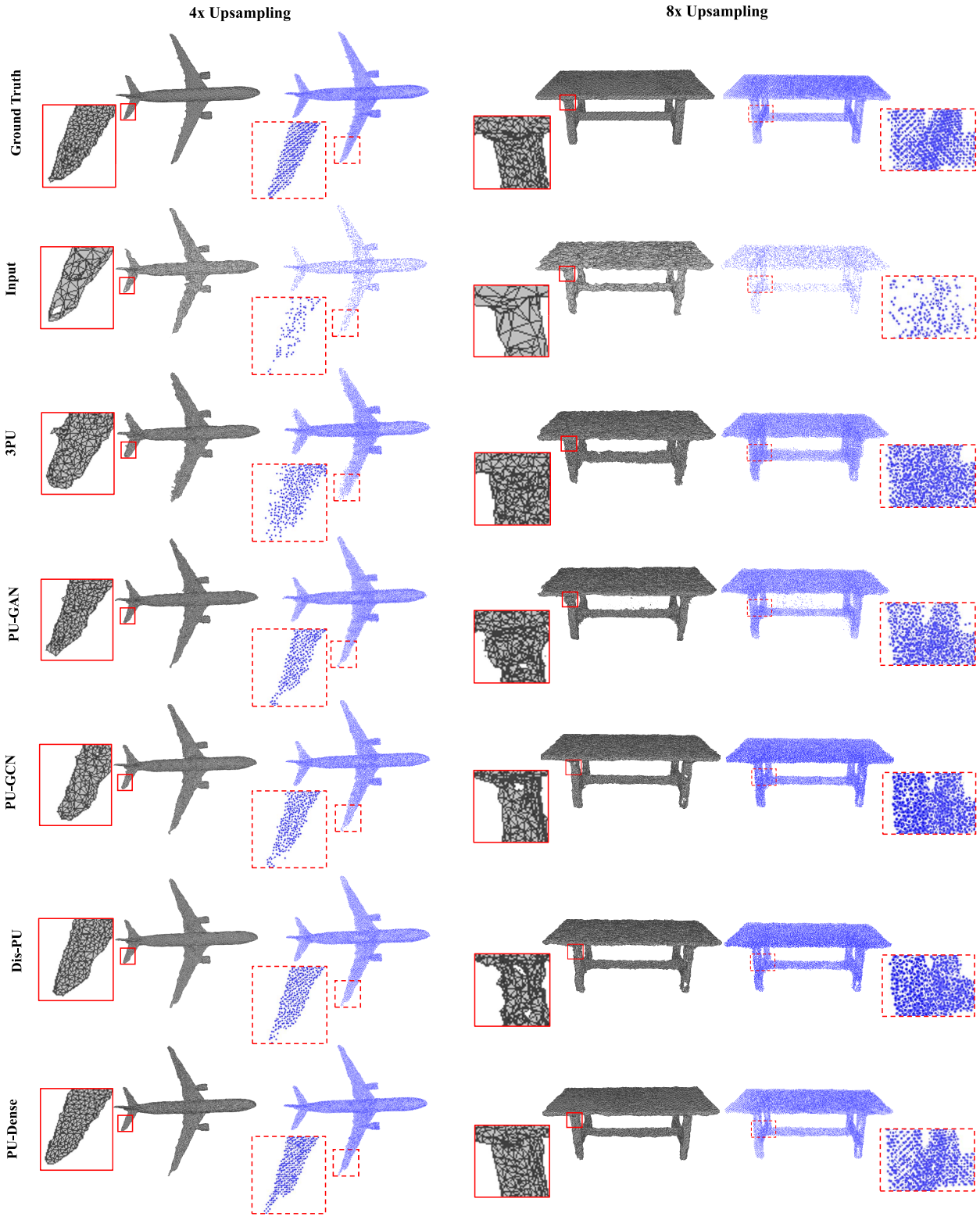

Fig. 7. Visual comparisons for mesh reconstruction using ball-pivoting algorithm for 4× and 8× upsampling on ShapeNet dataset.

the original *FE Unit*. This network is shown as *PU-Dense w/o IRB, 3D-DPB* in Table VII and contains three *ResNet* blocks per *FE Unit*. We can see that the complete PU-Dense pipeline gives the best performance and removing any one individual component greatly reduces the results, meaning that each component contributes to PU-Dense's efficacy. This also shows that employing both *3D-DPB* as well as *IRB* units for feature extraction yields the best results.

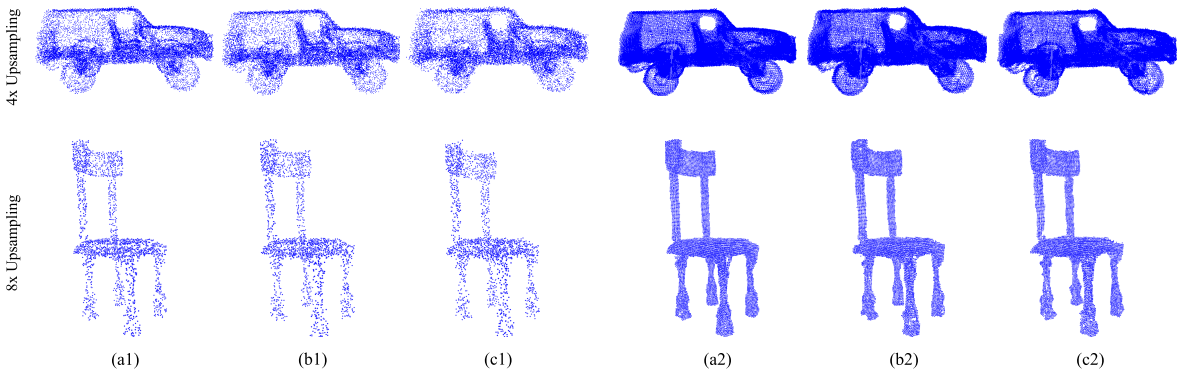

Fig. 8. Visual results for PU-Dense on noisy data for 4× and 8× upsampling. Left: (a1), (b1), and (c1) are the sparse inputs with 0%, 1%, and 2% Gaussian noise, respectively. Right: (a2), (b2), and (c2) are the upsampled results from (a1), (b1), and (c1) respectively.

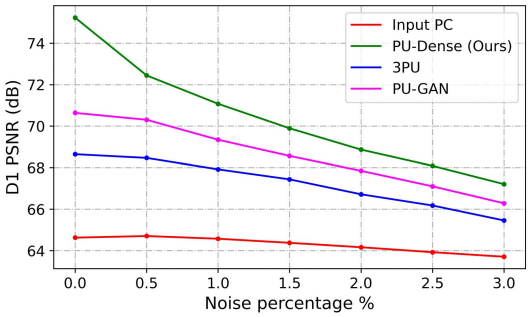

Fig. 9. Quantitative comparisons on data with various levels of noise for 4× upsampling using ShapeNet dataset.

TABLE VII

ABLATION STUDY: REMOVING SPECIFIC COMPONENTS FROM THE PU-DENSE PIPELINE FOR 4× UPSAMPLING

Dataset	Upsampling Method	CD (10^{-2})	MSE PSNR (dB)
8iVFB	PU-Dense w/o IRB, 3D-DPB	26.14	74.01
	PU-Dense w/o IRB	23.88	74.50
	PU-Dense w/o 3D-DPB	23.99	74.42
	PU-Dense	**19.38**	**75.05**

V. LIMITATIONS

In this section, we discuss the limitations of our proposed method and the future improvements that could be made to PU-Dense. We employ sparse tensors with 3D sparse convolutions along with a binary voxel classification loss. This makes PU-Dense computationally efficient allowing us to build a much larger network, process a dynamic number of input points, as well as allow PU-Dense to process high-resolution dense photo-realistic point clouds with millions of points. However, there is a downside to this approach. As a preprocess, we convert point clouds into sparse tensors by converting them into a 3D volumetric representation through voxelization. Sparse tensors allow us to only operate 3D convolutions on the occupied voxels while ignoring the empty voxels. We can populate the empty voxels using TG-Conv around an already occupied voxel and then prune them using binary voxel classification loss. The size of the kernel (ks) determines how far the newly occupied voxel could be from an already occupied voxel. This method works well for upsampling dense photo-realistic point clouds as well as sparse point clouds. However, this method would not be able to fill out larger holes that cannot be covered by the kernel size (ks). PU-Dense is not able to produce points at a location that has no nearby points within ks distance. For this reason, the current architecture would not work for inpainting tasks. However, this limitation can be easily rectified by employing multiple TG-Conv and pruning layers. We can simply replace all the T-Conv with TG-Conv + Pruning to be able to adapt this architecture for inpainting tasks.

VI. CONCLUSION

In this paper, we propose a novel point cloud upsampling method called *PU-Dense* that can upsample both synthetic as well as real-world captured point clouds including LiDAR scanned as well as dense high-resolution point clouds. PU-Dense incorporates hierarchical learning via progressive rescaling and multiscale feature extraction. PU-Dense utilizes a voxelized 3D representation with sparse convolutions backbone and introduces a novel *feature extraction (FE)* unit that contains *Inception-Residual Blocks* and a *3D Dilated Pyramid Block* to extract 3D multiscale features with different field-of-view in a computationally efficient manner. PU-Dense employs a memory-efficient voxel-based binary occupancy classification loss that can better reconstruct point clouds from features. Experimental results show that PU-Dense outperforms other state-of-the-arts by a significant margin on synthetic datasets, real-world LiDAR scanned datasets, as well as dense photo-realistic point clouds for immersive communication. While the other state-of-the-arts struggle to efficiently process high-resolution point clouds with a large number of points, PU-Dense can effectively learn 3D features and produce higher level-of-detail upsampled point clouds for both synthetic as well as real-world datasets. PU-Dense generates limited outliers while populating points very close to the surface of the ground truth point cloud resulting in a high-resolution point cloud with considerably better quality than the other state-of-the-arts. Furthermore, PU-Dense is more robust against Gaussian noise compared to other methods.

ACKNOWLEDGMENT

Project website: https://aniqueakhtar.github.io/publications/PU-Dense/

REFERENCES

[1] H. Fuchs, A. State, and J. C. Bazin, "Immersive 3D telepresence," *Computer*, vol. 47, no. 7, pp. 46–52, Jul. 2014.

[2] S. Orts-Escolano *et al.*, "Holoportation: Virtual 3D teleportation in real-time," in *Proc. 29th Annu. Symp. User Interface Softw. Technol.*, Oct. 2016, pp. 741–754.

[3] A. Akhtar *et al.*, "Low latency scalable point cloud communication in VANETs using V2I communication," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[4] A. Akhtar, B. Kathariya, and Z. Li, "Low latency scalable point cloud communication," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 2369–2373.

[5] L. Li, Z. Li, V. Zakharchenko, J. Chen, and H. Li, "Advanced 3D motion prediction for video-based dynamic point cloud compression," *IEEE Trans. Image Process.*, vol. 29, pp. 289–302, 2019.

[6] S. Schwarz *et al.*, "Emerging MPEG standards for point cloud compression," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 133–148, Mar. 2019.

[7] S. Perry, *JPEG Pleno Point Cloud Coding Common Test Conditions V3*, Standard ISO/IEC JTC1/SC29/WG1 N 86044, 2020.

[8] N. Haala, M. Peter, J. Kremer, and G. Hunter, "Mobile LiDAR mapping for 3D point cloud collection in urban areas—A performance test," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 37, pp. 1119–1127, Jul. 2008.

[9] X. Wen, T. Li, Z. Han, and Y.-S. Liu, "Point cloud completion by skip-attention network with hierarchical folding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1939–1948.

[10] X. Wang, M. H. Ang, Jr., and G. H. Lee, "Cascaded refinement network for point cloud completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 790–799.

[11] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le, "PF-Net: Point fractal network for 3D point cloud completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 7662–7670.

[12] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese, "TopNet: Structural point cloud decoder," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 383–392.

[13] P. H. Casajus, T. Ritschel, and T. Ropinski, "Total denoising: Unsupervised learning of 3D point cloud cleaning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 52–60.

[14] H. Zhou, K. Chen, W. Zhang, H. Fang, W. Zhou, and N. Yu, "DUP-Net: Denoiser and upsampler network for 3D adversarial point clouds defense," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1961–1970.

[15] C. Dinesh, G. Cheung, and I. V. Bajić, "Point cloud denoising via feature graph Laplacian regularization," *IEEE Trans. Image Process.*, vol. 29, pp. 4143–4158, 2020.

[16] J. Zeng, G. Cheung, M. Ng, J. Pang, and Y. Cheng, "3D point cloud denoising using graph Laplacian regularization of a low dimensional manifold model," *IEEE Trans. Image Process.*, vol. 29, pp. 3474–3489, 2019.

[17] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Trans. Vis. Comput. Graphics*, vol. 9, no. 1, pp. 3–15, Jan. 2003.

[18] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," *ACM Trans. Graph.*, vol. 26, no. 3, p. 22, Jul. 2007.

[19] H. Hui, L. Dan, Z. Hao, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–7, 2009.

[20] R. Preiner, O. Mattausch, M. Arikan, R. Pajarola, and M. Wimmer, "Continuous projection for fast L1 reconstruction," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–47, 2014.

[21] H. Huang, S. Wu, M. Gong, D. Cohen-Or, and H. Zhang, "Edge-aware point set resampling," *ACM Trans. Graph.*, vol. 32, no. 1, pp. 1–12, 2013.

[22] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2790–2799.

[23] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "EC-Net: An edge-aware point set consolidation network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 386–402.

[24] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-based progressive 3D point set upsampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5958–5967.

[25] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-GAN: A point cloud upsampling adversarial network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7203–7212.

[26] Y. Qian, J. Hou, S. Kwong, and Y. He, "PUGeo-Net: A geometry-centric network for 3D point cloud upsampling," in *Proc. Eur. Conf. Comput. Vis (ECCV)*. Springer, 2020, pp. 752–769.

[27] A. Akhtar, W. Gao, X. Zhang, L. Li, Z. Li, and S. Liu, "Point cloud geometry prediction across spatial scale using deep learning," in *Proc. IEEE Int. Conf. Vis. Commun. Image Process. (VCIP)*, Dec. 2020, pp. 70–73.

[28] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Intervent.* Springer, 2015, pp. 234–241.

[29] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3075–3084.

[30] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338–4364, Dec. 2021.

[31] Z. Wu *et al.*, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[32] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "SEGCloud: Semantic segmentation of 3D point clouds," in *Proc. Int. Conf. 3D Vis. (3DV)*, Oct. 2017, pp. 537–547.

[33] A. Kanezaki, Y. Matsushita, and Y. Nishida, "RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5010–5019.

[34] H. Su *et al.*, "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2530–2539.

[35] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12697–12705.

[36] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 945–953.

[37] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun, "GRNet: Gridding residual network for dense point cloud completion," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2020, pp. 365–381.

[38] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[39] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," 2017, *arXiv:1706.02413*.

[40] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on χ-transformed points," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 828–838.

[41] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.

[42] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[43] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.

[44] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4558–4567.

[45] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based convolutional neural networks for 3D shape analysis," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–11, 2017.

[46] B. Graham, M. Engelcke, and L. Van Der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9224–9232.

[47] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," 2017, *arXiv:1706.01307*.

[48] G. Qian, A. Abualshour, G. Li, A. Thabet, and B. Ghanem, "PU-GCN: Point cloud upsampling using graph convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11683–11692.

[49] R. Li, X. Li, P.-A. Heng, and C.-W. Fu, "Point cloud upsampling via disentangled refinement," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 344–353.
[50] S. Ye, D. Chen, S. Han, Z. Wan, and J. Liao, "Meta-PU: An arbitrary-scale upsampling network for point cloud," *IEEE Trans. Vis. Comput. Graphics*, early access, Feb. 9, 2021, doi: 10.1109/TVCG.2021.3058311.
[51] Y. Qian, J. Hou, S. Kwong, and Y. He, "Deep magnification-flexible upsampling over 3D point clouds," *IEEE Trans. Image Process.*, vol. 30, pp. 8354–8367, 2021.
[52] J. Gwak, C. Choy, and S. Savarese, "Generative sparse detection networks for 3D single-shot object detection," 2020, *arXiv:2006.12356*.
[53] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. AAAI Conf. Artif. Intell.*, 2017, vol. 31, no. 1, pp. 1–7.
[54] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale point cloud geometry compression," in *Proc. DCC*, Mar. 2021, pp. 73–82.
[55] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2014.
[56] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.
[57] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 801–818.
[58] W. Jia, L. Li, A. Akhtar, Z. Li, and S. Liu, "Convolutional neural network-based occupancy map accuracy improvement for video-based point cloud compression," *IEEE Trans. Multimedia*, vol. 24, pp. 2352–2365, 2021.
[59] A. X. Chang *et al.*, "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.
[60] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, *8i Voxelized Full Bodies—A Voxelized Point Cloud Dataset*, document WG11M40059/WG1M74006, 2017, vol. 7, p. 8.
[61] M. Krivokuca, P. A. Chou, and P. Savill, *8i Voxelized Surface Light Field (8iVSLF) Dataset*, document m42914, 2018.
[62] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1588–1597.
[63] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3354–3361.
[64] S. Schwarz, G. Martin-Cocher, D. Flynn, and M. Budagavi, *Common Test Conditions for Point Cloud Compression*, document ISO/IEC JTC1/SC29/WG11 w17766, Ljubljana, Slovenia, 2018.
[65] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3460–3464.
[66] A. Javaheri, C. Brites, F. Pereira, and J. Ascenso, "A generalized Hausdorff distance based quality metric for point cloud geometry," in *Proc. 12th Int. Conf. Quality Multimedia Exper. (QoMEX)*, May 2020, pp. 1–6.

Anique Akhtar (Student Member, IEEE) received the B.S. degree in electrical engineering from the Lahore University of Management Sciences (LUMS), Lahore, Pakistan, in 2013, and the M.S. degree from Koc University, Istanbul, Turkey, in 2015. He is currently pursuing the Ph.D. degree with the University of Missouri–Kansas City (UMKC). He worked on wireless communication in the past and is currently working at the Multimedia and Communications Laboratory, UMKC. His research interests include point cloud compression and communication, and deep learning solutions for point cloud processing. He is also working on computer vision tasks.

Zhu Li (Senior Member, IEEE) received the Ph.D. degree in electrical & computer engineering from Northwestern University in 2004. He was the AFRL Summer Faculty at the US Air Force Academy, UAV Research Center, in 2016, 2017, and 2018. He was a Senior Staff Researcher/Senior Manager with Samsung Research America's Multimedia Core Standards Research Laboratory, Dallas, from 2012 to 2015, a Senior Staff Researcher at FutureWei from 2010 to 2012, an Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University from 2008 to 2010, and a Principal Staff Research Engineer with the Multimedia Research Laboratory (MRL), Motorola Labs, Schaumburg, IL, USA, from 2000 to 2008. He is currently an Associate Professor with the Department of CSEE, University of Missouri–Kansas City (UMKC), Kansas City, MO, USA, directs the NSF I/UCRC Center for Big Learning, UMKC. His research interests include image/video analysis, compression, and communication and associated optimization, and machine learning problems. He has 46 issued or pending patents, more than 100 publications in book chapters, journals, conference proceedings, and standards contributions in these areas. He has been an Associate Editor-in-Chief (AEiC) of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM FOR VIDEO TECHNOLOGY since 2020, and served and serving as an Associate Editor for IEEE TRANSACTIONS ON IMAGE PROCESSING since 2019, IEEE TRANSACTIONS ON MULTIMEDIA from 2015 to 2019, and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM FOR VIDEO TECHNOLOGY from 2016 to 2019.

Geert Van der Auwera (Senior Member, IEEE) received the Belgian M.S.E.E. degree from Vrije Universiteit Brussel (VUB), Brussels, Belgium, in 1997, and the Ph.D. degree in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2007. In 2000, he joined the IWT-Flanders after researching wavelet video coding at IMEC's Electronics and Information Processing Department (VUB-ETRO), Brussels, Belgium. Until December 2004, he was a Scientific Advisor with IWT-Flanders, the Institute for the Promotion of Innovation by Science and Technology, Flanders, Belgium. Until January 2011, he was with Samsung Electronics, Irvine, CA, USA. In 1998, his M.S.E.E. thesis on motion estimation in the wavelet domain received the Barco and IBM prizes by the Fund for Scientific Research of Flanders, Belgium. He is currently the Director of the Multimedia Research and Development & Standards Group, Qualcomm Technologies Inc., San Diego, CA, USA, where he actively contributes to the standardization efforts on MPEG's point cloud compression and previously on JVET's Versatile Video Coding (VVC). His research interests include point cloud compression, XR, video coding, video traffic and quality characterization, and video streaming mechanisms and protocols.

Li Li (Member, IEEE) received the B.S. and Ph.D. degrees in electronic engineering from the University of Science and Technology of China (USTC), Hefei, Anhui, China, in 2011 and 2016, respectively. He was a Visiting Assistant Professor with the University of Missouri–Kansas City from 2016 to 2020. He joined the Department of Electronic Engineering and Information Science, USTC, as a Research Fellow, in 2020, and became a Professor in 2022. His research interests include image/video/point cloud coding and processing. He received the Best 10% Paper Award at the 2016 IEEE Visual Communications and Image Processing (VCIP) and the 2019 IEEE International Conference on Image Processing (ICIP).

Jianle Chen, photograph and biography not available at the time of publication.