3D Adaptive Structural Convolution Network for Domain-Invariant Point Cloud Recognition

Younggun Kim[1*] and Soomok Lee[2**]

[1] University of Central Florida, Florida, USA
[2] Ajou University, 206 World cup-ro, Suwon, Korea
younggun.kim@ucf.edu, soomoklee@ajou.ac.kr

Abstract. Adapting deep learning networks for point cloud data recognition in self-driving vehicles faces challenges due to the variability in datasets and sensor technologies, emphasizing the need for adaptive techniques to maintain accuracy across different conditions. In this paper, we introduce the 3D Adaptive Structural Convolution Network (3D-ASCN), a cutting-edge framework for 3D point cloud recognition. It combines 3D convolution kernels, a structural tree structure, and adaptive neighborhood sampling for effective geometric feature extraction. This method obtains domain-invariant features and demonstrates robust, adaptable performance on a variety of point cloud datasets, ensuring compatibility across diverse sensor configurations without the need for parameter adjustments. This highlights its potential to significantly enhance the reliability and efficiency of self-driving vehicle technology.

Keywords: Self-Driving System · LiDAR Point Cloud Classification · Domain-Invariant Feature Representation.

1 INTRODUCTION

Self-driving vehicles have ushered in a new era of transportation, promising to improve safety, efficiency, and convenience with their advanced capabilities. Central to their operation is the accurate perception of the surrounding environment, a task crucially supported by Light Detection and Ranging (LiDAR) sensors. Distinct from cameras and radar, LiDAR excels by providing high-precision 3D spatial data. Operating across various channels, such as 16, 32, or 64, and rotating 360 degrees, LiDAR is able to create a comprehensive point cloud of obstacles over substantial distances, a capability that is crucial for navigation in urban areas. This feature is particularly beneficial for the effective detection of 3D objects and navigating complex environments filled with numerous objects and distinct features.

Adapting deep learning networks for point cloud data recognition introduces distinct challenges, especially with domain variability illustrated in Fig. 1 (a),

* This work was carried out while he was affiliated with Ajou University.
** Corresponding author: Soomok Lee.

such as differences in dataset characteristics or sensor manufacturers. Research in this area primarily explores how diversity in datasets affects deep learning models. It is observed that models trained on point cloud data under one set of conditions or with a specific sensor often face performance issues when applied to data from different conditions or sensors. For instance, a shift from training on high-density point cloud data to testing on lower-density data, due to cost considerations, can lead to a noticeable drop in model performance. This scenario highlights the sensitivity of deep learning models to the specific attributes of point cloud data, emphasizing the necessity for adaptive techniques capable of handling variations in data domains and acquisition methods with different platforms or sensor suites.

PointNet [18], the first network to encode point cloud data, revolutionized its treatment by organizing it into one-dimensional arrays for processing through a multi-layer perceptron (MLP). This approach preserves the data's unstructured nature via max-pooling across MLP layers without converting it to structured formats. However, despite its pioneering method and improvements by PointNet++ [19], it overlooks the inherent 3D structure of point cloud data. Instead, it uses a rotation matrix to approximate spatial relationships, which can lead to overfitting in certain sensor setups. This limitation hampers its ability to handle the dynamic nature of point cloud data across various environments. Additionally, these encoding methods often fail to capture the full 3D context by over-simplifying environmental variations.

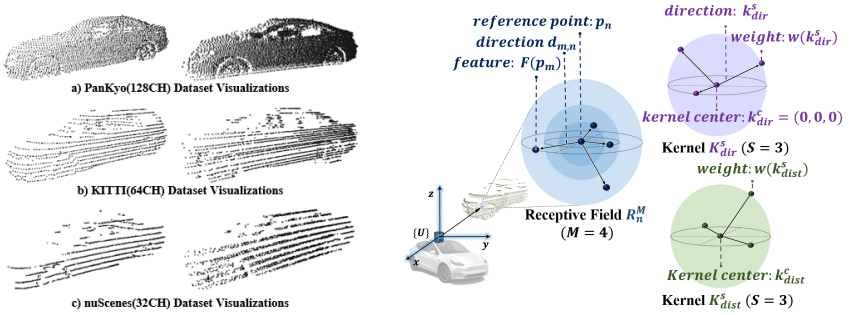
Meanwhile, graph-based methods such as Dynamic Graph CNN (DGCNN) [25] can explain 3D geometric information more deeply, utilizing graph-based convolution networks alongside a k-nearest neighbor strategy to form dynamic graphs that enhance local feature understanding. This approach facilitates a nuanced comprehension of point-to-point relationships, contributing to more sophisticated feature extraction. However, DGCNN's complexity introduces computational challenges and a dependency on the precision of input graphs, where suboptimal graph constructions can undermine the outcomes. Addressing these complexities is in need of ensuring rigid and comprehensive feature extraction.

In this paper, we present a 3D Adaptive Structural Convolution Network (3D-ASCN) for encoding in LiDAR-based 3D recognition tasks within the self-driving dataset. By selecting adaptive neighborhoods of each point, applying 3D convolution kernels in combination with a 3D geometric tree structure, and utilizing cosine similarity and Euclidean distance, our approach effectively extracts geometric features that are more appropriate and maintains consistency in features extracted from LiDAR data, irrespective of regional variations or changes in LiDAR sensors. Through our proposed method of structural encoding, we have achieved significant improvements in performance across different domains, including changes in datasets or LiDAR sensor configurations. The main contributions of this study can be summarized in three folds as follows:

– We propose novel LiDAR-invariant 3D convolution kernels to train 3D structural perspective by combining Cosine similarity and Euclidean distance

term. Our network shows stability and consistent performance in varying point cloud datasets.
– An adaptive neighborhood sampling method is proposed, based on principal components of the 3D covariance ellipsoid, which allows us to exploit highly notable geometric features by selecting the appropriate neighborhood numbers within a specific sampling range.
– The proposed 3D-ASCN method demonstrates domain-invariant feature extraction across diverse types of LiDAR data and platforms. The robust performance is demonstrated with a variety of real-world point cloud datasets, showcasing the structural features' domain invariance. When tested on high-resolution point clouds from vastly different areas and resolutions, the model maintains its highest robustness and adaptability to various sensor configurations without necessitating parameter adjustments.

The subsequent sections of this paper are organized as follows. Section 2 presents a comprehensive literature review of the point cloud encoding networks with respect to LIDAR object classification. Section 3 illustrates the proposed 3D-ASCN configuration and classification network architecture. In Section 4, the paper demonstrates the evaluation of the proposed adaptation concerning varying LiDAR channels. Lastly, Section 5 concludes the paper.



(a) Point Cloud Visualizations in different source domain

(b) Drawing of receptive field, direction-based kernel, and distance-based kernel.

Fig. 1: Our proposed convolution concept and its target for different point cloud domain

2 RELATED WORK

2.1 3D point cloud-based Networks

Pointwise MLP networks PointNet [18] processes unordered 3D point cloud data using shared fully connected layers and channel-wise max-pooling to extract global features. However, it primarily learns key point representations and

does not encode local geometric information, making it sensitive to variations in translation and scaling. To address this, researchers have proposed sorting 3D points into ordered lists where neighboring points have smaller Euclidean distances. For example, one study [7] sorts points along different dimensions and uses Recurrent Neural Networks (RNN) to extract features, while others [4,8] use a kd-tree to convert 3D points into a 1D list followed by 1D CNNs. However, this approach can struggle to preserve local geometric information. PointMLP [16] asserts that detailed local geometric information might not be essential for point cloud analysis. It utilizes a pure residual network without a complex local geometry extractor. Instead, it includes a lightweight geometric affine module, which significantly improves inference speed. PointMLP shows outstanding performance on both the ModelNet40 dataset [27], which is ideal for CAD-based analysis, and the ScanObjectNN dataset [22], which represents real-world indoor environments. PointNet++ [19] based on a hierarchical feature learning paradigm to recursively capture local geometric structures demonstrates promising results and has become the cornerstone of modern point cloud methods due to its local point representation. Based on PointNet++, PointNeXt [20] improves the training and training strategies to enhance the performance of PointNet++ and introduces a separable MLP along with an inverted residual bottleneck design within the PointNet++ framework. PointNeXt surpasses state-of-the-art methods in 3D classification.

Graph convolution networks The Dynamic Graph CNN (DGCNN) [25] represents a significant advancement in 3D point cloud processing with its graph-based convolutional networks. It uses a k-nearest neighbor algorithm to create a dynamic graph that captures local point-to-point relationships for improved feature extraction. However, DGCNN is computationally intensive due to the complexity of dynamic graph adjustments. Its performance also hinges on the quality of the input graph, where suboptimal or noisy graphs can affect results, and refining the graph construction process is complex. To improve graphical point cloud relationships in convolution, Lin *et al.* propose a 3D Graph Convolutional Network [14] to learn geometric properties of point clouds. Their approach stands out for considering geometric values and adjusting network parameters through directional analysis. The model uses deformable graph kernels to handle translation, scale, and z-axis rotation. However, its effectiveness is mainly evaluated on ideal CAD-based datasets, which do not fully represent realistic occlusion scenarios in point cloud data. Additionally, it does not adequately account for the distances between points, potentially missing detailed interval nuances during parameter training, which is crucial for self-driving datasets with consistent but small-scale patterns despite variations in LiDAR technology. Furthermore, although both DGCNN and GCN adopt the number of the neighborhoods as a constant value when selecting neighborhoods, each point in the point cloud will probably require a different number of neighborhoods. Accordingly, not only may this approach be able to extract distorted local region features, but if the neighborhood is set to be large, the degree of distortion will increase as the network becomes more advanced. Therefore, the appropriate number of neigh-

borhoods must be selected for every point by variably adjusting the number of neighborhoods.

2.2 Adaptive neighborhood sampling

To describe the 3D structure surrounding a point using geometric features, it is essential to define a local neighborhood that encompasses all considered 3D points. Various strategies exist for defining these local neighborhoods around a specific 3D point. The most commonly used neighborhood definitions include Spherical Neighborhood [10], Cylindrical Neighborhood [3], and K-Nearest Neighborhood (K-NN) [15]. Spherical neighborhood approach involves defining the neighborhood as all 3D points within a sphere of a predetermined radius centered on the point. Cylindrical neighborhood method consists of all 3D points whose 2D projections fall within a circle of a fixed radius around the point's projection. The k-nearest neighbors definition uses a fixed number of the closest neighbors to the point in 3D space.

These methods rely on a constant scale parameter to select neighborhoods. The parameter is either a fixed radius or a set number of neighbors. However, the choice of scale parameter often depends on heuristic or empirical knowledge across point cloud datasets. To circumvent making strong assumptions about the local 3D neighborhoods of each point, more recent studies have aimed at determining the appropriate neighborhood size for each individual point, thereby enhancing the uniqueness of the derived features. Most approaches optimize the number of closest neighbors (k) for each point, which could be done through the methods that consider factors like curvature, point density, and noise in normal estimation. [9] These factors are especially pertinent for densely sampled, nearly continuous surfaces. In addition, several studies have proposed advanced sampling strategies for varying point cloud densities. For instance, Arief et al. [6] introduced a density-adaptive method for object segmentation in autonomous vehicles, while Wang et al. [23] and Xu et al. [28] proposed adaptive approaches to address data corruption and improve 3D scene understanding.

Other techniques consider variations in the surface [17] or are based on the dimensionality [2] or eigenentropy [26] to select adaptively the neighborhood size within the search range. While methods based on surface variation and dimensionality require heuristic judgment, the eigenentropy method can automatically determine the neighborhood size within the search range. We further refined this eigenentropy approach to make it more suitable for application in our 3D-ASCN.

3 3D ADAPTIVE STRUCTURAL CONVOLUTION NETWORK (3D-ASCN)

Our 3D adaptive structural convolution network (3D-ASCN) is specifically designed to process data captured by LIDAR point cloud sensors. This network

ingests in 3D point cloud data and analyzes it to identify important geometric structures. Specifically, 3D-ASCN extracts 3D structural context features, including geometric information, ensuring performance does not depend on the point density of the dataset and enables domain-invariant point cloud recognition.

3.1 Receptive Field for 3D-ASCN

A 3D point cloud object is referred to as P, consisting of N points represented by $P = (p_n | n = 1, 2, \ldots, N)$. p_n represents a 3D coordinate within the three-dimensional space, thus belonging to the set $\mathbb{R}^3$. For each point in the point cloud, a specific derived feature, $F(p)$, is associated, which is a vector of dimension D. This vector, $F(p) \in \mathbb{R}^D$, encapsulates certain characteristics or attributes of the point. In order to grasp the local structural and geometric features associated with each point p_n, we define the receptive field, denoted as R_n^M, which consists of M neighborhood points. This receptive field for the point p_n, with a size of M, is established as shown in Fig. 1 (b) and defined as:

$$R_n^M = \{p_n, p_m | \forall p_m \in \mathcal{N}(p_n, M)\} \tag{1}$$

where $\mathcal{N}(p_n, M)$ formally represents the M nearest neighbor points of p_n, and the directional vectors $d_{m,n} = p_m - p_n$ will be used for later farthest neighborhood selection among points p_m in a receptive field and for structural convolution purposes. The features encompassed within the input points p_n, with a size of M, can be expressed as $\{F(p_n), F(p_m) | \forall p_m \in \mathcal{N}(p_n, M)\}$. These features are computed during the structural convolution operation.

3.2 3D Structural Kernels

To perform convolution in 3D point cloud structures, we compose direction-based kernels and distance-based kernels.

Direction-based kernel Similar to other 3D Graph Convolution Kernel concepts [14], Kernel K^S is computed for graph computation. We design it as a direction-based kernel, denoted as K_{dir}^S, where S indicates the number of branches. More precisely, it consists of S branches to train the directions and weights of the directional vectors. In order to define the direction, K_{dir}^S requires the center kernel $k_{dir}^c = (0, 0, 0)$, from which each support includes k_{dir}^1, k_{dir}^2, $\ldots$, k_{dir}^S. K_{dir}^S is composed of the center k_{dir}^c and supports for the directional vectors $k_j \in \mathbb{R}^3$ and its combination $K_{dir}^S = \{k_{dir}^c, k_{dir}^1, k_{dir}^2, \ldots, k_{dir}^S\}$. Then, with direction-based weight vectors defined as $w(k_{dir}) \in \mathbb{R}^D$ for each kernel point k_{dir}, we achieve a part of the convolution operations through the weighted sum of features corresponding to the directional weights of $F(p)$. Since it is designed to train the structures in 3D, the directional vectors, represented by $k_{dir}^s - k_{dir}^c = k_{dir}^s$, are among the trained kernel parameters.

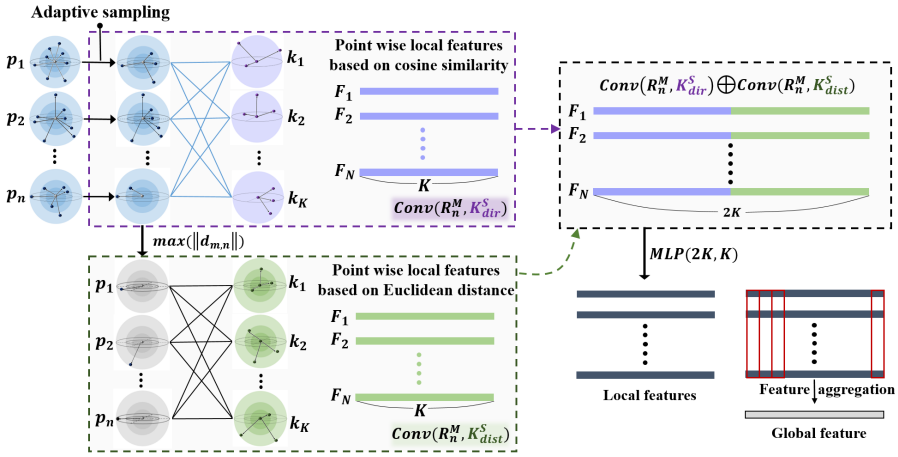


Fig. 2: Adaptive sampling and structural convolution operation: All points select adaptively their neighborhoods based on k-nearest neighborhood selection and by minimizing Shannon entropy. Then, structural convolution operations are performed to extract neighborhood features of all points based on cosine similarity and Euclidean distance. Features based on cosine similarity and Euclidean distance are concatenated and passed through a multi-layer perceptron. These features are refined as they are used as inputs for this network. Finally, we can extract a global feature that includes structural information of a point cloud by aggregating all local features.

Distance-based kernel We propose a distance-based kernel, denoted as K^S_{dist}, to reflect the distance between each point as a learning parameter. Simillar to the direction kernel, S denotes the number of branches, and each branch contains weights for the distance relationship between points, defined as $w(k_{dist}) \in \mathbb{R}^D$. Then, we perform a part of convolution operations via the weighted sum of features corresponding to the distance-based weights of $F(p)$. Therefore, with both the direction-based kernel and distance-based kernel included, the total trainable kernels in this network are defined as $\{w(k^c_{dir}), (k^s_{dir}, w(k^s_{dir})), w(k^c_{dist}), w(k^s_{dist}) \mid s = 1, 2, \ldots, S\}$.

3.3 Adaptive neighborhood sampling

Since our network obtains local region geometric information through the distance and directional relationships between each point p_n in the point cloud and the neighborhoods, all p_n require different neighborhood sizes based on geometric properties. Additionally, the point density in the captured 3D point cloud data varies with the type of LiDAR sensor, and this density significantly influences the selection of the number of neighborhoods. Our 3D-ASCN is not only capable of extracting excellent geometric information from a point cloud by adaptively selecting the number of neighborhood points for each point p_n, but it also re-

mains invariant in LiDAR channel changes.

K-NN [15] algorithm is applied to each point as the neighborhood search method for p_n, allowing more flexibility concerning the geometric size of the neighborhood. Instead of arbitrarily choosing a fixed number for the receptive field parameter M, we opt for an automatic approach to determine the best value for M. Considering a point cloud composed of N points in three-dimensional space, with M as a variable from the set of natural numbers N, each point $p_n = (X, Y, Z)^T$ lies in $\mathbb{R}^3$, and its M-neighborhood defines the extent of each receptive field $\mathcal{R}_n^M$. To capture the local structure around each point p_n, we compute the covariance-based 3D structure tensor S, which is a symmetric positive definite matrix in $\mathbb{R}^{3\times3}$. This tensor always has three non-negative eigenvalues $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}$ that are ordered such that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ and correspond to an orthogonal set of eigenvectors. To determine the neighborhood size for each point p_n adaptively, eigenvalues representing the principal components [26] of the 3D covariance ellipsoid are obtained. The neighborhood sizes M of p_n minimize the entropy of these eigenvalues, known as eigenentropy, defined as Eq. (2), which serves as a quantification of the spatial coherence within the ellipsoid.

$$E_\lambda = -(e_1 \ln(e_1) + e_2 \ln(e_2) + e_3 \ln(e_3)) \tag{2}$$

where $e_j = \lambda_j / \sum_{i=1}^{3} \lambda_i, (j = 1, 2, 3)$, representing each normalized eigenvalue. Specifically, we measure the eigenentropy in $[M_{\min}, M_{\max}]$, which indicates the number of potential neighbors of p_n, and all integer values within the range.

3.4 Structural Convolution Operation

Using the specified inputs and kernel definitions for 3D point cloud data, we propose the 3D Structural Convolution process, as shown in Fig. 2. We determine the similarity between the receptive field R_n^M and the direction-based kernel K_{dir}^S, represented as $Conv_{dir}(R_n^M, K_{dir}^S)$, to derive point-specific features through cosine similarity. Additionally, we calculate the product of the distance to the farthest neighbor in R_n^M and the weights in K_{dist}^S, represented as $Conv_{dist}(R_n^M, K_{dist}^S)$, to obtain point-specific local features via Euclidean distance. These extracted features are then concatenated and processed through an MLP, enabling accurate feature extraction of the structural context without introducing bias towards either distance or directional context during training.

Unlike 2D CNNs, where kernels and image patches both have grid structures, our method involves combinations of 3D vectors instead of grids. For the purpose of convolution in 3D point clouds, we achieve our structural convolution operation with direction-based kernels and distance-based kernels. Initially, we assess the similarity between features within the receptive field of p_n (i.e., $F(p_n)$, $F(p_m)$ for all $p_m \in N(p_n, M)$, as specified in Eq. (1)) and the weight vectors

of the directional kernel K_{dir}^{S}, centered around k_{dir}^{C} with S supports (namely, $w(k_{dir}^{c})$, $w(k_{dir}^{s})$ for all $s = 1, 2, \ldots, S$). We consider every pairing of (p_m, k_{dir}^{s}). Consequently, the direction-based convolution between a receptive field and a direction-based kernel can be described as follows:

$$Conv_{dir}(R_n^M, K_{dir}^S) = \langle F(p_n), w(k_{dir}^c) \rangle + \sum_{s=1}^{S} \max_{m \in (1,M)} sim(p_m, k_{dir}^s) \tag{3}$$

where the symbol $\langle \cdot \rangle$ denotes the inner product operation, and the function sim calculates the inner product between the features $F(p_m)$ and the directional weights $w(k_{dir}^{s})$, utilizing cosine similarity [14] to define this interaction:

$$sim(p_m, k_{dir}^s) = \langle F(p_m), w(k_{dir}^s) \rangle \frac{\langle d_{m,n}, k_{dir}^s \rangle}{||d_{m,n}||\, ||k_{dir}^s||} \tag{4}$$

Moreover, to account for the influence of spatial relationships among neighboring points on structural characteristics, we assess the Euclidean distances from point p_n to its most distant neighbor and then multiply them with the weights of the kernel K_{dist}^{S} centered at k_{dist}^{C} (precisely, $w(k_{dist}^{c}), w(k_{dist}^{s}), \forall s = 1, 2, \ldots, S$). we consider all conceivable combinations of (p_m, k_{dist}^{s}) to perform the distance-based convolution that merges the a receptive field with the kernel, expressed as:

$$Conv_{dist}(R_n^M, K_{dist}^S) = \sum_{s=1}^{S} w_{dist}^s \times \max_{m \in (1,M)} (||d_{m,n}||) \tag{5}$$

Note that, since Eq. (3) indicates the convolution between a receptive field and a direction-based kernel, and Eq. (5) indicates the convolution between a receptive field and a distance-based kernel, the convolution operations in Eq. (3) and Eq. (5) must be applied to all receptive fields, all direction-based kernels, and all distance-based kernels. Finally, by concatenating the outputs from the above convolution operations and passing them through an MLP, 3D structural convolution is achieved as follows:

$$\begin{aligned} &Str - Conv(R_n^M, K_{dir}^S, K_{dist}^S) \\ &= MLP(Conv_{dir}(R_n^M, K_{dir}^S) \oplus Conv_{dist}(R_n^M, K_{dist}^S)) \end{aligned} \tag{6}$$

Concatenating distance-based features and direction-based features, and then processing them through a Multi-Layer Perceptron (MLP), enables learning without bias towards either direction-based or distance-based kernels.

3.5 Classification Architecture

In the design of the 3D-ASCN, we have modified the convolution and pooling layers. The framework consists of five convolution layers and three max-pooling layers, with adjustments made to the number of parameters to suit our needs

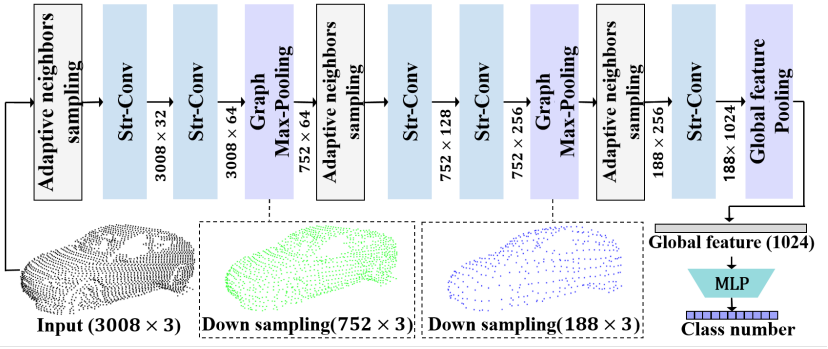


Fig. 3: Illustration of our classification architecture: Str-Conv blocks perform structural convolution. Graph Max-Pooling blocks perform channel-wise max-pooling from the features and then randomly sample a subset following the sampling rate r. During this process, the Adaptive neighborhood sampling block adaptively finds the neighborhoods and passes them to the Str-Conv block.

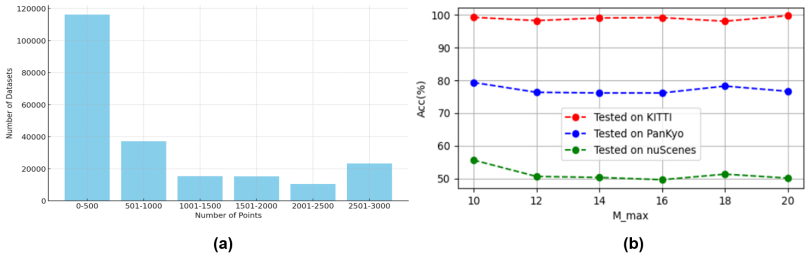


Fig. 4: (a) The number of points per object. (b) Model performance trained on KITTI dataset along M_{max}

for LiDAR-based point cloud data, as illustrated in Fig. 3. In this model, the computation of the convolution layer follows the Eq. (6) previously mentioned. First, the range of the neighborhood is used to be searched as input. The convolution process then uses a fixed number of kernels, each with a specific number of supports. These kernels are applied to a 3D point cloud and its associated D-dimensional features, generating output features for each network layer. The feature value for each point in the point cloud and the kernel weights are set to one.

Following the convolution step, the 3D Max-Pooling layer identifies the receptive field for each point and performs channel-wise max-pooling to compile features. It then samples a portion of the point cloud, effectively reducing the point count by a rate of 4. This pooling layer is crucial for efficiently learning features from the 3D point cloud by decreasing the dimensionality of both the input and output point clouds and their features by a quarter.

Intra-Domain classification ($S \rightarrow S$)					Cross-Domain classification ($S \rightarrow S^*$)						
Methods	KITTI	PanKyo	nuScnees	Average	KITTI PanKyo	KITTI nuScenes	PanKyo KITTI	PanKyo nuScenes	nuScenes KITTI	nuScenes PanKyo	Average
PointNet [18]	99.5	92.9	92.5	95.0	17.6	54.8	6.1	24.0	40.4	**44.4**	31.2
PointNeXt [20]	99.6	93.6	95.5	96.2	**67.6**	31.4	5.8	17.3	31.6	18.6	28.7
PointMLP [16]	**99.9**	**96.6**	94.7	97.1	17.7	**80.4**	**87.7**	64.0	47.1	15.2	52.0
DGCNN [25]	99.4	96.3	95.9	97.2	19.9	49.7	14.7	46.2	66.3	17.4	35.7
PointTransformer [29]	99.3	**98.5**	**96.2**	**97.9**	7.9	88.1	77.3	**73.3**	**79.1**	8.8	**55.8**
PointMamba [13]	**100**	95.1	**97.9**	**97.7**	6.6	**90.1**	11.5	37.5	61.6	30.6	39.7
3D-ASCN(ours)	99.2	94.2	92.2	95.2	**55.6**	79.3	**89.8**	**71.4**	**92.6**	**74.9**	**77.3**

Table 1: Intra-domain and cross-domain classification results on KITTI [5], nuScenes [1], and PanKyo [11]. The bold **black**, **blue** indicate the first and the second best performance on each scenario, respectively.

Methods	KITTI PanKyo nuScenes	KITTI nuScenes PanKyo	PanKyo nuScenes KITTI	Avg
Pointnet	**84.9**	8.5	61.0	51.5
DGCNN	84.2	8.4	39.4	44.0
3D-ASCN	71.6	**70.1**	**76.6**	**72.8**

Table 2: Cross-Validatioin results. The sizes of all datasets have been rearranged to be the same as the size of the PanKyo [11] dataset.

4 EVALUATION

4.1 Implementation Detail

Our model incorporates three pivotal components, including adaptive neighborhood sampling, direction-based kernel, and distance-based kernel. In the beginning, the 10 nearest neighbors is sampled for each point using the kNN algorithm [15]. The 3D-ASCN then evaluates the eigenentropy for each possible neighborhood size, from 3 to 10, to select the number of neighbors. Since the points are highly sparse, as depicted in Figure4-(a), leading the algorithm to frequently select smaller neighborhood sizes, increasing M_{max} beyond 10 does not result in significant performance differences as shown in Figure4-(b).

To accommodate kernels requiring a consistent size, the maximum neighborhood size previously determined is used. For cases with fewer than 10 neighbors, blanks are processed as the central location to ensure uniformity in input size, with the distance and cosine similarity of the central point set to zero. The 3D-ASCN applies both direction-based and distance-based kernels across all layers, with each layer featuring a fully connected (FC) layer for extracting distance features and another for direction features. This setup includes stacking five convolutional layers in a manner akin to 3D-GCN [14].

4.2 Classification

Classification experiments are conducted with our model on three outdoor point cloud datasets: KITTI [5], nuScenes [1], and PanKyo [11]. These datasets are

Train	Test	Convolution operation	Acc.(%)
PanKyo	KITTI	$Conv_{dir}$	52.6
		$Conv_{dir} + Conv_{dist}$	86.3
		$Str - Conv$	**88.7**
	nuScenes	$Conv_{dir}$	55.7
		$Conv_{dir} + Conv_{dist}$	69.0
		$Str - Conv$	**71.2**
	PanKyo	$Conv_{dir}$	92.4
		$Conv_{dir} + Conv_{dist}$	93.9
		$Str - Conv$	**94.2**

Table 3: Performance with different convolution operations on KITTI [5], nuScenes [1], and PanKyo [11] datasets. These comparisions were performed under the fixed number of neighborhood. The best results are given in **boldface**.

Train	Test	The number of neighbors	Acc.(%)
PanKyo	KITTI	Fixed M	88.7
		Adaptive M	**89.8**
	nuScenes	Fixed M	71.2
		Adaptive M	**71.4**
	PanKyo	Fixed M	**94.2**
		Adaptive M	**94.2**

Table 4: Performance comparison according to neighborhood sampling methods on KITTI [5], nuScenes [1], and PanKyo [11] datasets. The best results are given in **boldface**.

provided in the given link[3] and originate from various countries and manufacturers. This diversity is crucial for assessing not only the model's inferencing capabilities but also its robustness across multiple domain shifts. Especially, they all have different channels. nuScenes has 32 channels, KITTI has 64 channels, while PanKyo has 128 channels. We evaluated the robustness of our model using these channel shifts. In addition, to seamlessly align the existing classes, we reduce the number of classes to three: Pedestrian, Car, and Truck. All methods are evalauted through the object classification accuracy and the evaluation metric [12] is shown as follows:

$$\text{Accuracy}(\%) = \frac{\text{the \# of correctly classified classes}}{\text{the \# of total ground truths}} \times 100$$

Classification on single-domain A detailed evaluation was conducted to assess the algorithm's performance, particularly its behavior under conditions that might lead to overfitting, by examining it from within-domain perspectives. Tab. 1 on the left side ($S \rightarrow S$) displays the classification performance within a single domain. While 3D-ASCN shows a tendency towards underfitting in the general automotive dataset, a consequence of removing scale invariance and possessing low inductive bias, 3D-ASCN demonstrates performance on par with models known to overfit in specific domains.

[3] https://sites.google.com/site/cvsmlee/dataset

	PV-RCNN(w/ 3D-ASCN)			PV-RCNN(w/ PointNet) [21]		
	Easy	Mod.	Hard	Easy	Mod.	Hard
Car(IoU 0.7)	92.32	83.24	80.81	92.10	84.39	82.49
Pedestrian(IoU 0.5)	70.05	64.91	60.38	62.72	54.51	49.86
Cyclist(IoU 0.5)	88.15	66.25	63.55	89.09	70.41	66.01
3D mAP	**83.51**	**71.47**	**68.24**	81.30	69.77	66.12

Table 5: Intra-domain 3D object detection performance comparision on the KITTI dataset. The results are evaluated by the mean Average Precision with 40 recall positions.

Classification on cross-domain Evaluations were conducted across different domains to assess the models' robustness in various contexts. The experiments specifically aimed to examine the performance and robustness of models by training them in one domain and testing them in another, with a focus on domain invariance. Tab. 1 on the right side ($S \rightarrow S^*$) shows the performance comparison between 3D-ASCN and other methods for the cross-domain classification task of autonomous driving datasets. Our method demonstrates highly consistent classification performance across all domain shift scenarios. It especially outperforms the second-best methods by 13.5% when shifting from nuScenes to KITTI and by 30.5% when transitioning from nuScenes to Pankyo. Moreover, the average accuracy of 3D-ASCN surpasses the second-best method, PointTransformer, by 21.5%, as illustrated in Tab. 1 right side.
Moreover, we conducted cross-validation experiments, the results of which are presented in Tab. 2 We trained the model using two of the train datasets as the training set and used the remaining dataset as the test set. As a result, our 3D-ASCN demonstrated a 21.3% performance improvement over the second-best model. Consequently, these two type experiments indicate that both the kernels based on cosine similarity, the kernels based on Euclidean distance, and adaptive neighborhood selection methods were effective in enabling structural learning capabilities amid general changes in datasets.

Ablation study Structural Convolution combines Cosine similarity and Euclidean distance to extract 3D structural features, enhancing model robustness when the LiDAR sensor changes. Tab. 3 shows three convolution methods. Direction-only kernels degrade performance under sensor changes, but combining direction- and distance-based kernels improves robustness. Concatenating these features and processing them through an MLP ensures balanced learning between direction and distance features. As a result, our structural convolution achieves the best performance in both fixed and changing domain conditions.

Tab. 4 shows that adaptive neighborhood sampling improves performance, especially with domain changes, compared to a fixed M of three. When training and testing within the same domain (e.g., PanKyo), performance is unchanged as the number of neighbors remains near three. These results highlight the advantages of adaptive M in cross-domain scenarios.

Methods	PV-RCNN	PV-RCNN†	PV-RCNN(w/3D-ASCN)
BEV mAP	51.84	40.03	**68.10**
3D mAP	17.92	21.23	**43.83**

Table 6: Cross-domain object detection performance when all methods were trained on nuScenes dataset and tested on KITTI(Moderate) dataset. 3D mAP and BEV mAP are calculated by 40 recall positions and IoU=0.7. † mark indicates PV-RCNN with statistical normalization [24].

4.3 Object detection

We evaluated object detection performance in both intra-domain and cross-domain scenarios. Our 3D-ASCN demonstrates exceptional performance in these tasks due to its ability to capture domain-invariant feature representations, making it highly effective in handling variations between different domains. Tab. 5 illustrates the effectiveness of 3D-ASCN in object detection tasks. In this table, PV-RCNN(w/3D-ASCN) refers that utilizng 3D-ASCN instead of PointNet in vanilla PV-RCNN [21] outperforms PV-RCNN in intra-domain 3D object detection tasks. Notably, Tab. 6 shows that our feature extraction method maintains very robust performance over other methods in domain changing scenarios. These results indicate that our geometrical encoding approach is highly effective in extracting domain-invariant features.

5 CONCLUSION

In this study, we introduced 3D-ASCN, a novel feature extraction model that addresses the scale-invariant bias of GCNs by incorporating Distance Feature Extraction and an Adaptive Nearest Neighbor approach. 3D-ASCN demonstrates strong robustness to domain shifts, making it highly adaptable for sensor-dependent point cloud classification and object detection. Our experiments showed significant improvements in cross-domain object recognition, emphasizing its potential in autonomous driving technologies. Future work will explore efficient learning methods for small datasets, focusing on self-supervised learning with raw point clouds.

Acknowledgements We would like to express sincere gratitude to Beomsik Cho and Seonghoon Ryoo (Ajou University) for their invaluable assistance with the conceptualization and experiments in this work. Their contributions greatly supported the development of this research. This work was supported in 2024 by Korea National Police Agency(KNPA) under the project "Development of autonomous driving patrol service for active prevention and response to traffic accidents"(RS-2024-00403630), Institute of Information communications Technology Planning & Evaluation (IITP) under the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2023-No.RS-2023-00255968) grant funded by the Korea government(MSIT), and the BK21 FOUR program of the National Research Foundation Korea funded by the Ministry of Education(NRF5199991014091).

References

1. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (2020) 11, 12
2. Demantké, J., Mallet, C., David, N., Vallet, B.: Dimensionality based scale selection in 3d lidar point clouds. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. **XXXVIII**(5/W12), 97–102 (2011) 5
3. Filin, S., Pfeifer, N.: Neighborhood systems for airborne laser data. Photogrammetric Engineering & Remote Sensing **71**(6), 743–755 (2005) 5
4. Gadelha, M., Wang, R., Maji, S.: Multiresolution tree networks for 3d point cloud processing. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018) 4
5. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE (2012) 11, 12
6. Hasan, A.A., Arief, M., Bhat, M., Indahl, U., Tveite, H., Zhao, D.: Density-adaptive sampling for heterogeneous point cloud object segmentation in autonomous vehicle applications. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (June 2019) 5
7. Huang, Q., Wang, W., Neumann, U.: Recurrent slice networks for 3d segmentation of point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 4
8. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In: Proceedings of the IEEE international conference on computer vision (ICCV) (2017) 4
9. Lalonde, J.F., Unnikrishnan, R., Vandapel, N., Hebert, M.: Scale selection for classification of point-sampled 3d surfaces. In: Proceedings of the International Conference on 3-D Digital Imaging and Modeling (3DIM). pp. 285–292 (2005) 5
10. Lee, I., Schenk, A.F.: Perceptual organization of 3d surface points. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **XXXIV**(3A), 193–198 (2002) 5
11. Lee, R., Ryoo, S., Lee, S.: Domain-invariant 3d structural convolutional network for autonomous driving point cloud dataset. In: IEEE Intelligent Vehicles Symposium (IV) (2024) 11, 12
12. Lee, S., Seo, S.W.: Probabilistic context integration-based aircraft behaviour intention classification at airport ramps. IET Intelligent Transport Systems **16**(6), 725–738 (2022) 12
13. Liang, D., Zhou, X., Xu, W., Zhu, X., Zou, Z., Ye, X., Tan, X., Bai, X.: Pointmamba: A simple state space model for point cloud analysis. arXiv preprint arXiv:2402.10739v4 (2024) 11
14. Lin, Z.H., Huang, S.Y., Wang, Y.C.F.: Learning of 3d graph convolution networks for point cloud analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(8), 4212–4224 (2021) 4, 6, 9, 11
15. Linsen, L., Prautzsch, H.: Local versus global triangulations. In: Proceedings of Eurographics. pp. 257–263 (2001) 5, 8, 11
16. Ma, X., Qin, C., You, H., Ran, H., Fu, Y.: Rethinking network design and local geometry in point cloud: A simple residual mlp framework. In: International Conference on Learning Representations (ICLR) (2022) 4, 11

17. Pauly, M., Keiser, R., Gross, M.: Multi-scale feature extraction on point-sampled surfaces. Computer Graphics Forum **22**(3), 81–89 (2003) 5
18. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). pp. 652–660 (2017) 2, 3, 11
19. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems (NIPS). vol. 30 (2017) 2, 4
20. Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H.A.A.K., Elhoseiny, M., Ghanem, B.: Pointnext: Revisiting pointnet++ with improved training and scaling strategies. arXiv preprint arXiv:2206.04670 (2022) 4, 11
21. Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Xiaogang Wang, H.L.: Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. arXiv preprint arXiv:1912.13192v2 (2021) 13, 14
22. Uy, M.A., Pham, Q.H., Hua, B.S., Nguyen, D.T., Yeung, S.K.: Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In: International Conference on Computer Vision (ICCV) (2019) 4
23. Wang, J., Ding, L., Xu, T., Dong, S., Xu, X., Bai, L., Li, J.: Sample-adaptive augmentation for point cloud recognition against real-world corruptions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14330–14339 (2023) 5
24. Wang, Y., Chen, X., You, Y., Li, L.E., Hariharan, B., Campbell, M., Q.Weinberger, K., Chao, W.L.: Train in germany, test in the usa: Making 3d object detectors generalize. In: CVPR. pp. 11713–11723 (2020) 14
25. Wang, Y., Sun, Y., Liu, Z., Sanjay E Sarma, M.M.B., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) **38**(5), 1–12 (2019) 2, 4, 11
26. Weinmann, M., Jutzi, B., Mallet, C.: Semantic 3d scene interpretation: A framework combining optimal neighborhood size selection with relevant features. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci. **II**(3), 181–188 (2014) 5, 8
27. Wu, M., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: Shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015) 4
28. Xu, M., Chen, P., Liu, H., Han, X.: To-scene: A large-scale dataset for understanding 3d tabletop scenes. In: ECCV (2022) 5
29. Zhao, H., Jiang, L., Jia, J., Torr, P., Koltun, V.: Point transformer. arXiv preprint arXiv:2012.09164v2 (2021) 11