A Conditional Denoising Diffusion Probabilistic Model for Point Cloud Upsampling Supplementary Materials

1. Additional Ablation Study

We provide additional ablation studies to deepen the analysis and understanding for our method.

Rate label form. Our method achieves high-quality arbitrary-rate sampling during inference by parameterizing a rate factor. Therefore, we explore the impact of different rate label forms for the model performance. We provide additional information regarding the scale of points, such as the number of points. As shown in Tab 1, the performance difference for the model is relatively small (the variation < 0.02 for CD). This reason is that the rate label is solely modeled to identify the scale difference between the sparse point cloud and the dense point cloud, thus it can not significantly improve the performance of the model.

Rate Label Forms	CD↓	HD↓	P2F↓
[256,1024,3]	0.143	1.258	**1.907**
[256,1024]	0.145	1.352	1.913
[0.256,1.024]	0.141	1.289	1.954
[3,]	**0.131**	**1.220**	1.912

Table 1. The ablation study of the rate label form at $4\times$ on PUGAN [6]. Our method performs optimally, when $r = [3,]$ (we set $r = [0,]$ to represent $1\times$).

Sampling intervals. Our method is formally based on conditional DDPM, thus inevitably lagging behind existing non-DDPM-based point cloud upsampling methods in terms of sampling speed. Therefore, we conduct the ablation study between the quality and time of generating point clouds under different sampling intervals. This only changes the sampling interval during inference, without retraining the model.

Tab 2 shows the trade-off between the performance and the generating time of our method under different sampling intervals. Surprisingly, although the overall performance of the model decreases with increasing sampling intervals, this does not follow a linear trend, presenting an irregular state. We believe that the performance variation of the model is not only related to the sampling interval but also to the distance from x_1 to x_0. When the distance of time step between x_1 and x_0 is smaller, the model performs better, such as the sampling interval $= 12$ (with a distance of 3 time steps between x_1 and x_0). In other words, when given some close time intervals, we should choose the sampling interval that brings x_1 closest to x_0, rather than the one that evenly divides the total time steps.

Intervals	Distance	CD↓	HD↓	P2F↓	Times(s)↓
50	49	0.320	2.738	3.145	0.410
40	39	0.303	2.608	3.124	0.473
30	9	0.221	2.234	2.054	0.584
20	19	0.248	2.380	2.064	0.784
12	3	0.202	1.976	1.984	1.302
10	9	0.210	2.001	2.087	1.392
1	1	**0.131**	**1.220**	**1.934**	14.773

Table 2. Ablation study of different sampling interval at $4\times$ on PUGAN [6]. "Distance" means the distance of time steps between x_1 and x_0. When the sampling interval $= 12$, the performance and sampling speed of our methods keep a favorable balance. Meanwhile, we showcase the visual results for the sampling interval $= 12$ and $= 30$ in Fig 5.

2. Additional Comparative Experiments

Sampling speed. We conduct the evaluation of sampling speed. We believe speed comparisons are more intuitive compared to Params and FLOPs, as most methods adopt an iterative approach to achieve point cloud upsampling (all experiments were conducted using an NVIDIA 3090 GPU). To constrain under DDPM formulation and accelerate inference, we adopted interval sampling acceleration (Sec 1). Simultaneously, we also provide the results using DDIM [13] in Tab 3 (without retraining). We found that incorporating the interval sampling trick (Sec 1) into DDIM yields significant results (large interval sampling in the first half and normal inference in the second half). In the second half of inference, interpolation points as guiding information can frequently and meticulously guide the noise generation direction (Eq 2 and Eq 9 in the main text), reducing the impact of rough inference in the first half, maintaining high-quality generation.

Other Noise. We conduct additional other noise experiments, to reveal the issue of the non-robustness of DDPM to non-modeled distributions. Despite the Tab 6 in main text showing the SOTA performance for PUDM on uniform noise, this is solely due to the high-performance baseline. Simultaneously, we provide results for approximating

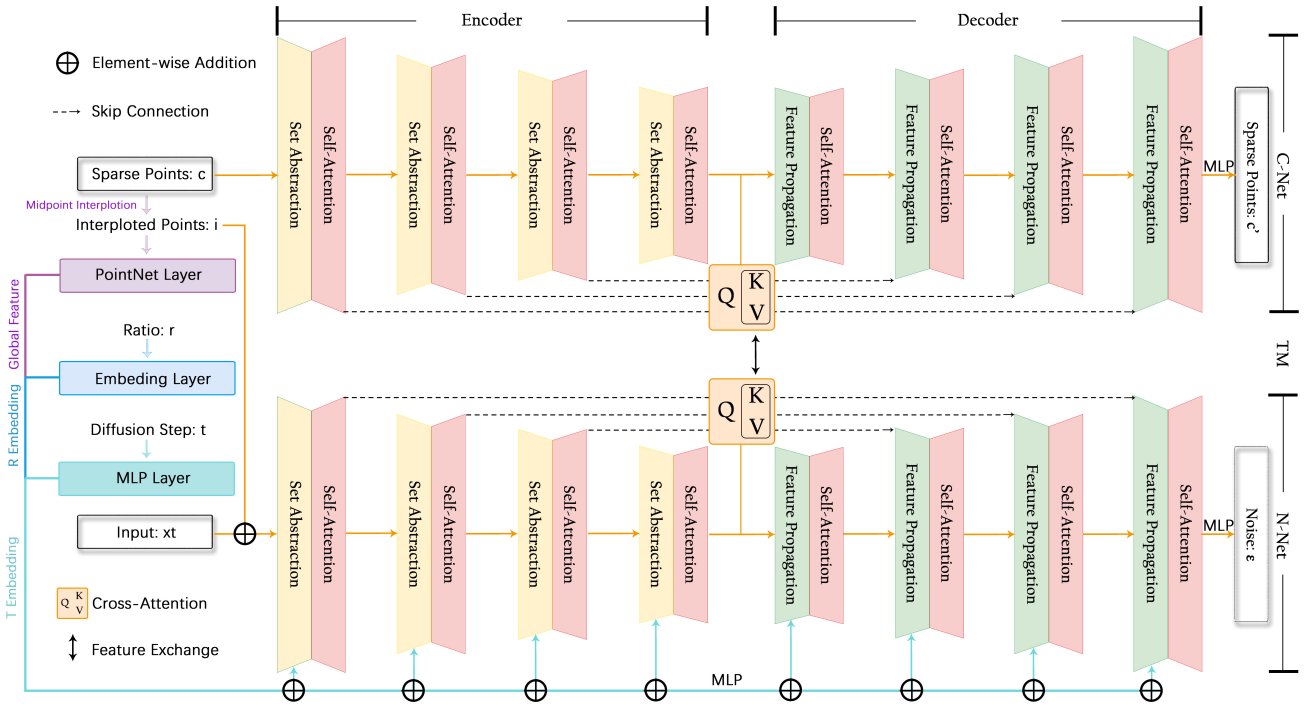

Figure 1. The network detail of PUDM. The encoder of the N-Net and the C-Net consists of set abstraction (SA) layers and self-attention layers. Simultaneously, the decoder consists of feature propagation (FP) layers and self-attention layers. The transfer module used for interaction between the N-Net and the C-Net consists of two cross-attention modules. Compared to the C-Net, the N-Net requires additional information for modeling the diffusion step: the global features, the R embedding, and the T embedding.

Methods	CD↓	HD↓	P2F↓	Params↓	FLOPs↓	Time↓
PU-Net [15]	0.529	6.805	4.460	0.814M	4.982G	0.446s
MPU [14]	0.292	6.672	2.822	0.076M	2.897G	0.487s
PU-GAN [12]	0.282	5.577	2.016	0.684M	0.974G	0.618s
Dis-PU [7]	0.274	3.696	1.943	0.105M	3.276G	0.724s
PU-EVA [8]	0.277	3.971	2.524	0.287M	10.377G	0.587s
PU-GCN [12]	0.268	3.201	2.489	0.076M	**0.410G**	0.531s
NePS [2]	0.259	3.648	1.935	0.664M	9.135G	0.479s
Grad-PU [4]	0.245	2.369	**1.893**	**0.067M**	9.135G	0.479s
Ours-30	0.132	1.311	1.998	16.034M	16.457G	0.507s
Ours-20	0.133	1.317	2.016	16.034M	16.457G	0.390s
Ours-10	0.145	1.349	2.102	16.034M	16.457G	0.266s
Ours-6	0.283	1.979	2.556	16.034M	16.457G	**0.219s**
Ours-1000	**0.131**	**1.220**	1.912	16.034M	16.457G	14.773s

Table 3. The results of $4\times$ on PUGAN. "Ours-X" indicates inferring X steps using DDIM. "FLOPs" means the computational cost for one inference step. PUDM meets the speed requirements.

Noise	Laplace ($\tau = 0.05$)			Poisson ($\tau = 0.05$)		
Methods	CD↓	HD↓	P2F↓	CD↓	HD↓	P2F↓
NePS [2]	1.045	9.114	17.845	1.698	14.01	25.484
Grad-PU [4]	0.964	8.364	16.915	**1.637**	13.784	25.157
Ours	**0.810**	**7.501**	**15.345**	1.644	**8.014**	**24.145**

Table 4. The results of the Laplace noise and the Poisson noise at $4\times$ on PUGAN.

Gaussian distribution noise (Laplace noise, $\mu = 0, b = 1$) and far from Gaussian distribution noise (Poisson noise, $\lambda = 3$) in Tab 4. This further validated the conclusion for Sec 5.3 in the main text. We believe that finding a distribution or a training strategy to ensure the robustness of DDPM across multiple or even all noise distributions is highly meaningful for the applications of DDPM.

Point Cloud Part Segmentation. We evaluate the quality of point cloud upsampling on point cloud part segmentation. Tab 5 displays extremely poor results for all point cloud upsampling methods (class accuracy $< 40\%$).

Existing point cloud upsampling methods (including our methods) are not proficient in semantic-related downstream tasks for the evaluation metrics such as point cloud segmentation and point cloud detection. This is because the semantic order of points is disrupted during the point cloud upsampling process, making each point struggle to align with the semantic label. However, this does not mean that point cloud upsampling cannot be applied to semantic-related downstream tasks entirely, since precise semantic labels are not required in practical applications.

In fact, because the semantic label is uniquely mapped to each point in point clouds, their distributions are inherently similar. Therefore, the model may be able to maintain the semantic order of points during upsampling point clouds.

Datasets	PointNet [10] (%)				PointNet++ [11] (%)			
Methods	IA↑	CA↑	Im↑	Cm↑	IA↑	CA↑	Im↑	Cm↑
Low-res	92.16	81.12	77.13	81.11	92.97	84.99	81.10	83.13
High-res	93.47	83.05	79.01	83.99	94.34	86.27	82.91	85.61
PU-Net [15]	51.92	36.05	32.69	35.79	52.04	36.25	32.66	35.90
MPU [14]	52.01	**36.16**	32.76	35.89	52.14	36.28	32.71	36.04
PU-GAN [12]	52.01	35.94	**32.78**	35.90	52.25	36.08	32.73	36.10
Dis-PU [7]	**52.38**	36.02	32.79	35.96	**52.71**	36.16	32.78	36.16
PU-EVA [8]	51.80	35.94	32.69	35.82	52.04	36.13	32.65	35.94
PU-GCN [12]	51.67	35.87	32.58	35.58	51.89	36.03	32.65	35.79
NePS [2]	51.71	35.91	32.63	35.61	52.01	36.11	32.67	35.87
Grad-PU [4]	52.22	36.02	32.75	**36.07**	52.46	35.29	32.97	36.31
Ours	51.88	36.08	32.67	36.00	52.11	**36.48**	**32.99**	**36.34**

Table 5. The results of point cloud part segmentation on ShapeNet [1]. "Low-res" refers to the point cloud subsampled with 512 points, while "High-res" denotes the original test point cloud with 2048 points. Meanwhile, "IA" stands for instance accuracy, and "CA" denotes class accuracy. "Im" means instance mIoU, and "Cm" denotes class mIoU. All methods yield very poor results, as existing point cloud upsampling methods struggle to maintain the semantic order of points.

3. Implementation

The network detail of PUDM is shown in Fig 1. We employ the same training configuration for PUGAN and PU1K. Specifically, we configure batch size $=$ 28, and conduct 1000 epochs using an NVIDIA 3090 GPU, taking approximately 5 days for PUGAN. For the C-Net and the N-Net, the sampling points/the channel dimensions are (1024, 256, 64, 16)/(64, 128, 256, 512) and (1024, 256, 64, 16)/(128, 256, 256, 512), respectively. Meanwhile, in the TM, we set the number $= 4$ of head to improve the modeling capacity of our model, and the latent dimension $= 64$. In addition, the global features with a dimension of 1024 are extracted from the interpolated point cloud i through a two-stage PointNet [10]. The parameters of the R embedding layer are $(256, 128)$, indicating that our method can upsample a point cloud to a maximum of 256 times. The time step t is embedded dimension $= 512$ via MLPs [5].

Encoder. In the Encoder, the SA first uses iterative farthest point sampling (FPS) to subsample the input points $p_e^l \in \mathbb{R}^{N^l \times 3}$ and the feature $f_e^l \in \mathbb{R}^{N^l \times C_e^l}$ into $p_e^l \in \mathbb{R}^{N^{l+1} \times 3}$ and $f_e^l \in \mathbb{R}^{N^{l+1} \times C_e^l}$ at level $l+1$ ($N^l > N^{l+1}$). Subsequently, we locate K nearest neighbors in p_e^l, and aggregate the feature f_e^l and neighbors into $g_{in} \in \mathbb{R}^{N^{l+1} \times C_e^l \times K}$. Next, we further extract the neighborhood features by transforming g_{in} into $g_{out} \in \mathbb{R}^{N^{l+1} \times C_e^{l+1} \times K}$ through MLPs. Simultaneously, to preserve more details, we use the residual connection to aggregate $MLP(g_{in})$ and g_{out}. Finally, unlike PointNet++ [11] using the max-pooling layer to filter features, we consider using the self-attention layer to retain more fine-grained information [9, 16], $f_e^{l+1} \in \mathbb{R}^{N^{l+1} \times C_e^{l+1}}$.

Decoder. In the Decoder, the FP is similar to the SA, while the FP transforms the input feature $f_d^l \in \mathbb{R}^{N^l \times C_d^l}$ into $f_d^l \in \mathbb{R}^{N^{l+1} \times C_d^l \times K}$ through upsampling ($N_l < N_{l+1}$). Subsequently, we feed f_d^l into a self-attention layer to obtain $f_d^l \in \mathbb{R}^{N^{l+1} \times C_d^l}$. Simultaneously, to propagate features, we aggregate f_d^l with the points p_d^l and the features f_e^l from the SA at the same level. Finally, we transform f_d^l into $f_d^{l+1} \in \mathbb{R}^{N^{l+1} \times C_d^{l+1}}$ through MLPs.

4. Formula Derivation of DDPM for PCU

In this section, we provide the theoretical foundation for the application of DDPM in PCU. Due to the page limitations, our derivation process focuses more on the overall logic, overlooking some details.

The forward process. The forward process q is modeled as a Markov chain, while the each step follows an independent Gaussian distribution. This gradually adds noise to x until x degrades to z. The process is irrelevant of the condition c (i.e. the sparse point cloud). Formally, given a time step $t \sim \mathcal{U}(T)$ and the dense point cloud $x_0 \sim P_{data}$, we can compute the forward process by the conditional distribution $q(x_{1:T}|x_0)$:

$$
\begin{aligned}
q(x_{1:T}|x_0) &= \frac{q(x_{0:T})}{q(x_0)} \\
&= \frac{q(x_T|x_{0:T-1})q(x_{0:T-1})}{q(x_0)} \\
&\textit{Markov Property}: \\
&= \frac{q(x_T|x_{T-1})q(x_{T-1}|x_{0:T-2})q(x_{0:T-2})}{q(x_0)} \\
&= \frac{q(x_T|x_{T-1})q(x_{T-1}|x_{T-2})...q(x_1|x_0)\cancel{q(x_0)}}{\cancel{q(x_0)}} \\
&= \prod_{t=1}^{T} q(x_t|x_{t-1})
\end{aligned}
\tag{1}
$$

where $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$. β_t is a predefined and increasing variance term ($\beta_t \in [0.0001, 0.02]$ in this paper).

Meanwhile, to enable the sampling-differentiable training, we utilize the reparameterization trick[5] : $x_t = \mu + \sigma\epsilon_t$, $\epsilon_t \sim \mathcal{N}(\epsilon_t; 0, I)$. Next, we can obtain a more simplified formulation of computing x_t by setting $\alpha_t = 1 - \beta_t$, and $\overline{\alpha}_t = \prod_{t=1}^{T} \alpha_t$:

$$
\begin{aligned}
x_t &= \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t \\
&= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_t \\
&= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_{t-1}) + \sqrt{1-\alpha_t}\epsilon_t \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\epsilon_{t-1} + \sqrt{1-\alpha_t}\epsilon_t \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon \\
&...
\end{aligned}
$$

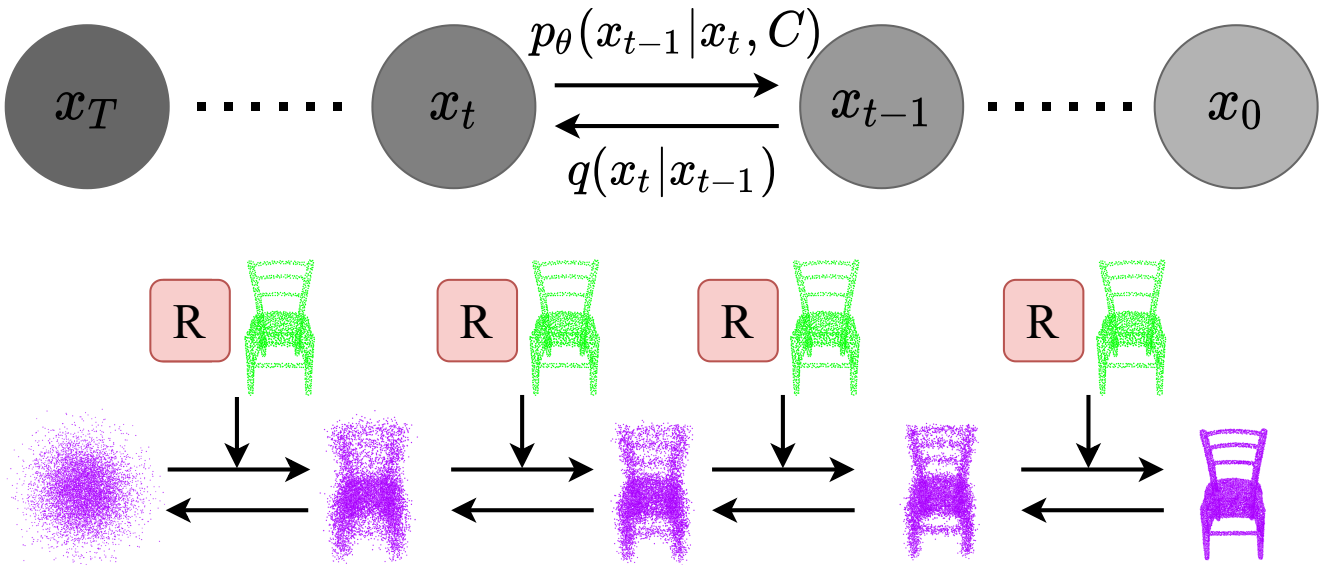

Figure 2. Visualization of the forward process and the reverse process of PUDM. For the forward process, the dense point cloud x_0 is gradually added noise according to $q(x_t|x_{t-1})$, until x_0 degrades to x_T. Simultaneously, for the reverse process, x_T is slowly removed noise according to $p_\theta(x_{t-1}|x_t, C)$, until x_T recovers to x_0. We consider adding the conditions at each time step in the reverse process, the sparse point cloud c and the rate prompt R, to control the generation of the dense point cloud.

$$= \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon \tag{2}$$

where ϵ represents the combination of multiple Gaussian noise terms.

Therefore, x_t is only related to the dense point cloud x_0 and the time step t in the forward process.

The reverse process. Similarly, the reverse process p is also modeled as a Markov chain, while the each step is assumed to follow an independent Gaussian distribution. This slowly removes noise from z until z recovers to x. Formally, given a set of conditions $C = \{c_i|i = 1..S\}$ ("S" means the number of conditions), we can compute the reverse process by the joint distribution $p_\theta(x_{0:T}, C)$:

$$\begin{aligned}
&p_\theta(x_{0:T}, C) = p_\theta(x_0|x_{1:T}, C)p_\theta(x_{1:T}, C) \\
&\textit{Markov Property}: \\
&\quad = p_\theta(x_0|x_1, C)p_\theta(x_1|x_{2:T}, C)p_\theta(x_{2:T}, C) \\
&\quad = p_\theta(x_0|x_1, C)...p_\theta(x_T|x_{T-1}, C)p(x_T, C) \\
&\quad = p(x_T, C)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t, C) \\
&x_T \sim \mathcal{N}(x_T; 0, I) \\
&\quad = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t, C)
\end{aligned} \tag{3}$$

When $C = \emptyset$, Eq 3 transforms into the reverse process of standard DDPM (i.e. unconditional DDPM).

Then, as DDPM is modeled to be reversible, we can directly compute the posterior distribution $q(x_{t-1}|x_t, x_0)$ in the forward process:

$$\begin{aligned}
q(x_t|x_{t-1}, x_0) &= \frac{q(x_{t-1}|x_t, x_0)q(x_t, x_0)}{q(x_{t-1}, x_0)} \\
q(x_{t-1}|x_t, x_0) &= \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)} \\
&\textit{Markov Property}: \\
&= \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}
\end{aligned} \tag{4}$$

where $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I)$, $q(x_{t-1}|x_0) = \mathcal{N}(x_{t-1}; \sqrt{\overline{\alpha}_{t-1}}x_0, (1-\overline{\alpha}_{t-1})I)$, and $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)I)$. In fact, each time step in the reverse process aims to gradually fit the posterior distribution under corresponding time step in the forward process (the posterior distribution represents the inverse process of the forward process, not the reverse process of DDPM), i.e. $p_\theta(x_{t-1}|x_t, C) \approx q(x_{t-1}|x_t, x_0)$. Therefore, deriving $p_\theta(x_{t-1}|x_t, C)$ is equated to deriving $q(x_{t-1}|x_t, x_0)$.

Subsequently, by substituting $q(x_t|x_{t-1})$, $q(x_{t-1}|x_0)$ and $q(x_t|x_0)$ into Eq 4, we obtain the mean $\widetilde{\mu}_t$ and the variance $\widetilde{\sigma}_t$ of $q(x_t|x_{t-1})$:

$$\widetilde{\mu}_t = \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}(1-\alpha_t)}{1-\overline{\alpha}_t}x_0$$
$$\widetilde{\sigma}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}(1-\alpha_t) \tag{5}$$

We can clearly realize that $\widetilde{\sigma}_t$ is a constant.

Next, we substitute $x_0 = \frac{x_t - \sqrt{1-\overline{\alpha}_t}\epsilon}{\sqrt{\overline{\alpha}_t}}$ into $\widetilde{\mu}_t$ to the new expression of the mean $\widetilde{\mu}_t$:

$$\widetilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon) \tag{6}$$

Therefore, for the posterior distribution $q(x_{t-1}|x_t, x_0)$, we can compute x_{t-1} solely by providing x_t and ϵ (where x_0 must be considered as prior knowledge in forward process).

Although according to Eq 5 and Eq 6, the posterior distribution $q(x_{t-1}|x_t, x_0)$ is known, we can not directly utilize it to deriving x_0 due to involving $q(x_t|x_0)$, which requires obtaining x_t and ϵ from the forward process.

Typically, the network $f(x_t, t)$ fits ϵ during the training process of DDPM, as x_t is known in the reverse process ($x_t \sim \mathcal{N}(x_t; 0, I)$). Simultaneously, to introduce the condition set C during in the reverse process (Fig 2), the network increases additionally inputs, i.e. $f(x_t, t, C)$. In PUDM, the condition set $C = \{c, r\}$ represent the sparse point cloud and the rate prompt between the sparse point cloud and the dense point cloud.

Training objective under specific conditions. The training objective of DDPM under specific conditions is to maximize a **E**vidence **L**ower **BO**und (ELBO), due to directly optimize the log-likelihood $\log p_\theta(x_0, C)$ is intractable.

We directly convert the log-likelihood $\log p_\theta(x_0, C)$ into a loss form $-\log p_\theta(x_0, C)$. We first add a KL divergence item $D_{kl}[q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0, C)]$ to $-\log p_\theta(x_0, C)$:

$$\begin{aligned}
&-\log p_\theta(x_0, C) \\
&\leq -\log p_\theta(x_0, C) + D_{kl}[q(x_{1:T}|x_0)||(p_\theta(x_{1:T}|x_0, C))] \\
&\leq -\log p_\theta(x_0, C) + \\
&\quad \int q(x_{1:T}|x_0) \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0, C)} dx_{1:T} \\
&\leq -\log p_\theta(x_0, C) + \\
&\quad \int q(x_{1:T}|x_0) \log \frac{q(x_{1:T}|x_0)}{\frac{p_\theta(x_{1:T}, x_0, C)}{p_\theta(x0, C)}} dx_{1:T} \\
&\leq -\log p_\theta(x_0, C) + \\
&\quad \int q(x_{1:T}|x_0)(\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T}, C)} + \log p_\theta(x_0, C)) dx_{1:T} \\
&\leq -\log p_\theta(x_0, C) + \\
&\quad \mathbb{E}_{q(x_{1:T}|x_0)} \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T}, C)} + \log p_\theta(x_0, C) \\
&\leq \mathbb{E}_{q(x_{1:T}|x_0)} \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T}, C)}
\end{aligned}$$

Adding $\mathbb{E}_{q(x_0)}$ to the both sides :

$$-\mathbb{E}_{q(x_0)} \log p_\theta(x_0, C) \leq \mathbb{E}_{q(x_{0:T})} \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}, C)} \tag{7}$$

Then, leveraging the Markov property and Bayes' theorem, we can obtain the loss form L_{ELBO} of the ELBO:

$$\begin{aligned}
L_{ELBO} &= \mathbb{E}_q \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}, C)} \\
&= \underbrace{\mathbb{E}_q[D_{KL}(q(x_T|x_0)||p(x_T))}_{①} \\
&\quad + \underbrace{\sum_{t=2\in T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t, C))}_{②} \\
&\quad \underbrace{- \log p_\theta(x_0|x_1, C)]}_{③}
\end{aligned} \tag{8}$$

Subsequently, we can eliminate the constant term ①, and combine ② and ③ (where ② = ③, when $t = 1$) to obtain a more simplified expression:

$$L_{ELBO} = \sum_{t=1\in T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t, C)) \tag{9}$$

In order to approximate $p_\theta(x_{t-1}|x_t, C)$ to $q(x_{t-1}|x_t, x_0)$, we represent the mean $\mu_\theta(x_t, t, C)$ through a neural network with parameter θ. Meanwhile, we can further expand L_{ELBO} to obtain a more simplified form:

$$\begin{aligned}
&q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon), \widetilde{\sigma}_t I) \\
&p_\theta(x_{t-1}|x_t, C) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t, C), \widetilde{\sigma}_t I) \\
&\textit{where } \widetilde{\sigma}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}(1-\alpha_t) \\
&L_{ELBO} \\
&= \mathbb{E}_{q(x_{0:T})}(\frac{1}{2\widetilde{\sigma}_t^2}||\widetilde{\mu}_t - \mu_\theta(x_t, t, C)||^2)
\end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}_{q(x_{0:T}),\epsilon}\Big(\frac{1}{2\widetilde{\sigma}_t^2}||\Big(\frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon\Big) - \\
&\mu_\theta(x_t, t, C)||^2\Big) \\
&= \mathbb{E}_{q(x_{0:T}),\epsilon}\Big(\frac{(1-\alpha_t)^2}{2\widetilde{\sigma}_t^2\alpha_t(1-\overline{\alpha}_t)}||\Big(\frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon\Big)\Big) - \\
&\Big(\frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t, t, C)\Big)\Big)||^2\Big) \\
&= \mathbb{E}_{q(x_{0:T}),\epsilon}||\epsilon - \epsilon_\theta(x_t, t, C)||^2
\end{aligned}
\tag{10}
$$

Next, given $t \sim \mathcal{U}(T)$ and $\epsilon \sim \mathcal{N}(0, I)$, we can obtain the training objective $L(\theta)$ for DDPM under specified conditions:

$$
\begin{aligned}
&L(\theta) = \\
&\mathbb{E}_{t\sim U(T),\epsilon\sim\mathcal{N}(0,I)}||\epsilon - \epsilon_\theta(\sqrt{1-\overline{\alpha}_t}\epsilon + \sqrt{\overline{\alpha}_t}x_0, t, C)||^2
\end{aligned}
\tag{11}
$$

Similar to the reverse process, when $C = \emptyset$, $L(\theta)$ means the training objective of the standard DDPM.

We provide a general derivation process concerning both conditional and unconditional DDPM. Therefore, this is not only applicable to PUDM but also to other tasks employing DDPM.

5. More Visualization

We display additional visual results of upsampled point clouds in Fig 3, Fig 4 and Fig 5.

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 3

[2] Wanquan Feng, Jin Li, Hongrui Cai, Xiaonan Luo, and Juyong Zhang. Neural points: Point cloud representation with neural fields for arbitrary upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18633–18642, 2022. 2, 3

[3] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. pages 1231–1237. Sage Publications Sage UK: London, England, 2013. 8, 9

[4] Yun He, Danhang Tang, Yinda Zhang, Xiangyang Xue, and Yanwei Fu. Grad-pu: Arbitrary-scale point cloud upsampling via gradient descent with learned distance functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2023. 2, 3

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3

[6] 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, 7

[7] 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. 2, 3

[8] 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. 2, 3

[9] Liang Pan, Xinyi Chen, Zhongang Cai, Junzhe Zhang, Haiyu Zhao, Shuai Yi, and Ziwei Liu. Variational relational point completion network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8524–8533, 2021. 3

[10] 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. 3

[11] 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. 3

[12] 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. 2, 3

[13] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 1

[14] 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, 3

[15] 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. 2, 3

[16] 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. 3

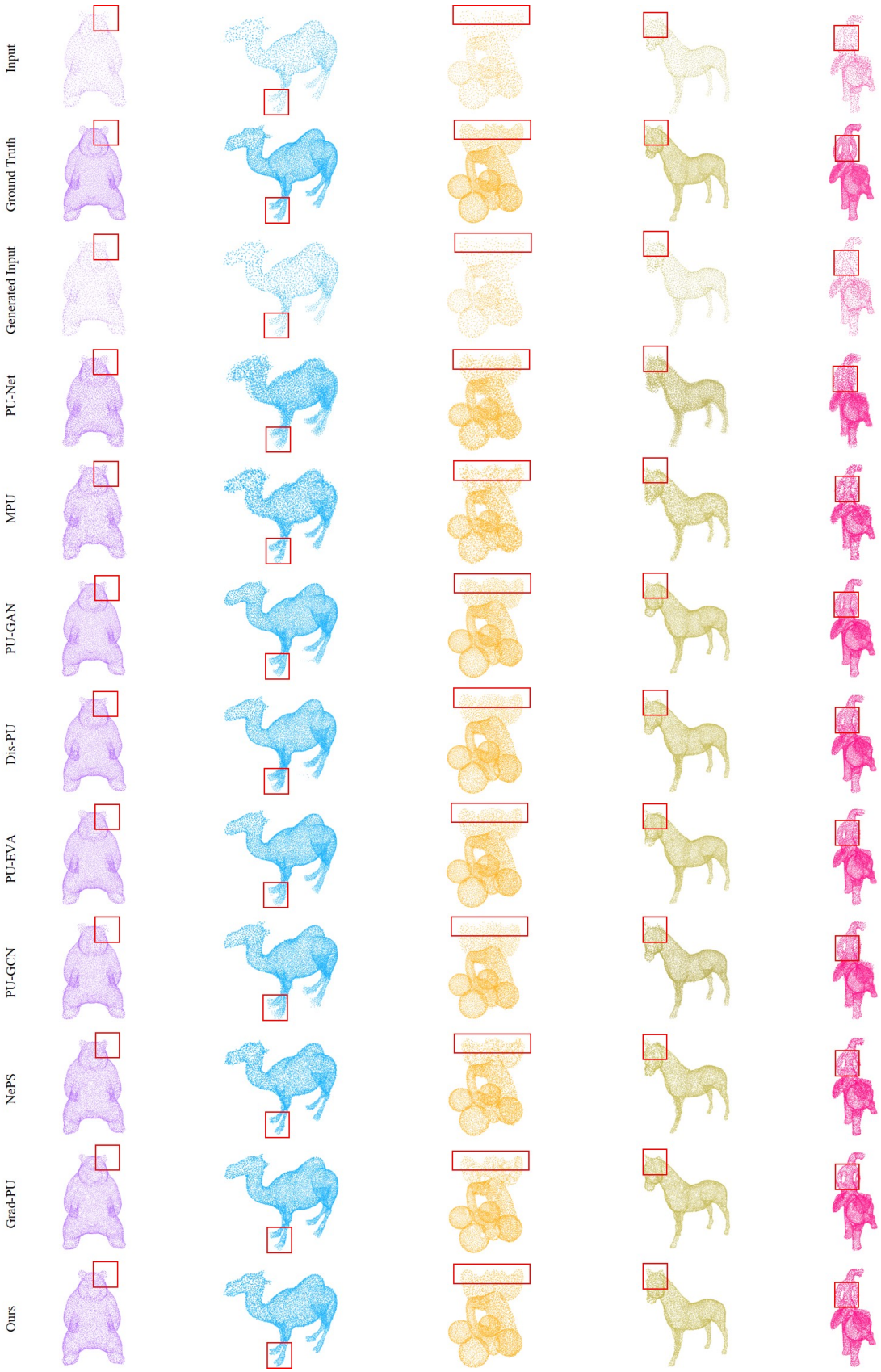

Figure 3. Visualization of point cloud upsampling at $4\times$ on PUGAN [6].

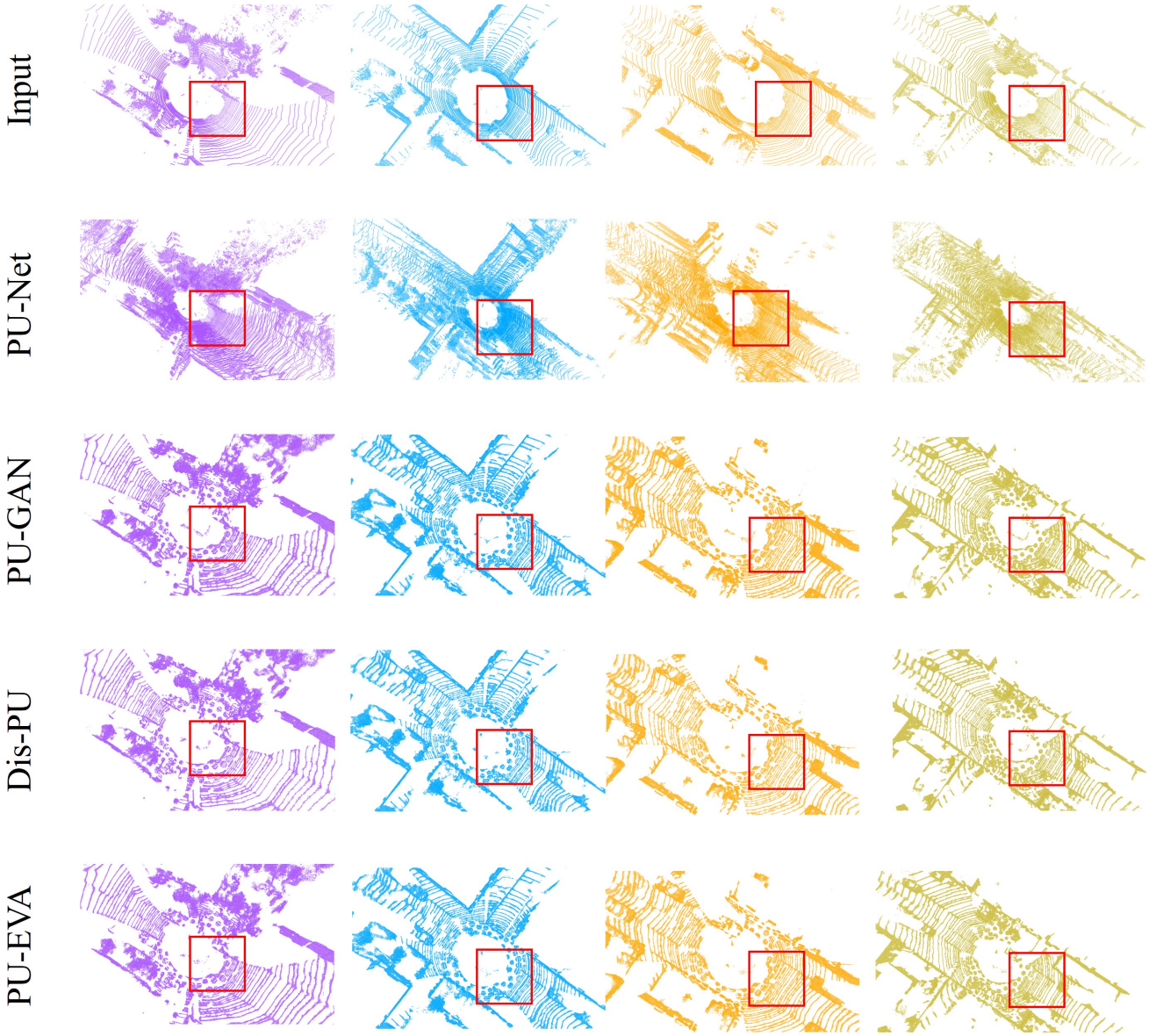

Figure 4. Visualization of point cloud upsampling at $4\times$ on KITTI [3].

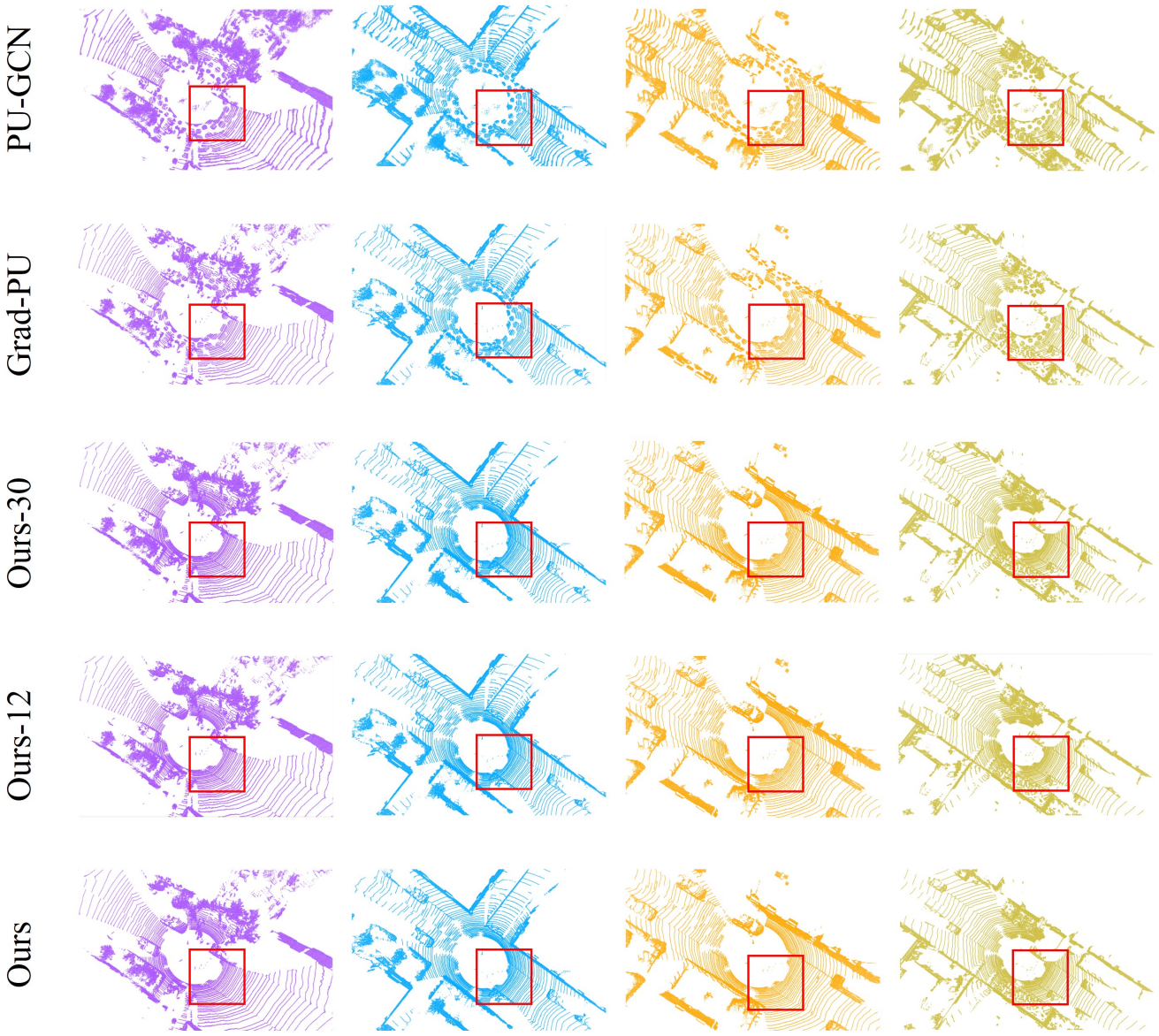

Figure 5. Visualization of point cloud upsampling at $4\times$ on KITTI [3]. "Ours-12" and "Ours-30" represent the results of our method at sampling intervals $= 12$ and $= 30$, respectively.