

Focal Sparse Convolutional Networks for 3D Object Detection

Yukang Chen¹, Yanwei Li¹, Xiangyu Zhang², Jian Sun², Jiaya Jia^{1,3}

¹The Chinese University of Hong Kong ²MEGVII Technology ³SmartMore

Abstract

Non-uniformed 3D sparse data, e.g., point clouds or voxels in different spatial positions, make contribution to the task of 3D object detection in different ways. Existing basic components in sparse convolutional networks (Sparse CNNs) process all sparse data, regardless of regular or submanifold sparse convolution. In this paper, we introduce two new modules to enhance the capability of Sparse CNNs, both are based on making feature sparsity learnable with position-wise importance prediction. They are focal sparse convolution (Focals Conv) and its multi-modal variant of focal sparse convolution with fusion, or Focals Conv-F for short. The new modules can readily substitute their plain counterparts in existing Sparse CNNs and be jointly trained in an end-to-end fashion. For the first time, we show that spatially learnable sparsity in sparse convolution is essential for sophisticated 3D object detection. Extensive experiments on the KITTI, nuScenes and Waymo benchmarks validate the effectiveness of our approach. Without bells and whistles, our results outperform all existing single-model entries on the nuScenes test benchmark. Code and models are at github.com/dvlab-research/FocalsConv.

1. Introduction

A key challenge in 3D object detection is to learn effective representations from the unstructured and sparse 3D geometric data such as point clouds. In general, there are two ways for this job. The first is to process point clouds [37, 51] directly, based on PointNet++ [33] networks. However, the neighbour sampling and grouping operations are time-consuming. This makes it improper for large-scale autonomous driving scenes that require real-time efficiency. The second is to convert point clouds into voxelizations and apply 3D sparse convolutional neural networks (Sparse CNNs) for feature extraction [11, 36]. 3D Sparse CNNs resemble 2D CNNs in structures, including several feature stages and down-sampling operations. They typically consist of *regular* and *submanifold* sparse convolutions [15].

Although regular and submanifold sparse convolutions have been widely used, they have respective limitations. Regular sparse convolution dilates all sparse features. It inevitably burdens models with considerable computations. That is why backbone networks commonly limit its usage only in down-sampling layers [36, 48]. In addition, detectors aim to distinguish target objects from massive background features. But regular sparse convolution reduces the sparsity sharply and blurs feature distinctions.

On the other hand, submanifold sparse convolutions avoid the computation issue by restricting the output feature positions to the input. But it misses necessary information flow, especially for the spatially disconnected features. The above issues on regular and submanifold sparse convolutions limit Sparse CNNs to achieve high representation capability and efficiency. We illustrate the submanifold and regular sparse convolutional operations in Fig. 1.

These limitations originate from the conventional convolution pattern: all input features are treated equally in the convolution process. It is natural for 2D CNNs, and yet is improper for 3D sparse features. 2D convolution is designed for structured data. All pixels in the same layer typically share receptive field sizes. But 3D sparse data is with varying sparsity and importance in space. It is not optimal to handle non-uniform data with uniform treatment. In terms of *sparsity*, upon the distance to LIDAR sensors, objects present large sparsity variance. In terms of *importance*, the contribution of features varies with different locations for 3D object detection, e.g., foreground or background. Although 3D object detection is achieved [11, 36, 37, 53], state-of-the-art methods still rely on RoI (region-of-interest) feature extraction. It corresponds to the idea that we should shoot arrows at the target in the feature extraction of 3D detectors.

In this paper, we propose a general format of sparse convolution by relaxing the conceptual difference between regular and submanifold ones. We introduce two new modules that improve the representation capacity of Sparse CNNs for 3D object detection. The first is focal sparse convolution (*Focals Conv*). It predicts *cubic importance* maps for the output pattern of convolutions. Features predicted as *important* ones are dilated into a *deformable* output shape,

Yukang’s work was done during internship in MEGVII Technology.

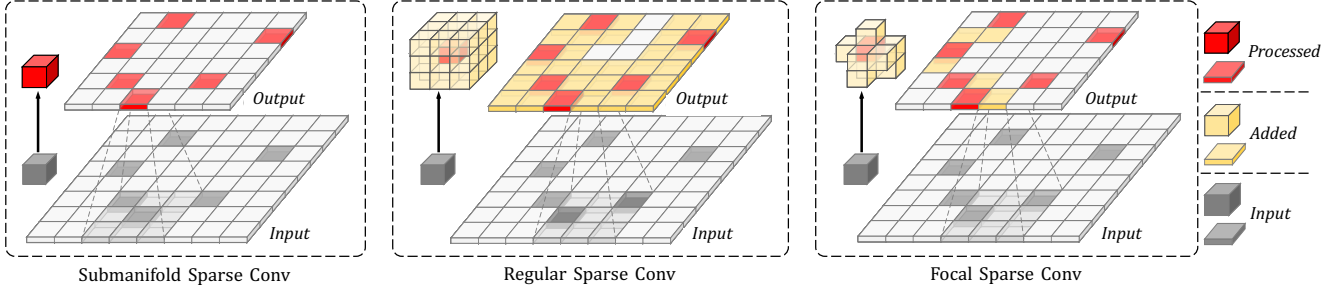


Figure 1. Process of different sparse convolution types. *Submanifold sparse convolution* fixes the output position identical to input. It maintains efficiency but disables information flow between disconnected features. *Regular sparse convolution* dilates all input features to its kernel-size neighbors. It encourages information communication with expensive computation, as it seriously increases feature density. The proposed *focal sparse convolution* dynamically determines which input features deserve dilation and dynamic output shapes, using predicted *cubic importance*. *Input* and *Output* are illustrated in 2D features for simplification. This figure is best viewed in color.

as shown in Fig 1. The importance is learned via an additional convolutional layer, dynamically conditioned on the input features. This module increases the ratio of valuable information among all features. The second is its multi-modal improved version of Focal sparse Convolution with Fusion (named as *Focals Conv-F*). Upon the LIDAR-only *Focals Conv*, we enhance importance prediction with RGB features fused, as image features typically contain rich appearance information and large receptive fields.

The proposed modules are novel in two aspects. First, *Focals Conv* presents a dynamic mechanism for learning spatial sparsity of features. It makes the learning process concentrated on the more valuable foreground data. With the down-sampling operations, valuable information increases in stages. Meanwhile, the large amount of background voxels are removed. Fig. 2 illustrates the learnable feature sparsity, including the common, crowded, and remote objects, where *Focals Conv* enriches the learned voxel features on the foreground without redundant voxels added in other areas. Second, both modules are lightweight. The importance prediction involves small overhead parameters and computation, as measured in Tab. 1. The RGB feature extraction of *Focals Conv-F* involves only *several layers*, instead of heady 2D detection or segmentation models [43].

The proposed modules of *Focals Conv* and *Focals Conv-F* can readily replace their original counterparts in sparse CNNs. To demonstrate the effectiveness, we build the backbone networks on existing 3D object detection frameworks [11, 36, 53]. Our method enables non-trivial enhancement with small model complexity overhead on both the KITTI [14] and nuScenes [2] benchmarks. These results manifest that learnable sparsity with focal points is essential. Without bells and whistles, our approach outperforms state-of-the-art ones on the nuScenes *test split* [2].

Convolutional dynamic mechanism adapts the operations conditioned on input data, *e.g.*, deformable convolu-

tions [10, 64] and dynamic convolutions [7, 49]. The key difference is that our approach makes use of the *intrinsic sparsity* of data. It promotes feature learning to be concentrated on more valuable information. We deem the non-uniform property as a great benefit. We discuss the relations and differences to previous literature in Sec. 2.

2. Related Work

2.1. Convolutional Dynamic Mechanism

Dynamic mechanisms have been widely studied in CNNs, due to their advantages of high accuracy and easy adaption in scenarios. We discuss two kinds of related methods, *i.e.*, kernel shape adaption [10, 41, 64], and input attention mask [34, 42, 45].

Kernel shape adaption. Kernel shape adaption methods [8, 10, 13, 64] adapt the effective receptive fields of networks. Deformable convolution [10] predicts offsets for feature sampling. Its variant [64] introduces an additional attention mask to modulate features. For 3D feature learning, KPConv [41] learns local offsets for kernel points. MinkowskiNet [8] generalizes sparse convolution to arbitrary kernel shape. Overall, these methods modify the input feature sampling process.

Deformable PV-RCNN [1] applies offset prediction for feature sampling in 3D object detection. In contrast, focal sparse convolution improves the output feature spatial sparsity and makes it learned, helpful for 3D object detection.

Attention mask on input. Methods of [39, 42, 45, 50] seek spatial-wise sparsity for efficient inference. These methods receive dense images and prune unimportant pixels based on attention masks. These methods aim to sparsify dense data while we make use of intrinsic data sparsity. Although SBNNet [34] also utilizes the sparse property, it limits application to 2D BEV (bird-eye-views) images, and shares the

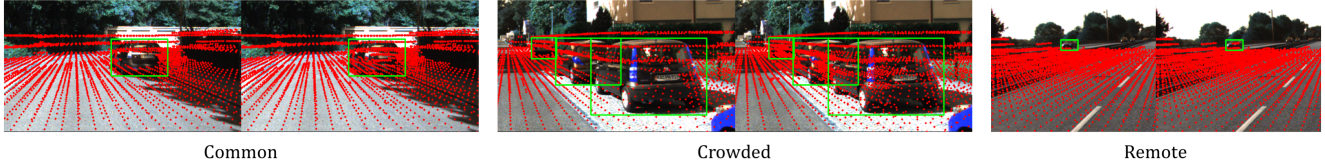


Figure 2. Illustrations on learnable feature sparsity. We project the 3D voxel centers from the backbone output onto 2D image planes. These cases include common, crowded, and remote objects. Left: plain Sparse CNN. Right: focal sparse CNN. Focal sparse convolution adaptively densifies object features without introducing redundant background features. This figure is best viewed in color and by zoom-in.

static masks over all layers in the network. In contrast, our improved convolution is more adaptive and is applicable to related tasks, e.g., 3D instance segmentation [9].

2.2. 3D Object Detection

LIDAR-only detectors. 3D object detection frameworks usually resemble 2D detectors, e.g., the R-CNN family [11, 28, 36, 37] and the SSD family [17, 51, 60, 61]. The main difference on 2D detectors lies in input encoders. VoxelNet [62] encodes voxel features using PointNet [32] and applies a RPN (region proposal network) [35]. SECOND [48] uses accelerated sparse convolutions and improves efficiency from VoxelNet [62]. VoTr [29] applies transformer architectures to voxels. Various detectors [11, 36, 53] have been presented based on feature encoders. We validate the proposed approach on backbones of frameworks of [11, 36, 53] on multiple datasets [2, 14, 40].

Completion-based detectors. Completion-based methods [16, 23, 46, 58] form another line of efforts in enriching foreground information. We focus on feature learning instead of point completion. PC-RGNN [58] has a point completion module by a graph neural network. SIENet [23] builds upon PCN [56] for point completion in a two-stage framework. The completion process relies on the prior generated proposals. GSDN [16] expands all features first through transposed convolutions and then by pruning. SPG [46] designs a semantic point generation module for domain adaption 3D object detection. It is applied during data preprocessing, complicating the detection pipelines.

Multi modal fusion. Multi-modal fusion methods [19, 25, 55] use more information than LIDAR-only ones. The KITTI [14] benchmark had been dominated by LIDAR-only methods until PointPainting [43] was proposed. It decorates raw point clouds with the corresponding image segmentation scores. PointAugmenting [44] further replaces the segmentation model with an 2D object detection one [12]. They are both decoration-based methods, which require image feature extraction on off-the-shelf 2D networks, before feeding into 3D detectors. Although promising results are achieved by these methods, the overall inference pipelines are complicated. Our multi-modal focal

sparse convolution differs from the above methods in two aspects. First, we only require several jointly trained layers for image feature extraction, rather than the heavy segmentation or detection models. Second, we only strengthen the predicted *important* features, instead of the uniform decoration [43, 44] for all LIDAR features.

3. Focal Sparse Convolutional Networks

In this section, we first review the formulation of sparse convolution in Sec. 3.1. Then, the proposed focal sparse convolution and its multi-modal extension will be elaborated in Sec. 3.2 and Sec. 3.3. We finally introduce the resulting focal sparse convolutional networks in Sec. 3.4.

3.1. Review of Sparse Convolution

Given an input feature x_p with number of c_{in} channels at position p in the d dimensional spatial space, we process this feature by a convolution with kernel weights $w \in \mathbb{R}^{K^d \times c_{in} \times c_{out}}$. For example, in the 3D coordinate space, w contains $c_{in} \times c_{out}$ spatial kernels with size 3 and $|K^d| = 3^3$. The convolution process is represented as

$$y_p = \sum_{k \in K^d} w_k \cdot x_{\bar{p}_k}, \quad (1)$$

where k enumerates all discrete locations in the kernel space K^d . $\bar{p}_k = p + k$ is the corresponding location around center p , where k is an offset distance from p .

This formulation accommodates most types of convolutions with simple modifications. When $p \in \mathbb{Z}$, the common convolution for dense input data is yielded. When \bar{p}_k is added with a learned offset $\Delta \bar{p}_k$, it includes the kernel shape adaption methods, e.g., deformable convolutions [10, 64]. Further, if W equals to a weighted sum $\sum \alpha_i W^i$, it generalizes to weight attention, e.g., dynamic convolution [7, 49]. Finally, when attention masks are multiplied to the input feature map x , this formulation makes input attention mask methods [34, 45].

For sparse input data, the feature position p does not belong to the dense discrete space \mathbb{Z} . The input and output feature spatial space is relaxed to P_{in} and P_{out} , respectively.

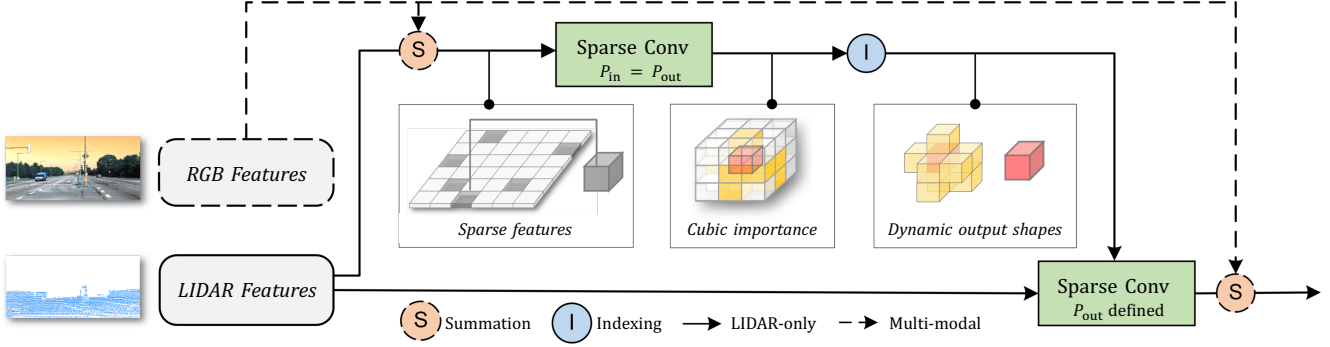


Figure 3. Framework of focal sparse convolution and its multi-modal extension. An additional branch predicts a cubic importance map for each input sparse feature, which determines the output feature positions. In the multi-modal version, the additional branch takes fusion of LIDAR and RGB features for better prediction. Output sparse features predicted as important are also fused with the RGB features.

The formulation is converted to

$$y_{p \in P_{\text{out}}} = \sum_{k \in K^d(p, P_{\text{in}})} w_k \cdot x_{\bar{p}_k}, \quad (2)$$

where $K^d(p, P_{\text{in}})$ is a subset of K^d , leaving out the empty position. It is conditioned on the position p and input feature space P_{in} as

$$K^d(p, P_{\text{in}}) = \{k \mid p + k \in P_{\text{in}}, k \in K^d\}. \quad (3)$$

If P_{out} includes a union of all dilated positions around P_{in} within K^d neighbours, this process is formulated as

$$P_{\text{out}} = \bigcup_{p \in P_{\text{in}}} P(p, K^d), \quad (4)$$

where

$$P(p, K^d) = \{p + k \mid k \in K^d\}. \quad (5)$$

On this condition, the formulation becomes regular sparse convolution. It acts at all positions where any voxels exist in its kernel space. It does not skip any information gathering in the total spatial space.

This strategy involves two drawbacks. (i) It introduces considerable computation cost. The number of sparse features is doubled or even tripled, increasing burden for following layers. (ii) We empirically find that continuously increasing the number of sparse features may harm 3D object detection (Tab. 2). Crowded and unpromising candidate features may blur the valuable information. It degrades foreground features and further declines the feature discrimination capacity of 3D object detectors.

When $P_{\text{in}} = P_{\text{out}}$, submanifold sparse convolution [15] is yielded. It happens only when the kernel centers locate at the input, restricting the active positions to input sets. This setting avoids the computation burden, but abandons necessary information flow between disconnected features. Note

that the flow is common in the irregular point cloud data. Thus, effective receptive field sizes are constrained by the feature disconnection, which degrade the model capability.

3.2. Focal Sparse Convolution

Regardless of regular or submanifold sparse convolution, output positions P_{out} are static across all $p \in P_{\text{in}}$, which is undesirable. In contrast, we perform adaptive determination of sparsity or receptive field sizes in a fine-grained manner. We relax output positions P_{out} to be dynamically determined by the sparse features. We illustrate this proposed process in Fig. 3 (via solid lines).

In our formulation, output positions P_{out} generalize to a union of all important positions with their dilated area and other unimportant positions. The dilated areas are deformable and dynamic to input positions. Eq. (5) becomes

$$P_{\text{out}} = \left(\bigcup_{p \in P_{\text{in}}} P(p, K_{\text{im}}^d(p)) \right) \cup P_{\text{in}/\text{im}}. \quad (6)$$

We factorize this process into three steps: (i) *cubic importance prediction*, (ii) *important input selection*, and (iii) *dynamic output shape generation*.

Cubic importance prediction. A cubic importance map I^p involves importance for candidate output features around the input feature at position p . Each cubic importance map shares the same shape K^d with the main processing convolution kernel weight, *e.g.*, $k^3 = 3 \times 3 \times 3$ with the kernel size 3. It is predicted by an additional submanifold sparse convolution with a sigmoid function. The latter steps depend on the predicted cubic importance maps.

Important input selection. In Eq. (5), P_{in} is a subset of P_{in} . It contains the positions of relatively important input features. We select P_{im} as

$$P_{\text{im}} = \{p \mid I_0^p \geq \tau, p \in P_{\text{in}}\}, \quad (7)$$

where I_0^p is the *center* of the cubic importance map at position p . And τ is a pre-defined threshold (Tab. 3 and 6). Our formulation becomes the regular or submanifold sparse convolution when τ is 0 or 1 respectively. We also find that using top-k ratio to select is an alternative of threshold.

Dynamic output shape generation. Features in P_{im} is dilated to a dynamic shape. The output around p is determined by the dynamic output shape $K_{\text{im}}^d(p)$. Note that our deformable output shapes are pruned inside the original dilation without offsets. It is computed similarly to Eq. (7) as

$$K_{\text{im}}^d(p) = \{k \mid p+k \in P_{\text{in}}, I_k^p \geq \tau, k \in K^d\}. \quad (8)$$

We analyze the dynamic output shape in Tab. 2. For the remaining unimportant features, their output positions are fixed as input, *i.e.*, submanifold. We found that directly removing them or using a fully dynamic manner without preserving them makes the training process unstable.

Supervision manners. In 3D object detection, we have a prior knowledge that foreground objects are more valuable information. Based on this prior, we apply focal loss [26] as an objective loss function to supervise the importance prediction. We construct the objective targets for the centers of feature voxels inside 3D ground-truth boxes. We keep its loss weight as 1 for the generality of our modules.

Additional supervision comes from multiplying the predicted cubic importance maps to output features as attention. It makes the importance prediction branch differentiable naturally. It shares motivation with the kernel weight sparsification methods [27] in the area of model compression. We empirically show that this attention manner benefits the performance for minor classes, *e.g.*, Pedestrian and Cyclist on KITTI (investigated in Tab. 4).

3.3. Fusion Focal Sparse Convolution

We provide a multi-modal version of focal sparse convolution, as illustrated in Fig. 3 (via dashed lines). This extension is conceptually simple but effective. We extract RGB features from images and align LIDAR features to them. The extracted features are fused to input and *important* output sparse features in focal sparse convolution.

Feature extraction. The fusion module is lightweight. It contains a conv-bn-relu layer and a max-pooling layer. It down-samples the input image to 1/4 resolutions. It is followed by 3 *conv-bn-relu* layers with residual connection [18]. The channel number is then reduced to be consistent with that of sparse features, with an MLP layer. This facilitates a simple summation of multi-modal features.

Feature alignment. A common issue during fusion is misalignment in the 3D-to-2D projection. Point cloud data is commonly processed by transformation and augmentation. Transformations include flip, re-scale, rotation, trans-

lation. The typical augmentation is ground-truth sampling [48], copying paste objects from other scenes. For these invertible transformations, we reverse the coordinates of sparse features with the recorded transformation parameters [44,57]. For ground-truth sampling, we copy the corresponding 2D objects onto images. Rather than using an additional segmentation model or mask annotations [57], we directly crop objects in bounding boxes for simplification.

Fusion manners. The aligned RGB features are directly fused to sparse features in *summation*, as they share the same channel numbers. Although other fusion methods, *e.g.*, concatenation or cross-attention, can be used, we choose the most concise summation for efficiency. The aligned RGB features are fused with sparse features twice in this module. It is first fused to input features for cubic importance prediction. Then we fuse RGB features only to *important* output sparse features, *i.e.*, the first part in Eq. (5), instead of all of them (investigated in Tab. 10).

Overall, the multi-modal layers are lightweight in terms of parameters and fusion strategies. They are jointly trained with detectors. It provides an efficient and economical solution for the fusion module in 3D object detection.

3.4. Focal Sparse Convolutional Networks

Both focal sparse convolution and its multi-modal extension can readily replace their counterparts in the backbone networks of 3D detectors. During training, we do not use any special initialization or learning rate settings for the introduced modules. The importance prediction branch is trained via back-propagation through the attention multiplication and objective loss function as introduced in Sec. 3.2.

The backbone networks in 3D object detectors [11, 36, 53] typically consist of one stem layer and 4 stages. Each stage, except the first one, includes a regular sparse convolution with down-sampling and two submanifold blocks. In the first stage, there are one [11, 36] or two [53] sparse convolutional layers. By default, each sparse convolution is followed by batch normalization [20] and ReLU activation.

We validate focal sparse convolution on the backbone networks of existing 3D detectors [11, 36, 53]. We directly apply focal sparse convolution at the last layer of certain stages. We analyze the stages for using our focal sparse convolution in experiments (ablated in Tab. 5 and 10).

4. Experiments

We conduct ablations and comparisons for *Focals Conv* and its multi-modal variant. More experiments, such as results on Waymo [40], are in the supplementary material.

4.1. Setup and Implementation

KITTI. The KITTI dataset [14] consists of 7,481 samples and 7,518 testing samples. The training samples are split

Table 1. Improvements on PV-RCNN in AP_{3D}(R40) on KITTI *val*.

Method	#Params	Runtime	Easy	Mod.	Hard
PV-RCNN [36]	–	–	92.57	84.83	82.69
PV-RCNN [◦]	13.16M	103ms	92.10	84.36	82.48
Focals Conv	13.44M	112ms	92.32	85.19	82.62
Focals Conv-F	13.70M	125ms	92.26	85.32	82.95

[◦] These results are evaluated on the official released model.

into a *train* set with 3,717 samples and a *val* set with 3,769 samples. Models are commonly evaluated in terms of the mean Average Precision (mAP) metric. mAP is calculated with recall 40 positions (R40). We perform ablation studies with AP_{3D} (R40) on the *val* split. We conduct main comparisons with AP_{3D} (R40) on *test* split and AP_{3D} (R11) on the *val* split. For the optional multi-modal settings, RGB features are extracted from single front-view for fusion.

nuScenes. The nuScenes [2] is a large-scale dataset, which contains 1,000 driving sequences in total. It is split into 700 scenes for training, 150 scenes for validation, and 150 scenes for testing. It is collected using a 32-beam synced LIDAR and 6 cameras with the complete 360° environment coverage. In evaluation, the main metrics are mAP and nuScenes detection score (NDS). In terms of multi-modal experiments, we use images of 6 views for fusion. For ablation study, models are trained on $\frac{1}{4}$ training data and evaluated on the entire validation set, *i.e.*, nuScenes $\frac{1}{4}$ split.

Implementation details. In experiments, we validate our modules on state-of-the-art frameworks of PV-RCNN [36], Voxel R-CNN [11] on KITTI [14], and CenterPoint [53] on nuScenes [2]. In LIDAR-only experiments, we apply Focals Conv in the first three stages of backbone networks. In multi-modal cases, we apply Focals Conv-F only in the first stage of the backbone network, for affordable memory and inference cost. We set the importance threshold τ to 0.5. We keep other settings intact. More experimental details are provided in the supplementary material.

4.2. Ablation Studies

Improvements on KITTI. We first evaluate our methods over PV-RCNN [36] in Tab. 1, as it is a high-performance, multi-class, and open-sourced framework. In Tab. 1, the 1st and 2nd lines show the reported results [36] and results tested from the released model. We take the latter as the baseline. Focal S-Conv and Focals Conv-F achieve non-trivial improvement over this strong baseline.

Dynamic output shape. In Focals Conv, the output shape from every single voxel is dynamically determined by the predicted importance maps. We ablate this by fixing output shapes as regular dilation, without any other change. Tab. 2 shows that dilating all sparse features is harmful. It dramati-

Table 2. Ablations on dynamic shape in AP_{3D} (R40) on KITTI *val*.

Method	Dynamic shape	Car			Ped.	Cyc.
		Easy	Mod.	Hard	Mod.	Mod.
Baseline	–	92.10	84.36	82.48	54.49	70.38
Focals Conv	\times	91.10	84.02	82.22	57.62	69.82
	\checkmark	92.32	85.19	82.62	61.61	72.76

Table 3. Ablations on input selection in AP_{3D} (R40) on KITTI *val*.

Method	Important selection	Car			Ped.	Cyc.
		Easy	Mod.	Hard	Mod.	Mod.
Baseline	–	92.10	84.36	82.48	54.49	70.38
Focals Conv	\times	91.36	82.77	82.12	57.86	71.77
	\checkmark	92.32	85.19	82.62	61.61	72.76

Table 4. Ablations on supervisions in AP_{3D} (R40) on KITTI *val*.

Method	Supervision	Car			Ped.	Cyc.
		Easy	Mod.	Hard	Mod.	Mod.
Baseline	–	92.10	84.36	82.48	54.49	70.38
Focals Conv	Attention	91.81	84.49	82.31	60.64	72.93
	Obj. loss	92.39	85.05	82.62	59.27	71.46
	Both	92.32	85.19	82.62	61.61	72.76

Table 5. Ablations on use stages in AP_{3D} (R40) on KITTI *val*.

Method	Stages	Car			Ped.	Cyc.
		Easy	Mod.	Hard	Mod.	Mod.
Baseline	–	92.10	84.36	82.48	54.49	70.38
Focals Conv	(1,.)	92.19	84.83	82.43	60.56	72.29
	(1, 2)	91.95	84.95	82.67	60.17	72.74
	(1, 2, 3)	92.32	85.19	82.62	61.61	72.76
	(1, 2, 3, 4)	91.96	84.42	82.31	60.33	72.53

Table 6. Ablations on the importance threshold τ on KITTI *val*.

Importance Threshold τ	0.1	0.3	0.5	0.7	0.9
AP _{3D} (R40) - Car	84.97	85.09	85.19	84.96	84.68

ically increases the number of unpromising voxel features.

Importance sampling. Focals Conv selects sparse features that need dilation with predicted importance. To ablate this module, we replace the importance selection (the important input selection step) with a random sample in Tab. 3 without other changes. It shows that large performance drop occurs without the guidance of importance. This validates that the importance prediction is necessary.

Supervision setting. The additional branch in Focals Conv is supervised by both attention multiplication and the objective loss. We ablate them in Tab. 4. Only using objective loss supervision is enough to ensure performance on

Table 7. Comparison on KITTI *test* split in AP_{3D} (R40) for *Car*.

Method	Fusion	Easy	Mod.	Hard
MV3D [6]		74.97	63.63	54.00
F-PointNet [31]		82.19	69.79	60.59
AVOD-FPN [21]		83.07	71.76	65.73
PointSIFT+SENet [59]	✓	85.99	72.72	64.58
MMF [24]		88.40	77.43	70.22
EPNet [19]		89.81	79.28	74.59
3D-CVF [55]		89.20	80.05	73.11
CLOCs [30]		88.94	80.67	77.15
PointPillars [22]		82.58	74.31	68.99
Point R-CNN [37]		86.96	75.64	70.70
Part-A ² [38]		87.81	78.49	73.51
STD [52]		87.95	79.71	75.09
SA-SSD [17]		88.75	79.79	74.16
PV-RCNN [36]	✗	90.25	81.43	76.82
Pyramid-PV [28]		88.39	82.08	77.49
VoTr-TSD [29]		89.90	82.09	79.14
Voxel R-CNN [11]	✗	90.90	81.62	77.06
Focals Conv	✗	90.20	82.12	77.50
Focals Conv-F	✓	90.55	82.28	77.59

Car. However, its performance on minor classes, *Ped.* and *Cyc.*, is not optimal. Attention multiplication is beneficial to *Ped.* and *Cyc.* We assume that minor classes cannot get balanced supervision from the objective loss like the long-tailed distribution. In contrast, attention multiplication is object-agnostic, relaxing the imbalance to some degree.

Stages for using focal sparse convolution. Tab. 5 shows results of using Focals Conv in different numbers of stages. (1) Applying Focals Conv in the first stage, which already obtains clear improvement. The performance enhances as the used stage increases until all stages are involved. Since Focals Conv adjusts output sparsity, it is reasonable to be used in early stages that make effects on subsequent feature learning. The spatial feature space in the last stage is down-sampled to a very limited size, which might not be large enough for sparsity adaptation. Empirically, usage in the last layer of the first three stages is the best choice. It is thus used as the default setting in our experiments.

Importance threshold. We ablate the importance threshold τ used in Focals Conv in Tab. 6. We run experiments with this value ranging from 0.1 to 0.9 and interval 0.2, without other change of settings. The accuracy AP_{3D} (R40) on *Car* serves as the metric in this ablation. The performance is stable as the threshold value τ varies.

Improvements over multi-modal baseline on nuScenes. We evaluate our multi-modal Focals Conv on the nuScenes [2] 1/4 dataset. More improvement is presented in Tab. 9. We build a multi-modal CenterPoint baseline by

Table 8. Comparison on KITTI *val* split in AP_{3D} (R11) for *Car*.

Method	Fusion	Easy	Mod.	Hard
F-PointNet [31]		83.76	70.92	63.65
PointSIFT+SENet [59]	✓	85.62	72.05	64.19
3D-CVF [55]		89.67	79.88	78.47
PointPillars [22]		86.62	76.06	68.91
Point R-CNN [37]		88.88	78.63	77.38
Part-A ² [38]		89.47	79.47	78.54
STD [52]		89.70	79.80	79.30
SA-SSD [17]	✗	90.15	79.91	78.78
Deform. PV-RCNN [1]		-	83.30	-
PV-RCNN [36]		89.35	83.69	78.70
VoTr-TSD [29]		89.04	84.04	78.68
Pyramid-PV [28]		89.37	84.38	78.84
Voxel R-CNN [11]	✗	89.41	84.52	78.93
Focals Conv	✗	89.52	84.93	79.18
Focals Conv-F	✓	89.82	85.22	85.19

Table 9. Improvement over multi-modal baseline on nuScenes $\frac{1}{4}$.

	#Params	Runtime	mAP	NDS
CenterPoint	9.0M	93ms	56.1	64.2
+ Fusion	9.24M	145ms	59.0 (+2.9)	65.6 (+1.4)
Focals Conv-F	9.25M	159ms	61.7 (+5.6)	67.2 (+2.9)

Table 10. Ablations on use stage and fusion scope on nuScenes $\frac{1}{4}$.

Stage	None	1		2	3	4	
Scope	-	None	Imp.	All	Imp.	Imp.	Imp.
mAP	56.1	58.6	61.7	60.9	60.7	55.0	54.8
NDS	64.2	66.2	67.2	66.4	66.5	63.5	63.3

fusing image features to the same fusion layer used in our methods, with the same fusion and feature extraction layers. This multi-modal CenterPoint enhances the LIDAR-only baseline from 56.1% to 59.0% mAP. Focals Conv-F improves to 61.7% mAP on this strong baseline.

Use stages and fusion scope for Focals Conv-F. We ablate the usage stages and fusion scope for Focals Conv-F in Tab. 10. Fusion scope is the scope of sparse features to fuse with RGB features at the output of Focals Conv-F. It shows that fusion in the early stages is beneficial, and becomes adverse in the last two stages. *Imp.* means only fusing onto important output features (judged by importance maps). When fusing in the first stage, it is better to fuse on *important* features, instead of all of them, making representation discriminative.

Model complexity and runtime. We report the model complexity and runtime comparisons in Tab. 1 and 9. The runtimes are evaluated on the same GPU machine. Focals

Table 11. Comparison with other methods on nuScenes *test* split.

<i>Method</i>	<i>Fusion</i>	mAP	NDS	Car	Truck	Bus	Trailer	C.V.	Ped	Mot	Byc	T.C.	Bar
PointPillars [22]	✗	30.5	45.3	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9
3DSSD [51]	✗	42.6	56.4	81.2	47.2	61.4	30.5	12.6	70.2	36.0	8.6	31.1	47.9
CBGS [63]	✗	52.8	63.3	81.1	48.5	54.9	42.9	10.5	80.1	51.5	22.3	70.9	65.7
HotSpotNet [5]	✗	59.3	66.0	83.1	50.9	56.4	53.3	23.0	81.3	63.5	36.6	73.0	71.6
CVCNET [4]	✗	58.2	66.6	82.6	49.5	59.4	51.1	16.2	83.0	61.8	38.8	69.7	69.7
PointPainting [43]	✓	46.4	58.1	77.9	35.8	36.2	37.3	15.8	73.3	41.5	24.1	62.4	60.2
3DCVF [55]	✓	52.7	62.3	83.0	45.0	48.8	49.6	15.9	74.2	51.2	30.4	62.9	65.9
FusionPainting [47]	✓	66.3	70.4	86.3	58.5	66.8	59.4	27.7	87.5	71.2	51.7	84.2	70.2
MVF [54]	✓	66.4	70.5	86.8	58.5	67.4	57.3	26.1	89.1	70.0	49.3	85.0	74.8
PointAugmenting [44]	✓	66.8	71.0	87.5	57.3	65.2	60.7	28.0	87.9	74.3	50.9	83.6	72.6
CenterPoint [53]	✗	58.0	65.5	84.6	51.0	60.2	53.2	17.5	83.4	53.7	28.7	76.7	70.9
CenterPoint [†]	✗	60.3	67.3	85.2	53.5	63.6	56.0	20.0	84.6	59.5	30.7	78.4	71.1
CenterPoint v2*	✓	67.1	71.4	87.0	57.3	69.3	60.4	28.8	90.4	71.3	49.0	86.8	71.0
Focals Conv	✗	63.8	70.0	86.7	56.3	67.7	59.5	23.8	87.5	64.5	36.3	81.4	74.1
Focals Conv-F	✓	67.8	71.8	86.5	57.5	68.7	60.6	31.2	87.3	76.4	52.5	84.6	72.3
Focals Conv-F [†]	✓	68.9	72.8	86.9	59.3	68.7	62.5	32.8	87.8	78.5	53.9	85.5	72.8
Focals Conv-F [‡]	✓	70.1	73.6	87.5	60.0	69.9	64.0	32.6	89.0	81.1	59.2	85.5	71.8

[†] Flip testing. [‡] Flip and rotation testing. * CenterPoint v2 includes PointPainting with Cascade R-CNN [3] and model-ensembling.

Conv and its multi-modal variant only add a small overhead to model parameters and computation, on KITTI [14]. This indicates that the performance improvement comes from the model capacity of sparsity learning, instead of increasing model sizes. On nuScenes [2], the overall runtime rises from 93 ms to 159 ms. But parameters are still limited. It is a common limitation in multi-view fusion methods. The multi-modal baseline also requires 145 ms. The reason is that there are 6-view images to process per frame.

4.3. Main Results

KITTI. We compare our Focals Conv modules upon Voxel R-CNN [11] with previous state-of-the-art methods on both the KITTI *test* and *val* split. In Tab. 7, we compare with both LIDAR-only and multi-modal methods. The original Voxel R-CNN [11] is comparable to PV-RCNN [36] and is inferior to Pyramid-PV [28] and VoTr-TSD [29]. Focals Conv improves it to surpass these two new methods. Using Focals Conv-F, the multi-modal Voxel R-CNN achieves 82.28% AP_{3D} on the KITTI *test* split. Tab. 8 shows comparisons on KITTI *val* split in AP_{3D} in recall 11 positions. Focals Conv and Focals Conv-F enhance this leading result to 84.93% and 85.22% respectively in *Car* class.

nuScenes. On the nuScenes dataset, we evaluate our models on the test server and compare them with both LIDAR-only and multi-modal methods, as in Tab. 11. Focals Conv improves CenterPoint [53] by a large margin to 63.8% mAP. Multi-modal methods present much better performance than LIDAR-only methods on the nuScenes

dataset. CenterPoint v2* includes PointPainting [43], Cascade R-CNN [3] instance segmentation models pre-trained on nuImages, and five-model ensembling. As the testing augmentations are not unified or stated in previous methods, we provide two results of our final model. Focals Conv-F achieves 67.8% mAP and 71.8% mAP without any ensembling or testing augmentation. Focals Conv-F[‡] further achieves 70.1% mAP and 73.6% NDS with test-time augmentations [53]. Both results outperform previous methods.

5. Conclusion and Discussion

This paper presents a focal sparse convolution and a multi-modal extension, which are simple and effective. They are end-to-end solutions for LIDAR-only and multi-modal 3D object detection. For the first time, we show that the learned sparsity with focal points is essential for 3D object detectors. Notably, focal and fusion sparse CNNs achieve leading performance on the large-scale nuScenes.

Limitations. In the multi-modal 3D detection that requires multiple views, *e.g.*, 6 high-resolution images per frame in nuScenes [2], computation cost increases, although the image branch is already largely simplified.

Boarder Impacts. The proposed method relies on the sparsity of data distribution. It might reflect biases in data collection, including the ones of negative societal impacts.

Acknowledgements. This work is in part supported by The National Key Research and Development Program of China (No. 2017YFA0700800) and Beijing Academy of Artificial Intelligence (BAAI).

References

- [1] Prarthana Bhattacharyya and Krzysztof Czarnecki. Deformable PV-RCNN: improving 3d object detection with learned deformations. In *ECCV Workshop*, 2020. 2, 7
- [2] 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 *CVPR*, pages 11618–11628, 2020. 2, 3, 6, 7, 8, 11
- [3] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: delving into high quality object detection. In *CVPR*, pages 6154–6162, 2018. 8
- [4] Qi Chen, Lin Sun, Ernest Cheung, and Alan L. Yuille. Every view counts: Cross-view consistency in 3d object detection with hybrid-cylindrical-spherical voxelization. In *NeurIPS*, 2020. 8
- [5] Qi Chen, Lin Sun, Zhixin Wang, Kui Jia, and Alan L. Yuille. Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots. In *ECCV*, volume 12366, pages 68–84, 2020. 8
- [6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, pages 6526–6534, 2017. 7
- [7] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *CVPR*, pages 11027–11036, 2020. 2, 3
- [8] Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, pages 3075–3084, 2019. 2
- [9] Ruihang Chu, Yukang Chen, Tao Kong, Lu Qi, and Lei Li. Icm-3d: Instantiated category modeling for 3d instance segmentation. *IEEE RAL*, 7(1):57–64, 2021. 3
- [10] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, pages 764–773, 2017. 2, 3
- [11] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel R-CNN: towards high performance voxel-based 3d object detection. In *AAAI*, pages 1201–1209, 2021. 1, 2, 3, 5, 6, 7, 8, 11, 12
- [12] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *ICCV*, pages 6568–6577, 2019. 3, 12
- [13] Hang Gao, Xizhou Zhu, Stephen Lin, and Jifeng Dai. Deformable kernels: Adapting effective receptive fields for object deformation. In *ICLR*, 2020. 2
- [14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *Int. J. Robotics Res.*, 32(11):1231–1237, 2013. 2, 3, 5, 6, 8, 11, 12, 13, 14
- [15] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, pages 9224–9232, 2018. 1, 4
- [16] JunYoung Gwak, Christopher B. Choy, and Silvio Savarese. Generative sparse detection networks for 3d single-shot object detection. In *ECCV*, volume 12349, pages 297–313, 2020. 3
- [17] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *CVPR*, pages 11870–11879, 2020. 3, 7
- [18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5
- [19] Tengpeng Huang, Zhe Liu, Xiwu Chen, and Xiang Bai. Epnnet: Enhancing point features with image semantics for 3d object detection. In *ECCV*, volume 12360, pages 35–52, 2020. 3, 7
- [20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, volume 37, pages 448–456, 2015. 5, 12
- [21] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L. Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IROS*, pages 1–8, 2018. 7
- [22] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, pages 12697–12705, 2019. 7, 8
- [23] Ziyu Li, Yuncong Yao, Zhibin Quan, Wankou Yang, and Jin Xie. Sienet: Spatial information enhancement network for 3d object detection from point cloud. *CoRR*, abs/2103.15396, 2021. 3
- [24] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, pages 7345–7353, 2019. 7
- [25] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, volume 11220, pages 663–678, 2018. 3
- [26] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *T-PAMI*, 42(2):318–327, 2020. 5, 12
- [27] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall F. Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *CVPR*, pages 806–814, 2015. 5
- [28] Jiageng Mao, Minzhe Niu, Haoyue Bai, Xiaodan Liang, Hang Xu, and Chunjing Xu. Pyramid R-CNN: towards better performance and adaptability for 3d object detection. In *ICCV*, 2021. 3, 7, 8
- [29] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *ICCV*, 2021. 3, 7, 8
- [30] Su Pang, Daniel D. Morris, and Hayder Radha. Clocs: Camera-lidar object candidates fusion for 3d object detection. In *IROS*, pages 10386–10393, 2020. 7
- [31] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. In *CVPR*, pages 918–927, 2018. 7
- [32] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 77–85, 2017. 3

- [33] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017. [1](#)
- [34] Mengye Ren, Andrei Pokrovsky, Bin Yang, and Raquel Urtasun. Sbnnet: Sparse blocks network for fast inference. In *CVPR*, pages 8711–8720, 2018. [2](#), [3](#)
- [35] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NeurIPS*, pages 91–99, 2015. [3](#)
- [36] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: point-voxel feature set abstraction for 3d object detection. In *CVPR*, pages 10526–10535, 2020. [1](#), [2](#), [3](#), [5](#), [6](#), [7](#), [8](#), [11](#), [12](#), [13](#), [14](#)
- [37] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, pages 770–779, 2019. [1](#), [3](#), [7](#)
- [38] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *T-PAMI*, 43(8):2647–2664, 2021. [7](#)
- [39] Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik G. Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. In *CVPR*, pages 11166–11175, 2019. [2](#)
- [40] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Etinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, pages 2443–2451, 2020. [3](#), [5](#), [13](#)
- [41] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, pages 6410–6419, 2019. [2](#)
- [42] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *CVPR*, pages 2317–2326, 2020. [2](#)
- [43] Sourabh Vora, Alex H. Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *CVPR*, pages 4603–4611, 2020. [2](#), [3](#), [8](#)
- [44] Chunwei Wang, Chao Ma, Ming Zhu, and Xiaokang Yang. Pointaugmenting: Cross-modal augmentation for 3d object detection. In *CVPR*, pages 11794–11803, 2021. [3](#), [5](#), [8](#), [11](#), [13](#)
- [45] Zhenda Xie, Zheng Zhang, Xizhou Zhu, Gao Huang, and Stephen Lin. Spatially adaptive inference with stochastic feature sampling and interpolation. In *ECCV*, volume 12346, pages 531–548, 2020. [2](#), [3](#)
- [46] Qiangeng Xu, Yin Zhou, Weiyue Wang, Charles R. Qi, and Dragomir Anguelov. SPG: unsupervised domain adaptation for 3d object detection via semantic point generation. *CoRR*, abs/2108.06709, 2021. [3](#)
- [47] Shaoqing Xu, Dingfu Zhou, Jin Fang, Junbo Yin, Bin Zhou, and Liangjun Zhang. Fusionpainting: Multimodal fusion with adaptive attention for 3d object detection. In *ITSC*, pages 3047–3054, 2021. [8](#)
- [48] Yan Yan, Yuxing Mao, and Bo Li. SECOND: sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. [1](#), [3](#), [5](#), [11](#)
- [49] Brandon Yang, Gabriel Bender, Quoc V. Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS*, pages 1305–1316, 2019. [2](#), [3](#)
- [50] Zetong Yang, Li Jiang, Yanan Sun, Bernt Schiele, and Jiaya Jia. A unified query-based paradigm for point cloud understanding. *CoRR*, abs/2203.01252, 2022. [2](#)
- [51] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *CVPR*, pages 11037–11045, 2020. [1](#), [3](#), [8](#)
- [52] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. STD: sparse-to-dense 3d object detector for point cloud. In *ICCV*, pages 1951–1960, 2019. [7](#)
- [53] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. In *CVPR*, pages 11784–11793, 2021. [1](#), [2](#), [3](#), [5](#), [6](#), [8](#), [11](#), [12](#), [13](#)
- [54] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Multi-modal virtual point 3d detection. In *NeurIPS*, 2021. [8](#)
- [55] Jin Hyeok Yoo, Yecheol Kim, Ji Song Kim, and Jun Won Choi. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. In *ECCV*, volume 12372, pages 720–736, 2020. [3](#), [7](#), [8](#)
- [56] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: point completion network. In *3DV*, pages 728–737, 2018. [3](#)
- [57] Wenwei Zhang, Zhe Wang, and Chen Change Loy. Exploring data augmentation for multi-modality 3d object detection. *CoRR*, abs/2012.12741, 2021. [5](#)
- [58] Yanan Zhang, Di Huang, and Yunhong Wang. PC-RGNN: point cloud completion and graph neural network for 3d object detection. In *AAAI*, pages 3430–3437, 2021. [3](#)
- [59] Xin Zhao, Zhe Liu, Ruolan Hu, and Kaiqi Huang. 3d object detection using scale invariant and feature reweighting networks. In *AAAI*, pages 9267–9274, 2019. [7](#)
- [60] Wu Zheng, Weiliang Tang, Sijin Chen, Li Jiang, and Chi-Wing Fu. CIA-SSD: confident iou-aware single-stage object detector from point cloud. In *AAAI*, pages 3555–3562, 2021. [3](#)
- [61] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. SE-SSD: self-ensembling single-stage object detector from point cloud. In *CVPR*, pages 14494–14503, 2021. [3](#)
- [62] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, pages 4490–4499, 2018. [3](#), [11](#), [12](#)
- [63] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *CoRR*, abs/1908.09492, 2019. [8](#)
- [64] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets V2: more deformable, better results. In *CVPR*, pages 9308–9316, 2019. [2](#), [3](#)

Table S - 12. Comparisons to Voxel R-CNN in R40 on KITTI *val*.

Method	AP _{BEV}			AP _{3D}		
	Easy	Mod.	Hard	Easy	Mod.	Hard
V. [11]	95.52	91.25	88.99	92.38	85.29	82.86
Ours	95.45	91.51	91.21	92.86	85.85	85.29

Appendix

A. More Implementation Details

Our implementation is based on the open-sourced OpenPCDet [11, 36], and the released code of CenterPoint [53].

A.1. Voxelization

KITTI. The 3D object detectors in this work convert point clouds into voxels as input data. On the KITTI [14] dataset, the range of point clouds is clipped into [0, 70.4m] for X axis, [-40m, 40m] for Y axis, and [-3, 1]m for Z axis. The voxelization size for input is (0.05m, 0.05m, 0.1m).

nuScenes. On the nuScenes [2], the detection range is set to [-54m, 54m] for both X and Y axes, and [-5m, 3m] for the Z axis. The voxel size is set as (0.075m, 0.075m, 0.2m).

A.2. Data Augmentations

KITTI. On the KITTI [14] dataset, data transformation and augmentations include random flipping, global scaling, global rotation, and ground-truth (GT) sampling [48]. The random flipping is conducted along the X axis. The global scaling factor is sampled from 0.95 to 1.05. The global rotation is conducted around the Z axis. The rotation angle is sampled from -45° and 45° . The ground-truth sampling is to copy-paste some new objects from other scenes to the current training data, which enriches objects in the environments. For the multi-modal setting, we do not transform images with the corresponding operations, except ground-truth sampling. We copy-paste the corresponding image crops from other scenes onto the current training images.

nuScenes. On the nuScenes [2] dataset, data augmentations includes random flipping, global scaling, global rotation, GT sampling [48], and an additional translation. The random flipping is conducted along both X and Y axes. The rotation angle is also randomly sampled in $[-45^\circ, 45^\circ]$. The global scaling factor is sampled in [0.9, 1.1]. The translation noise is conducted on all three axes, X , Y , and Z , with a factor independently sampled from 0 to 0.5. We also conduct the corresponding point-image GT sampling on the nuScenes. GT sampling is disabled in the last 4 epochs [44] for performance enhancement.

Table S - 13. Objective loss weight in AP_{3D} (R40) on KITTI *val*.

Method	Weight	Car			Ped.	Cyc.
		Easy	Mod.	Hard	Mod.	Mod.
Baseline	–	92.10	84.36	82.48	54.49	70.38
Focals Conv	0.1	91.59	84.63	82.42	60.62	71.33
	0.5	92.10	85.16	83.12	63.74	69.56
	1.0	92.32	85.19	82.62	61.61	72.76
	2.0	91.73	84.61	82.38	54.09	71.34

Table S - 14. Improvements upon CenterPoint on Waymo $\frac{1}{5}$.

Method	AP LEVEL 1			AP LEVEL 2		
	Veh.	Ped.	Cyc.	Veh.	Ped.	Cyc.
Baseline	70.9	71.5	69.1	62.8	63.5	66.5
Focals Conv	72.2	72.6	71.1	64.1	64.6	68.5

A.3. Training Settings

KITTI. For model training on the KITTI dataset, *i.e.*, PV-RCNN [36] and Voxel R-CNN [11], we train the network for 80 epochs with the batch size 16. We adopt the Adam optimizer. The learning rate is set as 0.01 and decreases in the cosine annealing strategy. The weight decay is set as 0.01. The momentum is set as 0.9. The gradient norms of training parameters are clipped by 10.

nuScenes. For models trained on the nuScenes datasets, *i.e.*, CenterPoint [53], we also train the network for 20 epochs with batch size 32. They are also trained with the Adam optimizer. The learning rate is initialized as $1e-3$ and decreases in the cosine annealing strategy to $1e-4$. The weight decay is set as 0.01. The gradient norms of training parameters are clipped by 35.

B. Backbone Networks

We illustrate the structure of the backbone networks in Fig. S - 4. In this illustration, *Reg block* and *Subm block* mean the regular sparse convolutional block and the sub-manifold sparse convolutional block, respectively. The backbone networks are based on VoxelNet [62]. It contains a stem layer and 4 stages. In the last three stages, Stage 1, 2, and 3, there a regular sparse convolutional block with stride as 2 for down-sampling. There are some detailed differences among different frameworks, as the following.

B.1. Architecture settings

PV-RCNN and Voxel R-CNN. In the backbones of PV-RCNN [36] and Voxel R-CNN [11], the channels for the stem and stages, $\{c_0, c_1, c_2, c_3, c_4\}$, are $\{16, 16, 32, 64, 64\}$. The numbers of *Subm blocks* in these stages, $\{n_1, n_2, n_3, n_4\}$, are $\{1, 2, 2, 2\}$. A *Reg* or *Subm block* is a

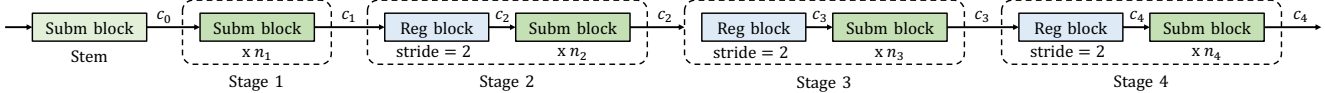


Figure S - 4. The illustration of the VoxelNet [62] backbone networks of PV-RCNN [36], Voxel R-CNN [11], and CenterPoint [53]. $\{c_0, c_1, c_2, c_3, c_4\}$ represent the output channels of the Stem, Stage 1, 2, 3, and 4. $\{n_1, n_2, n_3, n_4\}$ means the numbers of repeated submanifold blocks in these stages. In our approach, for the LIDAR-only setting, focal sparse convolutions (Focals Conv) are used in the last layer of Stage 1, 2, and 3. For the multi-modal setting, the focal sparse convolution with fusion (Focals Conv - F) is used in the last layer of Stage 1.

Table S - 15. Improvements over CenterPoint on the nuScenes *val* split.

Method	mAP	NDS	Car	Truck	Bus	Trailer	C.V.	Ped	Mot	Byc	T.C.	Bar
CenterPoint	59.0	66.4	85.6	57.2	71.2	37.3	16.2	85.1	58.4	41.0	69.2	68.2
Focals Conv	61.2	68.1	86.6	60.2	72.3	40.8	20.1	86.2	61.3	45.6	70.2	69.3
Focals Conv-F	63.9	69.4	86.5	58.5	72.4	41.2	23.9	86.0	69.0	55.2	76.9	69.1
- GT-S. last 4	65.6	70.4	87.1	62.1	72.7	42.6	27.1	86.5	72.4	61.0	78.1	66.3
+ Flip testing	67.1	71.5	87.7	62.9	72.4	42.6	28.1	87.8	74.4	65.5	78.9	70.1

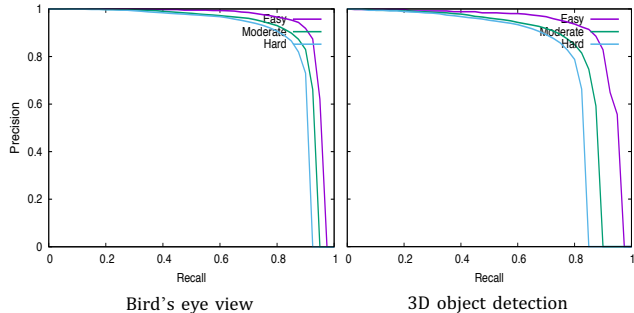


Figure S - 5. PR curves of Focals Conv-F on KITTI *test*.

conv-bn-relu layer, which includes a regular or submanifold convolution, a batch normalization layer [20], and a ReLU activation.

CenterPoint. In the backbone network of the CenterPoint [53] detector, the backbone network is larger. The channels for the stem and stages, $\{c_0, c_1, c_2, c_3, c_4\}$, equal to $\{16, 16, 32, 64, 128\}$. The numbers of repeated *Subm blocks* in these stages, $\{n_1, n_2, n_3, n_4\}$, are $\{2, 2, 2, 2\}$. Compared to that in the PV-RCNN [36] and Voxel R-CNN [11] detectors, the *Subm block* is more complicated in this backbone network. Except the stem, it contains two sequential conv-bn-relu layers, with a residual connection.

B.2. Focal Sparse Convolution Usage

In our approach, the above architecture-level settings are directly inherited from the original PV-RCNN [36], Voxel R-CNN [11], and CenterPoint [12] frameworks, without any adjustment, for a fair comparison. In the LIDAR-only task, we insert the *Focals Conv* in the last layer of Stage

1, 2, and 3. In the multi-modal task, we insert the *Focals Conv - F* only at the last layer of Stage 1. This relieves the efficiency and memory issues caused by the RGB feature extraction. Note that, in the CenterPoint [53] detectors, it is also used in the last *layer*, not the total *block*. In other words, although there are two conv-bn-relu layers in each *Subm block* in CenterPoint [53], we only apply it as the last layer. For simplification, we do not double it as a block.

C. Additional Experiments

C.1. Results on Bird's Eye View on KITTI

We report the accuracy for 3D object detection and Bird's Eye View (BEV) of Focals Conv-F upon Voxel R-CNN [11] on the KITTI [14] dataset in Tab. S - 12. The results are calculated by recall 40 positions with the IoU threshold of 0.7. It performs better than the strong Voxel R-CNN [11] baseline on both AP_{BEV} and AP_{3D} in moderate and hard cases. We also provide the Precision-Recall (PR) curves of Focals Conv-F on KITTI *test* split in Fig. S - 5.

C.2. Objective Loss Weight

The training of the focal sparse convolutional networks involves the objective loss function. We implement it as a focal loss [26] as in Eq (9).

$$L_{obj} = \frac{1}{|N|} \sum_{i \in N} -(1 - \bar{p}_i)^\gamma \log(\bar{p}_i). \quad (9)$$

where $i \in N$ enumerates all sparse features in the current feature space. Following the original focal loss [26], we directly set $\gamma = 2$ and it works well. For notational convenience,

Table S - 16. Ablations on ground-truth sampling on nuScenes $\frac{1}{4}$. GT Sampl. - Ground-truth Sampling. Trans. - Transformations.

Fusion	GT sampl.	Trans.	mAP	Car	Truck	Bus	Trail.	C.V.	Ped	Mot	Byc	T.C.	Bar
\times	\checkmark	\times	39.3	70.8	31.0	49.0	20.3	3.5	73.7	28.8	13.4	49.3	50.6
\checkmark			43.3	70.8	32.5	47.9	21.0	6.7	72.6	44.9	31.8	58.4	49.1
\times	\times	\checkmark	54.6	80.4	51.9	60.6	31.5	14.2	81.4	57.4	45.4	63.1	60.7
\checkmark			59.0	83.2	56.1	61.5	36.6	19.7	84.3	59.1	49.4	73.8	66.3

Table S - 17. Ablations on voxel size upon Focals Conv-F on the nuScenes *val* split.

Voxel size (m)	mAP	NDS	Car	Truck	Bus	Trailer	C.V.	Ped	Mot	Byc	T.C.	Bar
(0.05, 0.05, 0.2)	65.6	70.2	85.9	61.6	70.1	35.9	25.0	88.3	74.0	66.1	79.4	70.1
(0.075, 0.075, 0.2)	67.1	71.5	87.7	62.9	72.4	42.6	28.1	87.8	74.4	65.5	78.9	70.1
(0.1, 0.1, 0.2)	66.5	71.4	87.5	62.1	71.8	44.6	27.9	86.9	74.3	63.6	77.4	69.0
(0.125, 0.125, 0.2)	66.6	70.9	87.0	61.1	74.0	44.1	30.2	85.6	75.1	63.8	75.1	69.6
(0.15, 0.15, 0.2)	65.3	70.2	86.9	60.9	72.8	45.1	30.0	83.3	70.4	63.5	72.7	67.9

nience, we define \bar{p}_i as follow

$$\bar{p}_i = \begin{cases} p_i, & \text{if } y_i = 1, \\ 1 - p_i, & \text{otherwise,} \end{cases} \quad (10)$$

where $p_i \in [0, 1]$ is the estimated probability for the class with label $y_i = 1$. It is the estimation that whether the feature i contributes any foreground objects.

We analyze the loss weight for this objective loss in Tab. S - 13. This ablation study is conducted upon the PV-RCNN [36] detector on the KITTI [14] datasets. The results on AP_{3D} with 40 recall positions are reported as the metric. We change the loss weight values from {0.1, 0.5, 1.0, 2.0}. It shows that too large or too small loss weight values degrade the results. Loss weights 0.5 and 1.0 present competitive performance. We remain the 1.0 loss weight as a default setting for simplification.

C.3. Improvements on the Waymo Open Dataset.

To show our generalization capacity, we conduct further experiments on Waymo dataset. We use $\frac{1}{5}$ training data, following the default setting in the OpenPCdet codebase. As shown in Tab. S - 14, Focals Conv also brings non-trivial improvements on the Waymo [40] dataset.

C.4. Improvements on the nuScenes *val* split.

Tab. S - 15 presents the improvements over CenterPoint [53] on the nuScenes *val* split. The CenterPoint baseline in Tab. S - 15 is re-implemented in the same settings to Focals Conv and Focals Conv-F. It shows that both Focals Conv and Focals Conv-F bring non-trivial improvements. Notably, Focals Conv-F improves the plain CenterPoint [53] by 4.9% mAP on the nuScenes *val* split. We further apply some tricks for performance enhancement, *e.g.*,

<https://github.com/open-mmlab/OpenPCDet>

disabling ground-truth sampling in the last 4 epochs [44] and double-flip testing [53].

C.5. Accuracy loss on some categories after fusion.

A surprising case is that the multi-modal fusion make the performance stay the same or worse on some popular categories, *e.g.*, Car, Ped, Bar (from *Focals Conv* to *Focals Conv - F* in Tab. 11). The improvements over the baseline are consistent on all categories. To analyze this special case, we conduct ablations on augmentations on CenterPoint and the nuScenes $\frac{1}{4}$ training set. We find *ground-truth sampling* (GT Sampl.) is the keypoint. As in Tab. A - 16, when GT Sampl. is used, the performance on some popular categories (*e.g.*, Car, Bus, Ped, Bar) stays the same or worse. In contrast, when we disable GT Sampl. and apply all other transformations (flip, rotation, re-scaling, and translation), all categories are benefited from the fusion. We suppose that this is from the image-level copy-paste in GT Sampl. When other objects are pasted onto images, popular objects inevitably have more chance to be covered by the pasted, which degrades the performance on these categories.

C.6. Ablations on Voxel Size.

We ablate the effects of different voxel sizes upon Focals Conv-F on the nuScenes *val* split in Tab. S - 17. We change the voxel sizes in X and Y axes from 0.05m to 0.15m, with the interval 0.025m. The overall mAP achieves the best performance at the voxel size (0.075, 0.075, 0.2)m. However, the proper voxel sizes vary across different classes. This phenomenon deserves further analysis or a dynamic mechanism design in the future.

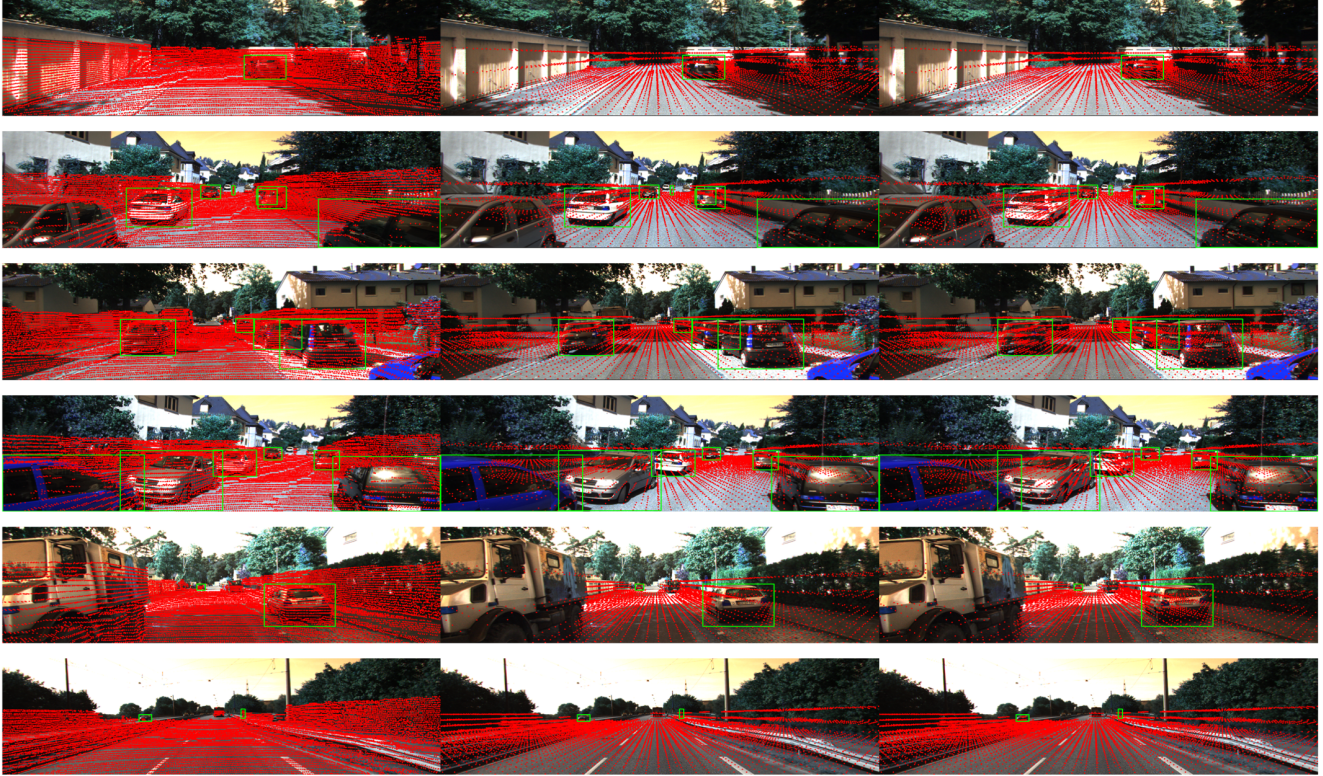


Figure S - 6. The illustration of visual comparisons between the plain and the focal sparse convolutional backbone networks.

D. Visualizations

We provide additional visual comparisons between the plain network and the focal sparse convolutional networks in Fig. S - 6. It shares the same settings to the Fig. 2 in the paper. These visualizations are based on the PV-RCNN [36] detectors and on the KITTI [14] dataset. In each visualization group, the top figure is the distribution of input voxels. The middle and the bottom figures are from the plain and the focal sparse convolutional networks, respectively. We project the coordinate centers of the output voxel features from the backbone networks onto the 2D image plane. The projection is based on the calibration matrices of KITTI [14].