LION: Linear Group RNN for 3D Object Detection in Point Clouds

Zhe Liu[1*] , **Jinghua Hou**[1*] , **Xinyu Wang**[1*] **Xiaoqing Ye**[3] , **Jingdong Wang**[3] , **Hengshuang Zhao**[2] , **Xiang Bai**[1†]

[1]Huazhong University of Science and Technology
[2]The University of Hong Kong   [3]Baidu Inc., China
https://happinesslz.github.io/projects/LION

Abstract

The benefit of transformers in large-scale 3D point cloud perception tasks, such as 3D object detection, is limited by their quadratic computation cost when modeling long-range relationships. In contrast, linear RNNs have low computational complexity and are suitable for long-range modeling. Toward this goal, we propose a simple and effective window-based framework built on **LI**near gr**O**up RNN (*i.e.*, perform linear RNN for grouped features) for accurate 3D object detection, called **LION**. The key property is to allow sufficient feature interaction in a much larger group than transformer-based methods. However, effectively applying linear group RNN to 3D object detection in highly sparse point clouds is not trivial due to its limitation in handling spatial modeling. To tackle this problem, we simply introduce a 3D spatial feature descriptor and integrate it into the linear group RNN operators to enhance their spatial features rather than blindly increasing the number of scanning orders for voxel features. To further address the challenge in highly sparse point clouds, we propose a 3D voxel generation strategy to densify foreground features thanks to linear group RNN as a natural property of auto-regressive models. Extensive experiments verify the effectiveness of the proposed components and the generalization of our LION on different linear group RNN operators including Mamba, RWKV, and RetNet. Furthermore, it is worth mentioning that our LION-Mamba achieves state-of-the-art on Waymo, nuScenes, Argoverse V2, and ONCE datasets. Last but not least, our method supports kinds of advanced linear RNN operators (*e.g.*, RetNet, RWKV, Mamba, xLSTM and TTT) on small but popular KITTI dataset for a quick experience with our linear RNN-based framework.

1 Introduction

3D object detection serves as a fundamental technique in 3D perception and is widely used in navigation robots and self-driving cars. Recently, transformer-based [56] feature extractors have made significant progress in general tasks of Natural Language Processing (NLP) and 2D vision by flexibly modeling long-range relationships. To this end, some researchers have made great efforts to transfer the success of transformers to 3D object detection. Specifically, to reduce the computation costs, SST [15] and SWFormer [53] divide point clouds into pillars and implement window attention for pillar feature interaction in a local 2D window. Considering some potential information loss of the pillar-based manners along the height dimension, DSVT-Voxel [60] further adopts voxel-based formats and implements set attention for voxel feature interaction in a limited group size.

---

[*]Equal contribution.
[†]Corresponding author.
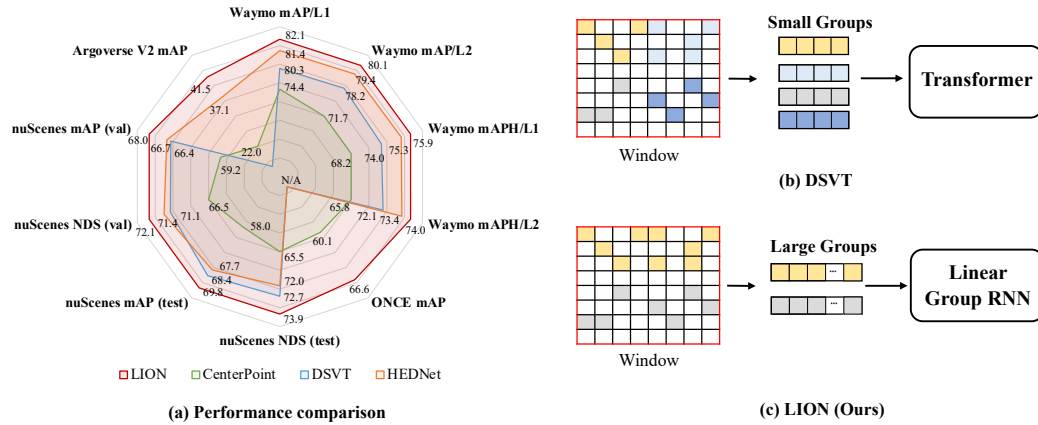
https://doi.org/10.52202/079017-0435

Figure 1: (a) Comparison of different 3D backbones in terms of detection performance on Waymo [52], nuScenes [4], Argoverse V2 [62] and ONCE [37] datasets. Here, we adopt Mamba [22] as the default operator of our LION. Besides, we present the simplified schematic of DSVT (b) [60] and our LION (c) for implementing feature interaction in 3D backbones.

Although the above methods have achieved some success in 3D detection, they perform self-attention for pillar or voxel feature interaction with only a small group size due to computational limitations, locking the potential of transformers for modeling long-range relationships. Moreover, it is worth noting that modeling long-range relationships can benefit from large datasets, which will be important for achieving foundational models in 3D perception tasks in the future. Fortunately, in the field of large language models (LLM) and 2D perception tasks, some representative linear RNN operators such as Mamba [22] and RWKV [40] with linear computational complexity have achieved competitive performance with transformers, especially for long sequences. Therefore, a question naturally arises: can we perform long-range feature interaction in larger groups at a lower computation cost based on linear RNNs in 3D object detection?

To this end, we propose a window-based framework based on **LI**near gr**O**up R**N**N (*i.e.*, perform linear RNN for grouped features in a window-based framework) termed **LION** for accurate 3D object detection in point clouds. Different from the existing method DSVT (b) in Figure 1, our LION (c) could support thousands of voxel features to interact with each other in a large group for establishing the long-range relationship. Nevertheless, effectively adopting linear group RNN to construct a proper 3D detector in highly sparse point cloud scenes remains challenging for capturing the spatial information of objects. Concretely, linear group RNN requires sequential features as inputs. However, converting voxel features into sequential features may result in the loss of spatial information (*e.g.*, two features that are close in 3D spatial position might be very far in this 1D sequence). Therefore, we propose a simple 3D spatial feature descriptor and decorate the linear group RNN operators with it, thus compensating for the limitations of linear group RNN in 3D local spatial modeling.

Furthermore, to enhance feature representation in highly sparse point clouds, we present a new 3D voxel generation strategy based on linear group RNN to densify foreground features. A common manner of addressing this is to add an extra branch to distinguish the foregrounds, as seen in previous methods [53, 17, 70]. However, this solution is relatively complex and rarely used in 3D backbone due to its lack of structural elegance. Instead, we simply choose the high response of the feature map in the 3D backbone as the areas for voxel generation. Subsequently, the auto-regressive property of linear group RNN can be effectively employed to generate voxel features.

Finally, as shown in Figure 1 (a), we compare LION with the existing representative methods. We can clearly observe that our LION achieves state-of-the-art on a board autonomous datasets in terms of detection performance. To summarize, our contributions are as follows: **1)** We propose a simple and effective window-based 3D backbone based on the linear group RNN named LION to allow long-range feature interaction. **2)** We introduce a simple 3D spatial feature descriptor and integrate it with the linear group RNN, compensating for the lack of capturing 3D local spatial information. **3)** We provide a new 3D voxel generation strategy to densify foreground features, producing a more discriminative feature representation in highly sparse point clouds. **4)** We verify the generalization of our LION with different linear group RNN mechanisms (*e.g.*, Mamba, RWKV, RetNet). In particular,
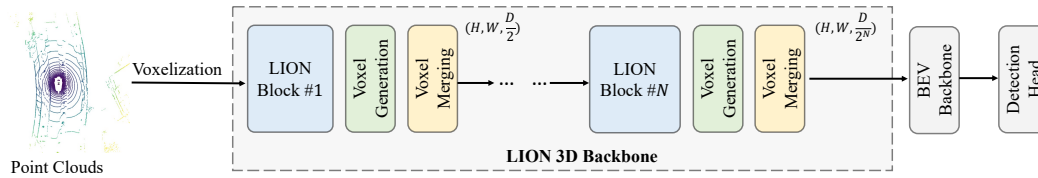
13602

Figure 2: The illustration of LION, which mainly consists of N LION blocks, each paired with a voxel generation for feature enhancement and a voxel merging for down-sampling features along the height dimension. (H, W, D) indicates the shape of the 3D feature map, where H, W, and D are the length, width, and height of the 3D feature map along the X-axis, Y-axis, and Z-axis. N is the number of LION blocks. In LION, we first convert point clouds to voxels and partition these voxels into a series of equal-size groups. Then, we feed these grouped features into LION 3D backbone to enhance their feature representation. Finally, these enhanced features are fed into a BEV backbone and a detection head for final 3D detection.

our LION-Mamba achieves state-of-the-art on challenging Waymo [52], nuScenes [4], Argoverse V2 [62], and ONCE [37] dataset, which further illustrates the superiority of LION.

2 Related Work

3D Object Detection in Point Clouds. 3D object detectors in point clouds can be roughly divided into point-based and voxel-based. For point-based methods [6, 68, 42, 10, 32, 39, 24, 50, 73, 67, 43, 65, 5], they usually sample point clouds and adopt point encoder [44, 45] to directly extract point features. However, the point sampling and grouping utilized by point-based methods is time-consuming. To avoid these problems, voxel-based methods [13, 12, 34, 48, 49, 51, 23, 59, 63, 69, 64, 70] convert the input irregular point clouds into regular 3D voxels and then extract 3D features by 3D sparse convolution. Although these methods achieve promising performance, they are still limited by the local receptive field of 3D convolution. Therefore, some methods [8, 36] adopt the large kernel to enlarge the receptive field and achieve better performance.

Linear RNN. Recurrent Neural Networks (RNNs) are initially developed to address problems in Natural Language Processing (NLP), such as time series prediction and speech recognition, by effectively capturing temporal dependencies in sequential data. Recently, to overcome the quadratic computational complexity of transformers, significant advancements have been made in time-parallelizable data-dependent RNNs (called linear RNNs in this paper) [46, 38, 40, 41, 55, 11, 66, 22, 54, 3]. These models retain linear complexity while offering efficient parallel training capabilities, allowing their performance to match or even surpass that of transformers. Due to their scalability and efficiency, linear RNNs are poised to play an increasingly important role in various fields and some works [1, 14, 29, 72] have applied linear RNNs to 2D/3D vision filed. In this paper, we aim to further extend linear RNNs to 3D object detection tasks thanks to their long-range relationship modeling capabilities.

Transformers in 3D Object Detection. Transformer [56] has achieved great success in many tasks, motivating numerous works to adopt attention mechanisms in 3D object detection to achieve better performance. However, the application of transformers is non-trivial in large-scale point clouds. Many works [16, 53, 35, 60] apply transformers to extract features by partitioning pillars or voxels into several groups based on local windows. Although these approaches achieve promising performance, they usually adopt small groups for feature interaction due to the quadratic computational complexity of transformers, hindering them from capturing long-range dependencies in 3D space. In contrast, we propose a simple and effective framework based on linear RNNs named LION to achieve long-range feature interaction for accurate 3D object detection thanks to their linear computational complexity.

3 Method

Due to computational limitations, some transformer-based methods [15, 60, 35] usually convert features into pillars or group small size of voxel features to interact with each other within small groups, limiting the advantages of transformers in long-range modeling. More recently, some linear RNN operators [22, 40, 55] that maintain linear complexity with the length of the input sequence
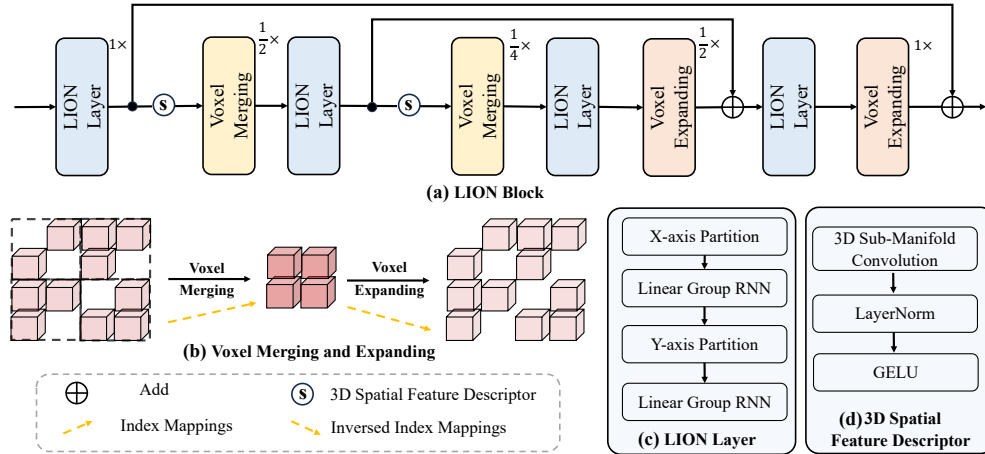
Figure 3: (a) shows the structure of LION block, which involves four LION layers, two voxel merging operations, two voxel expanding operations, and two 3D spatial feature descriptors. Here, $1\times$, $\frac{1}{2}\times$, and $\frac{1}{4}\times$ indicate the resolution of 3D feature map as (H, W, D), $(H/2, W/2, D/2)$ and $(H/4, W/4, D/4)$, respectively. (b) is the process of voxel merging for voxel down-sampling and voxel expanding for voxel up-sampling. (c) presents the structure of LION layer. (d) shows the details of the 3D spatial feature descriptor.

are proposed to model long-range feature interaction. More importantly, the linear RNN operators such as Mamba [22] and RWKV [40] have even shown comparable performance with transformers in LLM thanks to their low computation cost in long-range feature interaction. This further motivates us to adopt linear RNNs to construct a 3D detector for long-range modeling.

3.1 Overview

In this paper, we propose a simple and effective window-based framework based on **LI**near gr**O**up RNN (*i.e.*, perform linear RNN for grouped features in a window-based framework) named **LION**, which can group thousands of voxels (dozens of times more than the number of previous methods [15, 60, 35]) for feature interaction. The pipeline of our LION is presented in Figure 2. LION consists of a 3D backbone, a BEV backbone, and a detection head, maintaining a consistent pipeline with most voxel-based 3D detectors [63, 60, 69]. In this paper, our contribution lies in the design of the 3D backbone based on linear group RNN, which will be introduced in the following.

3D Sparse Window Partition. Our LION is a window-based 3D detector. Thus, before feeding voxel features into our LION block, we need to implement a 3D sparse window partition to group them for feature interaction. Specifically, we first convert point clouds into voxels with the total number of L. Then, we divide these voxels into non-overlapping 3D windows with the shape of (T_x, T_y, T_z), where T_x, T_y and T_z denote the length, width, and height of the window along the X-axis, Y-axis, and Z-axis. Next, we sort voxels along the X-axis for the X-axis window partition and along the Y-axis for the Y-axis window partition, respectively. Finally, to save computation cost, we adopt the equal-size grouping manner in FlatFormer [35] instead of the classic equal-window grouping manner in SST [15]. That is, we partition sorted voxels into groups with equal size K rather than windows of equal shapes for feature interaction. Due to the quadratic computational complexity of transformers, previous transformer-based methods [15, 60, 35] only achieve feature interaction using a small group size. In contrast, we adopt a much larger group size K to obtain long-range feature interaction thanks to the linear computational complexity of the linear group RNN operators.

3.2 LION Block

The LION block is the core component of our approach, which involves LION layer for long-range feature interaction, 3D spatial feature descriptor for capturing local 3D spatial information, voxel merging for feature down-sampling and voxel expanding for feature up-sampling, as shown in Figure 3 (a). Besides, LION block is a hierarchical structure to better extract multi-scale features due to the gap of different 3D objects in size. Next, we introduce each part of LION block.
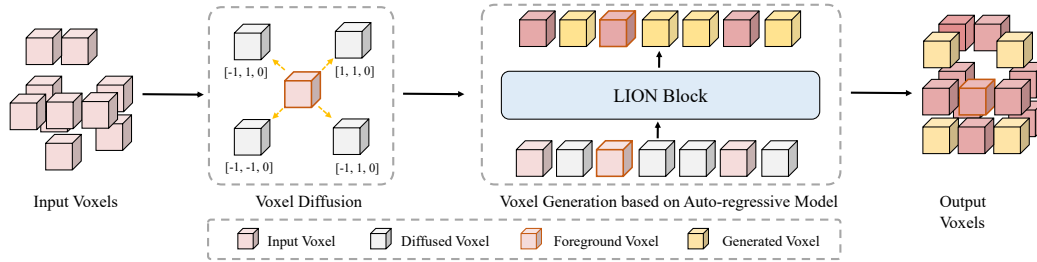
Figure 5: The details of voxel generation. For input voxels, we first select the foreground voxels and diffuse them along different directions. Then, we initialize the corresponding features of the diffused voxels as zeros and utilize the auto-regressive ability of the following LION block to generate diffused features. Note that we do not present the voxel merging here for simplicity.

LION Layer. In LION block, we apply LION layer to model a long-range relationship among grouped features with the help of the linear group RNN operator. Specifically, as shown in Figure 3 (c), we provide the structure of LION layer, which is composed of two linear group RNN operators. The first one is used to perform long-range feature interaction based on the X-axis window partition and the second one can extract long-range feature information based on the Y-axis window partition. Taking advantage of two different window partitions, LION layer can obtain more sufficient feature interaction, producing more discriminative feature representation.

3D Spatial Feature Descriptor. Although linear RNNs have the advantages of long-range modeling with low computation cost, it is not ignorable that the spatial information might be lost when input voxel features are flattened into 1D sequential features. For example, as shown in Figure 4, there are two adjacent features (*i.e.*, indexed as 01 and 34) in 3D space. However, after they are flattened into 1D sequential features, the distance between them in 1D space is very far. We regard this phenomenon as a loss of 3D spatial information. To tackle this problem, an available manner is to increase the number of scan orders for voxel features such as VMamba [31]. However, the order of scanning is too hand-designed. Besides, as the scanning orders increase, the corresponding computation cost also increases significantly. Therefore, it is not appropriate in large-scale sparse 3D point clouds to adopt this manner. As shown in Figure 3 (d), we introduce a 3D spatial feature descriptor, which consists of a 3D sub-manifold convolution, a LayerNorm layer, and a GELU activation function. Naturally, we can leverage the 3D spatial feature descriptor
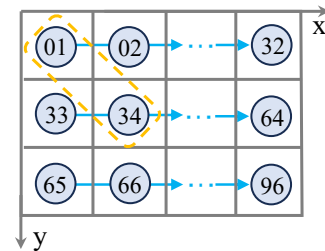

Figure 4: The illustration of spatial information loss when flattening into 1D sequences. For example, there are two adjacent voxels in spatial position (indexed as 01 and 34) but are far in the 1D sequences along the X order.

to provide rich 3D local position-aware information for the LION layer. Besides, we place the 3D spatial feature descriptor before the voxel merging to reduce spatial information loss in the process of voxel merging. We provide the corresponding experiment in our appendix.

Voxel Merging and Voxel Expanding. To enable the network to obtain multi-scale features, our LION adopts a hierarchical feature extraction structure. To achieve this, we need to perform feature down-sampling and up-sampling operations in highly sparse point clouds. However, it is worth mentioning that we cannot simply apply max or average pooling or up-sampling operations as in 2D images since 3D point clouds possess irregular data formats. Therefore, as shown in Figure 3 (b), we adopt voxel merging for feature down-sampling and voxel expanding for feature up-sampling in highly sparse point clouds. Specifically, for voxel merging, we calculate the down-sampled index mappings to merge voxels. In voxel expanding, we up-sample the down-sampled voxels by the corresponding inversed index mappings.

3.3 Voxel Generation

Considering the challenge of feature representation in highly sparse point clouds and the potential information loss of implementing voxel merging in Figure 2, we propose a voxel generation strategy to address these issues with the help of the auto-regressive capacity of the linear group RNN.

Distinguishing Foreground Voxels without Supervision. In voxel generation, the first challenge is identifying which regions of voxel features need to be generated. Different from previous methods [53, 17, 70] that employ some supervised information based on well-learned BEV features to obtain the foreground region for feature diffusion. However, these approaches may be unsuitable for a 3D backbone and may even compromise its elegance. Interestingly, inspired by [34, 30], we notice that the corresponding high values of feature responses along the channel dimension in the 3D backbone (Refer to Figure 6 in our appendix) are usually the foregrounds. Therefore, we compute the feature response F_i^* for the output feature F_i of the ith LION block, where $i = 1, 2, ..., N$ indicates the index of LION block in the 3D backbone. Thus, this above process can be formulated as:

$$F_i^* = \frac{1}{C} \sum_{j=0}^{C} F_i^j, \qquad (1)$$

where C is the channel dimension of F_i. Next, we sort the feature responses F_i^* in descending order and select the corresponding Top-m voxels as the foregrounds from the total number L of non-empty voxels, where $m = r * L$ and r is the ratio of foregrounds. This process can be computed as:

$$F_m = \text{Top}_m(F_i^*), \qquad (2)$$

where $\text{Top}_m(F_i^*)$ means selecting Top-m voxel features from F_i^*. F_m are the selected foreground features, which will serve for the subsequent voxel generation.

Voxel Generation with Auto-regressive Property. The previous method [53] adopts a K-NN manner to obtain generated voxel features based on their K-NN features, which might be sub-optimal to enhance feature representation due to the redundant features and the limited receptive field. Fortunately, the linear RNN is well-suited for auto-regressive tasks in addition to its advantage of handling long sequences. Therefore, we leverage the auto-regressive property of linear RNN to effectively generate the new voxel features by performing sufficient feature interaction with other voxel features in a large group. Specifically, for convenience, we define the corresponding coordinates of selected foreground voxel features F_m as P_m. As shown in Figure 5, we first obtain diffused voxels by diffusing P_m with four different offsets (*i.e.*, [-1,-1, 0], [1,1, 0], [1,-1, 0], and [-1,1, 0]) along the X-axis, Y-axis, and Z-axis, respectively. Then, we initialize the corresponding features of diffused voxels by all zeros. Next, we concatenate the output feature F_i of the ith LION block with the initialized voxel features, and feed them into the subsequent $(i + 1)^{th}$ LION block. Finally, thanks to the auto-regressive ability of the LION block, the diffused voxel features can be effectively generated based on other voxel features in large groups. This process can be formulated as:

$$F_p = F_i \oplus F_{[-1,-1,0]} \oplus F_{[1,1,0]} \oplus F_{[1,-1,0]} \oplus F_{[-1,1,0]}, \qquad (3)$$

$$F_p^{'} = \text{Block}(F_p), \qquad (4)$$

where $F_{[x,y,z]}$ denotes the initialized voxel features with diffused offsets of x, y, and z along the X-axis, Y-axis, and Z-axis. The $\oplus$ and Block denote the concatenation and LION block respectively.

4 Experiments

4.1 Datasets and Evaluation Metrics

Waymo Open Dataset. Waymo Open dataset (WOD) [52] is a well-known benchmark for large-scale outdoor 3D perception, comprising 1150 scenes which are divided into 798 scenes for training, 202 scenes for validation, and 150 scenes for testing. Each scene includes about 200 frames, covering a perception range of $150m \times 150m$. For evaluation metrics, WOD employs 3D mean Average Precision (mAP) and mAP weighted by heading accuracy (mAPH), each divided into two difficulty levels: L1 is for objects detected with more than five points and L2 is for those at least one point.

nuScenes Dataset. nuScenes [4] is a popular outdoor 3D perception benchmark with a perception range of up to 50 meters. Each frame in the scene is annotated with 2Hz. The dataset includes 1000 scenes, which is divided into 750 scenes for training, 150 scenes for validation, and 150 scenes for testing. nuScenes adopts mean Average Precision (mAP) and the NuScenes Detection Score (NDS) as evaluation metrics.

Argoverse V2 Dataset. Argoverse V2 [62] is a outdoor 3D perception benchmark with a long-range perception of up to 200 meters. It contains 1000 sequences in total, 700 for training, 150 for validation, and 150 for testing. Each frame in the scene is annotated with 10Hz. For the evaluation metric, Argoverse v2 adopts a similar mean Average Precision (mAP) metric with nuScenes [4].

4.2 Implementation Details

Network Architecture. In our LION, we provide three representative linear RNN operators (*i.e.*, Mamba [22], RWKV [40], and RetNet [55]). Each of operator adopts a bi-directional structure to better capture 3D geometric information inspired by [21]. On WOD, we keep the same channel dimension $C = 64$ for all LION blocks in LION-Mamba, LION-RWKV, and LION-RetNet. For the large version of LION-Mamba-L, we set $C = 128$. We follow DSVT-Voxel [60] to set the grid size as (0.32m, 0.32m, 0.1875m). The number of LION blocks N is set to 4. For these four LION blocks, the window sizes (T_x, T_y, T_z) are set to $(13, 13, 32)$, $(13, 13, 16)$, $(13, 13, 8)$, and $(13, 13, 4)$, and the corresponding group sizes K are 4096, 2048, 1024, 512, respectively. Besides, we adopt the same center-based detection head and loss function as DSVT [60] for fair comparison. In the voxel generation, we set the ratio $r = 0.2$ to balance the performance and computation cost. For the nuScenes dataset, we replace DSVT [60] 3D backbone with our LION 3D backbone except for changing the grid size to $(0.3m, 0.3m, 0.25m)$. For the Argoverse V2 dataset, we replace the 3D backbones of VoxelNext [9] or SAFDNet [70] with our LION 3D backbone except for setting the grid size to $(0.4m, 0.4m, 0.25m)$. Moreover, it is noted that we only add three extra LION layers to further enhance the 3D backbone features, rather than applying the BEV backbone to obtain the dense BEV features.

Training Process. On the WOD, we adopt the same point cloud range, data augmentations, learning rate, and optimizer as the previous method [60]. We train our model 24 epochs with a batch size of 16 on 8 NVIDIA Tesla V100 GPUs. Besides, we utilize the fade strategy [58] to achieve better performance in the last epoch. For the nuScenes dataset, we adopt the same point cloud range, data augmentations, and optimizer as previous method [2]. Moreover, we find that LION converges faster than previous methods on nuScenes dataset. Therefore, we only train our model for 36 epochs without CBGS [77]. The learning rate and batch size are set to 0.003 and 16, respectively. It is worth noting that the CBGS strategy extends training iterations about 4.5 times, which means that our training iterations are much fewer than previous methods [2] (*i.e.*, 20 epochs with CBGS). For the Argoverse V2 dataset, we adopt the same training process with SAFDNet [70] and SECOND [63], respectively.

4.3 Main Results

In this section, we provide a board comparison of our LION with existing methods on WOD, NuScenes and Argoverse V2 datasets for 3D object detection. Furthermore, in the section A.1 of our appendix, we present the experiment in ONCE dataset [37] and provide more types of linear RNN operators (*e.g.*, RetNet, RWKV, Mamba, xLSTM, and TTT) based on our LION framework for 3D detection on a small but popular dataset KITTI [20] for a quick experience.

Results on WOD. To illustrate the superiority of our LION, we provide the comparison with existing representative methods on the WOD in Table 1. Here, we also conduct the experiments on our LION with different linear group RNN operators, including LION-Mamba, LION-RWKV and LION-RetNet. Compared with the transformer-based methods [15, 53, 60], our LION with different linear group RNN operators outperforms the previous state-of-the-art (SOTA) transformer-based 3D backbone DSVT-Voxel [60], illustrating the generalization of our proposed framework. To further scale up our LION, we present the performance of LION-Mamba-L by doubling the channel dimension of LION-Mamba. It can be observed that LION-Mamba-L significantly outperforms DSVT-Voxel with 1.9 mAPH/L2, leading to a new SOTA performance. The above promising results effectively demonstrate the superiority of our proposed LION.

In Table 2, we also provide the results with multiple frames as inputs on the WOD *test* split. For three frames, our LION-L outperforms the representative method PillarNeXt [20] by 3.3 (77.4 vs. 74.1) mAPH/L2, which clearly illustrates the superiority of our methods.

Results on nuScenes. We also evaluate our LION on nuScenes *validation* and *test* set [4] further to verify the effectiveness of our LION. As shown in Table 3, on nuScenes *validation* set, our LION-RetNet, LION-RWKV, and LION-Mamba achieves 71.9, 71.7, and 72.1 NDS, respectively, which outperforms the previous advanced methods DSVT [60] and HEDNet [71]. Besides, our LION-Mamba even brings a new SOTA on nuScenes *test* benchmark, which beats the previous advanced method DSVT with 1.2 NDS and 1.4 mAP, illustrating the superiority of our LION. Note that all results of our LION are conducted without any test-time augmentation and model ensembling.

Table 1: Performances on the Waymo Open Dataset *validation* set (train with 100% training data). ‡ denotes the two-stage method. **Bold** denotes the best performance of all methods. "-L" means we double the dimension of channels in LION 3D backbone. RNN denotes the linear RNN operator. All results are presented with single-frame input, no test-time augmentation, and no model ensembling.

Methods	Present at	Operator	Vehicle 3D AP/APH L1	L2	Pedestrian 3D AP/APH L1	L2	Cyclist 3D AP/APH L1	L2	mAP/mAPH L2
SECOND [63]	Sensors 18	Sparse Conv	72.3/71.7	63.9/63.3	68.7/58.2	60.7/51.3	60.6/59.3	58.3/57.0	61.0/57.2
PointPillars [27]	CVPR 19		72.1/71.5	63.6/63.1	70.6/56.7	62.8/50.3	64.4/62.3	61.9/59.9	62.8/57.8
CenterPoint [69]	CVPR 21		74.2/73.6	66.2/65.7	76.6/70.5	68.8/63.2	72.3/71.1	69.7/68.5	68.2/65.8
PV-RCNN‡ [48]	CVPR 20		78.0/77.5	69.4/69.0	79.2/73.0	70.4/64.7	71.5/70.3	69.0/67.8	69.6/67.2
PillarNet-34 [47]	ECCV 22		79.1/78.6	70.9/70.5	80.6/74.0	72.3/66.2	72.3/71.2	69.7/68.7	71.0/68.5
FSD‡ [17]	NeurIPS 22		79.2/78.8	70.5/70.1	82.6/77.3	73.9/69.1	77.1/76.0	74.4/73.3	72.9/70.8
AFDetV2 [26]	AAAI 22		77.6/77.1	69.7/69.2	80.2/74.6	72.2/67.0	73.7/72.7	71.0/70.1	71.0/68.8
PillarNeXt [28]	CVPR 23		78.4/77.9	70.3/69.8	82.5/77.1	74.9/69.8	73.2/72.2	70.6/69.6	71.9/69.7
VoxelNext [9]	CVPR 23		78.2/77.7	69.9/69.4	81.5/76.3	73.5/68.6	76.1/74.9	73.3/72.2	72.2/70.1
CenterFormer[76]	ECCV 22		75.0/74.4	69.9/69.4	78.6/73.0	73.6/68.3	72.3/71.3	69.8/68.8	71.1/68.9
PV-RCNN++‡ [49]	IJCV 22		79.3/78.8	70.6/70.2	81.3/76.3	73.2/68.0	73.7/72.7	71.2/70.2	71.7/69.5
TransFusion [2]	CVPR 22		–/–	–/65.1	–/–	–/63.7	–/–	–/65.9	–/64.9
ConQueR [78]	CVPR 23		76.1/75.6	68.7/68.2	79.0/72.3	70.9/64.7	73.9/72.5	71.4/70.1	70.3/67.7
FocalFormer3D [7]	ICCV 23		–/–	68.1/67.6	–/–	72.7/66.8	–/–	73.7/72.6	71.5/69.0
HEDNet [71]	NeurIPS 23		81.1/80.6	73.2/72.7	84.4/80.0	76.8/72.6	78.7/77.7	75.8/74.9	75.3/73.4
SEED [33]	ECCV 24		79.7/79.2	71.8/71.4	83.1/78.3	75.5/70.8	80.0/78.8	77.3/76.1	74.9/72.8
SST_TS‡ [15]	CVPR 22	Transformer	76.2/75.8	68.0/67.6	81.4/74.0	72.8/65.9	–/–	–/–	–/–
SWFormer [53]	ECCV 22		77.8/77.3	69.2/68.8	80.9/72.7	72.5/64.9	–/–	–/–	–/–
OcTr [74]	CVPR 23		78.1/77.6	69.8/69.3	80.8/74.4	72.5/66.5	72.6/71.5	69.9/68.9	70.7/68.2
DSVT-Pillar [60]	CVPR 23		79.3/78.8	70.9/70.5	82.8/77.0	75.2/69.8	76.4/75.4	73.6/72.7	73.2/71.0
DSVT-Voxel [60]	CVPR 23		79.7/79.3	71.4/71.0	83.7/78.9	76.1/71.5	77.5/76.5	74.6/73.7	74.0/72.1
LION-RetNet (Ours)	–	RNN	79.0/78.5	70.6/70.2	84.6/80.0	77.2/72.8	79.0/78.0	76.1/75.1	74.6/72.7
LION-RWKV (Ours)	–		79.7/79.3	71.3/71.0	84.6/80.0	77.1/72.7	78.7/77.7	75.8/74.8	74.7/72.8
LION-Mamba (Ours)	–		79.5/79.1	71.1/70.7	84.9/80.4	77.5/73.2	79.7/78.7	76.7/75.8	75.1/73.2
LION-Mamba-L (Ours)	–		**80.3/79.9**	**72.0/71.6**	**85.8/81.4**	**78.5/74.3**	**80.1/79.0**	**77.2/76.2**	**75.9/74.0**

Table 2: Result of our LION with multiple frames as input on the Waymo Open Dataset *test* set.

Methods	Present at	Frames	Vehicle 3D AP/APH L1	L2	Pedestrian 3D AP/APH L1	L2	Cyclist 3D AP/APH L1	L2	mAP/mAPH L2
PV-RCNN++‡ [49]	IJCV 22	1	81.6/81.2	73.9/73.5	80.4/75.0	74.1/69.0	71.9/70.8	69.3/68.2	72.4/70.2
AFDetV2 [26]	AAAI 22	1	–/–	–/–	–/–	–/–	–/–	–/–	72.2/70.3
PillarNeXt [28]	CVPR 23	1	–/–	–/–	–/–	–/–	–/–	–/–	72.2/69.6
FSD‡ [17]	NeurIPS 22	1	82.7/82.3	74.4/74.1	82.9/77.9	75.9/71.3	75.6/74.4	72.9/71.8	74.4/72.4
PillarNeXt [28]	CVPR 23	1	–/–	–/–	–/–	–/–	–/–	–/–	–/72.0
CenterPoint++ [69]	CVPR 21	3	82.8/82.3	75.5/75.1	81.1/78.2	75.1/72.4	74.4/73.3	72.0/71.0	74.2/72.8
PillarNeXt [28]	CVPR 23	3	83.3/82.8	76.2/75.8	84.4/81.4	78.8/76.0	73.7/72.7	71.6/70.6	75.5/74.1
LION-Mamba-L (ours)	–	3	84.7/84.3	77.2/76.9	87.2/84.5	82.0/79.3	79.2/78.3	76.8/75.9	78.7/77.4

Results on Argoverse V2. To further verify the effectiveness of our LION on the long-range perception, we evaluate the experiments on Argoverse V2 *validation* set. For a fair comparison, we adopt the same detection head [69] with VoxelNext [9] and SAFDNet [70] for long-range perception. As shown in Table 4, our LION-RetNet, LION-RWKV, and LION-Mamba achieve the detection performance with 40.7 mAP, 41.1 mAP and 41.5 mAP, all three of which have outperformed the previous SOTA method SAFDNet [70], leading to new SOTA results. These superior results clearly illustrate the effectiveness of our LION.

4.4 Ablation Study

In this section, we conduct ablation studies of LION on the WOD *validation* set with 20% training data. If not specified, we adopt LION-Mamba as our default model and train our model with 12 epochs in the following ablation studies. For more experiments, please refer to our appendix.

Ablation Study of LION. To illustrate the effectiveness of our proposed LION, we conduct the ablation study for each component, including the design of large group size, 3D spatial feature descriptor, and voxel generation in Table 5. Here, our baseline is proposed LION that removes the design of large group size, 3D spatial feature descriptor, and voxel generation. In Table 5, we observe that the design of large group size even brings 1.1 mAPH/L2 performance improvement, which illustrates the benefits of performing long-range feature interaction with the help of linear RNN. Then, we integrate the 3D spatial feature descriptor, which further produces an obvious performance improvement with 1.7 mAPH/L2. This demonstrates the superiority of the 3D spatial feature descriptor in compensating for the lack of capturing spatial information of linear RNNs. Furthermore, we notice that the 3D spatial feature descriptor is very helpful to small objects (*e.g.*, Pedestrians) thanks to its capability of extracting the local information of 3D objects. To address

Table 3: Performances on the nuScenes *validation* and *test* set. 'T.L.', 'C.V.', 'Ped.', 'M.T.', 'T.C.', and 'B.R.' are short for trailer, construction vehicle, pedestrian, motor, traffic cone, and barrier, respectively. All results are reported without any test-time augmentation and model ensembling.

Method	Present at	NDS	mAP	Car	Truck	Bus	T.L.	C.V.	Ped.	M.T.	Bike	T.C.	B.R.
Performances on the validation set													
CenterPoint [69]	CVPR 21	66.5	59.2	84.9	57.4	70.7	38.1	16.9	85.1	59.0	42.0	69.8	68.3
VoxelNeXt [9]	CVPR 23	66.7	60.5	83.9	55.5	70.5	38.1	21.1	84.6	62.8	50.0	69.4	69.4
Uni3DETR [61]	NeurIPS 23	68.5	61.7	–	–	–	–	–	–	–	–	–	–
TransFusion-LiDAR [2]	CVPR 22	70.1	65.5	86.9	60.8	73.1	43.4	25.2	87.5	72.9	57.3	77.2	70.3
QTNet [25]	NeurIPS 23	70.9	66.5	87.2	61.5	75.8	43.0	25.7	87.8	75.5	**61.5**	75.4	71.4
DSVT [60]	CVPR 23	71.1	66.4	87.4	62.6	75.9	42.1	25.3	88.2	74.8	58.7	77.9	71.0
HEDNet [71]	NeurIPS 23	71.4	66.7	87.7	60.6	77.8	**50.7**	**28.9**	87.1	74.3	56.8	76.3	66.9
LION-RetNet (Ours)	–	71.9	67.3	87.9	64.3	**78.7**	44.6	27.6	88.9	73.5	56.6	79.2	**72.1**
LION-RWKV (Ours)	–	71.7	66.8	**88.1**	59.0	77.6	46.6	28.0	**89.7**	74.3	56.2	80.1	68.3
LION-Mamba (Ours)	–	**72.1**	**68.0**	87.9	**64.9**	77.6	44.4	28.5	89.6	**75.6**	59.4	**80.8**	71.6
Performances on the test set													
TransFusion-LiDAR [2]	CVPR 22	70.2	65.5	86.2	56.7	66.3	58.8	28.2	86.1	68.3	44.2	82.0	78.2
DSVT [60]	CVPR 23	72.7	68.4	86.8	58.4	67.3	63.1	**37.1**	88.0	73.0	47.2	84.9	78.4
HEDNet [71]	NeurIPS 23	72.0	67.7	87.1	56.5	**70.4**	63.5	33.6	87.9	70.4	44.8	85.1	78.1
LION-Mamba (Ours)	–	**73.9**	**69.8**	**87.2**	**61.1**	68.9	**65.0**	36.3	**90.0**	**74.0**	**49.2**	**87.3**	**79.5**

Table 4: Comparison with prior methods on Argoverse V2 *validation* set. 'Vehicle', 'C-Barrel', 'MPC-Sign', 'A-Bus', 'C-Cone', 'V-Trailer', 'MBT', 'W-Device' and 'W-Rider' are short for regular vehicle, construction barrel, mobile pedestrian crossing sign, articulated bus, construction cone, vehicular trailer, message board trailer, wheeled device, and wheeled rider.

Method	mAP	Vehicle	Bus	Pedestrian	Stop Sign	Box Truck	Bollard	C-Barrel	Motorcyclist	MPC-Sign	Motorcycle	Bicycle	A-Bus	School Bus	Truck Cab	C-Cone	V-Trailer	Sign	Large Vehicle	Stroller	Bicyclist	Truck	MBT	Dog	Wheelchair	W-Device	W-Rider
CenterPoint [69]	22.0	67.6	38.9	46.5	16.9	37.4	40.1	32.2	28.6	27.4	33.4	24.5	8.7	25.8	22.6	29.5	22.4	6.3	3.9	0.5	20.1	22.1	0.0	3.9	0.5	10.9	4.2
HEDNet [71]	37.1	78.2	47.7	67.6	46.4	45.9	56.9	67.0	48.7	46.5	58.2	47.5	23.3	40.9	27.5	46.8	27.9	20.6	6.9	27.2	38.7	21.6	0.0	30.7	9.5	28.5	8.7
VoxelNeXt [9]	30.7	72.7	38.8	63.2	40.2	40.1	53.9	64.9	44.7	39.4	42.4	40.6	20.1	25.2	19.9	44.9	20.9	14.9	6.8	15.7	32.4	16.9	0.0	14.4	0.1	17.4	6.6
FSDv1 [17]	28.2	68.1	40.9	59.0	29.0	38.5	41.8	42.6	39.7	26.2	49.0	38.6	20.4	30.5	14.8	41.2	26.9	11.9	5.9	13.8	33.4	21.1	0.0	9.5	7.1	14.0	9.2
FSDv2 [18]	37.6	**77.0**	47.6	70.5	43.6	41.5	53.9	58.5	56.8	39.0	60.7	49.4	28.4	41.9	**30.2**	44.9	33.4	16.6	7.3	32.5	45.9	**24.0**	**1.0**	12.6	**17.1**	26.3	17.2
SAFDNet [70]	39.7	78.5	**49.4**	70.7	51.5	44.7	65.7	72.3	54.3	**49.7**	60.8	50.0	**31.3**	**44.9**	24.7	55.4	31.4	22.1	7.1	31.1	42.7	23.6	0.0	26.1	1.4	30.2	11.5
LION-RetNet	40.7	74.7	41.0	72.7	47.5	44.2	66.9	**77.0**	57.1	48.3	63.7	55.1	27.0	42.5	25.2	57.9	29.7	22.0	6.9	39.3	47.3	19.9	0.0	28.8	12.8	**37.7**	**12.6**
LION-RWKV	41.1	76.3	44.6	**74.0**	52.1	**46.0**	**68.1**	75.8	55.8	49.4	62.8	55.3	27.1	42.9	25.9	**60.1**	30.9	22.2	**9.3**	36.5	**55.3**	23.2	0.0	27.8	7.1	37.6	11.4
LION-Mamba	**41.5**	75.1	43.6	73.9	**53.9**	45.1	66.4	74.7	**61.3**	48.7	**65.1**	**56.2**	21.7	42.7	25.3	58.4	28.9	**23.6**	8.3	**49.5**	47.3	19.0	0.0	**31.4**	8.7	37.6	11.8

Table 5: Ablation study for each component in LION. Here, the large group size means that we set it as (4096, 2048, 1024, 512) for four blocks (also refer to the section of our implementation details), otherwise, we set a small group size as (256, 256, 256, 256).

Large Group Size	3D Spatial Feature Descriptor	Voxel Generation	3D AP/APH (L2)			mAP/mAPH (L2)
			Vehicle	*Pedestrian*	*Cyclist*	
–	–	–	65.6/65.2	72.3/65.0	68.3/67.2	68.8/65.8
✓	–	–	66.2/65.7	73.7/67.2	68.7/67.6	69.5/66.9
✓	✓	–	66.5/66.1	74.8/69.6	70.9/70.0	70.8/68.6
✓	–	✓	66.4/66.0	73.5/67.4	70.4/69.3	70.1/67.6
✓	✓	✓	**67.0/66.6**	**75.4/70.2**	**71.9/71.0**	**71.4/69.3**

the challenge of feature representation in highly sparse point clouds, we adopt voxel generation to enhance the features of foregrounds, which brings a promising gain of 0.7 mAPH/L2 (67.6 *vs.* 66.9). By combining all components, our LION achieves a superior performance of 69.3 mAPH/L2, which outperforms the baseline of 3.5 mAPH/L2.

Superiority of 3D Spatial Feature Descriptor. To further verify the necessity of 3D spatial feature descriptor, we provide the comparison with two available manners including the MLP and linear RNN to replace our descriptor in Table 6. Here, we set our LION without 3D spatial feature descriptor as the baseline in this part. We observe that MLP even does not bring promising performance improvement in terms of mAPH/L2 since MLP lacks the ability to capture local 3D spatial information. Furthermore, considering the limited receptive field of MLP, we adopt a linear group RNN operator to replace MLP. We find that there is only slight performance improvement with 0.3 mAPH/L2, which indicates that the linear group RNN might not be good at modeling local spatial relationships although it has the strong capability to establish long-range relationships. In contrast, our 3D spatial feature

Table 6: Ablation study for 3D Spatial Feature Descriptor (3D SFD) in LION.

Methods	3D AP/APH (L2)			mAP/mAPH (L2)
	Vehicle	*Pedestrian*	*Cyclist*	
Baseline	66.4/66.0	73.5/67.4	70.4/69.3	70.1/67.6
MLP	66.6/66.2	74.1/68.1	70.0/69.0	70.2/67.7
Linear Group RNN	66.4/66.0	74.0/68.2	70.5/69.5	70.3/67.9
3D SFD (Ours)	**67.0/66.6**	**75.4/70.2**	**71.9/71.0**	**71.4/69.3**

Table 7: Ablation study for voxel generation in LION. "Baseline" indicates no voxel generation. "Zero Feats" and "K-NN Feats" indicate initializing features to all zeros and K-NN features, respectively. "Auto-Regressive" uses the LION block based on linear group RNN for its auto-regressive property. "Sparse-Conv" maintains the same structure as the LION block but replaces the linear group RNN with 3D sub-manifold convolution.

Index	Methods	3D AP/APH (L2)			mAP/mAPH (L2)
		Vehicle	*Pedestrian*	*Cyclist*	
I	Baseline	66.5/66.1	74.8/69.6	70.9/70.0	70.8/68.6
II	Zero Feats + Sparse-Conv	64.6/64.2	72.8/67.4	69.3/68.3	68.9/66.6
III	K-NN Feats + Auto-Regressive	66.5/66.1	74.0/68.7	71.1/70.1	70.5/68.3
IV	Zero Feats + Auto-Regressive (Ours)	**67.0/66.6**	**75.4/70.2**	**71.9/71.0**	**71.4/69.3**

descriptor brings obvious performance improvement, which boosts the baseline of 1.7 mAPH/L2. This effectively illustrates the superiority of the 3D spatial feature descriptor in compensating for the lack of local 3D spatial-aware modeling in the linear group RNN.

Effectiveness of Voxel Generation. Voxel generation is applied to enhance the feature representation of objects in highly sparse point clouds for accurate 3D object detection. Therefore, to explore the effectiveness of our proposed voxel generation, we present the comparison with several available manners in Table 7. First, we compare our results of IV with II by only replacing the operator of linear group RNN in LION block with 3D sub-manifold convolution to generate the diffused features. We find that the manner of IV (69.3 *vs.* 66.6) significantly outperforms the performance of II in terms of mAPH/L2. This benefits from the linear group RNN's ability to model long-range feature interactions, generating a more reliable feature representation through its auto-regressive capacity, demonstrating the superiority of voxel generation with the linear group RNN. To further illustrate that the effectiveness of voxel generation is from its auto-regressive property of LION block rather than a strong feature extractor, we initialize the diffused features of the foreground voxels by K-NN operation (III) instead of the manner of all-zeros features (IV) and then feed them to the same following LION block for voxel generation. In Table 7, we find that the manner of III is inferior to IV by 1.0 mAPH/L2. This clearly illustrates that our voxel generation is benefiting from its auto-regressive property of LION block. Finally, compared with the baseline (I), our voxel generation (IV) can obtain a promising performance improvement, which verifies its effectiveness.

5   Conclusion

In this paper, we have presented a simple and effective window-based framework termed LION, which can capture the long-range relationship by adopting linear RNN for large groups. Specifically, LION incorporates a proposed LION block to unlock the great potential of linear RNNs in modeling a long-range relationship and a voxel generation strategy to obtain more discriminative feature representation in sparse point clouds. Extensive ablation studies demonstrate the effectiveness of our proposed components. Additionally, the generalization of our LION is verified by performing different linear group RNN operators. Benefiting from our well-designed framework and the proposed superior components, our LION-Mamba achieves state-of-the-art performance on the challenging Waymo and nuScenes datasets.

Acknowledgements

References

[1] Benedikt Alkin, Maximilian Beck, Korbinian Pöppel, Sepp Hochreiter, and Johannes Brandstetter. Vision-lstm: xlstm as generic vision backbone. *arXiv:2406.04303*, 2024. 3

[2] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *CVPR*, 2022. 7, 8, 9

[3] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv:2405.04517*, 2024. 3, 15

[4] 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*, 2020. 2, 3, 6, 7

[5] Chen Chen, Zhe Chen, Jing Zhang, and Dacheng Tao. Sasa: Semantics-augmented set abstraction for point-based 3d object detection. In *AAAI*, 2022. 3

[6] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *ICCV*, 2019. 3

[7] Yilun Chen, Zhiding Yu, Yukang Chen, Shiyi Lan, Anima Anandkumar, Jiaya Jia, and Jose M Alvarez. Focalformer3d: Focusing on hard instance for 3d object detection. In *ICCV*, 2023. 8

[8] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Largekernel3d: Scaling up kernels in 3d sparse cnns. In *CVPR*, 2023. 3

[9] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Voxelnext: Fully sparse voxelnet for 3d object detection and tracking. In *CVPR*, 2023. 7, 8, 9

[10] Bowen Cheng, Lu Sheng, Shaoshuai Shi, Ming Yang, and Dong Xu. Back-tracing representative points for voting-based 3d object detection in point clouds. In *CVPR*, 2021. 3

[11] Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv:2402.19427*, 2024. 3

[12] 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*, 2021. 3

[13] Shaocong Dong, Lihe Ding, Haiyang Wang, Tingfa Xu, Xinli Xu, Jie Wang, Ziyang Bian, Ying Wang, and Jianan Li. Mssvt: Mixed-scale sparse voxel transformer for 3d object detection on point clouds. In *NeurIPS*, 2022. 3

[14] Yuchen Duan, Weiyun Wang, Zhe Chen, Xizhou Zhu, Lewei Lu, Tong Lu, Yu Qiao, Hongsheng Li, Jifeng Dai, and Wenhai Wang. Vision-rwkv: Efficient and scalable visual perception with rwkv-like architectures. *arXiv:2403.02308*, 2024. 3

[15] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. In *CVPR*, 2022. 1, 3, 4, 7, 8

[16] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. In *CVPR*, 2022. 3

[17] Lue Fan, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Fully sparse 3d object detection. In *NeurIPS*, 2022. 2, 6, 8, 9

[18] Lue Fan, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Fsd v2: Improving fully sparse 3d object detection with virtual voxels. *arXiv:2308.03755*, 2023. 9

[19] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *IJRS*, 2013. 16

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 7, 15

[21] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013. 7

[22] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv:2312.00752*, 2023. 2, 3, 4, 7, 15, 17

[23] Tianrui Guan, Jun Wang, Shiyi Lan, Rohan Chandra, Zuxuan Wu, Larry Davis, and Dinesh Manocha. M3detr: Multi-representation, multi-scale, mutual-relation 3d object detection with transformers. In *WACV*, 2022. 3

[24] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *CVPR*, 2020. 3

[25] Jinghua Hou, Zhe Liu, Zhikang Zou, Xiaoqing Ye, Xiang Bai, et al. Query-based temporal fusion with explicit motion for 3d object detection. In *NeurIPS*, 2023. 9

[26] Yihan Hu, Zhuangzhuang Ding, Runzhou Ge, Wenxin Shao, Li Huang, Kun Li, and Qiang Liu. Afdetv2: Rethinking the necessity of the second stage for object detection from point clouds. In *AAAI*, 2022. 8

[27] 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*, 2019. 8, 15, 16

[28] Jinyu Li, Chenxu Luo, and Xiaodong Yang. Pillarnext: Rethinking network designs for 3d object detection in lidar point clouds. In *CVPR*, 2023. 8

[29] Dingkang Liang, Xin Zhou, Xinyu Wang, Xingkui Zhu, Wei Xu, Zhikang Zou, Xiaoqing Ye, and Xiang Bai. Pointmamba: A simple state space model for point cloud analysis. *arXiv:2402.10739*, 2024. 3

[30] Jianhui Liu, Yukang Chen, Xiaoqing Ye, Zhuotao Tian, Xiao Tan, and Xiaojuan Qi. Spatial pruned sparse convolution for efficient 3d object detection. In *NeurIPS*, volume 35, pages 6735–6748, 2022. 6

[31] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *arXiv:2401.10166*, 2024. 5

[32] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. In *ICCV*, 2021. 3

[33] Zhe Liu, Jinghua Hou, Xiaoqing Ye, Tong Wang, Jingdong Wang, and Xiang Bai. Seed: A simple and effective 3d detr in point clouds. In *ECCV*, 2024. 8

[34] Zhe Liu, Xin Zhao, Tengteng Huang, Ruolan Hu, Yu Zhou, and Xiang Bai. Tanet: Robust 3d object detection from point clouds with triple attention. In *AAAI*, 2020. 3, 6, 15

[35] Zhijian Liu, Xinyu Yang, Haotian Tang, Shang Yang, and Song Han. Flatformer: Flattened window attention for efficient point cloud transformer. In *CVPR*, 2023. 3, 4

[36] Tao Lu, Xiang Ding, Haisong Liu, Gangshan Wu, and Limin Wang. Link: Linear kernel for lidar-based 3d perception. In *CVPR*, 2023. 3

[37] Jiageng Mao, Minzhe Niu, Chenhan Jiang, Hanxue Liang, Jingheng Chen, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, et al. One million scenes for autonomous driving: Once dataset. *arXiv:2106.11037*, 2021. 2, 3, 7, 15

[38] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *ICML*, 2023. 3

[39] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *CVPR*, 2021. 3

[40] Bo Peng, Eric Alcaide, Quentin Gregory Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Nguyen Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *EMNLP*, 2023. 2, 3, 4, 7, 15

[41] Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, Kranthi Kiran GV, Jan Kocoń, Bartłomiej Koptyra, Satyapriya Krishna, Ronald McClelland Jr. au2, Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Stanisław Woźniak, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv:2404.05892*, 2024. 3

[42] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019. 3

[43] 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*, 2018. 3, 15

[44] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 3

[45] 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*, 2017. 3

[46] Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. In *NeurIPS*, 2023. 3

[47] Guangsheng Shi, Ruifeng Li, and Chao Ma. Pillarnet: High-performance pillar-based 3d object detection. In *ECCV*, 2022. 8

[48] 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*, 2020. 3, 8, 16

[49] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *IJCV*, 2021. 3, 8

[50] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 3, 15, 16

[51] 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. *IEEE TPAMI*, 2020. 3

[52] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 2, 3, 6, 15, 19

[53] Pei Sun, Mingxing Tan, Weiyue Wang, Chenxi Liu, Fei Xia, Zhaoqi Leng, and Drago Anguelov. Swformer: Sparse window transformer for 3d object detection in point clouds. In *ECCV*, 2022. 1, 2, 3, 6, 7, 8

[54] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv:2407.04620*, 2024. 3, 15

[55] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv:2307.08621*, 2023. 3, 7, 15

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 3

[57] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *CVPR*, 2020. 16

[58] Chunwei Wang, Chao Ma, Ming Zhu, and Xiaokang Yang. Pointaugmenting: Cross-modal augmentation for 3d object detection. In *CVPR*, 2021. 7

[59] Haiyang Wang, Lihe Ding, Shaocong Dong, Shaoshuai Shi, Aoxue Li, Jianan Li, Zhenguo Li, and Liwei Wang. Cagroup3d: Class-aware grouping for 3d object detection on point clouds. In *NeurIPS*, 2022. 3

[60] Haiyang Wang, Chen Shi, Shaoshuai Shi, Meng Lei, Sen Wang, Di He, Bernt Schiele, and Liwei Wang. Dsvt: Dynamic sparse voxel transformer with rotated sets. In *CVPR*, 2023. 1, 2, 3, 4, 7, 8, 9, 15, 17, 19

[61] Zhenyu Wang, Ya-Li Li, Xi Chen, Hengshuang Zhao, and Shengjin Wang. Uni3detr: Unified 3d detection transformer. In *NeurIPS*, 2024. 9

[62] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *NeurIPS*, 2023. 2, 3, 6

[63] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 3, 4, 7, 8, 15, 16

[64] Honghui Yang, Wenxiao Wang, Minghao Chen, Binbin Lin, Tong He, Hua Chen, Xiaofei He, and Wanli Ouyang. Pvt-ssd: Single-stage 3d object detector with point-voxel transformer. In *CVPR*, 2023. 3

[65] Jinrong Yang, Lin Song, Songtao Liu, Weixin Mao, Zeming Li, Xiaoping Li, Hongbin Sun, Jian Sun, and Nanning Zheng. Dbq-ssd: Dynamic ball query for efficient 3d object detection. In *ICLR*, 2022. 3

[66] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv:2312.06635*, 2023. 3

[67] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *CVPR*, 2020. 3

[68] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *CVPR*, 2019. 3

[69] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021. 3, 4, 8, 9, 16

[70] Gang Zhang, Junnan Chen, Guohuan Gao, Jianmin Li, Si Liu, and Xiaolin Hu. Safdnet: A simple and effective network for fully sparse 3d object detection. In *CVPR*, 2024. 2, 3, 6, 7, 8, 9, 16

[71] Gang Zhang, Chen Junnan, Guohuan Gao, Jianmin Li, and Xiaolin Hu. HEDNet: A hierarchical encoder-decoder network for 3d object detection in point clouds. In *NeurIPS*, 2023. 7, 8, 9

[72] Guowen Zhang, Lue Fan, Chenhang He, Zhen Lei, Zhaoxiang Zhang, and Lei Zhang. Voxel mamba: Group-free state space models for point cloud based 3d object detection. *arXiv preprint arXiv:2406.10700*, 2024. 3

[73] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. In *CVPR*, 2022. 3

[74] Chao Zhou, Yanan Zhang, Jiaxin Chen, and Di Huang. Octr: Octree-based transformer for 3d object detection. In *CVPR*, 2023. 8

[75] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 15

[76] Zixiang Zhou, Xiangchen Zhao, Yu Wang, Panqu Wang, and Hassan Foroosh. Centerformer: Center-based transformer for 3d object detection. In *ECCV*, 2022. 8

[77] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv:1908.09492*, 2019. 7

[78] Benjin Zhu, Zhe Wang, Shaoshuai Shi, Hang Xu, Lanqing Hong, and Hongsheng Li. Conquer: Query contrast voxel-detr for 3d object detection. In *CVPR*, 2023. 8

A Appendix

The appendix is organized as follows. First, in section A.1, we provide more types of linear RNN operators (*e.g.*, RetNet, RWKV, Mamba, xLSTM, and TTT) based on our LION framework for 3D detection on a small but popular dataset KITTI for a quick experience. Second, we present extra experiments on the WOD [52] *validation* set in section A.3, including the placement of the 3D spatial feature descriptor, the impact of different window sizes and different group sizes in inference, and the ratio r in voxel generation. Third, we provide the comparison of computation cost and parameter size in section A.4, and detailed information of LION structure in section A.5. Forth, in section A.6, we visualize the feature maps of different LION blocks to illustrate the rationality of distinguishing foreground voxels based on feature response. Finally, we provide the comparison of qualitative results with DSVT [60] and qualitative results of LION to demonstrate the superiority of our LION in section A.7 and section A.8. Besides, we discuss the broader impacts in section A.9.

A.1 Experiments on KITTI dataset

Table 8: Effectiveness on the KITTI *validation* set for *Car*, *Pedestrian*, and *Cyclist*. * represents our reproduced results by keeping the same configures except for their 3D backbones for a fair comparison. Our LION supports different representative linear RNN operators (TTT, xLSTM, RetNet, RWKV, and Mamba). mAP is calculated by all categories and all difficulties with recall 11.

Method	Car			Pedestrian			Cyclist			mAP
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	
VoxelNet [75]	77.5	65.1	57.7	39.5	33.7	31.5	61.2	48.4	44.4	51.0
SECOND [63]	83.1	73.7	66.2	51.1	42.6	37.3	70.5	53.9	46.9	58.4
PointPillars [27]	79.1	75.0	68.3	52.1	43.5	41.5	75.8	59.1	52.9	60.8
PointRCNN [50]	85.9	75.8	68.3	49.4	41.8	38.6	73.9	59.6	53.6	60.8
TANet [34]	83.8	75.4	67.7	54.9	46.7	42.4	73.8	59.9	53.5	62.0
DSVT-Pillar* [60]	87.3	77.4	76.2	61.4	56.8	51.8	82.3	67.1	63.7	69.3
DSVT-Voxel* [60]	87.8	77.8	76.8	66.1	59.7	55.2	83.5	66.7	63.2	70.8
LION-TTT	87.9	78.0	76.7	63.4	58.6	53.7	84.0	69.6	64.5	70.7
LION-xLSTM	87.7	77.9	76.8	66.6	59.3	54.0	82.4	67.4	63.4	70.6
LION-RetNet	88.0	77.9	76.7	67.4	60.2	55.8	83.6	69.6	64.6	71.5
LION-Mamba	**88.6**	**78.3**	77.2	67.2	60.2	55.6	83.0	68.6	63.9	71.4
LION-RWKV	88.5	**78.3**	**77.1**	**68.9**	**62.2**	**58.1**	**89.6**	**71.2**	**66.9**	**73.4**

KITTI Dataset. KITTI [20] is a popular benchmark dataset for autonomous driving, which consists of 7481 training frames and 7518 test frames for 3D object detection. We follow the dataset splitting protocol in [43] and further split the 7481 training frames into 3712 frames for *training* set and 3769 frames for *validation* set. For the 3D detection task, KITTI dataset mainly detects Car, Pedestrian, and Cyclist for three difficulty levels, *i.e.*, Easy, Moderate, and Hard. And the mean Average Precision (mAP) using 11 recall positions is adopted as the evaluation metric.

Results on KITTI. We conduct experiments on the KITTI *validation* set to illustrate the generalization of LION for different linear RNN operators. We select some representative linear RNN operators (TTT [54], xLSTM [3], RetNet [55], Mamba [22], and RWKV [40]) for LION. We adopt the same training parameters (*i.e.*, number of epochs, learning rate, optimizer) with SECOND [63]. Besides, we use the same BEV backbone and the detection head with SECOND [63]. For a fair comparison, we keep the same configure of DSVT [60] and all our LION methods except 3D backbones. As shown in Table 8, LION-RetNet, LION-Mamba, and LION-RWKV outperforms DSVT-Voxel by 0.7 mAP, 0.6 mAP, and 2.6 mAP. These experiments demonstrate the generalization and effectiveness of our linear RNN-based framework LION.

A.2 Experiments on ONCE dataset

ONCE Dataset. ONCE [37] is another representative autonomous driving dataset, which consists of 5000, 3000, and 8000 frames for training, validation, and testing set, respectively. Each frame is annotated with 5 classes (Car, Bus, Truck, Pedestrian, and Cyclist). Besides, ONCE merges the car, bus, and truck class into a super-class called vehicle following WOD [52]. For the detection metric,

Table 9: Comparison with previous methods on ONCE *validation* set. For a fair comparison, we replace the SECOND 3D backbone in the ONCE dataset with our LION 3D backbone, maintaining a grid size of $(0.4m, 0.4m, 0.25m)$, and adopt the same training process as SAFDNet [70] and SECOND [63] while utilizing the center head of CenterPoint [69].

Method	Vehicle				Pedestrian				Cyclist				mAP
	overall	0-30m	30-50m	50m-inf	overall	0-30m	30-50m	50m-inf	overall	0-30m	30-50m	50m-inf	
PointRCNN [50]	52.1	74.5	40.9	16.8	4.3	6.2	2.4	0.9	29.8	46.0	20.9	5.5	28.7
PointPillars [27]	68.6	80.9	62.1	47.0	17.6	19.7	15.2	10.2	46.8	58.3	40.3	25.9	44.3
SECOND [63]	71.2	84.0	63.0	47.3	26.4	29.3	24.1	18.1	58.0	70.0	52.4	34.6	51.9
PV-RCNN [48]	77.8	**89.4**	**72.6**	58.6	23.5	25.6	22.8	17.3	59.4	71.7	52.6	36.2	53.6
CenterPoint [69]	66.8	80.1	59.6	43.4	49.9	56.2	42.6	**26.3**	63.5	74.3	57.9	41.5	60.1
PointPainting [57]	66.2	80.3	59.8	42.3	44.8	52.6	36.6	22.5	62.3	73.6	57.2	40.4	57.8
LION-RetNet	78.1	88.7	72.4	**58.5**	52.4	60.5	43.6	**26.3**	68.3	**79.4**	62.9	46.1	66.3
LION-RWKV	**78.3**	89.2	**72.6**	56.7	50.6	60.0	40.4	24.2	68.4	**79.4**	**63.2**	45.7	65.8
LION-Mamba	78.2	89.1	**72.6**	57.5	**53.2**	**62.4**	**44.0**	24.5	**68.5**	79.2	**63.2**	**47.1**	**66.6**

ONCE extends [19] by taking the object orientations into special consideration and evaluating the final performance by mAP for three classes.

Results on ONCE. We also evaluated our LION on ONCE *validation* set to further verify the effectiveness of our LION. As shown in Table 9, our LION-RetNet, LION-RWKV, and LION-Mamba produces advanced detection performance with 66.3 mAP, 65.8 mAP, and 66.6 mAP, respectively. It is worth mentioning that our LION-Mamba outperforms the previous SOTA method CenterPoint [69] with 6.5 mAP, leading to a new SOTA result. These experiments illustrate the superiority of our LION.

A.3 Extra Experiments

Table 10: The Placement of 3D Spatial Feature Descriptor.

Methods	3D AP/APH (L2)			mAP/mAPH (L2)
	Vehicle	*Pedestrian*	*Cyclist*	
Baseline	66.4/66.0	73.5/67.4	70.4/69.3	70.1/67.6
Placement 1	66.5/66.1	74.8/69.1	71.1/70.2	70.1/68.6
Placement 2 (Ours)	**67.0/66.6**	**75.4/70.2**	**71.9/71.0**	**71.4/69.3**

Table 11: The ratio r in voxel genenration.

Ratio	3D AP/APH (L2)			mAP/mAPH (L2)
	Vehicle	*Pedestrian*	*Cyclist*	
0	66.5/66.1	74.8/69.6	70.9/70.0	70.8/68.6
0.2 (Ours)	67.0/66.6	**75.4/70.2**	71.9/71.0	71.4/**69.3**
0.5	**67.2/66.8**	75.3/70.0	**72.1/71.1**	**71.5/69.3**

Table 12: Comparison of group sizes on WOD *validation* set (train with 20% training data).

#	Group Size	3D AP/APH (L2)			mAP/mAPH (L2)
		Vehicle	*Pedestrian*	*Cyclist*	
I	[256, 256, 256, 256]	65.6/65.2	72.3/65.0	68.3/67.2	68.8/65.8
II	[1024, 512, 256, 256]	66.9/66.5	74.9/69.6	70.8/69.8	70.9/68.6
III	[2048, 1024, 512, 256]	66.7/66.3	74.9/69.7	**72.2/71.2**	71.3/69.1
IV	[4096, 2048, 1024, 512]	**67.0/66.6**	**75.4/70.2**	71.9/71.0	**71.4/69.3**
V	[8192, 4096, 2048, 1024]	66.5/66.1	74.6/69.5	71.6/70.6	70.9/68.7

The Placement of 3D Spatial Feature Descriptor. We conduct experiments about the placement of the 3D spatial feature descriptor, as shown in Table 10. We regard the manner that does not adopt the 3D SFD of our LION as the baseline. Here, we provide two available manners: Placement 1 and Placement 2. For Placement 1, we place the 3D SFD after voxel merging. For Placement 2, we place 3D SFD before the voxel merging. Compared to the baseline, Placement 1 brings 1.0 mAPH/L2 improvement, which demonstrates the effectiveness of 3D SFD in compensating for the lack of local

Table 13: Comparison of different window and group sizes in inference on WOD *validation* set (train with 100% training data). **Bold** denotes the result of LION with the default settings in the main paper.

Window Size	Group Size	mAP/mAPH (L2)
$[7, 7, 32], [7, 7, 16], [7, 7, 8], [7, 7, 4]$	[4096, 2048, 1024, 512]	73.24
$[13, 13, 32], [13, 13, 16], [13, 13, 8], [13, 13, 4]$	[4096, 2048, 1024, 512]	**73.24**
$[25, 25, 32], [25, 25, 16], [25, 25, 8], [25, 25, 4]$	[4096, 2048, 1024, 512]	73.25
$[13, 13, 32], [13, 13, 16], [13, 13, 8], [13, 13, 4]$	[2048, 1024, 512, 256]	73.18
$[13, 13, 32], [13, 13, 16], [13, 13, 8], [13, 13, 4]$	[4096, 2048, 1024, 512]	**73.24**
$[13, 13, 32], [13, 13, 16], [13, 13, 8], [13, 13, 4]$	[8192, 4096, 2048, 1024]	73.02

3D spatial-aware modeling in linear RNNs. Moreover, Placement 2 further brings 0.7 mAPH/L2 improvement over Placement 1, which demonstrates the effectiveness of 3D SFD for reducing spatial information loss in the process of voxel merging.

The Ratio in Voxel Generation. We conduct the ablation study for foreground selection ratio r in voxel generation. As shown in Table 11, compared with baseline ($r = 0$), the manner of setting $r = 0.2$ brings 0.7 mAPH/L2 performance improvement. When we set a larger ratio $r = 0.5$, the performance is improved slightly. Therefore, we set $r = 0.2$ to balance the performance and computation cost.

Different Group Sizes in Training. We provide a comparison of different group sizes in Table 12. We set a minimum group size of 256 for four LION blocks as the baseline. We could find that the manners with large group sizes (i.e., II, III, IV) bring consistent performance over the small ones. However, there is a drop in performance when the group size is increased from IV to V. This might lead to less effective retention of important information in excessively long sequences due to the limited memory capacity of linear RNNs.

Different Window Sizes and Group Sizes in Inference. To analyze the impact of window size and group size in the inference process, we evaluate the results under the cases of different window sizes and group sizes with the same trained model of LION-Mamba (*i.e.*, window size={$(13, 13, 32)$, $(13, 13, 16)$, $(13, 13, 8)$, and $(13, 13, 4)$} and group size={4096, 2048, 1024, 512}) on WOD 100% training data. As shown in Table 13, surprisingly, we find that using different window sizes or group sizes during inference still does not significantly affect performance. This indicates that LION might decrease the strong dependence on hand-crafted priors and have good extrapolation ability.

A.4 Comparison of Computation Cost, Parameter Size and Latency

Table 14: Comparison of computation cost, parameter size and latency of different methods on the WOD *validation* set.

Method	Operator	mAP/mAPH (L2)	FLOPs (G)	Params (M)	Latency (ms)
DSVT-Voxel [60]	Transformer	74.0/72.1	100.8	2.7	136.7
LION (Ours)	Mamba	75.1/73.2	58.5	1.4	146.2

We compare our LION with the representative transformer-based method DSVT-Voxel [60] in terms of computation cost, parameter size and latency on the WOD *validation* set. As shown in Table 14, LION with Mamba [22] achieves a more superior performance (73.2 mAPH/L2), less computational cost (58.5 GFLOPs), fewer parameters than DSVT-Voxel. However, our LION still has more latency in inference stage. These properties clearly illustrate the superiority of our proposed LION and might have great potential to further optimize the speed of inference in the future.

A.5 Architecture Specifications

As shown in Table 15, the architecture specifications of the LION models (LION-RWKV, LION-RetNet, LION-Mamba, and LION-Mamba-L) on Waymo Open dataset are detailed in terms of

Table 15: Detailed architecture specifications on Waymo Open dataset.

	LION-RWKV	LION-RetNet	LION-Mamba	LION-Mamba-L
Block	Window Shape Dim, Group Size	Window Shape Dim, Group Size	Window Shape Dim, Group Size	Window Shape Dim, Group Size
Block 1	[13, 13, 32] 64, 4096	[13, 13, 32] 64, 4096	[13, 13, 32] 64, 4096	[13, 13, 32] 128, 4096
Block 2	[13, 13, 16] 64, 2048	[13, 13, 16] 64, 2048	[13, 13, 16] 64, 2048	[13, 13, 16] 128, 2048
Block 3	[13, 13, 8] 64, 1024	[13, 13, 8] 64, 1024	[13, 13, 8] 64, 1024	[13, 13, 8] 128, 1024
Block 4	[13, 13, 4] 64, 512	[13, 13, 4] 64, 512	[13, 13, 4] 64, 512	[13, 13, 4] 128, 512

window shape, dimension, and group size. For LION-Mamba-L, we set the dimension to 128 to double the channel of LION.

Figure 6: Visualization of feature map of different blocks. We highlight the foreground annotated by red GT boxes. The color map represents the magnitude of the feature response.

A.6 Visualization for Feature Map

As shown in Figure 6, we visualize feature maps of different LION blocks. We can observe that as the features pass through more blocks, the magnitude of the foreground's feature response becomes larger, demonstrating the rationality of distinguishing foreground voxels by feature response. Besides, we find that the foreground features become more dense and more distinguished, which also demonstrates the effectiveness of the voxel generation operation.
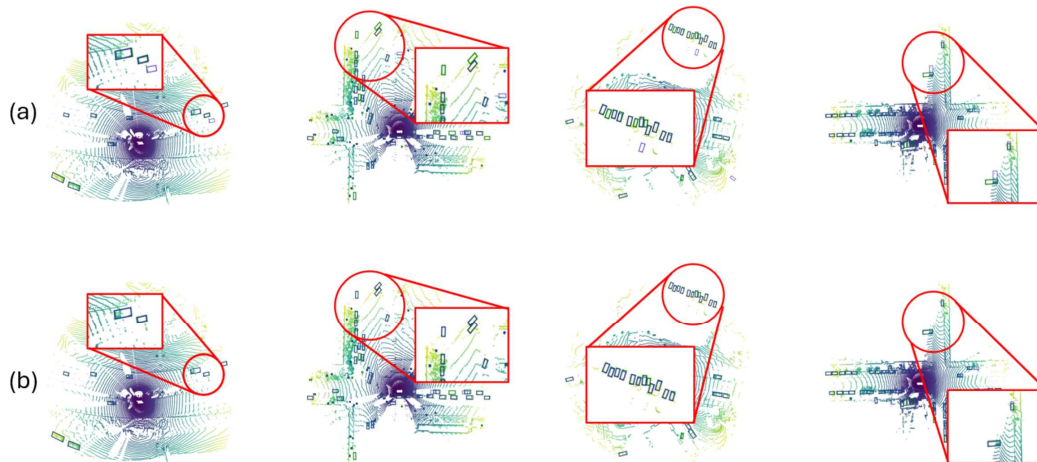
Figure 7: Comparison of DSVT and LION on the WOD *validation* set from the BEV perspective. Blue and green boxes are the prediction and ground truth boxes. It can be seen that LION can achieve better results compared to DSVT, demonstrating the superiority of LION.

A.7 Comparison of Qualitative Results with DSVT

To illustrate the superiority of LION, we present the visualization of the qualitative results of DSVT [60] (a) and LION (b) on the WOD [52] *validation* set, as shown in Figure 7. Specifically, in the first and third columns, our LION can reduce more false positives compared with DSVT. In the second column, our LION even detects some hard objects at a distance. In the last column, our LION can achieve more accurate localization. These qualitative results demonstrate the superior performance of our LION.

A.8 Qualitative Results

As shown in Figure 8, we visualize the qualitative results of LION on the WOD *validation* set. As shown in the first column, LION can still achieve satisfactory results even in crowded 3D scenes. However, as shown in the second and third columns, LION misses some objects at a distance with sparse point clouds. Therefore, we will further improve the performance of distant objects by fusing the image features in the future.

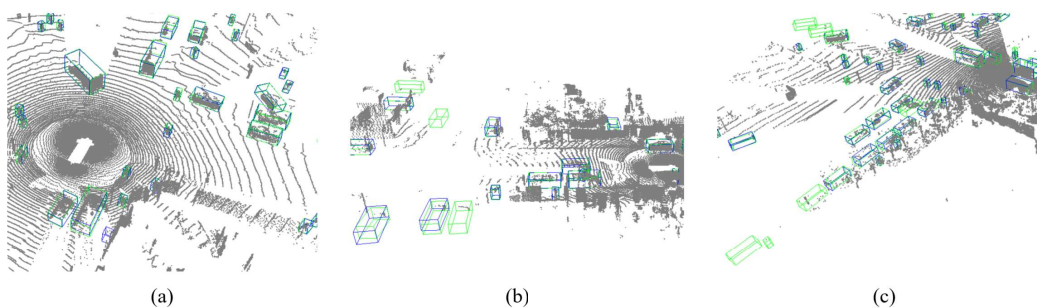(a)   (b)   (c)

Figure 8: Qualitative results of LION on the WOD *validation* set. Green and blue boxes denote ground truth and predicted bounding boxes, respectively.

A.9 Broader Impacts

LION achieves promising performance for 3D object detection, enhancing the safety of autonomous driving. However, LION has relatively high requirements on computing resources to achieve faster running speed, which puts forward higher requirements for the hardware of autonomous driving. Future research could focus on optimizing LION to improve bottlenecks in running speed while maintaining high detection accuracy, making it more accessible and practical for autonomous driving.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist",**
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction accurately reflect the our contributions.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We have discussed limitations in paper.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper primarily introduces a new architecture and proposes several modules based on this architecture. It does not include theoretical assumptions and proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide detailed information to reproduce the main experimental results of the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We plan to provide all code for reproducing the results after the manuscript is accepted. Additionally, all datasets used in our experiments are publicly available, ensuring that the main experimental results can be faithfully reproduced.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experimental details in paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper provides detailed experimental setups and procedures to ensure the reproducibility of the results. However, it does not include error bars or other statistical significance information.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide experiments compute resources in paper.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: Our research conducted in the paper conform with the NeurIPS code of ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: These impacts are discussed in the appendix.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper focuses on foundational research and does not release any data or models that pose a high risk for misuse. Therefore, no specific safeguards are necessary.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All existing assets used in the paper, such as code and datasets, are properly credited. The licenses and terms of use for these assets are explicitly mentioned and respected.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The new models introduced in the paper are thoroughly documented. Comprehensive documentation is provided alongside the models to ensure clarity and reproducibility.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing or research with human subjects. Therefore, it does not include participant instructions, screenshots, or compensation details.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects. Therefore, IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.