An End-to-End Robust Point Cloud Semantic Segmentation Network with Single-Step Conditional Diffusion Models Supplementary Material

Wentao Qu[1], Jing Wang[2], YongShun Gong[3], Xiaoshui Huang[4*], Liang Xiao[1*]
NJUST[1], THU[2], SDU[3], SJTU[4]
{quwentao, jwang}@njust.edu.cn, huangxiaoshui@163.com, xiaoliang@mail.njust.edu.cn

Due to the space limitation of the main text, we include additional experiments, derivations, implementations, and discussions in the supplementary material. We first conduct additional ablation (Sec. 1) and comparison (Sec. 2) experiments. Then, we logically derive DDPMs from a modeling perspective, explaining some issues about applying DDPMs to 3D tasks (Sec. 3). Next, the implementation details (Sec. 4) and optimization process (Sec. 5) of our method are presented. Finally, we discuss the limitations of CNF (Sec. 6) and visualize additional results (Sec. 7).

1. Additional Ablation Study

1.1. Selection of FFM

Since excessive noise perturbations from the Noise Network (NN) may harm the performance of the Conditional Network (CN), the aim of the Feature Fusion Module (FFM) is to adaptively filter the noise information, making the feature augmentation in a reasonable way. To better achieve the aim, we consider several ways of FFM: 1) Channel Mapping (CM) [13]. This preserves the channel information of features from CN and NN, but lacks the effective information filtering in the feature space. 2) Channel Cross-attention (CCA) [25]. This filters information along the channel dimension, but compresses the search space, making filtering out effective information difficult at the point level. 3) Spatial Cross-Attention (SCA) [23]. This precisely searches for similar elements in the spatial dimension, but the quadratic complexity for the input points. Fortunately, the input point number at the bottleneck stage of the U-Net is usually less than a thousand.

Tab. 1 exhibits the results on ScanNet. Benefiting from the effective filtering for perturbations, spatial cross-attention significantly outperforms other two ways.

*Corresponding Author. https://github.com/QWTforGithub/CDSegNet

1.2. Inference Modes

Although CDSegNet can be considered a non-DDPM during inference, CDSegNet can still follow the iterative inference approach of DDPMs (the output is still dominated by CN). Therefore, this can be divided into three inference modes: 1) Single-Step Inference (SSI), semantic labels are generated by CN through a single-step iteration in NN. 2) Multi-Step Average Inference (MSAI), MSAI conducts T step iterations in NN and averages T outputs produced by CN. 3) Multi-Step Final Inference (MSFI), MSFI is determined by the output from the final iteration of CN.

Tab. 2 shows that the difference in performance among SSI, MSAI, and MSFI is negligible. Thanks to CNF, the impact of iterations is no longer significant for the results. Meanwhile, since the noise system of DDPMs is retained during training, CDSegNet still maintains the robustness to data noise and sparsity.

Methods	Performance			Training Latency	Inference Latency
	mIoU	mAcc	allAcc		
CM [13]	77.4	84.8	91.9	**268ms**	**101ms**
CCA [25]	77.5	85.0	92.3	272ms	107ms
SCA [23]	**77.9**	**85.2**	**92.2**	278ms	112ms

Table 1. Ablation study of FFM on ScanNet. The spatial cross-attention exhibits the most optimal selection.

Methods	mIoU	mAcc	allAcc
MSAI-100	**77.9**	**85.3**	92.1
MSAI-50	77.8	85.1	91.9
MSAI-20	77.8	85.1	91.9
MSFI-100	77.8	85.2	**92.3**
MSFI-50	77.7	85.1	92.1
MSFI-20	77.7	85.1	92.1
SSI	**77.9**	85.2	92.2

Table 2. Ablation study of different inference modes on ScanNet. SSI demonstrates a better trade-off between performance and efficiency.

1.3. Input for NN

According to Sec. 4.1 of the main text, NN is modeled as a noise-feature generator to enhance the semantic features

Input type	mIoU	mAcc	allAcc
Semantic Label	77.7	85.1	**92.2**
Point Coordinate	77.6	85.0	92.1
Color+Normal	**77.9**	**85.2**	**92.2**

Table 3. Ablation study of input for NN on ScanNet. Color+Normal, consistent with the input of CN, demonstrates the best performance.

in CN. Nevertheless, we can still input semantic labels or point coordinates into NN for the diffusion modeling (the color and the normal as the inputs of CN).

Tab. 3 shows that using the color and the normal as the inputs of NN, consistent with the input of CN, achieves the best results. This demonstrates that the feature perturbations from NN can effectively enhance the semantic features in CN.

Methods	Performance			Robustness		
	mIoU	mAcc	allAcc	τ=0.1	τ=0.5	τ=1.0
ScanNet [6]						
PTv3 [24]	77.6	**85.0**	**92.0**	**77.5**	45.8	12.9
PTv3 + CNF	**77.7**	84.8	91.7	77.4	**60.1**	**33.2**
ScanNet200 [18]						
PTv3 [24]	35.3	**46.0**	83.3	34.0	10.2	1.0
PTv3 + CNF	**35.9**	45.3	**83.4**	**35.5**	**21.2**	**7.1**
nuScenes [3]						
PTv3 [24]	80.3	87.2	94.6	63.9	1.1	1.1
PTv3 + CNF	**81.0**	**87.9**	**94.8**	**67.8**	**1.3**	**1.3**

Table 4. The results of introducing CNF to PTv3. PTv3+CNF shows a significant improvement in noise robustness.

2. Additional Comparative Experiments

2.1. Generalization for CNF on Indoor Benchmark

We also conduct the generalization experiments of CNF on indoor benchmarks (ScanNet [6], ScanNet200 [18]). Following the same setup as in Sec. 5.5 of the main text, we simply consider PTv3 as CN and add NN and FFM of CDSegNet to PTv3.

Tab. 4 shows this result. By introducing CNF, PTv3 has significantly improved noise robustness. Simultaneously, we can see that although the noise robustness has increased significantly on ScanNet, the performance has only slightly improved. This is because ScanNet represents a purer and denser scene compared to ScanNet200 and nuScenes. This verifies that CNF is more suitable for disturbed and sparse scenes but is slightly inferior in relatively pure scenes.

2.2. Comparison on Test Set

We further provide the results on the test sets of ScanNet, ScanNet200 and nuScenes. Sincerely and honestly, all models are uniformly trained on the training set and validated on the validation set. The all comparison methods come from the official released checkpoints. Due to the limitations of the submission number and the checkpoint releases, we provided only a small set of results (**all results on the test set are from previous checkpoints**).

On ScanNet and ScanNet200 test set. Tab. 5 shows the results of our method compared with PonderV2 [26] and PTv3 [24] on the test sets of ScanNet and ScanNet200. We can clearly observe that, compared to the results on the validation set, our method performs better on the test set. This demonstrates that reasonable noise perturbations can effectively enhance the generalization ability of models.

On nuScenes test set. As shown in Tab. 6, CDSegNet significantly outperforms PTv3 on both the validation and test sets. Notably, we found that PTv3 with CNF introduced achieves a significant improvement on the test set, even surpassing CDSegNet. This further demonstrates that CNF enables models to inherit the sparse robustness from DDPMs, enhancing the generalization ability of models in sparse scenes (we sincerely reaffirm that we only trained on the training set and validated on the validation set. Meanwhile, all checkpoints are downloaded directly from the official website).

Methods	Val (mIoU)	Test (mIoU)	Only Training Set?	URL
ScanNet [6]				
PonderV2 [26]	77.0	73.9	no	Here
https://github.com/OpenGVLab/PonderV2/blob/main/docs/model_zoo.md				
PTv3 [24]	77.6	73.6	yes	Here
https://huggingface.co/Pointcept/PointTransformerV3/tree/main/scannet-semseg-pt-v3m1-0-base/model				
Ours	**77.9**	**74.5**	yes	-
ScanNet200 [18]				
PTv3 [24]	35.3	33.2	yes	Here
https://huggingface.co/Pointcept/PointTransformerV3/tree/main/scannet200-semseg-pt-v3m1-0-base/model				
PTv3+CNF	35.5	33.7	yes	-
Ours	**36.0**	**34.1**	yes	-

Table 5. The results on the ScanNet and ScanNet200 test set. Our method demonstrates better performance on the validation set and the test set.

Methods	Val (mIoU)	Test (mIoU)	Only Training Set?	URL
PTv3 [24]	80.3	81.2	yes	Here
https://huggingface.co/Pointcept/PointTransformerV3/tree/main/nuscenes-semseg-pt-v3m1-0-base/model				
PTv3+CNF	80.8	**82.8**	yes	-
Ours	**81.2**	82.0	yes	-

Table 6. The results on the nuScenes test set. PTv3+CNF demonstrates a significant improvement compared with PTv3 on the test set.

Methods	Performance			Robustness (mAcc)	
	mAcc	CAmIoU	IAmIoU	τ=0.5	25%
PointNet [15]					
Without CNF	93.2	77.9	83.2	37.3	70.6
With CNF	**94.1**	**78.5**	**83.9**	**42.2**	**73.1**
PointNet++ [16]					
Without CNF	94.2	82.7	85.1	39.0	71.1
With CNF	**95.1**	**83.5**	**86.0**	**44.6**	**73.9**

Table 7. The instance segmentation results on ShapeNet [4]. With the introduction of CNF, the performance and robustness of PointNet and PointNet++ have been improved significantly.

Other 3D tasks. We also introduce CNF into instance segmentation task. Similar to the classification task in the main text, PointNet [15] and PointNet++ [16] are selected as backbones. Tab. 7 show that by introducing CNF, they exhibit significantly improve the performance and robustness.

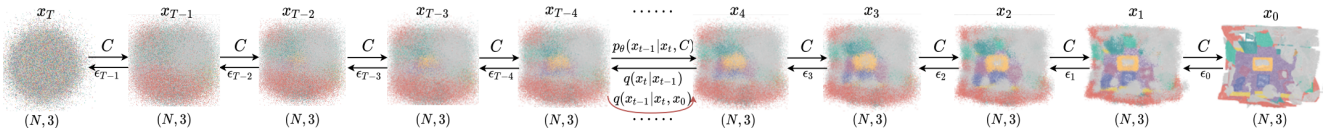

Figure 1. The visualization of the predefined diffusion process $q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})$, the inverse of the diffusion process $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$, and the trainable conditional generation process $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, C)$. In the diffusion process $q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})$, the task target $\boldsymbol{x_0}$ is gradually noised until $\boldsymbol{x_0}$ degrades to $\boldsymbol{z}$ ($\boldsymbol{x_T}$). Meanwhile, the inverse of the diffusion process (the true Ground Truth in DDPMs) $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$ can be calculated by the predefined distribution in the diffusion process. Furthermore, the generation process $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, C)$ gradually fits the inverse of the diffusion process $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$ until $\boldsymbol{z}$ ($\boldsymbol{x_T}$) is restored to $\boldsymbol{x_0}$ conditioned on $C = \{\boldsymbol{c}, t\}$ (unconditional generation, $\boldsymbol{c} = \emptyset$, in the formula derivation of DDPMs, the time label t is usually ignored.).

3. Formula Derivation of DDPMs for 3D Tasks

In this section, we provide the theoretical support for applying DDPMs to most existing 3D vision tasks. This focuses on the logical modeling process of DDPMs, omitting some derivation details that we consider unnecessary. For example, to better understand the modeling process of DDPMs intuitively, we believe that the derivation of the **E**vidence **L**ower **BO**und (ELBO) can be skipped. Interested readers can refer to [17] for the derivation of the ELBO under specific conditions.

For a 3D task, given a data sample pair $(\boldsymbol{c}, \boldsymbol{x_0})$, we aim to train a generalized model f_θ that takes $\boldsymbol{c}$ as the input and produces an output $\boldsymbol{x_0'}$ approximating $\boldsymbol{x_0}$. We can transform the task into a conditional generation problem using DDPMs. This performs an auto-regressive process [17, 21]: a predefined diffusion process q ($\boldsymbol{x_0} \rightarrow \boldsymbol{z}$) and a trainable conditional generation process p_θ ($\boldsymbol{z} \rightarrow \boldsymbol{x_0}$) under the guidance of the condition $\boldsymbol{c}$ (see Fig. 1). Here, $\boldsymbol{z}$ represents an implicit variable sampled from a predefined prior distribution P_{noise} in DDPMs.

3.1. Diffusion Process

The modeled distribution and noise-adding pattern. The diffusion process q is more critical in DDPMs, as this defines the type of DDPMs. For example, [8] can be referred to as Gaussian or continuous DDPMs. Similarly, we can also use the categorical distribution to model the diffusion process, referred to as categorical or discrete DDPMs [1]. Theoretically, any distribution can be used to model the diffusion process, such as the Laplace distribution or the Poisson distribution. Meanwhile, the noise addition pattern is not limited to element-wise addition and multiplication. This can also involve snowification and masking [2].

We derive this diffusion process based on [8], which involves adding perturbations using a Gaussian distribution through element-wise addition and multiplication.

The detailed derivation. The diffusion process q is structured as a Markov chain, with each step governed by an independent Gaussian distribution. This gradually destroys the essential information until $\boldsymbol{x_0}$ degrades to $\boldsymbol{z}$. Meanwhile, this process is only related to the Ground Truth (task target) $\boldsymbol{x_0}$ and is independent of the condition (task input) $\boldsymbol{c}$. Formally, given a time label $t \sim \mathcal{U}(T)$ that controls the noise level, the diffusion process can be computed by multiple conditional distributions $q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0})$:

$$\begin{aligned} q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}) &= \frac{q(\boldsymbol{x_{0:T}})}{q(\boldsymbol{x_0})} \\ &= \frac{q(\boldsymbol{x_T}|\boldsymbol{x_{0:T-1}})q(\boldsymbol{x_{0:T-1}})}{q(\boldsymbol{x_0})}. \end{aligned} \tag{1}$$

Meanwhile, according to the Markov property, the current term x_T is determined only by the previous term $\boldsymbol{x_{T-1}}$:

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

where $q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}) = \mathcal{N}(\boldsymbol{x_t}; \sqrt{1-\beta_t}\boldsymbol{x_{t-1}}, \beta_t \boldsymbol{I})$. β_t is a predefined and increasing variance factor.

The intuitive explanation. We can intuitively understand this diffusion process. According to Fig. 1, $\boldsymbol{x_1}$ is determined by $\boldsymbol{x_0}$, $\boldsymbol{x_1} \sim q(\boldsymbol{x_1}|\boldsymbol{x_0})$. Similarly, $\boldsymbol{x_2}$ is determined by both $\boldsymbol{x_1}$ and $\boldsymbol{x_0}$, $\boldsymbol{x_2} \sim q(\boldsymbol{x_2}|\boldsymbol{x_1}, \boldsymbol{x_0})$. In this way, $\boldsymbol{x_T} \sim q(\boldsymbol{x_T}|\boldsymbol{x_{0:T-1}})$. Due to the independent and identically distributed (i.i.d.) and Markov properties, this process can be described as:

$$\begin{aligned} &\textit{i.i.d.}: \\ &\quad q(\boldsymbol{x_1}|\boldsymbol{x_0})q(\boldsymbol{x_2}|\boldsymbol{x_1}, \boldsymbol{x_0})...q(\boldsymbol{x_T}|\boldsymbol{x_{0:T-1}}) \\ &\textit{Markov}: \\ &= q(\boldsymbol{x_1}|\boldsymbol{x_0})q(\boldsymbol{x_2}|\boldsymbol{x_1})...q(\boldsymbol{x_T}|\boldsymbol{x_{T-1}}) \\ &= \prod_{t=1}^{T} q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}). \end{aligned} \tag{3}$$

The sampling differentiable. Furthermore, to make the sampling process differentiable and ensure the sampling results under a specific distribution, a reparameterization trick is applied [8]: $\boldsymbol{x_t} = \boldsymbol{\mu_t} + \boldsymbol{\sigma_t}\boldsymbol{\epsilon_{t-1}}, \boldsymbol{\epsilon_{t-1}} \sim \mathcal{N}(\boldsymbol{\epsilon_{t-1}}; \mathbf{0}, \boldsymbol{I})$ (this can be verified by the properties of random variable that $\boldsymbol{x_t}$ follows a Gaussian distribution with mean $\boldsymbol{\mu_t}$ and variance $\sigma_t^2\boldsymbol{I}$). Next, we can further simplify to compute $\boldsymbol{x_t}$ by setting $\alpha_t = 1 - \beta_t$, and $\overline{\alpha}_t = \prod_{t=1}^{T} \alpha_t$:

$$
\begin{aligned}
\boldsymbol{x_t} &= \sqrt{1-\beta_t}\boldsymbol{x_{t-1}} + \sqrt{\beta_t}\boldsymbol{\epsilon_{t-1}} \\
&= \sqrt{\alpha_t}\boldsymbol{x_{t-1}} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon_{t-1}} \\
&= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}\boldsymbol{x_{t-2}} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon_{t-1}}) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon_t} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x_{t-2}} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon_{t-1}} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon_t} \\
&\textit{Gaussian Variable Additivity :} \\
&\Rightarrow \epsilon \sim \mathcal{N}(0, (a_t - a_t a_{t-1}) + (1 - a_t)) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x_{t-2}} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} \\
&\cdots \\
&= \sqrt{\overline{\alpha}_t}\boldsymbol{x_0} + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}.
\end{aligned}
\tag{4}
$$

Therefore, according to Eq. 4, $\boldsymbol{x_t}$ is only related to the task target $\boldsymbol{x_0}$ and the time label t in the diffusion process, while $\boldsymbol{x_t} \sim q(\boldsymbol{x_t}|\boldsymbol{x_0}) = \mathcal{N}(\boldsymbol{x_t}; \sqrt{\overline{\alpha}_t}\boldsymbol{x_0}, (1-\overline{\alpha}_t)\boldsymbol{I})$.

That is, during the diffusion process, we can obtain $q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})$ and $q(\boldsymbol{x_t}|\boldsymbol{x_0})$.

The inverse of the diffusion process. The inverse of the diffusion process $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$ does not imply the generation process of DDPMs, which serves as the true Ground Truth in DDPMs. This inverse process is determined by the diffusion process, defining the true posterior distribution that the generation process is required to fit, i.e., $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, C) \approx q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$ (see Fig. 1). $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$ also indicates that the computation of the inverse process necessarily involves the task target $\boldsymbol{x_0}$. Consistent with the diffusion process, the inverse process also follows i.i.d. and Markov properties. Meanwhile, the inverse process $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$ can directly be calculated by the inverse probability formula (Bayesian formula):

$$
\begin{aligned}
q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}, \boldsymbol{x_0}) &= \frac{q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})q(\boldsymbol{x_t}, \boldsymbol{x_0})}{q(\boldsymbol{x_{t-1}}, \boldsymbol{x_0})} \\
q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}) &= \frac{q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}, \boldsymbol{x_0})q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0})}{q(\boldsymbol{x_t}|\boldsymbol{x_0})} \\
&\textit{Markov :} \\
&= \frac{q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0})}{q(\boldsymbol{x_t}|\boldsymbol{x_0})},
\end{aligned}
\tag{5}
$$

where $q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}) = \mathcal{N}(\boldsymbol{x_t}; \sqrt{\alpha_t}\boldsymbol{x_{t-1}}, (1-\alpha_t)\boldsymbol{I})$, $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0}) = \mathcal{N}(\boldsymbol{x_{t-1}}; \sqrt{\overline{\alpha}_{t-1}}\boldsymbol{x_0}, (1-\overline{\alpha}_{t-1})\boldsymbol{I})$, and $q(\boldsymbol{x_t}|\boldsymbol{x_0}) = \mathcal{N}(\boldsymbol{x_t}; \sqrt{\overline{\alpha}_t}\boldsymbol{x_0}, (1-\overline{\alpha}_t)\boldsymbol{I})$. These distributions are known in the derivation of the diffusion process. Subsequently, by substituting $q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})$, $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0})$ and $q(\boldsymbol{x_t}|\boldsymbol{x_0})$ into Eq 5, the mean $\boldsymbol{\mu_t}$ and the variance $\sigma_t^2\boldsymbol{I}$ of $q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}, \boldsymbol{x_0}) = \mathcal{N}(\boldsymbol{x_t}; \boldsymbol{\mu_t}, \sigma_t^2\boldsymbol{I})$ can be obtain:

$$
\begin{aligned}
\boldsymbol{\mu_t} &= \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\boldsymbol{x_t} + \frac{\sqrt{\overline{\alpha}_{t-1}}(1-\alpha_t)}{1-\overline{\alpha}_t}\boldsymbol{x_0}, \\
\boldsymbol{\sigma_t^2} &= \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}(1-\alpha_t)\boldsymbol{I}.
\end{aligned}
\tag{6}
$$

Meanwhile, in Eq. 6, we can observe that the variance $\sigma_t^2\boldsymbol{I}$ is a constant term (some works also set the variance as a fitting term [7, 14]).

Next, due to the better performance observed in experiment [8], $\boldsymbol{x_0}$ is considered to be replaced by $\boldsymbol{\epsilon}$, i.e., $\boldsymbol{x_0} = \frac{\boldsymbol{x_t} - \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}}{\sqrt{\overline{\alpha}_t}}$:

$$
\boldsymbol{\mu_t} = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x_t} - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}).
\tag{7}
$$

Notably, the initial value $\boldsymbol{x_T}$ of $\boldsymbol{x_t}$ can be directly sampled from a prior distribution P_{noise} (Gaussian distribution, $\boldsymbol{x_T} \sim \mathcal{N}(\boldsymbol{x_T}; \mathbf{0}, \boldsymbol{I})$) during inference. Therefore, the only unknown term is $\boldsymbol{\epsilon}$ in Eq. 7.

3.2. Generation Process

The fitting target. The generation process defines the generation mode of DDPMs: unconditional generation and conditional generation. This takes the inverse of the diffusion process as the fitting target, i.e., $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, C) \approx q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$. To better fit the inverse process, each step of the generation process is characterized by i.i.d. and Markov properties.

We directly derive the fitting target of conditional DDPMs, as unconditional generation ($\boldsymbol{c} = \emptyset$) can be viewed as a special case of conditional DDPMs solely conditioned on the time label t ($C = \{\boldsymbol{c}, t\}$).

The detailed derivation. Formally, given a set of conditions $C = \{\boldsymbol{c_i}, t | i = 1..n\}$ ("n" means the number of conditions, this means that conditional DDPMs can perform the multi-conditional generation), we can compute the reverse process by the joint distribution $p_\theta(\boldsymbol{x_{0:T}}, C)$:

$$
\begin{aligned}
p_\theta(\boldsymbol{x_{0:T}}, C) &= p_\theta(\boldsymbol{x_0}|\boldsymbol{x_{1:T}}, C)p_\theta(\boldsymbol{x_{1:T}}, C) \\
&\textit{Markov :} \\
&= p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1}, C)p_\theta(\boldsymbol{x_1}|\boldsymbol{x_{2:T}}, C)p_\theta(\boldsymbol{x_{2:T}}, C) \\
&= p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1}, C)...p_\theta(\boldsymbol{x_T}|\boldsymbol{x_{T-1}}, C)p(\boldsymbol{x_T}, C) \\
&= p(\boldsymbol{x_T}, C)\prod_{t=1}^{T} p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, C)
\end{aligned}
$$

$$
\begin{aligned}
&\boldsymbol{x_T} \sim \mathcal{N}(\boldsymbol{x_T}; 0, I) \\
&= p(\boldsymbol{x_T}) \prod_{t=1}^{T} p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, C),
\end{aligned} \tag{8}
$$

where $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, C) = \mathcal{N}(\boldsymbol{x_{t-1}}; \mu_\theta(\boldsymbol{x_t}, C), \sigma_t^2 \boldsymbol{I})$, as in Eq. 7, the mean $\boldsymbol{\mu_t}$ contains unknown variables during inference, while the variance factor σ_t^2 is a constant term.

Next, according to Eq. 7, we can logically further refine this fitting target:

$$
\begin{aligned}
&p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, C) \approx q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}), \\
&\Rightarrow \mu_\theta(\boldsymbol{x_t}, C) \approx \boldsymbol{\mu_t}, \\
&\Rightarrow \epsilon_\theta(\boldsymbol{x_t}, C) \approx \boldsymbol{\epsilon}.
\end{aligned} \tag{9}
$$

According to Eq. 9, we can clearly recognize that as long as the generation process can sufficiently fit the inverse of the diffusion process, DDPMs can achieve the multi-condition generation. Meanwhile, due to the inverse process with a total of T steps, the final training objective is:

$$
\begin{aligned}
L(\theta) &= \frac{1}{T} \sum_{t=1}^{T} D_{KL}(q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}) || p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, C)), \\
&= \frac{1}{T} \sum_{t=1}^{T} ||\boldsymbol{\mu_t} - \mu_\theta(\boldsymbol{x_t}, C)||^2, \\
&= \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})} ||\boldsymbol{\epsilon} - \epsilon_\theta(\boldsymbol{x_t}, C)||^2.
\end{aligned} \tag{10}
$$

Thus, we derive Eq. 1 of the main text.

The intuitive explanation. Similarly, we can also understand the generation process in a more intuitive way. The generation process aims to fit the reverse process to achieve the generalization of generation, due to the computation of $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$ involving $\boldsymbol{x_0}$. According to Eq. 5 and Eq. 7, the fitting target of the generation process can conduct two stages of simplification: fitting the distribution $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}) \rightarrow$ fitting the distribution hyperparameter $\boldsymbol{\mu_t} \rightarrow$ fitting the unknown variable $\boldsymbol{\epsilon}$. Meanwhile, due to the inverse process with a total of T steps, thus the fitting objective:

$$
L(\theta) = \frac{1}{T} \sum_{t=1}^{T} ||\boldsymbol{\epsilon} - \epsilon_\theta(\boldsymbol{x_t}, C)||^2. \tag{11}
$$

For a more intuitive description, we express the expectation as a summation in Eq. 10.

3.3. How do we introduce DDPMs to 3D tasks?

According to Sec. 3.1 and Sec. 3.2, we can apply DDPMs to most existing 3D tasks. For example, in the point cloud semantic segmentation task, $\boldsymbol{c}$ means the segmented point cloud, and $\boldsymbol{x_0}$ represents the semantic label (see Fig. 2). In the diffusion process, the semantic label $\boldsymbol{x_0}$ is gradually noised until $\boldsymbol{x_0}$ degenerates into $\boldsymbol{z}$. Meanwhile, in the generation process, the semantic label $\boldsymbol{x_0}$ is gradually reconstructed until $\boldsymbol{z}$ is restored to the desired $\boldsymbol{x_0}$ under the condition of the segmented point cloud $\boldsymbol{c}$. This achieves point cloud semantic segmentation tasks. DDPMs also can be introduced into other 3D tasks in a similar manner.

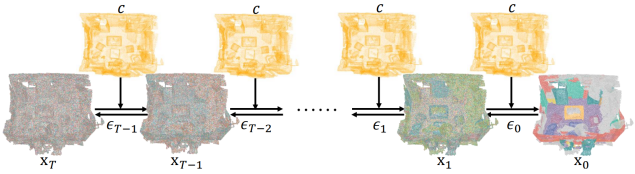

Figure 2. Applying DDPMs to the point cloud semantic segmentation task: the semantic label $\boldsymbol{x_0}$ is gradually perturbed with noise during the diffusion process and slowly reconstructed during the generation process conditioned on the segmented point cloud $\boldsymbol{c}$.

3.4. Why is the performance of DDPMs determined by the noise fitting quality?

According to Eq. 5, the generation process of DDPMs fits the reverse of the diffusion process at all time steps. Meanwhile, in Eq. 7, the noise $\boldsymbol{\epsilon}$ is an unknown in distribution mean $\boldsymbol{\mu_t}$. Therefore, the generation process of DDPMs essentially fits the unknown noise $\boldsymbol{\epsilon}$ (see Eq. 9). This means that the higher the the noise fitting quality, the better the generation result of DDPMs will be.

3.5. Why are DDPMs robust to noise in the modeled distribution?

From the gradient of the data distribution. [17] provides an intuitive explanation from the gradient of data distribution. Under stochastic differential equations (SDEs), the target noise in conditional DDPMs can be converted into and from the score that means the gradient of data distribution by a constant factor $\alpha = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}$ [22]:

$$
\alpha \epsilon_\theta(\boldsymbol{x_t}, C) = s_\theta(\boldsymbol{x_t}, C) \approx \nabla_{\boldsymbol{x_t}} \log P_t(\boldsymbol{x_t}). \tag{12}
$$

This means that the predicted noise guides the transformation between the two distributions (see Fig. 1), i.e., $\boldsymbol{x_t} \xrightarrow{\boldsymbol{\epsilon_{t-1}}} \boldsymbol{x_{t-1}}$, $\boldsymbol{\epsilon_{t-1}} = \epsilon_\theta(\boldsymbol{x_t}, C) \sim P_{noise}(\boldsymbol{\epsilon_{t-1}}|\boldsymbol{x_t}, C)$. Perturbing C inevitably affects the generation quality of $\boldsymbol{\epsilon_{t-1}}$, thus directly affecting the quality of the transformation between $\boldsymbol{x_t}$ and $\boldsymbol{x_{t-1}}$. When the distribution of the perturbation is identical or similar to the distribution of $\boldsymbol{\epsilon_{t-1}}$,

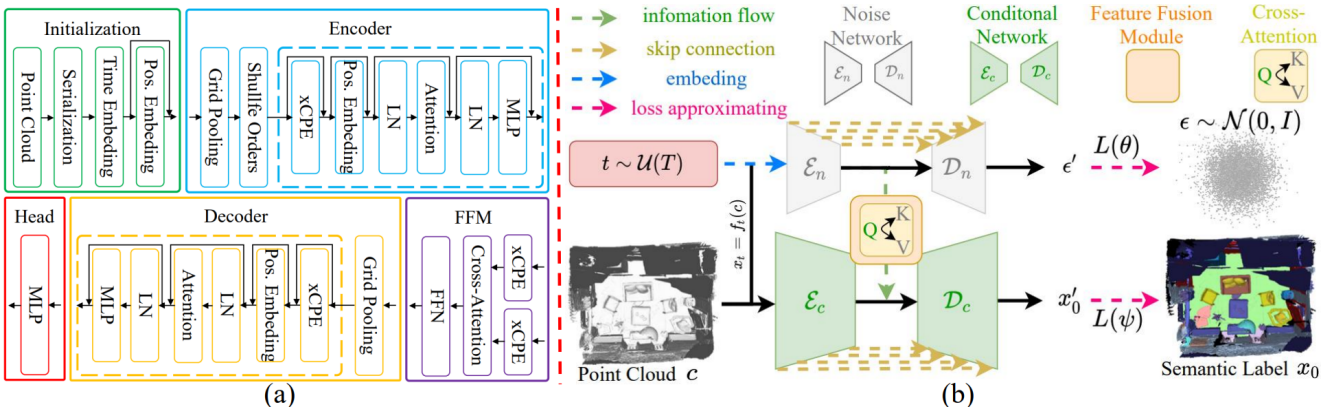

Figure 3. (a) describes the five components that make up CDSegNet: the Initialization Module, the Encoder, the Feature Fusion Module (FFM), the Decoder, and the Head. Meanwhile, (b) shows the overall framework of CDSegNet, where both the Noise Network (NN) and the Condition Network (CN) consist of three parts: the Encoder, the Decoder, and the Head.

the impact on the generation of $\boldsymbol{\epsilon_{t-1}}$ is relatively small; otherwise, it is greater. Therefore, DDPMs are robust to the modeled distribution noise. Meanwhile, the closer the noise distribution is to the modeled distribution, the better the noise robustness; otherwise, this deteriorates.

From the noise samples and the noise fitting. In the main text, we provide a simpler explanation that is consistent with the source. Since DDPMs can see multi-level noise samples $\boldsymbol{x_t}$ and fit the noise target $\boldsymbol{\epsilon}$ from the modeled distribution during training, DDPMs can adapt to related distribution noise during inference compared to non-DDPMs.

3.6. Why DDPMs require more training and inference iterations than non-DDPMs?

This is because DDPMs require fitting more intermediate samples than non-DDPMs. For a sample pair $\{\boldsymbol{c}, \boldsymbol{x_0}\}$, the training object of DDPMs is:

$$L_\theta = \frac{1}{T}\sum_{t=1}^{T} ||\boldsymbol{y_{t-1}} - f_\theta(\boldsymbol{x_t}, \boldsymbol{c})||^2, \tag{13}$$

where f_θ indicates a neural network with sufficient fitting ability. $\boldsymbol{y_{t-1}}$ represents $\boldsymbol{\epsilon}$ at corresponding the time label t in Eq. 11 (we omit the time label t as part of the input). This means that DDPMs require fitting T targets, i.e. $\boldsymbol{x_0} = \{\boldsymbol{y_{t-1}} | t = 1...T\}$, due to the significant error of fitting distributions with a large difference when using a single step or a small number of steps [12, 21, 22].

Meanwhile, this also makes that DDPMs require to iterate T steps to achieve the accurate result during inference:

$$\boldsymbol{y'_{t-1}} = f_\theta(\boldsymbol{x_t}, \boldsymbol{c}), \quad t \in [1, T], \tag{14}$$

where $\boldsymbol{y'_{t-1}}$ means the predicted noise in DDPMs. According to Eq. 14, DDPMs require T steps to converge ($\{\boldsymbol{y'_{i-1}} | i = 1...T\}$).

However, non-DDPMs only necessitate one step (the input $\boldsymbol{x_t}$=$\emptyset$, while T=1) for the training and inference:

$$L_\theta = ||\boldsymbol{y_0} - f_\theta(\emptyset, \boldsymbol{c})||^2, \quad \boldsymbol{y'_0} = f_\theta(\emptyset, \boldsymbol{c}), \tag{15}$$

where the target $\boldsymbol{y_0}$=$\boldsymbol{x_0}$, while $\boldsymbol{y'_0}$ means the predicted $\boldsymbol{x_0}$.

Therefore, under the same setting, DDPMs are bound to conduct more training and inference iterations than non-DDPMs.

3.7. Why can the CNF of DDPMs achieve single-step inference?

According to Sec. 3.4, the performance of DDPMs depends on the noise fitting quality in NN. This necessitates multiple iterations, as the significant errors occur in a single step. Although reducing the number of iterations can accelerate the sampling process [12, 21], this introduces two limitations: a loss of accuracy [12, 21, 22] and a decrease in diversity (less noise introduced resulting in less randomness [21]). Fortunately, except generation tasks, most 3D tasks focus on the certainty of the results. CNF considers CN rather than NN as the task-dominant network, cleverly avoiding extensive iterations from DDPMs. Meanwhile, since NN still see noise samples and fits the noise during training, models retain the robustness from DDPMs.

We do not deny NCF of DDPMs, which thrives in generative tasks. Our goal is to provide a new way of applying DDPMs to 3D tasks, hoping to expand the application scope of DDPMs. In fact, generation tasks require diversity, while other tasks focus on certainty. CNF actually sacrifices the result diversity to enhance the performance certainty.

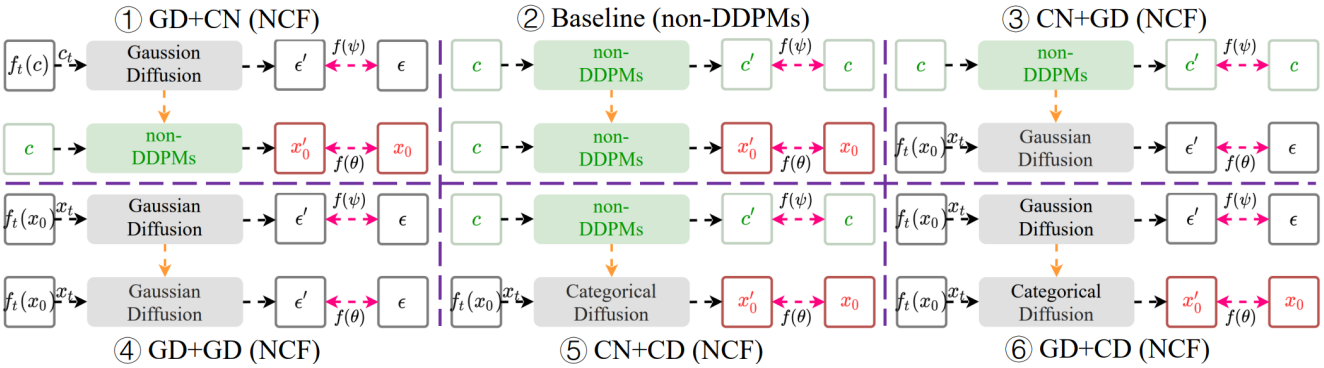

Figure 4. ① GD+CD (the auxiliary network as a Gaussian diffusion, the dominant network as a non-DDPM, CNF), ② Baseline (the auxiliary network as a non-DDPM, the dominant network as a non-DDPM, non-DDPMs), ③ CN+GD (the auxiliary network as a non-DDPM, the dominant network as a Gaussian diffusion, NCF), ④ GD+GD (the auxiliary network as a Gaussian diffusion, the dominant network as a Gaussian diffusion, NCF), ⑤ CN+CD (the auxiliary network as a non-DDPM, the dominant network as a categorical diffusion, NCF), and ⑥ GD+CD (the auxiliary network as a Gaussian diffusion, the dominant network as a categorical diffusion, NCF).

4. Implementation

4.1. Model Hyperparameters

In this section, we describe the implementation details of CDSegNet. CDSegNet is built on top of PTv3 and is depicted in Fig. 3. For subsequent pooling and denoising, the initialization module serializes the point cloud and conducts the time and position embeding. Meanwhile, both the Noise Network (NN) and the Conditional Network (CN), each composed of an encoder, a decoder, and a head, fit the noise and the task target (semantic labels), respectively. Moreover, FFM directs the noise information from NN to CN, enhancing the semantic features in CN. The detailed parameters of the network architecture of CDSegNet are described in Tab. 8, while the training hyperparameters for each benchmark (ScanNet, ScanNet200, nuScenes) are shown in Tab. 9.

We used 4 NVIDIA 4090 GPUs to train CDSegNet on ScanNet, ScanNet200 and nuScenes, which took approximately 21 hours, 21 hours and 29 hours, respectively. Additionally, we also tried to train CDSegNet on ScanNet using a single NVIDIA 3090 GPU, which took approximately 65 hours (batch size=2, this still can achieve 77.9 mIoU).

4.2. Combinations for Conditional DDPMs

As mentioned in Sec. 4.1 of the main text, CN and NN allow us to transcend the limitation of non-DDPMs and DDPMs. This means that NCF and CNF can use any type of DDPMs [2]. This only requires CN (the non-DDPM process) to be the dominant backbone in CNF, while NCF is dominated by NN (the DDPM process) (see Fig. 4 and Fig. 5). For example, in Fig. 5 of the main text, CD+DD (NCF) indicates that NN and CN are modeled as a Gaussian diffusion [8] and a categorical diffusion [1], respectively. Therefore, this can produce various combinations of conditional DDPMs. The above combination (NCF, ③, ④, ⑤, ⑥) is the same as the network framework of Baseline (②) and CDSegNet (CNF, ①). We provide implementations for all combinations.

Config	Parameter
Serialization Pattern	Z + TZ + H + TH
Patch Interaction	Shift Order + Shuffle Order
Time Embeding	Cos-Sin (128)
Positional Embeding	xCPE (32)
MLP Ratio	4
QKV Bias	True
Drop Path	0.3
NN Stride	[4,4]
NN Encoder Depth	[2,2]
NN Encoder Channels	[64,128]
NN Encoder Num Heads	[4,8]
NN Encoder Patch Size	[1024,1024]
NN Decoder Depth	[2,2]
NN Decoder Channels	[64,64]
NN Decoder Num Heads	[4,4]
NN Decoder Patch Size	[1024,1024]
NN Skip Connection	Element Addition
CN Stride	[2,2,2,2]
CN Encoder Depth	[2,2,6,6]
CN Encoder Channels	[64,128,256,512]
CN Encoder Num Heads	[4,8,16,32]
CN Encoder Patch Size	[1024,1024,1024,1024]
CN Decoder Depth	[2,2,2,2]
CN Decoder Channels	[64,64,128,256]
CN Decoder Num Heads	[4,4,8,16]
CN Decoder Patch Size	[1024,1024,1024,1024]
CN Skip Connection	Channel Concatenation
FFM Position Encoding	xCPE (128,512)
FFM Feat Scale	1.0
FFM Depth	[1,]
FFM Channels	[512,]
FFM Num Heads	[32,]
FFM Patch Size	[1024,]

Table 8. The parameters of network framework for CDSegNet.

ScanNet [6]		ScanNet200 [18]		nuScenes [3]	
Config	Parameter	Config	Parameter	Config	Parameter
Optimizer	AdamW	Optimizer	AdamW	Optimizer	AdamW
Scheduler	Cosine	Scheduler	Cosine	Scheduler	Cosine
LR	0.002	LR	0.002	LR	0.002
Block LR	0.0002	Block LR	0.0002	Block LR	0.0002
Weight De.	0.005	Weight De.	0.005	Weight De.	0.005
Batch Size	8	Batch Size	8	Batch Size	8
Epoch	800	Epoch	800	Epoch	50
Loss St.	GLS	Loss St.	GLS	Loss St.	GLS
Task Num	2	Task Num	2	Task Num	2
Target	ϵ	Target	ϵ	Target	ϵ
T	1000	T	1000	T	1000
Schedule	Cosine	Schedule	Linear	Schedule	Linear
Range	[0,1000]	Range	[1e-1,1e-5]	Range	[1e-2,1e-3]

Table 9. The training hyperparameters of CDSegNet for different benchmarks.

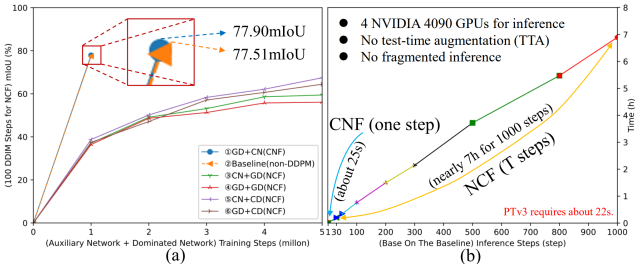

Figure 5. We try several combinations for conditional DDPMs built on the baseline on ScanNet in (a). The model of CNF (①) and baseline (②) only perform a step inference. Meanwhile, the models of NCF (③,④,⑤,⑥) use DDIM for 100 inference steps. (b) shows the inference time cost of CNF and NCF under the baseline. To effectively evaluate the inference cost, the models of CNF and NCF do not use test augmentation, and the voxel size of the input point cloud is 0.001m (i.e., no fragmentation inference). CNF achieves better performance with fewer iterations.

For modeling the Gaussian distribution, we follow [8], which simply employs MSE loss in NN to approximate the noise ϵ. Meanwhile, for modeling the categorical distribution, x_0 is used as the fitting target, and we employ KL divergence loss and cross-entropy loss in NN, similar to [1].

4.3. Integration of CNF into PointNet and PointNet++

Fig. 6 illustrates the framework of introducing CNF to PointNet and PointNet++. We use a additional PointNet++ to model the diffusion process and employ the FFM of CDSegNet as a noise filter. For PointNet, CNF is applied after the feature pooling stage. Meanwhile, for PointNet++, we introduce CNF at the bottleneck stage of the U-Net.

5. Optimization for CNF

Benefiting from a dual-branch framework, CNF has two fitting objectives: the noise fitting (NN) and the task-target fitting (CN). Therefore, CNF can be optimized from two perspectives: DDPMs and multi-task learning. Using PTv3+CNF on nuScenes as an example, we gradually exhibit the entire optimization process of CNF, aiming to pro-

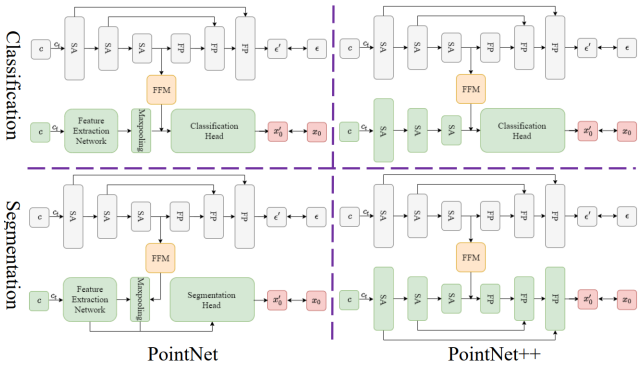

Figure 6. The framework of applying CNF to PointNet and PointNet++. We use a additional PointNet++ to model the diffusion process and FFM of CDSegNet as a noise filter.

vide a guidance for future applications (**All optimization processes are from previous checkpoint. However, this does not affect the performance improvement results**).

Baseline. Our baseline is PTv3+CNF, which is trained on nuScenes. This simply uses PTv3 as CN, with the additional inclusion of FFM and NN of CDSegNet. The network architecture and hyperparameter settings of CN in PTv3+CNF are entirely consistent with PTv3.

Tab. 10 shows the comparison results between the baseline and PTv3. The baseline (this directly introduces CNF onto PTv3 without any optimization) demonstrates a significant improvement in noise robustness. This is because the multi-level feature perturbations from NN enhance the noise adaptability of PTv3. Meanwhile, the baseline performs worse than PTv3 in terms of overall performance. This means that unreasonable noise perturbations harm the performance of CN.

Methods	Performance			Robustness
	mIoU	mAcc	allAcc	τ=0.1
PTv3 [24]	**80.3**	**87.2**	**94.6**	63.9
Baseline	79.6	86.9	94.1	**66.8**

Table 10. The results of the baseline and PTv3 on nuScenes.

The skip connection mode in the Decoder. We experimented with different skip connection modes in the Decoder: Element-Wise Addition (Baseline), Element-Wise Multiplication (EWM), and Channel Concatenation (CC).

Tab. 11 shows that CC achieves the better performance. Some works [9, 20] have demonstrated that in DDPMs, the Decoder typically generates high-frequency information, i.e., the details of the generated results. This requires feature fusion to retain as much information as possible. Channel concatenation effectively preserves information from the skip features and the backbone features though the expansion of the channel dimension. However, element-wise addition and multiplication causes the elements of the

skip features and the backbone features at the same spatial location to share the same number of channels, which may limit the ability of models to capture details.

We chose the model with the skip connection mode CC as the baseline.

Methods	Performance			Robustness
	mIoU	mAcc	allAcc	τ=0.1
Baseline	79.6	86.9	94.1	66.9
EWM	79.3	85.9	93.5	66.5
CC	**79.8**	**87.0**	**94.2**	**67.1**

Table 11. The results of the different skip connection modes in the Decoder on nuScenes.

The skip feature scaling in the Decoder. Varying values of t lead to oscillations in the loss values for DDPMs, which hinders the effective convergence of models. Some works [9, 19, 22] suggest that adjusting the skip feature scaling in the decoder can effectively address this issue:

$$F = cat(F_{SF} \times sf, F_{BF}) \tag{16}$$

where F_{SF} means the skip features from the Encoder, while F_{BF} represents the backbone features from the Decoder. sf indicates the scaling factor. Fig. 7(a) illustrates this detail of the skip feature scaling in the Decoder.

Tab. 12 shows that reducing the proportion of skip features in the decoder of NN makes training more stable and enhances the generative capability of models. Meanwhile, Fig. 7(b) further supports this viewpoint, as the training loss curve becomes smoother when using this trick.

We chose the model with the skip feature scaling $\sqrt{2}$ as the baseline.

Methods	Performance			Robustness
	mIoU	mAcc	allAcc	τ=0.1
Baseline	79.7	86.9	94.1	67.1
SL [9]	79.8	86.8	94.2	67.5
$\sqrt{2}$ [22]	**80.0**	**87.0**	**94.3**	**67.6**

Table 12. The result of the skip connection scaling in the Decoder on nuScenes.

The noise schedule range. We found in our experiments that the noise schedule range is critical to the performance of CNF. This is because the noise schedule range can control the perturbation degree from NN for CN.

Tab. 13 shows the results of different noise schedule ranges (the baseline means the noise schedule range is [0.0001,0.02]). As mentioned in Sec. 5.6 of the main text, in sparse and perturbed scenes, we should choose a noise schedule with a smaller range.

The loss strategy. CNF with two branches can also be optimized from a multi-task perspective. This can further

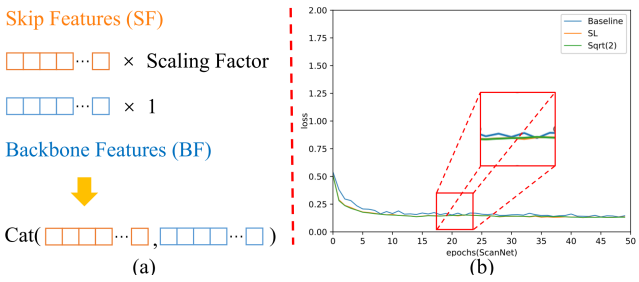

Figure 7. (a) shows the detail of the skip feature scaling. Meanwhile, (b) demonstrates a comparison of the training loss between the baseline, the skip feature scaling SL (ScaleLong) [9] and the skip feature scaling $\sqrt{2}$ [22].

Methods	Performance			Robustness
	mIoU	mAcc	allAcc	τ=0.1
Baseline	80.0	87.0	94.3	67.6
[0.001,0.005]	80.3	87.3	**94.4**	**67.7**
[0.002,0.003]	**80.5**	**87.5**	**94.4**	67.5

Table 13. The results of the different noise schedule ranges on nuScenes.

constrain unreasonable perturbations of NN, due to the convergence speed difference between NN and CN. As mentioned in Tab. 9 of the main text, we tried several loss strategies: : 1) Equal Weighting (EW, Baseline). This means that the losses of all tasks use the same weight. 2) Random Loss Weighting (RLW) [11]. The random weight are assigned to the losses of all tasks. 3) Uncertainty Weights (UW) [10]. This utilizes a learnable weight to balance the losses of all tasks. 4) Geometric Loss Strategy (GLS) [5]. This mitigates the convergence differences of multiple tasks through a geometric mean weight.

Tab. 14 shows that GLS achieves the best results. GLS can alleviate the difference in convergence speed between NN and CN, making the noise perturbation from NN more reasonable.

Methods	Performance			Robustness
	mIoU	mAcc	allAcc	τ=0.1
Baseline	80.5	87.5	94.4	67.5
RLW [11]	80.4	87.5	94.3	67.5
UW [10]	80.6	87.7	94.7	67.6
GLS [5]	**80.8**	**87.8**	**94.8**	**67.8**

Table 14. The results of the different loss strategies.

6. Limitations

In the main text, CNF demonstrates excellent results across various tasks, such as segmentation and classification. However, during the design and experimentation process, we also identified some limitations of CNF.

- **Indirectly inheriting the robustness from DDPMs.** To avoid the extensive training and inference iterations from

DDPMs, CNF uses CN as the task-dominant network. This also results in only indirectly inheriting the robustness from DDPMs. Therefore, under the same setting (e.g., both models of NCF and CNF achieve the same task performance), we believe that the robustness of CNF will be lower than that of NCF.

- **Limited robustness to noise.** As mentioned in Sec. 5.3 of the main text, the robustness of CNF is limited to noise from the modeled or approximate distribution, but sensitive to noise far from the modeled distribution.
- **Requiring more parameters.** CNF requires additional NN and FFM, resulting in requiring more parameters. Nevertheless, this also enhances the generalization of models, alleviating overfitting, as shown in Fig. 8. PTv3-big and Our-CN, with more parameters compared to PTv3, show lower training loss curves, but exhibit worse generalization performance on ScanNet. In comparison, our method shows the lower training loss curve and the better performance. Meanwhile, CNF can demonstrate outstanding performance in outdoor scenes, as outdoor scenes are more sparse and perturbed compared to indoor scenes. Tab. 15 shows that PTv3, with CNF introduced and amounting to only about half the parameter count of PTv3-big, exhibits better performance.
- **Difficult to apply to generative tasks.** Our goal is to propose a new network framework that lowers the threshold for applying DDPMs to various 3D tasks. However, this appears to be difficult to apply to generative tasks, as the introduction of randomness in CN is indirect. This may result in a lack of diversity in the generated outcomes (see Sec. 3.7). We suggest to still use NCF of DDPMs in generation tasks.

7. More Visualization Results

We display additional visual results of semantic segmentation in Fig. 9, Fig. 10 and Fig. 11.

References

[1] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021. 3, 7, 8

[2] Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold diffusion: Inverting arbitrary image transforms without noise. *Advances in Neural Information Processing Systems*, 36, 2024. 3, 7

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 2, 8

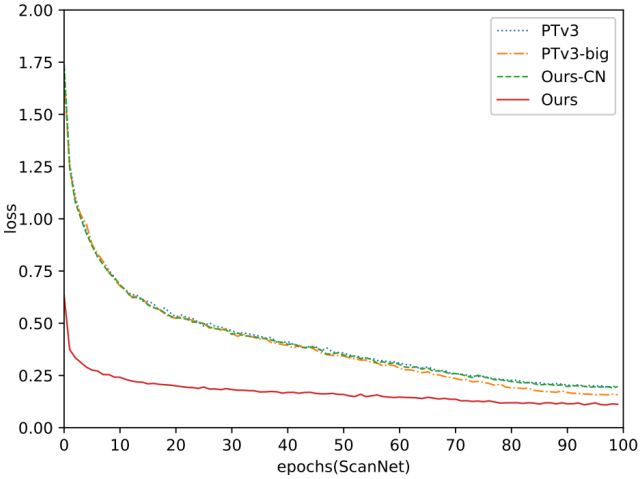

Figure 8. The training loss curves of PTv3, PTv3-big, Ours-CN and Ours. PTv3-big means using the same network architecture configuration as PTv3-PPT. Our-CN represents our method with NN entirely removed, retaining only CN. The loss curves of PTv3-big and Our-CN are almost identical to that of PTv3 on ScanNet. However, PTv3-big and Our-CN, with more parameters compared to PTv3, show poorer performance, demonstrating overfitting.

Methods	Performance			Params
	mIoU	mAcc	allAcc	
		ScanNet		
PTv3	77.6	85.0	92.0	46.2M
PTv3-big	76.8	85.0	91.8	97.3M
Ours-CN	76.6	84.6	91.6	88.1M
Ours-x0	77.7	84.7	91.6	101.4M
Ptv3+CNF	77.7	84.8	91.7	59.4M
Ours	**77.9**	**85.2**	**92.2**	101.4M
		ScanNet200		
PTv3	35.3	**46.0**	83.3	46.2M
PTv3-big	35.4	45.5	83.2	97.3M
Ours-CN	35.3	45.6	83.0	88.1M
Ours-x0	35.8	45.4	83.3	101.4M
Ptv3+CNF	35.9	45.3	83.4	59.4M
Ours	**36.3**	45.9	**83.9**	101.4M
		Nuscenes		
PTv3	80.3	87.2	94.6	46.2M
PTv3-big	80.4	87.2	94.5	97.3M
Ours-CN	80.4	87.1	94.2	88.1M
Ours-x0	80.7	87.6	94.7	101.4M
PTv3+CNF	81.0	**87.9**	**94.8**	59.4M
Ours	**81.2**	87.8	**94.8**	101.4M

Table 15. The results of PTv3, PTv3-big, Ours-CN and Ours on ScanNet, ScanNet200 and nuScenes. Our method achieves state-of-the-art performance on all benchmarks.

[4] 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. 2

[5] Sumanth Chennupati, Ganesh Sistu, Senthil Yogamani, and Samir A Rawashdeh. Multinet++: Multi-stream feature ag-

gregation and geometric loss strategy for multi-task learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019. 9

[6] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 2, 8

[7] Ying Fan and Kangwook Lee. Optimizing ddpm sampling with shortcut fine-tuning. *arXiv preprint arXiv:2301.13362*, 2023. 4

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3, 4, 7, 8

[9] Zhongzhan Huang, Zhou Pan, Shuicheng Yan, and Liang Lin. Scalelong: Towards more stable training of diffusion model via scaling network long skip connection. In *NeurIPS*, 2023. 8, 9

[10] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. 9

[11] Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *arXiv preprint arXiv:2111.10603*, 2021. 9

[12] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022. 6

[13] Zhaoyang Lyu, Zhifeng Kong, Xudong Xu, Liang Pan, and Dahua Lin. A conditional point diffusion-refinement paradigm for 3d point cloud completion. *arXiv preprint arXiv:2112.03530*, 2021. 1

[14] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021. 4

[15] 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. 2

[16] 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. 2

[17] Wentao Qu, Yuantian Shao, Lingwu Meng, Xiaoshui Huang, and Liang Xiao. A conditional denoising diffusion probabilistic model for point cloud upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20786–20795, 2024. 3, 5

[18] David Rozenberszki, Or Litany, and Angela Dai. Language-grounded indoor 3d semantic segmentation in the wild. In *European Conference on Computer Vision*, pages 125–141. Springer, 2022. 2, 8

[19] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE transactions on pattern analysis and machine intelligence*, 45(4):4713–4726, 2022. 9

[20] Chenyang Si, Ziqi Huang, Yuming Jiang, and Ziwei Liu. Freeu: Free lunch in diffusion u-net. In *CVPR*, 2024. 8

[21] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3, 6

[22] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. 5, 6, 9

[23] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 1

[24] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4840–4851, 2024. 2, 8

[25] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5728–5739, 2022. 1

[26] Haoyi Zhu, Honghui Yang, Xiaoyang Wu, Di Huang, Sha Zhang, Xianglong He, Tong He, Hengshuang Zhao, Chunhua Shen, Yu Qiao, and Wanli Ouyang. Ponderv2: Pave the way for 3d foundation model with a universal pre-training paradigm. *arXiv preprint arXiv:2310.08586*, 2023. 2

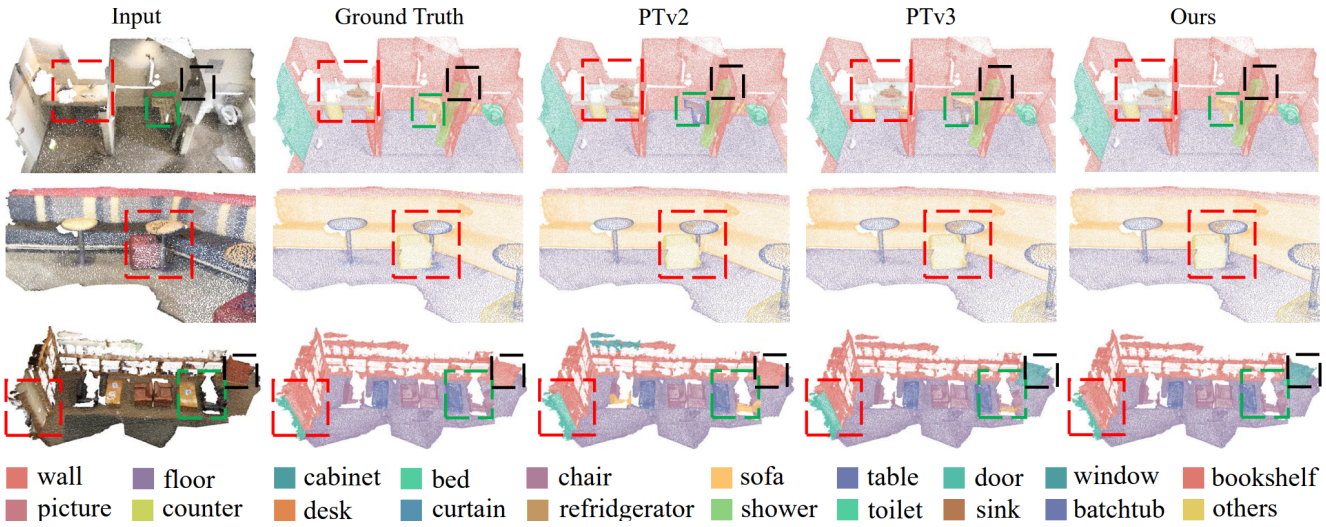

Figure 9. The visualization on ScanNet. The black dashed box indicates a misannotation in the Ground Truth that we believe exists.

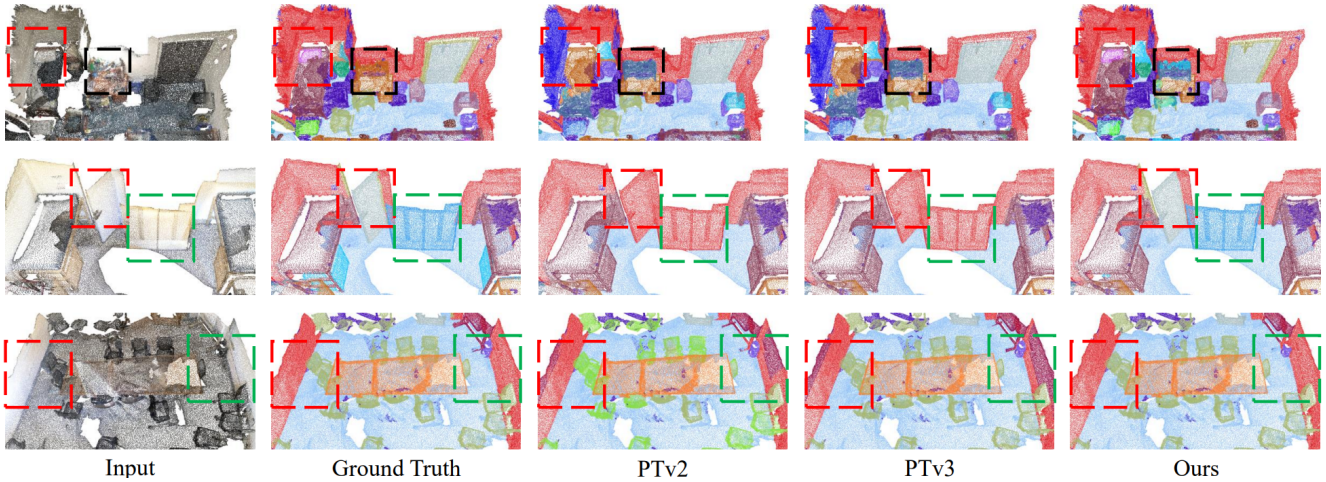

Figure 10. The visualization on ScanNet200. The black dashed box indicates a misannotation in the Ground Truth that we believe exists.

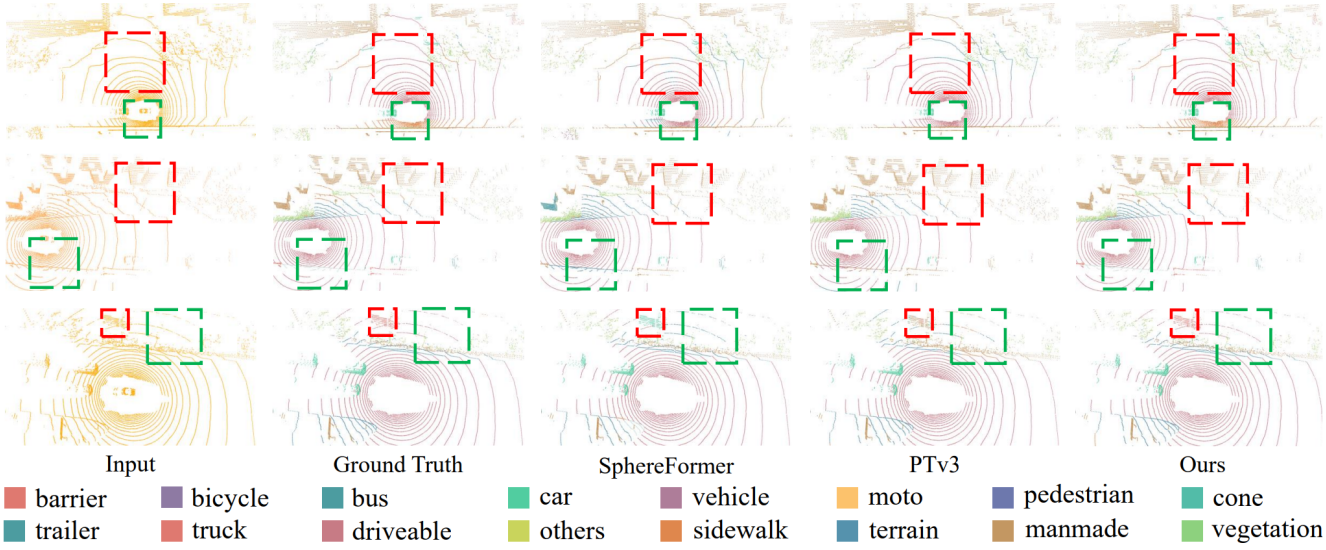

Figure 11. The visualization on nuScenes.