Adaptive SphereFormer: Dynamic Radial Windows for Better Sparse Learning

Haritheja Etukuru
NYU
hre7290@nyu.edu

Yaswanth Orru
NYU
yko207@nyu.edu

Rhitvik Sinha
NYU
rs8438@nyu.edu

Abstract

LiDAR 3D segmentation in LiDAR-based point clouds is essential for various applications, such as autonomous vehicles and robotics. But accurate and efficient segmentation of point clouds remains a critical challenge, specifically due to the issue of information disconnection and limited receptive fields for distant sparse points. The current state-of-the art model SphereFormer, employs radial window self-attention to partition the space into multiple non-overlapping, narrowly elongated windows. Building upon SphereFormer, in this work we introduce a modified approach 'Adaptive SphereFormer' characterized by its dynamic adjustment of the radial angle, θ, contingent upon the distance of a point from the origin. This dynamic adjustment approach not only adeptly manages memory usage associated with large static window sizes, but also enhances segmentation accuracy, achieving an increment of around 1% in the overall mean Intersection over Union(mIOU). Our results demonstrate that this adaptive strategy boosts the model's capacity to process distant, sparser points, while simultaneously optimizing computational resources. The GitHub code can be found here.

1. Introduction

The extensive use of LiDAR technology has significantly advanced 3D vision modeling, especially in point cloud segmentation. Despite notable advancements, a consistent obstacle remains in effectively managing distant, sparse points within LiDAR datasets. Due to their lower density, these points pose a challenge in gathering information from neighboring points, as there are fewer neighbors available. This situation leads to issues of information disconnection and a limited receptive field. Fig. 1 clearly shows the how density of points varies with distance.

Early methods in this field processed the point cloud by partitioning the 3D space into a series of 3D voxels with cubic windows as shown in Fig. 2a using variants of sparse convolutions. However, this approach faced challenges in determining the optimal voxel size. The smaller voxels led to inadequate information transfer, whereas the larger voxels presents memory constraints. To overcome these limitations, various techniques were employed to expand the receptive fields of these points, such as window self-attention [10], dilated self-attention [12], and large-kernel CNNs [3]. These methods primarily use a strategy of layering local operators to increase the receptive field. However, in areas far from the point of origin where the space is exceedingly sparse, these models, even with multiple layers, struggle to grasp the full scope of the scene. This highlights the importance of having sufficiently informative lower-level features to enhance the performance gained from layering multiple levels.

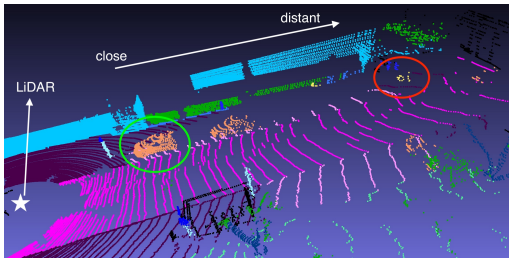
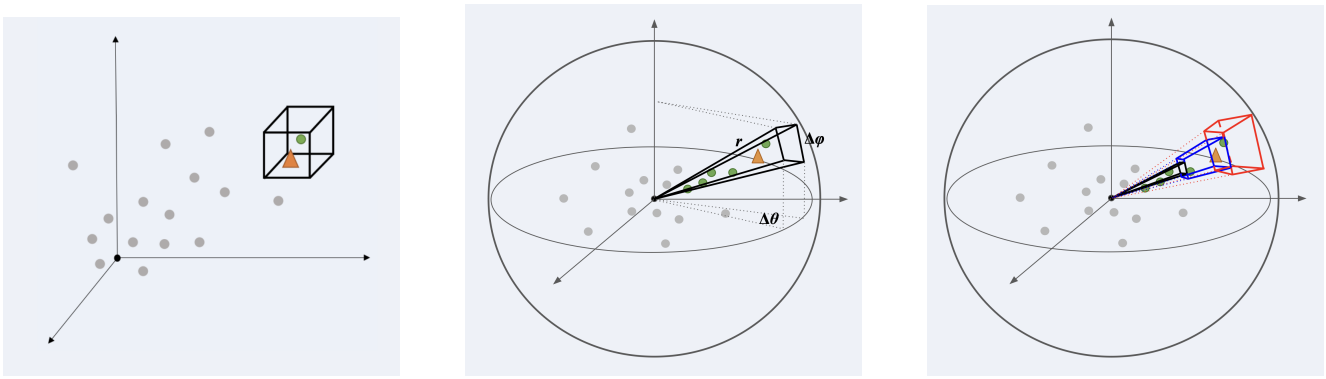


Figure 1. Varying-sparsity property of LiDAR point clouds. The dense close car is marked with a green circle and the sparse distant bicycle is marked with a red circle.

SphereFormer [9] addressed this issue by introducing radial windows as opposed to traditional cubic windows as shown in Fig. 2b. It utilizes spherical coordinates (r, θ, ϕ) to represent the 3D space, dividing the scene into several distinct, non-overlapping windows. Within each window, the model applies attention across all points to derive the final representation of a point. This design leads to long and narrow windows. As a result, when the model applies attention, even to points that are farther away, it can incorporate information from a broader range of points, leading to richer and more detailed representations.

Despite the improvement of using elongated windows in SphereFormer, a limitation arises from the fixed width of these radial windows, echoing the same challenge faced with cubic windows in determining the optimal window



(a) Cubic Window (b) Static Radial Window (c) Discrete Dynamic Radial Window

Figure 2. (a) Used in Both, (b) Used in SphereFormer, (c) Used in Adaptive SphereFormer.

width without compromising performance. To address this, in our approach we progressively expanded the window size with distance from the origin as shown in Fig. 2c. This adaptation results in windows that are not only longer, but also wider for points further away, thereby enabling the inclusion of an even greater number of points for analysis.

2. Related Work

Our work, which focuses on enhancing LiDAR-based 3D segmentation through dynamic radial window self-attention, is deeply rooted in the advancements of 3D LiDAR segmentation and the utilization of sparse convolutions. In this context, we review key contributions and methodologies that have shaped the current landscape of this field.

2.1. LiDAR 3D Segmentation

The segmentation of 3D point clouds, particularly those obtained from LiDAR, is a crucial task with various applications such as autonomous driving and robotics. Traditional methods in this domain can be broadly categorized into view-based, point-based, and voxel-based approaches.

View-based Methods: These methods involve transforming LiDAR point clouds into a range view [13, 16], or a bird-eye view [18], simplifying the three-dimensional data into a more manageable two-dimensional format for feature extraction. This approach simplifies the processing of 3D data but, it leads to a loss of critical geometric information.

Point-based Methods: This approach directly works with the raw point cloud data, leveraging point features and positions, focusing on capturing the intricate details present in the data [10, 14, 15].

Voxel-based Methods: Solutions involve dividing the 3D space into regular voxels (volumetric pixels) [4, 8] and then applying sparse convolutions. This approach is particularly effective in managing the sparsity of LiDAR data, allowing for efficient processing of large-scale point clouds.

2.2. Sparse Convolutions

Sparse convolutions have become a crucial technique in deep learning, particularly in the fields of computer vision and high-dimensional neural networks. They provide a computationally efficient alternative to traditional convolutions by focusing on non-zero elements in the data, which is advantageous for handling sparse inputs. The concept of sparse convolutions was initially introduced to address the inefficiency of standard convolutions in handling sparse data. It was shown that sparse structures in deep neural networks has significant computational savings without compromising performance [7]. The development of Sparse Convolutional Neural Networks (SCNNs) was a significant milestone, as it proposed an optimized SCNN that efficiently processed high-dimensional data [2], and demonstrated its effectiveness in tasks such as 3D object detection and segmentation. Furthermore, the use of sparse convolutions in processing 3D data, particularly in the context of LiDAR and point cloud processing, has also been worked on [14], leveraging sparse convolutions to significantly improve the efficiency and accuracy of 3D object detection models.

2.3. Vision Transformers

The utilization of Vision Transformers (ViT) in the field of LiDAR-based 3D point cloud recognition represents a significant shift from conventional convolutional approaches, offering new avenues for capturing complex spatial relationships. Our work with the Adaptive SphereFormer, which incorporates radial window self-attention, is deeply influenced by the evolution of Vision Transformers and their application in 3D point cloud processing.

In 2D Image Processing: Vision Transformers (ViT) [5] have revolutionized 2D image processing. Initially

designed for language tasks, ViT adapts to image data by tokenizing image patches and leveraging Transformer encoders. Variants like the Pyramid Vision Transformer (PVT) [17] and Swin Transformer [11] introduced hierarchical structures and window-based attention, enhancing the model's capability to capture long-range dependencies in 2D images.

Adaptation to 3D Point Clouds: The application of Transformers to 3D point cloud data, such as LiDAR, is challenging due to data sparsity and irregularity. As discussed above, initial adaptations of 2D Transformer models to 3D data often overlooked these unique aspects. However, recent methods have started to tailor Transformer architectures to better handle the varying sparsity of LiDAR point clouds.

Radial Window Self-Attention: The SphereFormer represents a significant advancement in this area, introducing radial window self-attention specifically for LiDAR-based 3D recognition. This technique partitions the space into elongated windows, effectively connecting dense and sparse regions of the point cloud. This approach is particularly effective in processing distant points, addressing a key challenge in LiDAR data interpretation.

3. Baseline Architecture

The SphereFormer module functions as a versatile plugin that seamlessly integrates into established mainstream models like SparseConvNet [7, 14], MinkowskiNet [4], and local window self-attention [10].

3.1. Transformer

As illustrated in Fig. 2, SphereFormer adopts a unique approach by partitioning the space using radial windows instead of traditional cubic windows. The space is divided into radial windows with widths defined by $\Delta\theta$ and $\Delta\phi$ in the (r, θ, ϕ) co-ordinate space. Consequently, a point p_i is associated with the window $(\frac{\theta_i}{\Delta\theta}, \frac{\phi_i}{\Delta\phi})$. Subsequently, all the points within the radial window undergo attention operations according to the following equations

$$\hat{q} = f \cdot W_q, \hat{k} = f \cdot W_k, \hat{v} = f \cdot W_v$$

where $f \in \mathbb{R}^{n\times c}$ denotes the input features of the window, $W_q, W_k, W_v \in \mathbb{R}^{c\times c}$ are the linear projection weights. The projected features $\hat{q}, \hat{k}, \hat{v} \in \mathbb{R}^{n\times c}$ are then split into h heads resulting in vectors $q, k, v \in \mathbb{R}^{h\times n\times d}$. For each head, a dot product is performed between queries and keys to obtain attention weights, and the final features are computed as the weighted sums of values,

$$attn_k = softmax(q_k \cdot k_k^T)$$

$$\hat{z}_k = attn_k \cdot v_k$$

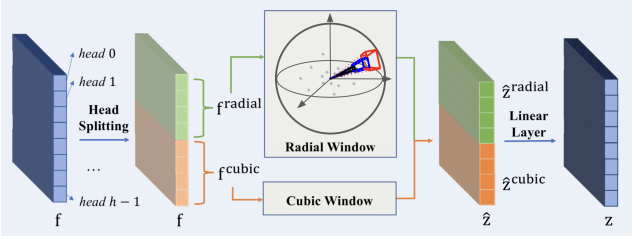


Figure 3. Adaptive SphereFormer Architecture

where $q_k, k_k, v_k \in \mathbb{R}^{n\times d}$ represent the features of k^{th} head, and $attn_k \in R^{n\times n}$ represents the corresponding weights. Finally, features from all the heads are concatenated and linear projected with Weights $W_{proj} \in \mathbb{R}^{c\times c}$ to obtain the final output $z \in \mathbb{R}^{n\times c}$

$$\hat{z} = concat(\hat{z_0}, \hat{z_1}, \cdots, \hat{z}_{h-1})$$

$$z = \hat{z} \cdot W_{proj}.$$

This structure is stacked multiple times to obtain richer representations.

3.2. Positional Embedding

In the context of the 3D point cloud network, the input features already include the absolute xyz positions, eliminating the need for the application of absolute position encoding. Stratified Transformer [10] attempts to add relative position embeddings, by dividing the window into uniform intervals. This division aims to convert continuous relative positions into integers. While this strategy is effective for smaller cubic windows, it encounters challenges when applied to large and narrow windows. So, the SphereFormer adopted an exponential splitting strategy which creates finer intervals as we move closer to the center as illustrated in Fig. 4 So, now given two points in a radial window (r_i, θ_i, ϕ_i) and (r_j, θ_j, ϕ_j) the relative position embedding $(idx_{ij}^r, idx_{ij}^\theta, idx_{ij}^\phi)$ is given by

$$idx_{ij}^r = \begin{cases} -\max(0, \log_2\left(\frac{-r_{ij}}{a}\right)) - 1 & \text{if } r_{ij} < 0 \\ 0 & \text{if } r_{ij} = 0 \\ \max(0, \log_2\left(\frac{r_{ij}}{a}\right)) & \text{if } r_{ij} > 0 \end{cases}$$

$$idx_{ij}^\theta = \left\lfloor \frac{\theta_{ij}}{interval_\theta} \right\rfloor, \quad idx_{ij}^\phi = \left\lfloor \frac{\phi_{ij}}{interval_\phi} \right\rfloor,$$

$$idx^x = idx^x + \frac{L}{2}, \quad x \in \{r, \theta, \phi\},$$

where **a** is a hyper-parameter to control the starting splitting interval, and L is the length of the positional embedding tables. $\frac{L}{2}$ is added to indices to make sure they are non-negative.

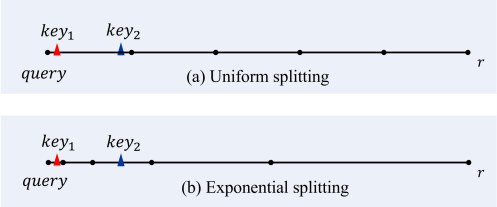


Figure 4. Exponential Splitting

The above indices $(idx_{ij}^{r}, idx_{ij}^{\theta}, idx_{ij}^{\phi})$ are then used to index their positional embedding tables $t_r, t_\theta, t_\phi \in \mathbb{R}^{L\times h\times d}$ to find the corresponding position encoding $p_{ij}^{r}, p_{ij}^{\theta}, p_{ij}^{\phi} \in \mathbb{R}^{h\times d}$, respectively. Then, we sum them up to yield the resultant positional encoding $\mathbb{P} \in \mathbb{R}^{h\times d}$, which then performs dot product with the features of q_i and k_j, respectively. Then this positional encoding is incorporated into attention mechanism such that weights are proportional to the distance between the points. The final attn equation is as follows

$$p = p_j^r + p_{ij}^\theta + p_{ij}^\phi, \tag{1}$$

$$\text{pos\_bias}_{k,i,j} = q_{k,i}^T \cdot p_k^T + k_{k,j}^T \cdot p_k^T, \tag{2}$$

$$\text{attn}_k = \text{softmax}(q_k \cdot k_k^T + \text{pos\_bias}_k), \tag{3}$$

where pos_bias $\in \mathbb{R}^{h\times n\times n}$ is the positional bias to the attention weight, $q_{k,i} \in \mathbb{R}^d$ means the the k-th head of the i-th query feature, and $p_k \in \mathbb{R}^d$ is the k-th head of the position encoding p.

Dynamic Feature Selection Prior studies have demonstrated that cubic windows are effective for points that are closely packed together. Building on this, the SphereFormer allocates half of the multi-head attention mechanism's heads to process fixed-size cubic attention. The remaining heads were tasked with calculating radial window attention, as previously detailed. This design allows the model to selectively focus on different aspects: it may opt for the more detailed radial window view for distant points, while preferring the cubic attention for points that are nearer.

3.3. Adaptive SphereFormer

To adapt to the variable sizes of the windows, the radial window was segmented into three ranges: from $r \in [0, 20)$, $r \in [20, 50)$, and $r \in [50, \infty)$. These specific intervals were chosen based on significant changes observed in the results Fig. 5 of the SphereFormer paper. A challenge arises with points that lie at the boundary of these radial segments since they would only be aligned with adjacent points that are only within the same segment. To counteract this, each window is extended by a margin s_{win} as implemented in Lai et al. (2022) [10], into adjacent windows for better integration of edge points.

As illustrated in Fig. 2, what was initially a single black interval has been subdivided into three distinct zones, which are marked by black, blue, and red. This division allows to follow a different width in θ and ϕ direction. The final structure of the model is represented in Fig. 2c, which showcases these three divided intervals.

4. Dataset

SemanticKITTI [1] is a large-scale outdoor scene data set for point-cloud semantic segmentation. It is derived from the KITTI Vision Odometry Benchmark [6] which it extends with dense point-wise annotations for the complete 360 field-of-view of the employed automotive LiDAR. The dataset consists of 22 sequences. Overall, the dataset provides 19130 point clouds for training and 4107 for testing.

5. Implementation & Training Details

The underlying model structure is a UNet-based structure with five levels with feature dimensions of [32, 64, 128, 256, 512]. At each stage, the SphereFormer is integrated into the system. For the baseline SphereFormer setup, the window sizes for both $\Delta\theta$ and $\Delta\phi$ are set at 2.5°. Training was carried out using a batch size of 32 degrees and a poly rate scheduler with a power of 0.9. The learning rate was set at 0.006, and the weight decay parameter was set to 0.01. In the Adaptive SphereFormer version of the model, the window sizes for the θ and ϕ axes were adjusted to 1.5° in $r \in [0, 20)$, 2° in $r \in [20, 50)$ and 2.5° in $r \in [50, \infty)$ with s_{win} of 0.5° in θ and ϕ directions and 10m in radial direction. Training was carried out on two RTX 8000 GPUs, each with 48GB of memory.

6. Experiments

Initially, we examined the correlation between the mean Intersection over Union (IoU) and the distance from the point of origin in Fig. 5. There was a noticeable decrease in IoU at a radial distance of 20 units, followed by a sharp decrease between radial distances of 40 to 50 units. These observations informed our decision on where to segment the radial window.

Subsequently, we analyzed how the performance of the model varied with increasing window size in both the θ and ϕ directions. Fig. 6 displays the variations in mean IOU corresponding to window sizes of 1.5°, 2°, and 2.5°, in regions classified as close ($r < 20$), medium ($r > 20 and r < 50$), and far ($r > 50$). However, an attempt to increase the window size further to 3° led to memory constraints. These findings were instrumental in determining the optimal width for the windows in three intervals.

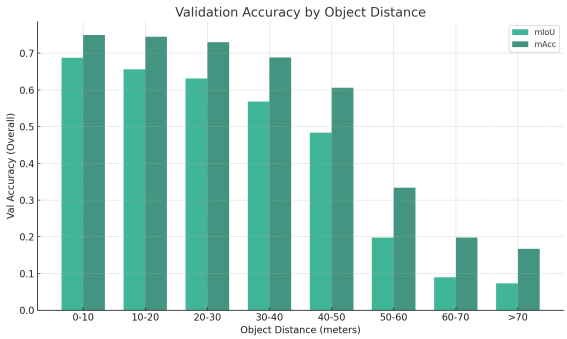


Figure 5. Overall Mean IOU and Mean Accuracy for the baseline model based on the distance of the object.

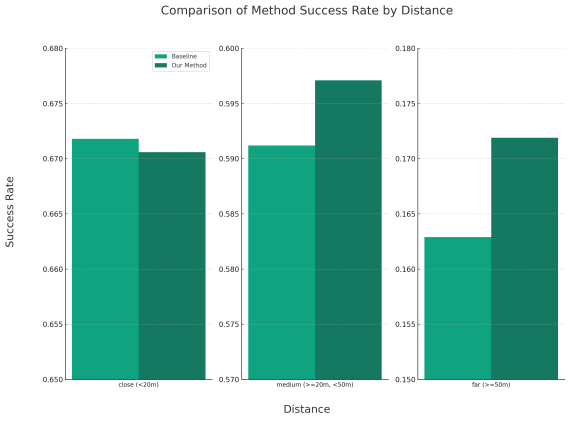


Figure 6. Our method vs. baseline. We trained from the SphereFormer papers. Our method is able to achieve nearly the same accuracy for close objects while outperforming the baseline on farther objects, as expected.

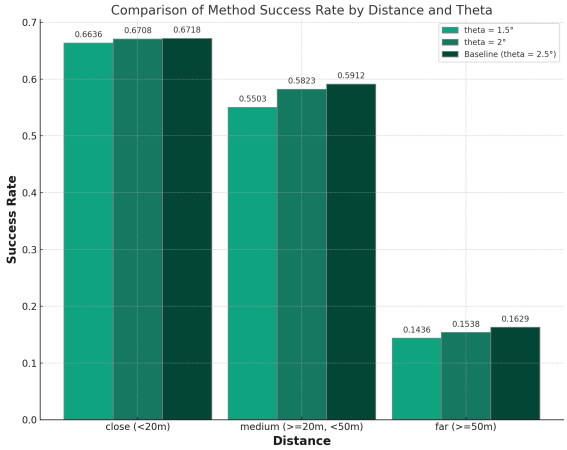


Figure 7. Baseline Model: mIoU vs θ vs distance

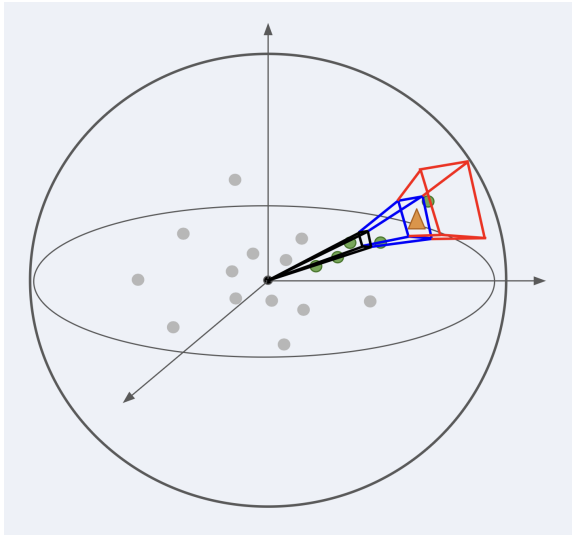


Figure 8. Proposed Future Work: Continuous Dynamic Radial Window

7. Results

Fig. 6 and Fig. 7 show that Adaptive SphereFormers achieve a better mIoU score (66.59%) than the baseline SphereFormers (65.64%) over 40 epochs of training and observed a significant increase for classes such as trucks, parking, fences, and buildings at the SemanticKITTI segmentation task. While either of the models have not reached convergence at the time of writing this report (owing to the substantial training time of 78 minutes per epoch), these preliminary results are a promising step towards an improved state-of-the-art in the 3D LiDAR segmentation task.

8. Conclusion & Future Work

The observed results confirm our empirically derived hypothesis that a larger radial angle (θ) is more conducive to a better segmentation result at greater distances from the sensor. As an extension to this idea, instead of increasing the window sizes at discrete intervals, we can define the window size as a continuous function of the distance from the sensor (see: Fig. 8). It also remains to be seen whether this change is task-agnostic, as we have not tested if 'Adaptive SphereFormer' improves upon the performance of the baseline SphereFormer at the 3D LiDAR Object Detection task.

References

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019. 4

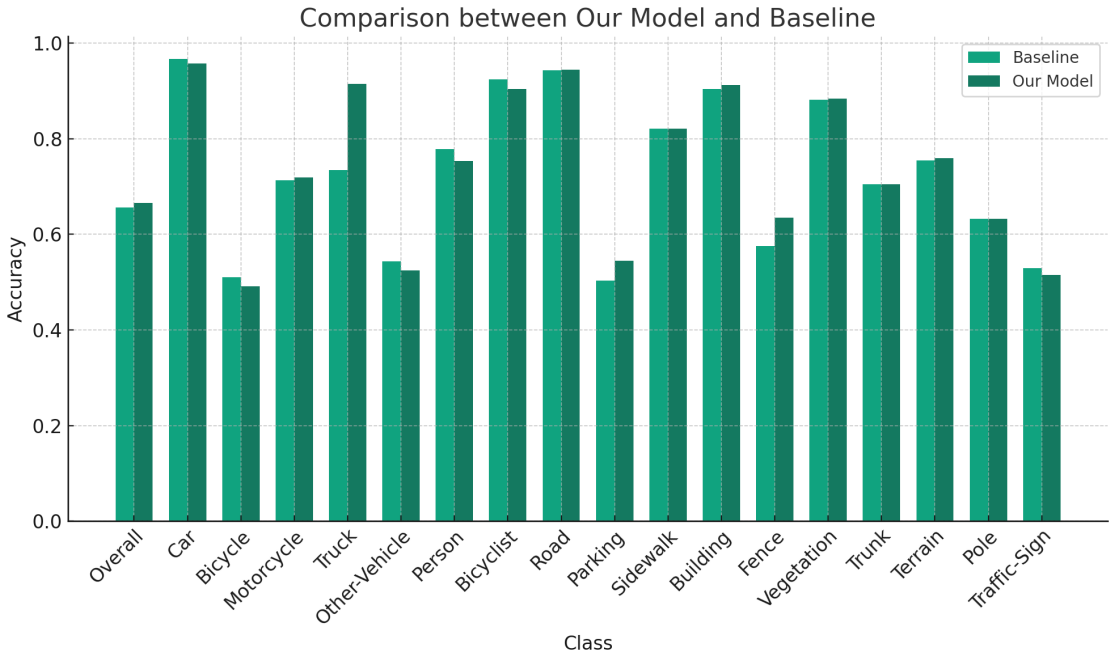


Figure 9. SphereFormer vs. Adaptive SphereFormer, Performance by Class.

[2] Xuhao Chen. Escoin: Efficient sparse convolutional neural network inference on gpus, 2019. 2

[3] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Largekernel3d: Scaling up kernels in 3d sparse cnns, 2023. 1

[4] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks, 2019. 2, 3

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 2

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 4

[7] Benjamin Graham. Spatially-sparse convolutional neural networks, 2014. 2, 3

[8] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks, 2017. 2

[9] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition, 2023. 1

[10] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation, 2022. 1, 2, 3, 4

[11] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. 3

[12] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection, 2021. 1

[13] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet ++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220, 2019. 2

[14] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 2, 3

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

[16] Ryan Razani, Ran Cheng, Ehsan Taghavi, and Liu Bingbing. Lite-hdseg: Lidar semantic segmentation using lite harmonic dense convolutions, 2021. 2

[17] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions, 2021. 3

[18] 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, 2020. 2