LinNet: Linear Network for Efficient Point Cloud Representation Learning

Hao Deng[1] **Kunlei Jing**[2,3*] **Shengmei Cheng**[1] **Cheng Liu**[1]
Jiawei Ru[1] **Jiang Bo**[1] **Lin Wang**[1*]

[1]State-Province Joint Engineering and Research Center of Advanced Networking and Intelligent Information Services, School of Information Science and Technology, Northwest University
[2]School of Software Engineering, Xi'an Jiaotong University
[3]Department of Computing, The Hong Kong Polytechnic University

denghao@stumail.nwu.edu.cn, kunlei.jing@xjtu.edu.cn
1615241805@qq.com, lc@nwu.edu.cn
rujiawei@stumail.nwu.edu.cn, {jiangbo, wanglin}@nwu.edu.cn

Abstract

Point-based methods have made significant progress, but improving their scalability in large-scale 3D scenes is still a challenging problem. In this paper, we delve into the point-based method and develop a *simpler*, *faster*, *stronger* variant model, dubbed as **LinNet**. In particular, we first propose the disassembled set abstraction (DSA) module, which is more effective than the previous version of set abstraction. It achieves more efficient local aggregation by leveraging spatial anisotropy and channel anisotropy separately. Additionally, by mapping 3D point clouds onto 1D space-filling curves, we enable parallelization of downsampling and neighborhood queries on GPUs with linear complexity. LinNet, as a purely point-based method, outperforms most previous methods in both indoor and outdoor scenes without any extra attention, and sparse convolution but merely relying on a simple MLP. It achieves the mIoU of 73.7%, 81.4%, and 69.1% on the S3DIS Area5, NuScenes, and SemanticKITTI validation benchmarks, respectively, while speeding up almost 10x times over PointNeXt. Our work further reveals both the efficacy and efficiency potential of the vanilla point-based models in large-scale representation learning. *Our code will be available at* https://github.com/DengH293/LinNet.

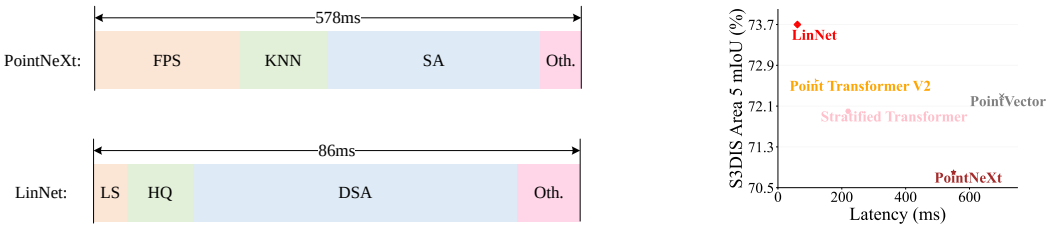
1 Introduction

Appealed by the ongoing evolution progress of technologies in robotics, autonomous driving, augmented reality, etc., LiDAR sensors are incrementally integrated into hardware-constrained devices such as mobile devices and AR headsets. This has led to a growing interest in efficient point cloud processing models. Given the limited computational power of mobile devices and embedded systems, the design of mobile-friendly point cloud representation learning algorithms should not only focus on performance but also pay attention to high computational efficiency.

Unlike images, point cloud data is irregular and unordered. There are various methods for processing point cloud data in 3D vision. Common approaches include multi-view methods [1, 2, 3] and voxel-based methods [4, 5]. Converting irregular data into the required formal representations often requires additional computation and memory and thus results in the loss of geometric information [6]. Therefore, point-based methods that directly operate on point clouds have emerged. PointNet [7]

*Corresponding Authors

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



(a) Latency decomposition. (b) Accuracy-speed tradeoff.

Figure 1: **Latency decomposition.** We show the inference run time decomposition. (a) SA: set abstraction; DSA: disassembled set abstraction; LS: linearization sampling; HQ: hash query; Oth.: others. (b) LinNet achieves the highest mIoU with extremely low latency compared to the comparative point-based approaches. The latency of each network is measured on a single Nvidia 3090 GPU, taking a batch of 80k points.

and PointNet++ [8] are the pioneers of this approach, introducing a general point cloud learning paradigm from local to global. The paradigm consists of two parts: the first part is a spatial neighborhood search strategy, which utilizes algorithms such as furthest point sampling (FPS) and K-nearest neighbors (KNN) to implement sub-sampling and neighborhood grouping of point clouds, respectively. The second part is a trainable local feature extractor. Following them, subsequent extensive research [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19] have shown promising results by focusing on the design of more sophisticated extractors.

Despite the impressive results that have been achieved in object recognition and semantic segmentation, most of these methods are limited to applications in small-scale 3D point clouds. The main reason is discouraged by the high time complexity of the neighborhood search strategy that the point-based methods adopted. As shown in Fig. 1a. FPS and KNN occupy 46% of the runtime. This draws forth the main motivation of this paper: *enhancing the scalability of point-based approaches in large-scale scenes, while maintaining their excellent performance in small-scale tasks.*

In this paper, we introduce a novel model, named ***Linear Net (LinNet)***. Our LinNet derives from inheriting the innovations and overcoming the drawbacks of the PointNet++ paradigm, including a disassembled set abstraction (DSA) module and an efficient point search strategy. Specifically, inspired by MobileNet [20], we first use two independent MLPs to separately learn depth-wise geometric features of the neighborhood and point-wise semantic features. Then, the geometric features are assigned as biases to the queried neighbors' semantic features, achieving spatially anisotropic neighborhood aggregation. Since the learning of high-dimensional semantic features is point-wise and does not involve the neighborhood, the required floating-point operations (FLOPs) are significantly lower than those needed for SA. Besides, a hash query and linearization sampling strategy are proposed for speeding up point searching. The core of our method is to map the 3D search space onto a segmented curve for acceleration. A sparse point cloud is ordered on that curve, and points adjacent to each other in the curve are also adjacent in space. For neighborhood queries, we store each segmented curve as a bucket in a hash table. When querying the neighborhood, we only need to search in the buckets corresponding to the neighboring curves, which drastically cuts down the search range. Linear sampling ensures uniform sampling by taking the point closest to the origin within each grid as the new sampling point. The method reduces the time complexity to be linear and supports GPU parallelism, resulting in very fast sampling. As shown in Fig. 1, the additional point cloud search operations take up less than 10% of the model's runtime. By employing these techniques, our method achieves efficient point cloud representation learning and scalability, providing significant performance improvements for large-scale point cloud analysis.

The contribution of our paper can be summarized in the following three folds:

- We analyze the feature aggregation of the vanilla SA module and introduce a novel efficient and effective DSA module. This strategy effectively reduces computational overhead and achieves performance gains. Moreover, we discuss the superiority of this method from the perspective of weight initialization, emphasizing how these adjustments crucially enhance the overall performance of the network.

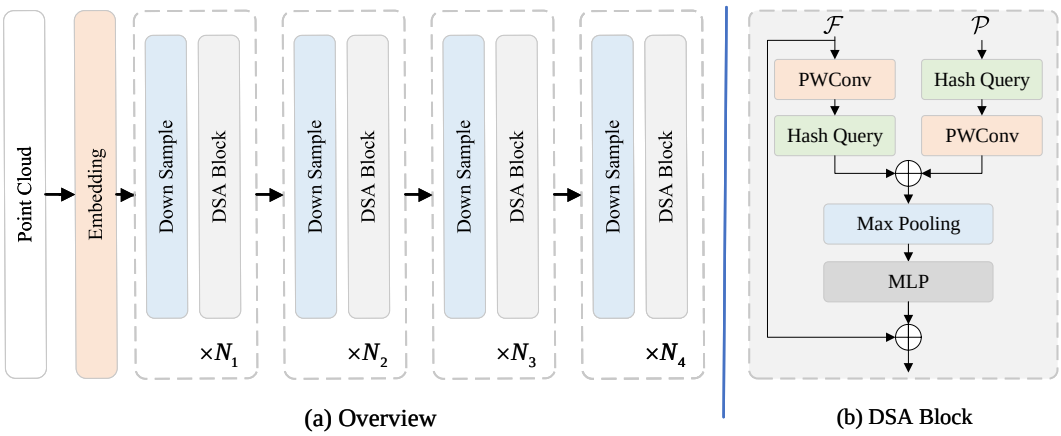


Figure 2: **Overall architecture.** (a) Overview of the framework. The whole network consists of an embedding layer and four stages, each containing a downsampling layer and N_i disassembled SA blocks. (b) Structure of the DSA blocks. Each DSA block consists of a DSA module and extra MLPs.

- To improve scalability in large-scale scenes, a linear complexity point cloud search strategy is introduced, mapping a 3D point cloud to a 1D space-filling curve. This approach drastically reduces the time consumption associated with sub-sampling and neighbors query.
- Experiments show that our approach achieves state-of-the-art performance on widely adopted 3D large-scale semantic segmentation benchmarks (S3DIS, NuScenes) and competitive results on small-scale classification tasks (ScanObjectNN and ModelNet40). Extensive ablation studies have also validated the effectiveness of our proposed components.

2 Related Works

Point cloud analysis. Point cloud analysis is primarily approached in two ways. Another approach, exemplified by the PointNet family, directly processes raw point clouds. They introduce a hierarchical feature learning paradigm to recursively capture local geometric structures. By adopting local point representation and multi-scale information, PointNet++ has demonstrated excellent performance and has become a cornerstone of modern point cloud methods [9, 16, 21, 17, 22, 23]. Our LinNet follows the design philosophy of PointNet++ but explores a simpler yet deeper network architecture.

Voxel-based methods. Instead of learning directly on discrete points, the sparse convolution [5, 24] first converts the point cloud into a regular grid and then constructs a full convolutional neural network using the discrete sparse tensor. By building a hash table of discrete rasters, neighborhood query and sampling can be performed efficiently with a constant time complexity of $\mathcal{O}(1)$. In addition, the hash table construction and query can be implemented in parallel on CUDA, which significantly improves the computational efficiency. However, even though sparse convolution performs well in many large-scale point cloud tasks, it still faces challenges in capturing the fine-grained patterns of point clouds. This is due to the quantization artifacts that may be introduced during voxelization, resulting in the extracted features being limited by the voxel size [25].

Efficient network in computer vision. In computer vision, an efficient network typically refers to a deep learning architecture designed to balance performance and operational efficiency, including aspects like latency, FLOPs, memory, and power consumption. MobileNet [20] use depthwise separable convolutions, making them particularly efficient for mobile and embedded devices. EfficientNet [26] systematically scales the network's width, depth, and resolution, achieving a balance between efficiency and accuracy. Many works in 3D vision [27, 18, 19, 25] are dedicated to optimizing the efficiency and performance of point cloud processing. While such networks often slightly compromise on performance for reduced computational load, our network, as detailed in Section 4, uniquely enhances both efficiency and accuracy.

3 Linear Net

3.1 Problem Formulation

Given a 3D point cloud $\mathcal{V} = (\mathcal{P}, \mathcal{F})$ consisting of n points $\boldsymbol{x}$. For the $\boldsymbol{i}$-th point $\boldsymbol{x}_i = (\boldsymbol{p}_i, \boldsymbol{f}_i)$, $\boldsymbol{p}_i \in \mathbb{R}^3$ and $\boldsymbol{f}_i \in \mathbb{R}^c$ are the space coordinates and features, respectively. The task of point cloud semantic segmentation involves assigning a class label to each point $\boldsymbol{x}_i$, while scene classification entails predicting a class label for the entire scene $\mathcal{C}$. The point-based methods usually employ several stages to classify the points or point clouds. In each stage, a downsample layer is first applied to sample the points, reducing the density of the point cloud.

3.2 Disassembled Set Abstraction

In this section, we progressively introduce the DSA modules. Initially, we explore the direct application of separable convolutions in 3D vision. Subsequently, we adopt a balanced approach to separate channel and spatial anisotropy. Finally, we explain the superiority of this method from the perspective of weight initialization, highlighting how these adjustments enhance the overall performance of the network.

Revisiting Local Aggregation of Computer Vision. In a standard convolutional kernel, the anisotropy of the weights plays a critical role in capturing local information. This anisotropy can be classified into two categories: spatial and channel anisotropy. Spatial anisotropy refers to the variations among the features of neighboring points within the same feature channel, while channel anisotropy reflects the differences across various feature channels. To improve efficiency, MobileNet [20] achieves speedup by decomposing the standard convolutional kernel: it divides the convolutional kernel into point-wise convolution (PWConv) for channel anisotropy and depth-wise convolution (DWConv) for spatial anisotropy. Due to the sparsity of point clouds, it is impractical to apply DWConv to handle them directly. Instead of achieving anisotropy through parameters, point-based methods approach this by manipulating the input directly and adding anisotropy to the input data. Given a point cloud $\boldsymbol{x}_i$, a typical local aggregation in 3D vision can be formulated as:

$$\boldsymbol{f}_i' = \mathcal{R}_{j:(i,j)\in\mathcal{N}}\{\text{PWConv}^{3+c\mapsto c}(\boldsymbol{f}_j||(\boldsymbol{p}_j - \boldsymbol{p}_i))\}, \tag{1}$$

where $\mathcal{R}$ is the aggregation function (usually max-pooling) that aggregates the local feature from the neighbors of anchor point $\boldsymbol{x}_i$ denoted as $\{j : (i,j) \in \mathcal{N}\}$. $||$ is the concatenate operation in channels. $\text{PWConv}^{3+c\mapsto c} : \mathbb{R}^{3+c} \mapsto \mathbb{R}^c$ is an MLP that consists of pointwise convolution, a batch normalization layer, and a ReLU activation function. Here, the neighborhood features of the different anchors come from the same query set, so they are isotropic. The coordinates are anisotropic as they are relative to their respective anchor. Concatenating together the isotropic features and anisotropic relative coordinates gives the input anisotropy.

Depth-wise Separate Set Abstraction. Corresponding to the separate weights, we initially chose to separate the inputs directly as follows:

$$\boldsymbol{f}_i' = \mathcal{R}_{j:(i,j)\in\mathcal{N}}\{\text{PWConv}^{c\mapsto c}(\boldsymbol{f}_j)\} + \mathcal{R}_{j:(i,j)\in\mathcal{N}}\{\text{PWConv}^{3\mapsto c}((\boldsymbol{p}_j - \boldsymbol{p}_i))\}. \tag{2}$$

Since all f_j' come from the same set, Eq. (2) is identity to Eq. (3):

$$\begin{aligned} \overline{\boldsymbol{f}}_i &= \text{PWConv}^{c\mapsto c}(\boldsymbol{f}_i); \\ \boldsymbol{f}_i' &= \mathcal{R}_{j:(i,j)\in\mathcal{N}}\{\overline{\boldsymbol{f}}_j\} + \mathcal{R}_{j:(i,j)\in\mathcal{N}}\{\text{PWConv}^{3\mapsto c}((\boldsymbol{p}_j - \boldsymbol{p}_i))\}. \end{aligned} \tag{3}$$

This separation is necessary for several reasons. First, since the features are derived from the same query set, there is no need to apply a shared-weight PWConv on the neighboring features [18]. If the feature dimension is c and the number of neighbors is k, the computational complexity would be kc^2. After separation, this complexity is reduced to c^2. Second, the distribution patterns of coordinates and features differ significantly, and using independent convolutional kernels allows for better capture of

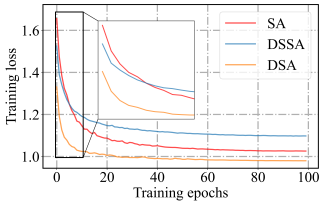
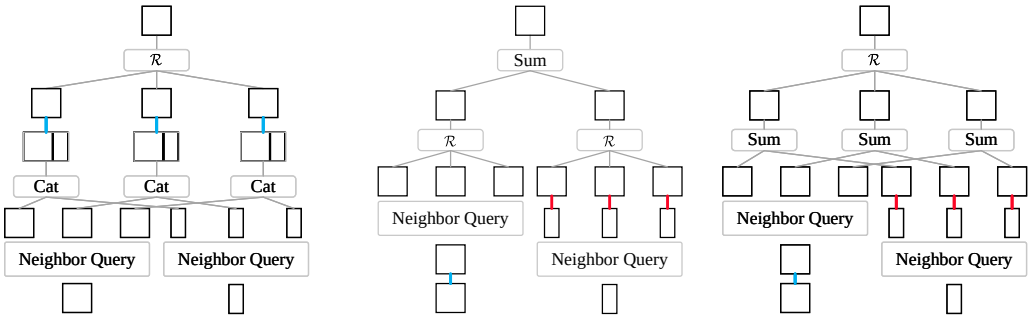


Figure 3: **Training on S3DIS**.



(a) SA (b) DSSA (c) DSA

Figure 4: **Comparison of various local aggregation.** Each square represents the semantic feature, while each rectangle represents relative coordinates. The number of neighbors is 3. The blue line indicates a mapping in a high-dimensional space (e.g., from $3+c$ to c, or c to c), and the red line indicates a mapping from a lower dimension to a higher dimension (e.g., from 3 to c). More blue lines indicate more computation.

these differences. However, as encountered with MobileNet, the speedup often comes at the cost of reduced accuracy. As illustrated in Fig. 3, DSSA accelerates the model's convergence during the initial few epochs. Yet, as training progresses, the convergence slows down, and the model's overall performance starts to degrade. The main reason lies in *the lack of spatial-wise anisotropy between neighbor features*. When features are processed independently of their spatial relationships, it can lead to a loss of contextual information critical for certain tasks, such as those involving complex spatial structures or detailed textural information.

Disassembled set abstraction. Building on the principles outlined above, we propose a novel method for lightweight local aggregation that addresses the inherent challenges of separating coordinates and features. The proposed disassembled set abstraction (DSA) module can be formulated as:

$$\begin{aligned} \overline{\boldsymbol{f}}_i &= \text{PWConv}^{c\mapsto c}(\boldsymbol{f}_i); \\ \boldsymbol{f}'_i &= \text{BN}\{\mathcal{R}_{j:(i,j)\in\mathcal{N}}\{\overline{\boldsymbol{f}}_j + \text{PWConv}^{3\mapsto c}((\boldsymbol{p}_j - \boldsymbol{p}_i))\}\}, \end{aligned} \tag{4}$$

where BN is a batch normalization layer [28]. The spatial anisotropy derived from relative positions is integrated into the neighborhood feature aggregation as a manner of bias. This integration ensures that the aggregation of neighborhood features is closely linked to the spatial distribution of the point cloud, thereby enhancing the model's robustness under varying spatial distributions. Interestingly, the Eq. (1) and Eq. (4) are actually mathematically equivalent during forward propagation. However, the DSA module exhibits faster convergence and lower loss compared to the SA modules (see Fig. 3). This phenomenon can be attributed to **variations in the initialization of weights** as follows.

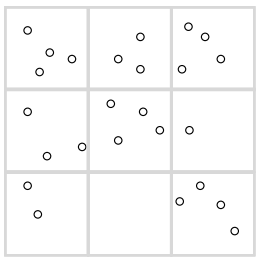
Excluding bias, Eq. (1) uses a combined weight matrix $\mathbf{W}$ (dimensions $c\times(c+3)$) to process semantic and geometric data simultaneously. For a given neighbor j, the input vector $\mathbf{x}_j = [\boldsymbol{f}_j, \boldsymbol{p}_j - \boldsymbol{p}_i]$ results in the output:

$$\mathbf{y}_j = \mathbf{W}\mathbf{x}_j^{\mathrm{T}}. \tag{5}$$

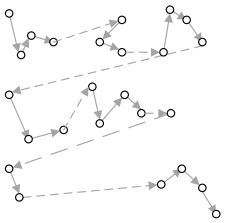
In this case, Kaiming initialization sets $\mathbf{W}$ as a normal distribution $\mathcal{N}(0, \sqrt{\frac{2}{c+3}})$, potentially reducing the impact of geometric data due to its smaller proportional weight. In stark contrast, the DSA module separates the processing of semantic and geometric data using two distinct weight matrices, $\mathbf{W}_f$ for semantic (dimensions $c \times c$) and $\mathbf{W}_p$ for geometric data (dimensions $c \times 3$), leading to:

$$\mathbf{y}_j = \mathbf{W}_f \boldsymbol{f}_j^{\mathrm{T}} + \mathbf{W}_p(\boldsymbol{p}_j - \boldsymbol{p}_i)^{\mathrm{T}}. \tag{6}$$

The specific initialization $\mathbf{W}_f \sim \mathcal{N}(0, \sqrt{\frac{2}{c}})$ and $\mathbf{W}_p \sim \mathcal{N}(0, \sqrt{\frac{2}{3}})$ allows for a more balanced influence of geometric data, thus enhancing the network's ability to extract and utilize geometric information effectively. It is noteworthy that the PWConv between high-dimensional semantic features is applied directly to the point cloud features, rather than to the neighborhood. Additionally, the number of input channels for PWConv applied to the neighborhood is only 3, which is significantly smaller than c (with a minimum of 64 in the segmentation model). As illustrated in Fig 4, DSA requires substantially fewer FLOPs compared to SA, thereby improving computational efficiency and making it more suitable for large-scale applications.



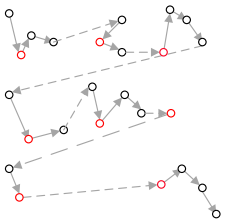
(a) Input points



(b) Space-filling curve

Cell	Start index	Count
0	0	4
1	4	3
2	7	4
3	11	3
...	...	...
6	19	2
7	21	4

(c) Hash table



(d) Linearization sampling

Figure 5: **Efficient point clouds searching for query and sampling.** (a) The input point cloud. (b) Point cloud after linearization by space-filling curves. Points connected by solid arrows are within the same grid, while dashed lines connect points between different grids. (c) Store each segment of the solid line as a hash table. (d) The point closest to the center of each region represented by segments connected by solid arrows is chosen as the new sampling point.

3.3 Point Searching Strategy

Recent literature [29, 30] employ space-filling curves to serialize point clouds, which are then uniformly divided into patches and fed into a transformer architecture. Inspired by this, we map points in 3D space onto a space-filling curve for accelerating point searching. Denote the coordinates $\boldsymbol{p}$ as (x, y, z) and the batch index as b. The shuffled key is defined as a 64-bit integer:

$$\text{Key} = (b \ll 54) \mid (\lfloor z/s \rfloor \ll 36) \mid (\lfloor y/s \rfloor \ll 18) \mid (\lfloor x/s \rfloor), \tag{7}$$

where $\ll$ denotes left bit-shift, s denotes the grid size, and $\mid$ denotes bitwise OR. It is worth noting that, unlike PTv3 [30], which contains only one point per grid cell, our approach allows multiple points to share the same key. We utilize shuffled keys to store these points in memory in an ordered manner, ensuring that elements sharing the same key are close to each other in memory. As shown in Fig. 5b, points on the same solid line are in the same grid. Through this, the point cloud $\mathcal{V}$ is partitioned into m sub-regions $[\mathcal{V}_1, \mathcal{V}_2, \dots \mathcal{V}_m]$.

Hash query. By storing coordinates in spatial order, a hash table can be constructed to efficiently manage and query neighbors. Each bucket represents a non-empty grid. The key is the shuffled key of the grid, and the value contains two parts: the index of the first point in the grid and the count of points in that grid. If the number of grids is m, the time complexity of building the hash table is $\mathcal{O}(m)$. During the query phase, for each point, we find the 27 (i.e., $3 \times 3 \times 3$) neighboring grids and query the hash table with these keys. Finally, the top k nearest points are selected. Assuming each point's 27-grid neighborhood contains p points on average, identifying the closest k points involves maintaining a heap with a complexity of $\mathcal{O}(p \log k)$ and a final sorting step costing $\mathcal{O}(k \log k)$. Thus, the total computational complexity is $\mathcal{O}(m + N(p \log k + k \log k))$, while that of kNN is $\mathcal{O}(kN^2)$.

Linearization sampling. To achieve uniform and fast sampling, we select a point from each subset according to the following rule:

$$\text{idx}_i = \arg \min_{j\,:\,(j) \in \mathcal{M}_i} \left(\|\boldsymbol{p}_j - \lfloor \boldsymbol{p}_j / s \rfloor \times s\|_2 \right), \tag{8}$$

where $(j) \in \mathcal{M}_i$ are the points of i-th sub-regions. This rule ensures that the newly sampled points are the closest to the origin within their respective grids, guaranteeing uniform sampling. The method has a linear time complexity and supports GPU parallelism, resulting in very fast sampling speeds.

3.4 Network Architecture

The overall architecture is illustrated in Fig. 2. For segmentation tasks, we use both encoders and decoders. To ensure a fair comparison, in the indoor dataset S3DIS, we configure the encoder depth as [4, 7, 4, 4], which is the same as PointNeXt. For the outdoor dataset, the encoder depth is set to [4, 4, 7, 4]. Specifically, the channel numbers for these stages are set to [C, 2C, 4C, 8C], with C being 64. For the classification task, only the encoder is used. Considering that the dataset for the classification task is small and PointNeXt is already capable of real-time response, we do not use the proposed linear search strategy, ensuring a fairer comparison between the DSA module and the SA module.

Table 1: Indoor sem. seg. on S3DIS Area 5.

Methods	Input	mIoU	OA	Acc
PointNet [7]	point	41.1	-	66.2
PointCNN [31]	point	-	-	75.6
PointWeb [32]	point	60.3	87.0	66.6
PointNet++ [8]	point	68.6	87.7	67.1
KPConv [33]	point	67.1	-	79.1
RandLA-Net [27]	point	-	-	82.0
PTv1 [9]	point	70.4	90.8	76.5
CBL [34]	point	69.4	90.6	-
PointMeta [17]	point	72.0	91.4	-
ASSANet [18]	point	66.8	-	-
Str. Trans. [21]	point	72.0	91.5	78.1
Fast PT [25]	point	70.1	-	77.3
PTv2‡ [11]	point	72.6	91.6	78.0
PTv3‡ [30]	point	73.4	-	-
ConDaFormer‡ [35]	point	73.5	**92.4**	78.9
PointVector [16]	point	72.3	91.0	78.1
PointNeXt [22]	point	70.8	91.7	77.5
LinNet (ours)	point	72.9	91.3	78.6
LinNet ‡ (ours)	point	**73.7**	91.9	**79.0**

Table 2: Outdoor sem. seg. on NuScenes.

Methods	Input	Val	Test
RandLA-Net [27]	point	-	-
KPConv [33]	point	-	-
RangeNet++ [36]	point	65.5	-
Salsanext [37]	hybrid	72.2	-
MinkUNet [4]	voxel	73.3	-
PolarNet [38]	point	71.0	-
PVKD [39]	hybrid	76.0	-
AMVNet [40]	-	76.1	-
Cylender3D [41]	cylender	76.1	77.2
SPVNAS [42]	hybrid	77.4	-
RPVNet [43]	hybrid	77.6	-
2DPASS‡ [44]	hybrid	-	80.8
RangeFormer [45]	-	78.1	80.1
SphereFormer [46]	voxel	78.4	81.9
WaffleIron‡ [47]	point	79.1	-
OACNN‡ [48]	voxel	78.9	-
PTv3‡ [30]	point	80.4	**82.7**
LinNet(ours)	point	80.4	-
LinNet ‡(ours)	point	**81.4**	82.3

4 Experiments

To validate the effectiveness of LinNet, we conduct experiments in 3D semantic segmentation and 3D object classification tasks. We also conduct an extensive ablation study to analyze each component in LinNet. More details of the experiments can be found in the Appendix.

4.1 Semantic Segmentation

Data and metric. S3DIS [49] (Stanford Large-Scale 3D Indoor Spaces) is a challenging benchmark that comprises 6 extensive indoor areas, 271 rooms, and 13 semantic categories, which represent different types of objects and room elements commonly found in indoor environments. Each point in the dataset is labeled with one of the 13 semantic categories, such as table, door, chair, column, and window, in addition to clutter. Following previous work [22], we subsample the grid before sending the point cloud to the network. The grid size and maximum number of points are set to 0.04m and 24000 respectively. During training, we crop the center point by the pre-set maximum number of points and discard the rest. During testing, the entire scene is processed. The experiment results are shown in Tab. 1. For evaluation metrics, we choose class-wise intersection over union (mIoU), mean of class-wise accuracy (mAcc), and overall point-wise accuracy (OA). Given that S3DIS is relatively small, we conduct further experiments on the NuScenes [50] dataset to validate the efficiency of our model. In this dataset, each point is annotated with one of the 16 semantic categories. The dataset encompasses 1,000 scenes collected in Boston and Singapore, reflecting diverse urban environments. We adhered to the official segmentation protocol, allocating 700 scenes for training, 150 for validation, and another 150 for testing, ensuring a balanced and comprehensive evaluation of our model's performance across varied scenes.

Performance. The results are shown in Tab. 1 and Tab. 2. Following Point Transformer v2 [11] and CondaFormer [35], we also employ the test time augmentation (TTA) strategy to achieve fairer comparisons, and results using the TTA strategy are labeled with ‡. In indoor dataset S3DIS, our LinNet outperforms the SoTA point-based method PointNeXt [22] by 2.9%, 0.2%, 2.5% in terms of mIoU, OA, and mAcc, respectively. We also visualize the segment result in Fig. 6. In large-scale dataset NuScenes, LinNet also outperforms all previous methods. It is worth noting that our approach utilizes a pure MLP network, employing solely point-wise convolutions. This highlights that sophisticated feature extractors, such as attention mechanisms and graph structures, are not essential for achieving robust segmentation capabilities. Moreover, our method is strictly based on point data, devoid of any sparse convolutions, which further underscores the scalability of point-based methods in managing large-scale point clouds effectively.

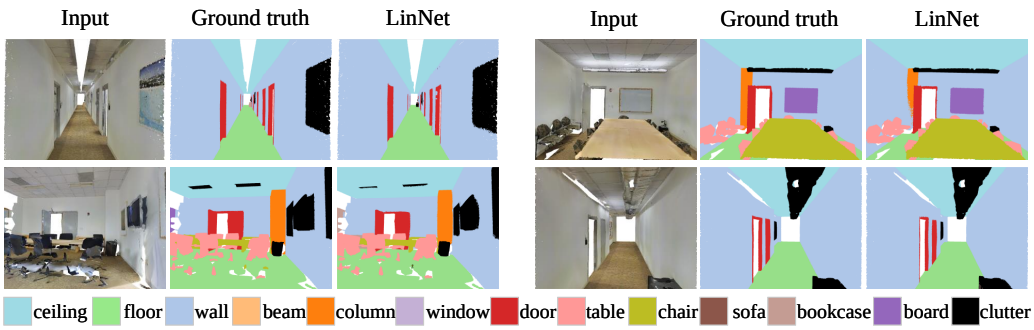


Figure 6: Comparative Visualization of Semantic Segmentation on S3DIS.

Table 3: **3D object classification in ScanObjectNN and ModelNet40.** Averaged results in three random runs using 1024 points as input without normals and without voting are reported.

Method	ScanObjectNN (PB_T50_RS) OA (%)	mAcc (%)	ModelNet40 OA (%)	mAcc (%)	Params. M	FLOPs G	Throughput (ins./sec.)
PointNet [7]	68.2	63.4	89.2	86.2	3.5	0.9	4199
PointCNN [14]	78.5	75.1	92.2	88.1	0.6	-	44
DGCNN [13]	78.1	73.6	92.9	90.2	1.8	4.8	458
DeepGCN [51]	-	-	93.6	90.9	2.2	3.9	-
KPConv [33]	-	-	92.9	-	14.3	-	-
ASSANet-L [18]	-	-	92.9	-	118.4	-	144
SimpleView [3]	80.5±0.3	-	93.0±0.4	90.5±0.8	0.8	-	-
MVTN [1]	82.8	-	93.5	92.2	3.5	1.8	2-
Point Cloud Transformer [10]	-	-	93.2	-	2.9	2.3	-
CurveNet [52]	-	-	93.8	-	2.0	-	-
PointMLP [23]	85.4±1.3	83.9±1.5	**94.1**	91.3	13.2	31.3	220
PointMetaBase [17]	87.9±0.2	-	-	-	-	0.6	-
PointNeXt [22]	87.7±0.4	85.8±0.6	93.7±0.3	90.9±0.5	1.4	1.6	2126
LinNet (ours)	**88.2±0.4**	**86.6±0.7**	93.6±0.2	**91.0±0.5**	1.4	0.6	1852

4.2 Object classification

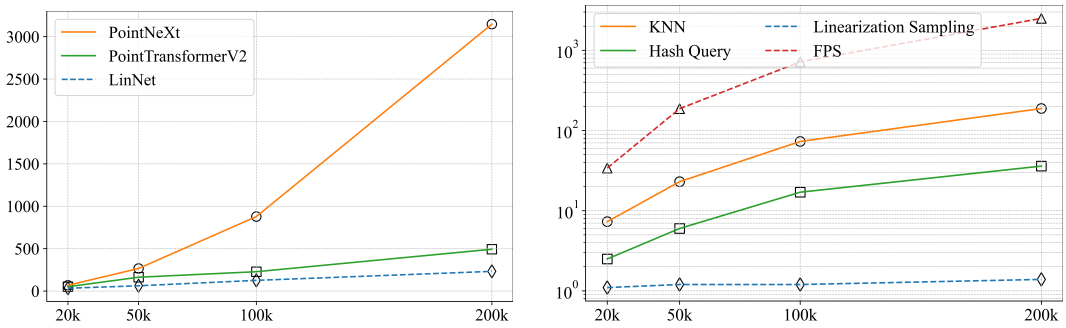
We chose the ScanObjectNN [53] and ModelNet40 [54] datasets to assess the classification capabilities of our model, and the results are shown in Fig. 3. We also report the parameters, FlOPs, and throughput. Following PointNeXt, the input channels of the models used in ScanobjectNN and ModelNet40 are 32 and 64 respectively. The model parameters are computed for C = 32.

ScanObjectNN. It contains approximately 15,000 real scanned objects divided into 15 categories with 2,902 instances, which presents substantial challenges due to occlusions and noise. We conduct experiments on PB_T50_RS, the most challenging and frequently used variant of ScanObjectNN. According to the report, the proposed LinNet significantly outperformed existing methods in Overall Accuracy (OA) and mean Accuracy (mAcc), using fewer model parameters and achieving faster processing speeds. LinNet achieved an OA of 88.6% and a mAcc of 87.3% on ScanObjectNN. Note that we employ the same training protocols and experimental conditions as the SOTA benchmark, PointNeXt. Nonetheless, we still achieve an OA improvement of 0.4%, while other PointNeXt style architectures (e.g., PointMetaBase [17]) using the same experimental setup only achieve an OA improvement of 0.1%.

ModelNet40. This dataset is a widely-used object classification dataset, comprises 12,311 3D computer graphics CAD models across 40 categories. Our results, as indicated, are highly competitive and exceed those of most previous methods. LinNet achieved an Overall Accuracy of 93.9% on ModelNet40, surpassing graph-based models like DGCNN [13], transformer-based models such as Point Transformer [9], and KPConv [33].

4.3 Model Efficiency

We evaluate the efficiency of our model at four different scales of points: 20k, 50k, 100k, and 200k. The down sampling rate is about 4. The experiments are performed on an RTX 3090. The models we compared include PointNeXt-XL [22] and Point Transformer v2 [11]. As shown in Fig. 7a, the



(a) Efficiency comparison of different models.

(b) Efficiency comparison of different components.

Figure 7: **Efficiency comparisons.** The horizontal axis represents the number of points in the input tensor and the vertical axis represents the running time in milliseconds.

Table 4: Model design ablation.

ID	LS	HQ	DDSA	DSA	mIoU	Latency(ms)
I					70.8	89
II	✓				72.0	45
III		✓			70.8	80
IV	✓	✓			72.0	38
V	✓	✓	✓		71.0	32
VI	✓	✓		✓	73.1	34

Table 5: Ablation on the DSA design.

ID	Methods	mIoU	Δ
(1)	Vanilla SA	72.0	-1.1
(2)	DSSA	71.0	-2.1
(3)	ASSA	70.5	-2.6
(4)	PosPool	69.9	-3.2
(5)	Avg. pooling	71.2	-1.9
(6)	DSA	73.1	-

latency of LinNet grows linearly with the scale of the point cloud. Point Transformer v2 employs grid pooling, a technique similar to our linearization sampling, both characterized by linear time complexity. However, thanks to the simplicity and efficiency of our disassembled set aggregation, our model exhibits only half the latency of Point Transformer v2. Notably, at the 200k level, our LinNet model operates 13 times faster than PointNeXt, demonstrating significant improvements in processing speed. We also explore the latency of the proposed point cloud search strategy. Note that the horizontal axis is on a logarithmic scale. As shown in Fig. 7b, linearization sampling and hash query based on space-filling curves leads to greater speedup as the point cloud size increases. The proposed linearization sampling can be up to a thousand times faster than FPS.

4.4 Ablation Study

We conduct ablation experiments of the model to verify the validity of each component, and all experimental results are averaged over three times in the S3DIS area 5 unless otherwise stated.

LinNet. We perform ablation experiments on different modules introduced in LinNet: linearization sampling (LS), hash query (HQ), depth-wise separate set abstraction (DSSA), and disassembled set abstraction (DSA), with the results shown in Tab. 4. The delay was measured on a 20k number of point clouds. The model used in Exp. I is PointNeXt, which is the baseline result of our design. In Exp. II through VI, we progressively incorporated each of our proposed components, improving the baseline accuracy to 73.1% and reducing the latency to 34 ms.

Disassembled set abstraction. In Tab. 5, we investigate the design of the feature aggregation module to improve the aggregation of semantic and geometric information. We utilize ASSA [18] and PosPool [19], instead of DSA module. Additionally, we evaluate the performance impact of replacing max pooling with average pooling. Exp. (1) and (2) show a 1% decrease in accuracy when using simple separation of inputs directly. Comparing Exp. (1) with Exp. (6) indicates that the proposed DSA module performs better than the SA module. Exp. (3) and (4), which employ ASSA and PosPool respectively, demonstrate performance degradation, highlighting that our data-driven approach outperforms the parameter-free strategy in merging low-dimensional coordinates and high-dimensional semantics. Exp. (5) shows that max pooling is more compatible with our network.

Model scalability. We refer to the default LinNet as LinNet-Base and designed two variants with different numbers of trainable parameters: LinNet-Small, which has one-tenth the parameters of LinNet-Base, and LinNet-Large, which has four times the parameters of LinNet-Base. We test the

Table 6: Model scalability. Latency and FLOPs are measured with 24k points.

Name	Channels	Depths	Param(M).	FLOPs (G)	Latency (ms)	mIoU(%)
Small	32	[2,2,2,2]	1.5	2.1	27	77.6
Base	64	[4,4,7,4]	16.5	7.8	34	80.4
Large	128	[4,4,7,4]	65.6	24.9	42	81.3

Table 7: Memory footprint during training and inference on the NuScenes dataset.

Model	Training Memory	Inference Memory
MinkUNet	2.6 GB	1.4 GB
PointNeXt	Out of Memory	Out of Memory
LinNet-Small	5.2 GB	4.9 GB
LinNet	16 GB	13 GB

performance of these models on the outdoor dataset NuScenes, and the results are summarized in Tab. 6. The mIoU on the validation set steadily improves with increasing model size. Additionally, since the models are linear, the increase in parameters does not result in significantly higher latency. Notably, without using TTA, our LinNet-Small achieves a validation accuracy of 77.6% with only 1.7M parameters, surpassing the 38M parameter sparse convolution method MinkUnet [4].

Memory footprint. We conduct experiments to evaluate memory usage during both training and inference phases on the NuScenes dataset, utilizing an RTX 4090 graphics card with all tests conducted at a batch size of 1. We include comparisons with the baseline model PointNeXt [22] and the sparse convolution method MinkUNet [4]. Our findings reveal that PointNeXt suffers from out-of-memory issues when handling large-scale scenes, highlighting scalability challenges. In contrast, our DSA module significantly reduces memory consumption by avoiding high-dimensional feature transformations on neighboring point clouds. Given that MinkUNet starts with 32 input channels, we conducted similar tests with our LinNet-Small model, which also has 32 initial feature channels, for a direct comparison. The results are shown in Tab. 7. Although LinNet-Small consumes more memory than MinkUNet, it is crucial to note that LinNet-Small, with only 1.7M parameters, achieves a validation accuracy of 77.6%, surpassing the 38M parameter sparse convolution method MinkUNet, which achieves 73.3%.

5 Conclusion

Conclusion. In this paper, we implement a point-based approach with linear complexity. Unlike current point-based methods, our framework uses space-filling curves to achieve neighbor query and downsampling with linear complexity. Additionally, we introduce a disassembled set aggregation module, which aggregates local features simply and elegantly, significantly reducing the redundant computations in neighborhoods and greatly enhancing scalability. Extensive experiments on multiple benchmarks demonstrate the efficiency and state-of-the-art performance of our method.

Limitation and Future Work. Although the proposed approach largely addresses the scalability challenges of point-based approaches in large-scale scenes, the distribution of point cloud data in memory in point-based approaches tends to be discontinuous, leading to inefficient memory access. This increases the cache miss rate, which in turn reduces the processing speed. Point-wise neighborhood aggregation also consumes a significant amount of memory. We hope that future work will address the significantly higher memory footprint than sparse convolution methods.

Acknowledgment. The authors would like to thank the reviewers of NeurIPS'24 for their constructive suggestions. This work was supported by the Key Research and Development Program of Shaanxi Province of China under Grant 2024GX-YBXM-149, in part by the National Natural Science Foundation of China under Grant 42271140, and in part by Northwest University Graduate Innovation Project under Grant CX2024194.

References

[1] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *CVPR*, 2021. 1, 8

[2] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015. 1

[3] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *ICML*, 2021. 1, 8

[4] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 1, 7, 10, 15

[5] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015. 1, 3

[6] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *NeurIPS*, 2019. 1

[7] 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. 1, 7, 8, 15

[8] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 2017. 2, 7, 15

[9] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *CVPR*, 2021. 2, 3, 7, 8, 15

[10] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *CVM*, 7:187–199, 2021. 2, 8

[11] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *NeurIPS*, 2022. 2, 7, 8

[12] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, 2019. 2

[13] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 38(5):1–12, 2019. 2, 8, 15

[14] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *NeurIPS*, 2018. 2, 8, 15

[15] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019. 2

[16] Xin Deng, WenYu Zhang, Qing Ding, and XinMing Zhang. Pointvector: A vector representation in point cloud analysis. In *CVPR*, 2023. 2, 3, 7

[17] Haojia Lin, Xiawu Zheng, Lijiang Li, Fei Chao, Shanshan Wang, Yan Wang, Yonghong Tian, and Rongrong Ji. Meta architecure for point cloud analysis. *arXiv:2211.14462*, 2022. 2, 3, 7, 8

[18] Guocheng Qian, Hasan Hammoud, Guohao Li, Ali Thabet, and Bernard Ghanem. Assanet: An anisotropic separable set abstraction for efficient point cloud representation learning. *NeurIPS*, 2021. 2, 3, 4, 7, 8, 9, 15

[19] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. A closer look at local aggregation operators in point cloud analysis. In *ECCV*, 2020. 2, 3, 9

[20] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 3, 4

[21] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *CVPR*, 2022. 3, 7

[22] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In *NeurIPS*, 2022. 3, 7, 8, 10, 15

[23] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *ICLR*, 2022. 3, 8

[24] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 3

[25] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. In *CVPR*, 2022. 3, 7

[26] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 3

[27] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Learning semantic segmentation of large-scale point clouds with random sampling. *PAMI*, 44(11):8338–8354, 2021. 3, 7

[28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[29] Peng-Shuai Wang. Octformer: Octree-based transformers for 3D point clouds. *ACM Transactions on Graphics (SIGGRAPH)*, 42(4), 2023. 6

[30] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *CVPR*, 2024. 6, 7

[31] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, 2019. 7

[32] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019. 7

[33] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019. 7, 8, 15

[34] Liyao Tang, Yibing Zhan, Zhe Chen, Baosheng Yu, and Dacheng Tao. Contrastive boundary learning for point cloud segmentation. In *CVPR*, 2022. 7

[35] Lunhao Duan, Shanshan Zhao, Nan Xue, Mingming Gong, Gui-Song Xia, and Dacheng Tao. Condaformer: Disassembled transformer with local structure enhancement for 3d point cloud understanding. In *NeurIPS*, 2023. 7

[36] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *IROS*, 2019. 7, 15

[37] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In *ISVC*, 2020. 7, 15

[38] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *CVPR*, 2020. 7, 15

[39] Yuenan Hou, Xinge Zhu, Yuexin Ma, Chen Change Loy, and Yikang Li. Point-to-voxel knowledge distillation for lidar semantic segmentation. In *CVPR*, 2022. 7, 15

[40] Venice Erin Liong, Thi Ngoc Tho Nguyen, Sergi Widjaja, Dhananjai Sharma, and Zhuang Jie Chong. Amvnet: Assertion-based multi-view fusion network for lidar semantic segmentation, 2020. 7, 15

[41] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *CVPR*, 2021. 7, 15

[42] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *ECCV*, 2020. 7, 15

[43] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. In *CVPR*, 2021. 7, 15

[44] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *ECCV*, 2022. 7, 15

[45] Lingdong Kong, Youquan Liu, Runnan Chen, Yuexin Ma, Xinge Zhu, Yikang Li, Yuenan Hou, Yu Qiao, and Ziwei Liu. Rethinking range view representation for lidar segmentation. In *CVPR*, 2023. 7, 15

[46] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *CVPR*, 2023. 7, 15

[47] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Using a waffle iron for automotive point cloud semantic segmentation. In *CVPR*, 2023. 7, 15

[48] Bohao Peng, Xiaoyang Wu, Li Jiang, Yukang Chen, Hengshuang Zhao, Zhuotao Tian, and Jiaya Jia. Oa-cnns: Omni-adaptive sparse cnns for 3d semantic segmentation. In *CVPR*, 2024. 7

[49] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, 2016. 7, 14

[50] 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. 7, 14

[51] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *CVPR*, 2019. 8, 15

[52] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the cloud: Learning curves for point clouds shape analysis. In *CVPR*, 2021. 8

[53] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, 2019. 8, 14

[54] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 8, 14

[55] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *CVPR*, 2019. 14

[56] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ToG*, 35(6):1–12, 2016.

[57] Haoxi Ran, Jun Liu, and Chengjie Wang. Surface representation for point clouds. In *CVPR*, 2022. 15

[58] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, 2018.

Appendix

In the appendix, we provide more experiment details in Sec. A, and more experiment results in Sec. B.

A Experimental Details

This section provides a detailed description of the experimental setup for each dataset.

Experimental environment.

- CUDA version: 11.3
- PyTorch version: 1.12.1
- GPU: Nvidia RTX 4090D $\times$ 4
- CPU: AMD EPYC 9754 128-Core

Training details. The specific model training settings are shown in Tab. 8 and Tab. 9. We used cross-entropy loss in all experiments.

Table 8: Data augmentation.

	Rotate	Flip	Scale	Jitter	Chromatic Drop	Height	Grid Size
ScanObjectNN	✓		✓			✓	
ModelNet40			✓				
S3DIS	✓		✓	✓	✓	✓	0.04
NuScenes	✓	✓	✓	✓			0.05
Sem. KITTI	✓	✓	✓	✓			0.05

Table 9: Training setting.

	Epoch	Learning Rate	Weight Decay	Scheduler	Optimizer	Batch Size
ScanObjectNN	250	0.002	0.05	Cosine	AdamW	32
ModelNet40	600	0.001	0.05	Cosine	AdamW	32
S3DIS	3000	0.01	0.0001	Cosine	AdamW	8
NuScenes	50	0.002	0.05	Cosine	AdamW	12
Sem. KITTI	50	0.002	0.05	Cosine	AdamW	12

Data license. Our experiments use open-source datasets widely applied for 3D recognition research. The ScanObjectNN [53], SemanticKITTI [55], dataset is under the MIT license, while S3DIS [49], NuScenes [50], and ModelNet40 [54] have custom licenses that only allow academic use.

B Additional Quantitative Results

In this section, we present additional quantitative results of SemanticKITT [55] for 3D semantic segmentation. In addition, we provide semantic segmentation results for each category of NuScenes (see Tab. 10) and S3DIS Area 5 (see Tab. 11).

SemanticKITTI. The SemanticKITTI dataset consists of sequences from the original KITTI dataset, comprising a total of 22 sequences. Each sequence contains approximately 1,000 LiDAR scans, amounting to around 20,000 individual frames. The result is shown in Tab. 12. The mIoU of validation set and test set are 69.1% and 70.4% respectively.

S3DIS 6-fold cross-validation. To evaluate the generalization capabilities, we perform 6-fold cross-validation on the S3DIS dataset to ensure a robust assessment of our model's performance across different subsets of data. The results are shown in Fig. 13.

Nomalization layer type. We conducted ablation studies on the S3DIS dataset to further assess the necessity and effectiveness of BN. As shown in Tab. 14, models with BN outperform those with Layer Normalization (LN) and without any normalization, indicating that BN is particularly effective for our specific architecture.

Table 10: Semantic segmentation results on NuScenes val set. ‡ denotes using rotation and translation testing-time augmentations.

Method	mIoU	barrier	bicycle	bus	car	construction	motorcycle	pedestrian	traffic cone	trailer	truck	driveable	other flat	sidewalk	terrain	manmade	vegetation
RangeNet53++ [36]	65.5	66.0	21.3	77.2	80.9	30.2	66.8	69.6	52.1	54.2	72.3	94.1	66.6	63.5	70.1	83.1	79.8
PolarNet [38]	71.0	74.7	28.2	85.3	90.9	35.1	77.5	71.3	58.8	57.4	76.1	96.5	71.1	74.7	74.0	87.3	85.7
Salsanext [37]	72.2	74.8	34.1	85.9	88.4	42.2	72.4	72.2	63.1	61.3	76.5	96.0	70.8	71.2	71.5	86.7	84.4
AMVNet [40]	76.1	79.8	32.4	82.2	86.4	**62.5**	81.9	75.3	**72.3**	**83.5**	65.1	**97.4**	67.0	**78.8**	74.6	90.8	87.9
Cylinder3D [41]	76.1	76.4	40.3	91.2	93.8	51.3	78.0	78.9	64.9	62.1	84.4	96.8	71.6	76.4	75.4	90.5	87.4
PVKD [39]	76.0	76.2	40.0	90.2	**94.0**	50.9	77.4	78.8	64.7	62.0	84.1	96.6	71.4	76.4	76.3	90.3	86.9
RPVNet [43]	77.6	78.2	43.4	92.7	93.2	49.0	85.7	80.5	66.0	66.9	84.0	96.9	73.5	75.9	76.0	90.6	88.9
2DPASS [44] ‡	79.4	78.8	49.6	95.6	93.6	60.0	84.1	82.2	67.5	72.6	88.1	96.8	72.8	76.2	**76.5**	89.4	87.2
SphereFormer [46]‡	79.5	78.7	46.7	95.2	93.7	54.0	88.9	81.1	68.0	74.2	86.2	97.2	74.3	76.3	75.8	91.4	89.7
LinNet(ours)	80.4	79.2	54.6	96.6	93.2	53.9	89.0	83.7	70.6	73.3	88.5	96.9	73.8	76.1	75.4	91.5	89.7
LinNet‡(ours)	**81.4**	**80.0**	**56.9**	**96.9**	**94.0**	58.4	**90.0**	**84.4**	72.1	74.2	**89.7**	97.0	**74.4**	76.8	76.0	**91.7**	**89.9**

Table 11: Semantic segmentation results on S3DIS Area 5.

Method	OA	mAcc	mIoU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet[7]	-	49.0	41.1	88.8	97.3	69.8	**0.1**	3.9	46.3	10.8	59.0	52.6	5.9	40.3	26.4	33.2
PointNet++[8]	83.0	-	53.5	-	-	-	-	-	-	-	-	-	-	-	-	-
PointCNN[14]	85.9	63.9	57.3	92.3	98.2	79.4	0.0	17.6	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
DGCNN[13]	83.6	-	47.9	-	-	-	-	-	-	-	-	-	-	-	-	-
DeepGCN[51]	-	-	52.5	-	-	-	-	-	-	-	-	-	-	-	-	-
KPConv[33]	-	72.8	67.1	92.8	97.3	82.4	0.0	23.9	58.0	69.0	81.5	91.0	75.4	75.3	66.7	58.9
ASSANet-L[18]	-	-	66.8	-	-	-	-	-	-	-	-	-	-	-	-	-
Point Trans.[9]	90.8	76.5	70.4	94.0	98.5	**86.3**	0.0	38.0	63.4	74.3	82.4	89.1	74.3	80.2	76.0	59.3
RepSurf-U[57]	90.2	76.0	68.9	-	-	-	-	-	-	-	-	-	-	-	-	-
PointVector[57]	91.0	78.1	72.3	**95.1**	**98.6**	85.1	0.0	41.4	60.8	76.7	84.4	92.1	82.0	77.2	**85.1**	61.4
PointNeXt [22]	90.7	77.5	70.8	94.2	98.5	84.4	0.0	37.7	59.3	74.0	83.1	91.6	77.4	77.2	78.8	60.6
LinNet(ours)	**91.9**	**79.0**	**73.7**	94.8	98.5	86.2	0.0	**45.5**	**61.6**	**82.8**	**85.1**	**92.3**	**85.5**	**80.0**	80.4	**65.4**

Table 12: Sem. seg. on Sem. KITTI.

Methods	Val	Test
SPVNAS [42]	64.7	66.4
Cylinder3D [41]	64.3	67.8
PVKD [39]	-	71.2
2DPASS [44]	69.3	72.9
WaffleIron [47]	68.0	70.8
SphereFormer [46]	67.8	74.8
RangeFormer [45]	67.6	73.3
MinkUNet [4]	63.8	-
LinNet (ours)	69.1	70.4

Table 13: S3DIS 6-fold cross-validation.

Methods	mIoU (%)	mAcc (%)	OA (%)
PointNeXt	74.9	83.0	90.3
LinNet	78.6	86.3	91.9

Table 14: Normalization layer.

None	BN	LN
71.8%	72.9%	71.9%

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: **[TODO]**

   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: **[TODO]**

   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: **[TODO]**

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: **[TODO]**

   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: [NA]

Justification: Our code will be available upon publication.

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, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: **[TODO]**

   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: [Yes]

   Justification: **[TODO]**

   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: **[TODO]**

   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: **[TODO]**

   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: **[TODO]**

    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 poses no such risks

    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: **[TODO]**

    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: **[TODO]**

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: **[TODO]**

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: **[TODO]**

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.