Published in AJSE, Vol:23, Issue: 1
Received on 1st May 2023
Revised on 19th February 2024
Published on 25th April 2024

AIUB JOURNAL OF SCIENCE AND ENGINEERING
ISSN: 1608 – 3679 (print) 2520 – 4890 (Online)
DOI: 10.53799/ajse.v23i1.763

GCN-Net: 3D Point Cloud Classification & Localization Using Graph-CNN

Ahmed Abdullah and Mehzabul Hoque Nahid

***Abstract*— This study aims to demonstrate the application of a graph convolutional neural network for the purpose of object detection in a LiDAR point cloud. To achieve efficient encoding of the point cloud, the research used a near-neighbours graph with a predetermined radius. The study created a graph convolutional neural network so that study can find out what kind of object and what class each vertex in a graph actually represents. The research developed a box merging and scoring operation to reliably integrate detections from multiple vertices into a single score, as well as an auto-registration technique to reduce the amount of internal translation errors. Based on the findings obtained from our experimentation using the KITTI benchmark, the research has arrived at the inference that the proposed technique demonstrates a commendable level of precision when compared to the point cloud. In fact, in some cases, it outperforms fusion-based methods. Based on the findings of our study, it can be concluded that the graph neural network has promising potential as a novel and efficient tool for the identification and recognition of three-dimensional objects.**

***Index Terms*—Graph Convolutional Neural Network, Point Clouds, Multilayer Perceptron, Object Detection, Localization, Semantic KITTI.**

I. Introduction

It is essential for robot perception [1] to have a three-dimensional understanding of the surrounding world. It is common practice for 3D sensors such as LiDAR to store their data in the form of point clouds, which are collections of points dispersed throughout space. It is vital for several applications, such as autonomous driving [2], to have precise object detection derived from a point cloud. In order to recognize objects in photographs, convolutional neural networks {Formatting Citation} must first perform the convolutional operation. The method of convolution is quite effective, but the input must be in the form of a regular grid. In contrast to an image, a point cloud often has a more dispersed and uneven distribution of its points. When a point cloud is superimposed on a regular grid, an uneven distribution of points results in each grid cell. If the same convolution technique is applied to such a grid, it is possible that there may be information loss in the cells that have a high population density and superfluous processing in the cells that are empty. Recent developments in neural network technology have made it possible to enter a set of points in any order. In the field of research, this particular type of neural network is applied to extract attributes of point clouds without first mapping the point clouds to a grid [5]. In most cases, however, in order to cons truct a representation of a point set, they are required to sample and aggregate points in an iterative manner. The process of repeatedly grouping and sampling a large point cloud might incur significant processing costs [6], [7]. Many modern 3D detection algorithms make use of a hybrid strategy that blends grid and set representations at different points of the detection process [8]. Despite the fact that hybrid approaches have shown some encouraging results, it is possible that they are prone to the limitations of both types of representation. In this paper, the study propose the use of a graph as a condensed representation of a point cloud and the creation of a graph neural network that the study will refer to as GCNnet for the purpose of object detection [9]. The study possess the capability to inherently represent the point cloud within a graph structure, accomplishing this by directly associating the points with the graph's vertices. The graph's edges establish connections between adjacent positions within a predefined radius, thereby facilitating the transfer of feature information among proximate neighbors. A point cloud can be represented by such a graph in such a way that it conforms directly to the structure of the point cloud, eliminating the need for any kind of regularization. A graph neural network avoids constantly grouping and sampling points by recycling graph edges at each layer and does so throughout the network. Research has been conducted on the topic of semantic segmentation and classification of point clouds with graph neural networks. However, graph neural networks have only been the subject of a small number of studies looking into 3D object detection in point clouds. Our study indicates that it is possible to use Graph-CNN for extremely precise object detection in a point cloud by utilizing it to analyze the data. The point graph is an acceptable kind of input for the GCNnet that the study have suggested. It will generate both the category and the bounding boxes associated with each linked item for every vertex. This method constitutes a single-step procedure capable of detecting numerous distinct elements within a single scan of the region. To mitigate the influence of translation-induced variations in a graph neural network, the study introduce a point cloud registration technique. When applied, this method facilitates the alignment of point coordinates based on their inherent characteristics. The KITTI [10] evaluation serves as a standard for the technique that has been suggested [11]. Our GCNnet illustrates the promise of a new form of 3D object detection approach based on graph neural networks, and it can serve as a solid foundation for further research. This method is

Ahmed Abdullah
Dept of Computer Science, Bishop University, Canada
Email: aabdullah20@ubishops.ca

Mehzabul Hoque Nahid
Dept of Management Information Systems, American International University-Bangladesh, Dhaka, Bangladesh, Email: mehzab.nahid@aiub.edu

based on graph neural networks.

II. RELATED WORD

Using a convolutional neural network (CNN), the authors of the publication [3] were able to identify individual points inside a point cloud. In order to evaluate the point cloud, a Bird's Eye View (BEV) [12] is first created, which is a 2D picture. Using a picture of the front elevation (FV) does the same thing. However, a phenomenon called quantization error may occur during the process of reducing a 3D picture to a 2D format. Some solutions to this problem include preserving the point cloud's 3D structure. VoxelNet is a technique for 3D object detection [13] that works by using 3D voxels (volumetric pixels) to represent points in the point cloud. However, as the resolution of the voxels becomes higher, the computational cost of employing 3D convolution rises. Sparse convolution [14] is one method that may be utilized to increase productivity. It's important to remember that transforming a point cloud into a 2D or 3D grid might result in a misalignment between the point cloud's organization and the grid's uniformity. The examples of PointNet[15] and DeepSet[16] demonstrate how neural networks can be used to directly extract attributes from an unsorted set of points. The data from each point is fed into a multi-layer perceptron (MLP) [17], which outputs a feature vector. All of these characteristics are represented in a single vector by a mean or maximum function. In order to aggregate point attributes in a hierarchical fashion, Pointnet++[15] takes samples from the neighborhood of nodes to generate local subsets of points. The study then generates new sets from these groups in order to extract more characteristics. Neural networks are used in several 3D object detection methods because of their ability to process point clouds without grids. More computational power is needed for large-scale point sampling and clustering. The neural network on sets is used exclusively in the object detection research. In order to predict 3D object bounding boxes, the paper [7] employs Pointnet++ to segment camera images into object and background points. Pointrcnn [18] proposes bounding boxes for point clouds using pointnet++ as its backbone network. The bounded zones are then fine-tuned by a point network. Incorporating both traditional and nontraditional elements SECOND, Point-Pillars [18] for 3D object detection in point clouds. By organizing local point sets on a regular grid, Pointnet is able to extract features and make convolution more manageable. When compared to the point cloud structure, the grid's regularity falls short, even after local smoothing. Graph neural networks attempt to generalize convolutional neural networks to graphs. Iterative edge summarization updates GNN vertex features[19]. GNNs' aggregation method is similar to deep learning on sets, but they're better at identifying periphery features. It rarely samples and labels vertices again. Some computer vision methods use a point cloud graph. As described, semantic segmentation involves decomposing a point cloud into simple geometric shapes and linking them with a network, GNNs can categorize point clouds. There hasn't been another approach like ours before because the study custom-made a GNN to find objects. By using a graph representation, the study is able to keep the original geometry of a point cloud intact, as opposed to trying to flatten it into an image or voxel grid. In contrast to methods that necessitate collecting and sorting a large amount of data into sets, the study only needs to construct the graph once. The proposed GCN-Net continuously refines vertex features on the same graph after the initial feature extraction. As opposed to the multi-stage procedures used in, our work only requires a single stage of detection.

III. METHODOLOGY

A. *Graph Formation:*

Let's assume, a point cloud is defined as $\mathbf{C = \{c_1, c_2, ..., c_N\}}$ where each individual point is described as a pair $\mathbf{(x_i, s_i)}$, and $x_i \in R^3$ denotes the 3-dimensional coordinates and $s_i \in R_k$ is a vector of length **k** representing the point's characteristics. To construct a graph **G(C,Ed)** from the point cloud **C**, the points **(C)** are utilized as vertices and connections **(Ed)** are made between neighboring points within a specified radius denoted as **(r):**

$$Ed = \{((c_i, c_j)| \|(x_i - x_j)\|_2) < r\} \quad (1)$$

Creating point-cloud graph presents a widely recognized challenge known as the fixed-radius near-neighbor search problem. To solve this problem efficiently, a cell list approach is used to identify points that are within a given cutoff distance. This results in a computational complexity of **O(cN);** here **c** is the highest point number that falls into the specified radius. Many real-world point clouds have hundreds or even several thousands of points, making graphs where all the points are vertices, computationally demanding. To fix this, the study employs a down-sampled point cloud **C′** with voxels that dilute point density.

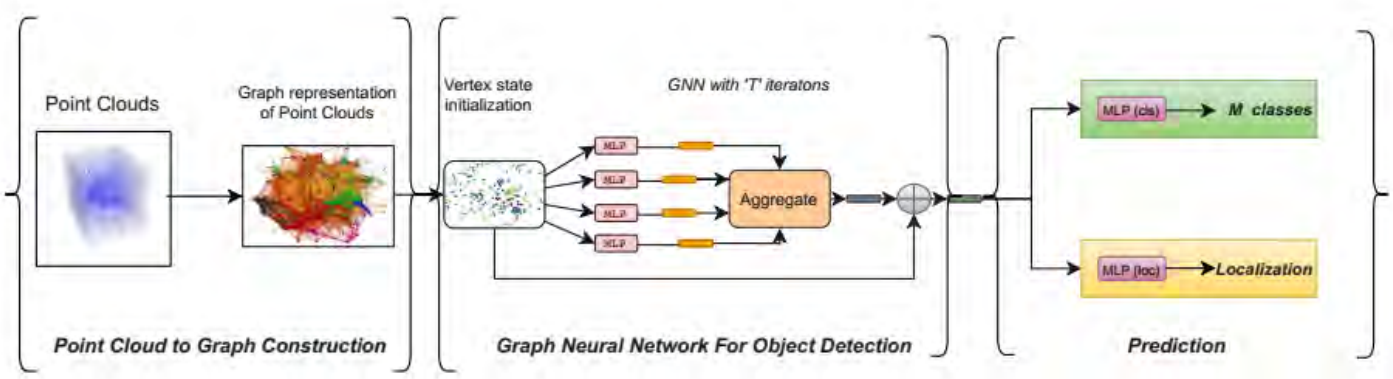

Figure 1: GCN-Net algorithm architecture with three major components: Graph formation from raw point cloud data, object detection using GNN network from constructed graph input and finally class prediction and localization.

But don't reflect the cloud. Downsampled clouds are graphed. Vertex initial state values include dense cloud information. A neural network extracts traits from source points within $\mathbf{r_0}$ radius of each vertex. Subsequently, a Multi-Layer Perceptron (MLP) integrates data from lidar reflection intensity and relative coordinates. The resultant features are then combined using the Max function to create the initial state value of the vertex. Following this, a Graph Neural Network (GNN) examines the structure of the graph.

B. Point Cloud Registration (GNN based):

A graph neural network (GNN) updates vertex features by consolidating feature information from the neighboring edges [20]. The updating process happens during each iteration, where updating vertex features happen in the $\mathbf{(n+1)^{th}}$ iteration using these equations:

$$(v_i)^{n+1} = q^n((\sigma(p^n(v_i^n, v_j^n)|(i,j)) \in Ed), v_i^n) \tag{2}$$

The edge features, denoted as $p^n(v_i^n, v_j^n)$ and vertex features, denoted as y^n are taken from the n^{th} iteration. Here, $p^n(.)$ performs calculation of the edge feature in-between vertices, and $\sigma(.)$ which is a set function, accumulates each vertex's edge features. The vertex features are then updated by $q^n(.)$ using the aggregated edge features. This process is repeated in subsequent iterations until the final output of the vertex features is produced. GNNs enhance vertex states to integrate object information in object detection. To achieve this target, the equation is modified to consider the states of the vertex's neighbors in refining the vertex's state:

$$s_i^{n+1} = q^n(\sigma(|(i,j) \in Ed), s_i^n) \tag{3}$$
$$p^n(x_j - x_i, s_j^n) = e_{ij}^n$$

Here, the the graph neural network's edge feature is denoted as e^n and edge feature function ($p^n(.)$) uses a vertex's neighbors' relative coordinates. The network's insensitivity to point cloud global shift provides translation invariance. It is nevertheless sensitive to local neighborhood translations, which increases input variance to $p^n(.)$. Instead of matching neighbors' coordinates with the central vertex's, structural characteristics might minimize variation, predict an alignment offset. Using this information, here a point cloud registration technique has been introduced in this work, using structural knowledge from prior iterations at the central vertex:

$$\Delta y_i^n = \{u^n(s_i)^n - x_i\} \tag{4}$$
$$q^n(\sigma(\ p(\Delta y_i^n + x_j, s_j^n)\,(s_i^n)) = s_i^{n+1}$$

$u^n(s_i)^n$ indicates vertex coordinate changes. $u^n(.)$ uses the previous center vertex state to calculate the offset. Setting $u^n(.)$ to zero disables the offset and returns the graph neural network (GNN) to Equation (3).

$$MLP_h^t\,(\,s_i^t\,) = \Delta x_i^t$$

The functions $p^n(.)$, $q^n(.)$, and $u^n(.)$ are modeled using multi-layer perceptrons (f_η) and an additional connection is added to $q_n(.)$. Max is chosen as $\sigma(.)$ due to its stability. A single iteration in the GNN is defined as follows:

$$\begin{aligned} f_{\eta_u}^n(s_i^n) &= \Delta y_i^n + x_i \\ f_{\eta_p}^n([\Delta y_i^n + x_j, s_j^n]) &= e_{ij}^n \\ f_{\eta_q}^n(Max((e_{ij}|(i,j)) \in Ed)) + (s_i)^n &= s_i^{n+1} \end{aligned} \tag{5}$$

The feed forward graph neural network iterates T times with a unique set of multi-layer perceptron denoted as (f_η^n) After T iterations, the vertex state value predicts the object's category and bounding box.

C. Loss:

The classification branch creates vertex probability distributions for M object classes, including the background class, denoted as p_{c_1}, ..., p_{c_M}. Vertices within bounding boxes are allocated object classes, whereas those outsides are assigned backdrop classes. Classification loss is average cross entropy [21]:

$$-\{\sum_{i=1}^{N}\sum_{j=1}^{M}(y_{c_j}^i)\log(p_{c_j}^i)\}/N = l_{cls} \tag{6}$$

Here, p_c^i is the expected probability and y_c^i 's each vertex's one-hot class label. The object bounding box's center position $(x\,,\,y\,,\,z)$ length (l) height (h), width (w), and yaw angle $(x\,,\,y,\,z)$ are predicted in a 7-degree-of-freedom format. Vertex coordinates $(x_v\,,\,y_v\,,\,z_v)$ used to encode the bounding box as the following equation:

$$\begin{aligned} (x - x_v)/l_m = \Delta x, (y - y_v)/h_m = \Delta y, (z - z_v)/w_m = \Delta z \\ \log(l/l_m) = \Delta h, \log(h/h_m) = \Delta w, \log(w/w_m) = \Delta l \\ \Delta\theta = (\theta - \theta_0)/\theta_m \end{aligned} \tag{7}$$

In the event that a vertex falls within a bounding box, the disparity between the projected value and the true value (ground truth) is computed through employment of the Huber loss. When a vertex is positioned beyond the bounds of any bounding boxes or pertains to a class not necessitating localization, the localization loss is assigned a value of zero. The average of all the localization losses are then taken, considering constant scale factors

$$\frac{1}{N}\sum_{i=1}^{N}\mathbf{1}(v_i \in b_{interest})\sum_{\delta\in\delta_{b_i}} l_{huber}(\delta - \delta^{gt}) = l_{loc} \tag{8}$$

To avoid overfitting, the study have added **L1** regularization to each multi-layer perceptron [22]. The final loss is (calculated by balancing the individual losses using constant weights **α, β**, and **γ**):

$$\alpha l_{cls} + \beta l_{loc} + \gamma l_{reg} = l_{total} \tag{9}$$

IV. EXPERIMENT

KITTI object detection [10] was used to test our design. 7380 training and 7619 testing samples both have a point cloud and camera picture. The study only processed the point cloud inside the image's field of view to match the dataset's annotations. The KITTI benchmark measures car, pedestrian, and cyclist accuracy. Cars and Pedestrians/Cyclists have distinct networks to manage scale. Filtered training samples included only items of interest. The graph neural network featured three iterations, limiting to 128 input edges per vertex during training and using

all input edges during inference. A two-layer MLP with 32 first-layer units and 3 second-layer units was used for auto-registration. The classification branch was a 32-unit single-layer MLP with the number of classes as output size. The localization branch featured a three-layer MLP with 32 units per layer and 7 outputs. The network was trained end-to-end with a batch size of 4 and loss weights of $\alpha = 0.1, \beta = 10, \gamma = 5e^{-7}$. Stochastic gradient descent with staircase learning-rate decay optimized. The Car network was trained for 1600K steps using a 0.15 initial learning rate and 0.12 decay rate per 500K steps. The Pedestrian and Cyclist network learned 0.38 and decayed 0.2 per 500K steps. It was trained for 1500K steps. Car bounding boxes (3.9m, 1.6m, 1.4m) set (l_m , h_m , w_m) to the median. Automobiles seen from the side with angles between $-\pi/4$ and $\pi/4$ are one class, whereas cars viewed from the front with angles between $\pi/4$ and $3\pi/4$ are another class. Angle threshold is $\pi/2$. Four classes, including "**Background**" and "**DoNotCare**", result. The graph has a minimum distance of 1m and a range of 4m. The point cloud is down-sampled to P′ with 0.9-meter voxels during training and 0.4 meters during inference. MLP and MLP g have identical sizes (350, 350). An MLP of size (64, 128, 256, 512) embeds the raw points, while another MLP of size (512, 512) does the Max aggregation. NMS cutoff is 0.012. The median bounding box size is (l_m , h_m , w_m), with Pedestrian at (0.88m, 1.77m, 0.65m) and Cyclist at 1.76m, 1.75m, 0.6m. The front view, side-view, Background, and Do not Care classes are classed similarly to the Car class, resulting in 6 projected classes. The graph has a radius of r = 1.6m and voxel down-sampling of 0.4 meters for training and 0.2 meters for inference. MLP_f and MLP_g have vertex state initializations of r_0 = 0.4m and MLP configurations of (256, 256). The embedding uses a MLP of (64, 128, 256, 512, 1024), whereas the aggregated feature uses (512, 512). NMS threshold is 0.22. The study augment training data to prevent overfitting. The study use global rotations, global flip, box translation, and vertex jitter to create novel ground truth boxes instead of complex methods. The point cloud undergoes random yaw rotations and x-axis flipping with a probability of 0.5 during training. Subsequently, each box and its associated points within a 110% area of the box undergo random shift by ($\Delta x \sim N(0, 3)$, $\Delta y = 0$, $\Delta z = 3$). Point selection uses a 10% larger box to avoid object truncation. The study also avoid box and background point collisions during translation. Vertex jitter is created by random voxel down-sampling during graph formation.

V. RESULT

The efficacy of the Bird's Eye View (BEV) object identification benchmark and the KITTI 3D object recognition benchmark has been subjected to scrutiny. Our findings are evaluated in the context of prior studies. The KITTI dataset is characterized by three different levels of difficulty, categorized as" easy,"" moderate," and "hard." The overall precision metric, referred to as the" Average Precision" (AP), is computed across all three difficulty levels. Our results demonstrate state-of-the-art performance across all three difficulty levels of the Car Detection problem, as well as the Cyclist Detection challenge.

Method	Mode		Bicyclist			Car			Pedestrian		
	LiDAR	Image	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
VoxelNet[4]	Yes	No	67.17	47.64	45.11	81.97	65.46	62.85	57.86	53.42	48.87
AVOD-FPN[21]	Combined		64.00	52.18	46.61	81.94	71.88	66.38	50.80	42.81	40.88
SECOND[22]	Yes	No	70.51	53.85	53.85	83.13	73.66	66.20	51.07	42.56	37.29
PointPillers[18]	Yes	No	75.78	59.07	52.92	79.05	74.99	68.30	52.08	43.53	41.49
PointRCNN[23]	Yes	No	73.93	59.60	53.59	85.94	75.76	68.32	49.43	41.78	38.63
F-PointNet[15]	Combined		71.96	56.77	50.39	81.20	70.39	62.19	51.21	44.89	40.23
GCN-Net (Ours)	Yes	No	79.36	62.45	58.09	87.32	79.55	73.19	51.92	43.77	40.14

Table 1: AP (average precision) comparison table of 3D object detection on KITTI(test) dataset.

Method	Mode		Bicyclist			Car			Pedestrian		
	LiDAR	Image	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
VoxelNet[4]	Yes	No	74.41	52.18	50.49	89.60	84.81	78.57	57.86	53.42	48.87
AVOD-FPN[21]	Combined		68.06	57.48	50.77	88.53	83.79	77.90	58.75	51.05	47.54
SECOND[22]	Yes	No	73.67	56.04	48.78	88.07	79.37	77.95	55.10	46.27	44.76
PointPillers[18]	Yes	No	79.14	62.25	56.00	88.35	86.10	79.83	58.66	50.23	47.19
PointRCNN[23]	Yes	No	81.04	65.32	57.85	89.66	87.76	86.89	60.99	51.39	45.89
F-PointNet[15]	Combined		75.38	61.96	54.68	88.70	84.00	75.33	58.09	50.22	47.20
GCN-Net (Ours)	Yes	No	80.28	68.23	58.69	92.12	88.29	84.92	56.63	48.12	45.53

Table 2: AP (average precision) comparison table of BE (bird's eye view) object detection on KITTI(test) dataset.

It is noteworthy that even in the" Easy" setting, the BEV Car Detection configuration exhibits remarkable performance. Our results, with the exception of pedestrian detection, were comparable with prior fusion-based systems in all other

categories. Our methodology also provides qualitative outcomes for each classification. The camera imagery and point cloud visualization serve as the sole source of information for our approach. Due to the absence of ground truth labels in the test dataset, the camera imagery is utilized solely for visual inspection. Despite not achieving a perfect score, our system exhibits a commendable degree of accuracy in recognizing pedestrians. The lower vertex density of the point cloud may contribute to the difficulties in constructing more accurate bounding boxes for pedestrian identification, thereby explaining its lower accuracy compared to vehicle and bicycle detection.

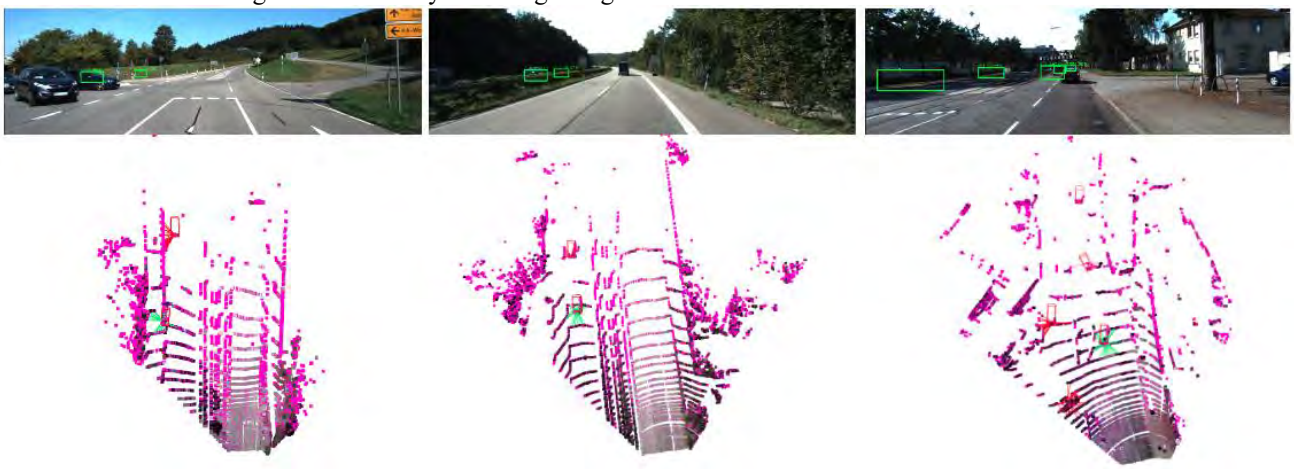

Figure 2: Pedestrian, Cyclist and Car are detected with their assigned bounding box which is green for Car, red for Pedestrian and blue for Cyclist. Object detection done on both images and point clouds.

VI. CONCLUSION

A graph neural network (GCN-Net) identifies three-dimensional objects in a point cloud. A graph model of the point cloud organizes the points without grid mapping or iterative sampling and grouping, resulting in a more compact data representation. CNNs and GNNs show great potential for applications in sectors such as drug development, fraud detection, traffic forecasting, and cybersecurity because of their capacity to evaluate intricate data structures and interactions efficiently. CNNs are used for forecasting drug-target interactions, whereas GNNs have been employed to anticipate drug-target affinity [20]. CNNs are used for image analysis tasks such as signature verification and document authentication in fraud detection, whereas GNNs are employed to identify fraudulent trends by analysing complicated connections in financial transaction data. CNNs are frequently used in cybersecurity for tasks such as detecting malware via file content analysis, whereas GNNs can scan network traffic data to identify abnormalities or intrusions. Our GCN-Net performs the best in KITTI benchmark object identification from both three-dimensional and bird's-eye views. Our research shows that our auto-registration approach, aided by a box merging and scoring procedure, may improve detection accuracy and reduce transition variation. The study wants to speed up inference by combining sensor data into a single stream.

REFERENCES

[1] M. Faisal, M. Algabri, and M. A. Mekhtiche, "Smart and Fully Functional Mobile-Robot Navigation System," in *2021 7th International Conference on Control, Automation and Robotics (ICCAR)*, 2021, pp. 76–83.

[2] Y. Chen, Y. Gao, and Y. Xu, "Research on Localization Method of Driverless Car Based on Fusion of GNSS and Laser SLAM," in *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2022, pp. 2134–2139.

[3] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, 2018.

[4] V. S. Rozario and P. Sutradhar, "In-Depth Case Study on Artificial Neural Network Weights Optimization Using Meta-Heuristic and Heuristic Algorithmic Approach," *AIUB J. Sci. Eng.*, vol. 21, no. 2, pp. 98–109, 2022.

[5] Y. Li *et al.*, "Deep learning for lidar point clouds in autonomous driving: A review," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 8, pp. 3412–3432, 2020.

[6] M. Zhang, Y. Wang, P. Kadam, S. Liu, and C.-C. J. Kuo, "Pointhop++: A lightweight learning model on point sets for 3d classification," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 3319–3323.

[7] S. Zheng and M. Castellani, "Primitive shape recognition from real-life scenes using the PointNet deep neural network," *Int. J. Adv. Manuf. Technol.*, vol. 124, no. 9, pp. 3067–3082, 2023.

[8] S. Hoque, M. Y. Arafat, S. Xu, A. Maiti, and Y. Wei, "A comprehensive review on 3D object detection and 6D pose estimation with deep learning," *IEEE Access*, vol. 9, pp. 143746–143770, 2021.

[9] R. Shahrear, M. A. Rahman, A. Islam, C. Dey, and M. S. R. Zishan, "An automatic traffic rules violation detection and number plate recognition system for Bangladesh," *AIUB J. Sci. Eng.*, vol. 19, no. 2, pp. 87–98, 2020.

[10] H. S. Gujjar, "A comparative study of VoxelNet and PointNet for 3D object detection in car by using KITTI benchmark," *Int. J. Inf. Commun. Technol. Hum. Dev.*, vol. 10, no. 3, pp. 28–38, 2018.

[11] J. Behley *et al.*, "Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset," *Int. J. Rob. Res.*, vol. 40, no. 8–9, pp. 959–967, 2021.

[12] Z. Wang, W. Zhan, and M. Tomizuka, "Fusing bird's eye view lidar point cloud and front view camera image for 3d

object detection," in *2018 IEEE intelligent vehicles symposium (IV)*, 2018, pp. 1–6.
[13] A. Ghasemieh and R. Kashef, "3D object detection for autonomous driving: Methods, models, sensors, data, and challenges," *Transp. Eng.*, vol. 8, p. 100115, 2022.
[14] A. Parashar *et al.*, "SCNN: An accelerator for compressed-sparse convolutional neural networks," *ACM SIGARCH Comput. Archit. news*, vol. 45, no. 2, pp. 27–40, 2017.
[15] A. Garcia-Garcia, F. Gomez-Donoso, J. Garcia-Rodriguez, S. Orts-Escolano, M. Cazorla, and J. Azorin-Lopez, "Pointnet: A 3d convolutional neural network for real-time object class recognition," in *2016 International joint conference on neural networks (IJCNN)*, 2016, pp. 1578–1584.
[16] M. Soelch, A. Akhundov, P. van der Smagt, and J. Bayer, "On deep set learning and the choice of aggregations," in *Artificial Neural Networks and Machine Learning--ICANN 2019: Theoretical Neural Computation: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17--19, 2019, Proceedings, Part I 28*, 2019, pp. 444–457.
[17] Y. Xu, F. Li, and A. Asgari, "Prediction and optimization of heating and cooling loads in a residential building based on multi-layer perceptron neural network and different optimization algorithms," *Energy*, vol. 240, p. 122692, 2022.
[18] J. Tu, P. Wang, and F. Liu, "PP-RCNN: Point-Pillars Feature Set Abstraction for 3D Real-time Object Detection," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.
[19] Y. Wu *et al.*, "Seastar: vertex-centric programming for graph neural networks," in *Proceedings of the Sixteenth European Conference on Computer Systems*, 2021, pp. 359–375.
[20] B. T. Yaseen, "Drug Target Interaction Prediction Using Convolutional Neural Network (CNN)," in *2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2023, pp. 1–5.

Mr. Ahmed Abdullah has a Master's degree in computer science from Bishop University in Canada and a Bachelor of Science degree in Electrical and Electronic Engineering from American International University-Bangladesh. He has accumulated eight years of teaching experience in the CSE department of Royal University Dhaka. He is certified in CCNA, CompTIA, and CEH. He has co-authored or produced several scientific publications and has taken part in many national and international conferences cantered on Data Science, Machine Learning, and IoT.

Mehzabul Hoque Nahid is an Assistant Professor in the Department of Management Information Systems at the American International University-Bangladesh. He has been a distinguished faculty member since 2015. He is now studying for a Doctor of Philosophy degree in Business and Management at Management Science University-Malaysia (MSU). He finished his postgraduate studies in Information Technology at Swinburne University of Technology in Australia, earning a master's degree. Prior to becoming an academic, he acquired professional experience via employment at several local and worldwide companies.