Adjustment of PointPillars for Effective LiDAR-Based Object Detection and Classification in Constrained Environments

Alif Ilman Nafian, *Member, IAENG*, Dian Anggraini, Mochamad Iqbal Ardimansyah, Arief Suryadi Satyawan, *Member, IAENG*

***Abstract*— Autonomous electric vehicles operating in confined environments face unique challenges, as the objects within these environments often have specific characteristics not adequately covered by general datasets. This condition is particularly relevant for the autonomous electric vehicle being developed by BRIN in Indonesia, which requires accurately recognizing and classifying objects to avoid collisions and navigate safely. To address these needs, our research proposes the development of an object detection and classification system utilizing the Velodyne VLP-16 LiDAR. Given the tendency of this LiDAR to produce sparse point clouds, we have adjusted the PointPillars method to better adapt to such data, showcasing the adaptability of our system. Our findings indicate that a backbone network model configuration based on a residual network (BaseBEVRESBackbone) outperforms the traditional PointPillars backbone configuration (BaseBEVBackbone). This superior performance is achieved even with a more straightforward layer configuration and smaller voxel size, demonstrating the effectiveness of our approach in enhancing LiDAR-based object detection and classification in constrained environments.**

***Index Terms*—Sparse Data, Adjusted PointPillars, Objects Classification, Constrained Environments, Autonomous Vehicle**

I. Introduction

SINCE 2020, we have been dedicated to developing autonomous electric vehicles designed to operate in limited environments such as office complexes, campuses, botanical gardens, and potentially for passenger transport at airports. This idea later also becomes an exciting theme in other works (see [1], [2]). Initially, our efforts focused on utilizing LiDAR technology as the primary sensor system to detect objects in front of the vehicle, ensuring it could brake to avoid collisions. This early system effectively prevented accidents by recognizing obstacles, but it quickly became evident that merely detecting objects was insufficient for safe and efficient navigation.

The current demands for autonomous vehicle systems extend beyond simple detection. It is now crucial for LiDAR technology to detect, recognize, and classify objects, enabling the vehicle to maneuver around obstacles when possible. Additionally, understanding the environment, including stationary and moving objects, is essential for planning safe routes. For example, an autonomous vehicle operating on a campus must differentiate between pedestrians, bicycles, and other vehicles and anticipate their movements to navigate safely. Similarly, in an airport setting, the vehicle must recognize and avoid luggage carts, other vehicles, and passengers. This research is at the forefront of innovation, focusing on developing an advanced LiDAR-based object detection and classification system.

By classifying objects detected by LiDAR, the system can provide the vehicle with detailed environmental awareness, allowing for sophisticated maneuvering strategies. For instance, if the vehicle detects a moving pedestrian, it can predict its path and adjust its route accordingly to avoid a collision. In scenarios involving multiple dynamic objects, the system must be capable of real-time decision-making to direct the vehicle on the safest path to its destination.

Our proposed system leverages cutting-edge advancements in LiDAR technology and data processing algorithms. One approach involves the refinement of the PointPillars method, which converts point cloud data into 2D pseudo-images that can be processed by convolutional neural networks (CNNs). This approach allows for efficient and accurate object detection and classification, even in sparse point cloud environments like those generated by the Velodyne VLP-16 LiDAR. By optimizing the number of network layers and adjusting voxel sizes, we aim to enhance the resolution and Accuracy of object detection, ensuring reliable performance in various confined settings.

Related works have demonstrated the potential of LiDAR technology in enhancing the capabilities of autonomous vehicles. For instance, the Apollo Autonomous Driving platform by Baidu integrates LiDAR data for robust object detection and path planning in complex urban environments [3]. Waymo's autonomous vehicles also utilize LiDAR for precise mapping and navigation, enabling safe operation in diverse settings, from city streets to parking lots. These projects highlight the critical role of LiDAR and

Manuscript received July 21, 2024; revised January 17, 2025.

This work was supported in part by the Indonesia Endowment Fund for Education (Lembaga Pengelola Dana Pendidikan) through the Riset dan Inovasi untuk Indonesia Maju (RIIM) program, organized by the National Research and Innovation Agency (BRIN).

A. I. Nafian is a Bachelor's graduate from the Software Engineering Department, Universitas Pendidikan Indonesia, Indonesia (e-mail: alifilmannafian6b@gmail.com).

D. Anggraini is a lecturer of the Software Engineering Department, Universitas Pendidikan Indonesia, Indonesia (e-mail: dian.anggraini@upi.edu).

M. I. Ardimansyah is a lecturer of the Software Engineering Department, Universitas Pendidikan Indonesia, Indonesia (e-mail: iqbalardimansyah@upi.edu).

A. S. Satyawan is a senior researcher of the National Research and Innovation Agency, Indonesia (e-mail: arie021@brin.go.id).

sophisticated data processing in making autonomous vehicles more adaptable and reliable [4]. In addition, Saha and Dhara's work on 3D LiDAR-based obstacle detection and tracking for autonomous navigation in dynamic environments focuses on enhancing robotic perception and navigation through advanced algorithms [5].

Their approach utilized LiDAR sensor data to detect and track obstacles in real time, which is crucial for autonomous vehicles operating in complex and changing settings. By integrating machine learning techniques with geometric analysis, their system could accurately identify and predict the movement of various obstacles, ensuring reliable navigation paths. Similarly, effort in [6] comprehensively evaluated contemporary 3D indoor scanning technologies and their point cloud generation capabilities. The study systematically compared various state-of-the-art methods, focusing on their Accuracy, efficiency, and suitability for indoor environments. The authors analyzed a range of 3D scanners and algorithms, highlighting their strengths and weaknesses in generating high-quality point clouds. By conducting rigorous experiments and assessments, this work offers valuable insights into the performance of these technologies, aiding practitioners in selecting the most appropriate tools for specific indoor scanning applications.

Our approach also benefits from advancements in the Simultaneous Localization and Mapping (SLAM) algorithms, as demonstrated in [7], describing an autonomous vehicle created by the Stanford Racing Team that won the 2005 DARPA Grand Challenge. A key component of Stanley's success was using LiDAR technology, which played a crucial role in the vehicle's perception system. The LiDAR sensors provided accurate, real-time 3D mapping of the environment, enabling the vehicle to detect and avoid obstacles, identify drivable paths, and make precise navigation decisions. This capability was instrumental in Stanley's ability to autonomously navigate complex terrains and overcome challenges such as rough roads, sharp turns, and varied obstacles, demonstrating the profound impact of LiDAR on autonomous vehicle performance. Integrating LiDAR with other sensor systems, such as GPS and cameras, and advanced algorithms for sensor fusion and path planning underscored the importance of a multi-faceted approach to autonomous navigation. This pioneering work significantly advanced the field of autonomous driving, highlighting the potential of LiDAR technology in real-world applications, further supporting our development goals.

By integrating cutting-edge technological advancements and leveraging research insights, our LiDAR-based system aims to revolutionize autonomous vehicle navigation in confined environments. The system is designed to generate high-definition maps enriched with object classifications and precise locations, predict the trajectories of dynamic objects, and plan optimal routes with enhanced efficiency. These improvements will enable autonomous vehicles to avoid obstacles, navigate complex scenarios, and reach their destinations safely and seamlessly. Ultimately, this innovation represents a significant leap forward in advancing autonomous electric vehicles tailored for operations in restricted areas.

II. The Proposed Method

PointPillars, introduced originally by [8] in 2019, is a state-of-the-art neural network architecture designed for 3D object detection using point cloud data. This technology is particularly effective in applications like autonomous driving and robotics. The key innovation in PointPillars is its method of converting 3D point clouds into 2D pseudo-images, which can then be processed using standard CNNs. This conversion significantly reduces computational complexity and improves processing speed, making real-time object detection feasible.

The core technology inside PointPillars involves several critical steps. Initially, the point cloud data is divided into vertical columns, or "pillars," each representing a segment of the 3D space. Each pillar contains a set of points within a defined grid cell. Features from these points are extracted and encoded into a feature map. This feature map is then transformed into a pseudo-image by collapsing the height dimension, allowing it to be processed by 2D CNNs. The resulting pseudo-image is used for further processing, including object classification and bounding box regression.

The mathematical model of PointPillars focuses on optimizing the loss function, which typically includes components for object classification and bounding box regression. The loss function L is defined as:

$$L = L_{cls} + \alpha L_{reg} \tag{1}$$

where L_{cls} is the classification loss, L_{reg} is the regression loss, and α is a weighting factor balancing the two components. The classification loss is usually computed using cross-entropy loss, while the regression loss, responsible for predicting the bounding box coordinates, often uses a combination of Smooth L1 loss or L2 loss. Using backpropagation and optimization methods like Adaptive Moment Estimation (Adam) or Stochastic Gradient Descent (SGD), the objective is to minimize this loss function [8]-[11].

The architecture of the PointPillars network significantly influences the performance of the model. One critical aspect is the number of layers in the 2D CNN backbone. Increasing the number of layers typically allows the network to capture more complex features, leading to better object detection and classification. However, it also increases the computational load and the risk of overfitting. Therefore, finding the optimal number of layers is crucial. Voxel size is another crucial parameter in the PointPillars architecture. Voxel size determines the resolution of the grid used to partition the point cloud data [11]. Balancing voxel size is essential to optimize both performance and efficiency.

Our proposed method aims to further enhance object detection and classification by modifying the PointPillars method to utilize LiDAR data better. By refining the construction of the number of layers and adjusting the voxel size for both BaseBEVBackbone and BaseBEVResBackbone types, we seek to optimize the handling of sparse point cloud data produced by the Velodyne VLP-16 LiDAR. This adjustment involves developing more sophisticated neural network architectures to better differentiate between objects and accurately predict

their movements.

BaseBEVBackbone is actually based on traditional neural networks, such as standard CNNs. These networks use a cascade of layers to handle input data, with each layer using pooling, activation, and convolution operations to change the input. Unlike ResNets, traditional networks do not include skip connections, which means each layer relies entirely on the previous layer's output. While these networks can be effective for many tasks, they can need help training very deep architectures due to issues like vanishing gradients.

In contrast, BaseBEVResBackbone adopts Residual Networks, or ResNets, introduced in 2015 [12]. These neural network types comprise shortcuts or skipping connections that go around one or more layers. These connections help mitigate the vanishing gradient problem, which often hampers the training of intense networks. Gradients can flow directly through a ResNet, making it easier to train far deeper designs and achieve better results on difficult problems. In the context of BaseBEVResBackbone, these residual connections improve feature learning and allow for more sophisticated representations of point cloud data.

ResNets have the advantage of improved gradient flow, enabling deeper networks and better performance on complex tasks. Although they are more adept at learning hierarchical features, the more connections they make, the higher their memory and computational complexity. Traditional networks, on the other hand, have a simpler architecture with lower computational overhead, making them easier to implement and optimize for smaller or less complex tasks. However, they can need help with training profound networks due to vanishing gradient problems and may not capture as rich features as ResNets.

Using these network architectures in object detection and classification for point cloud data has distinct advantages. ResNets (BaseBEVResBackbone) allow for better handling of complex, high-dimensional point cloud data due to their residual connections. This network leads to improved detection and classification Accuracy, as the network can learn more detailed and hierarchical features from the data. This condition benefits applications like autonomous driving, where understanding the environment in detail is crucial. Traditional networks (BaseBEVBackbone), while simpler to train and optimize, are efficient for less complex tasks or when computational resources are limited. They can still perform well in object detection and classification in point cloud data, especially when the task does not require extremely deep networks.

A comprehensive explanation of the proposed method regarding the modifications to PointPillars can be mathematically described as follows.

The process of Point Cloud representation to Pillars encompasses several key steps. Firstly, the raw point cloud data from Velodyne VLP-16 can be represented as a set of points in

$$P = \{p_i | p_i \in \mathbb{R}^4, i = 1, \dots, N\} \tag{2}$$

where each point $p_i = (x, y, z, \text{and } r)$. Here, x, y, and z are spatial coordinates, r represents reflectance value, and N is the total number of points.

Subsequently, the voxelization process is carried out. The 3D space is divided into a voxel grid of optimized size. The formula for voxel division is formulated as in

$$V = (x, y, z) = \{p_i | \lfloor x_i / v_x \rfloor = x, \lfloor y_i / v_y \rfloor = y, \lfloor z_i / v_z \rfloor = z\} \tag{3}$$

where v_x, v_y, and v_z are the voxel dimensions, and $\lfloor \cdot \rfloor$ denotes the floor function.

Finally, the transformation into Pillar representation is performed. Each pillar aggregates information from points in the vertical voxel using an encoding function. The representation is formulated as in

$$\phi(P) = \{D_{(x,y)} | x \in [1, X], y \in [1, Y]\} \tag{4}$$

where

$$D(x, y) = f_{encoder}(\{p_i | p_i \in V(x, y, *)\}) \tag{5}$$

The function $f_{encoder}$ maps the set of points into feature vectors, and X and Y are the dimensions of the BEV (Bird's Eye View) grid.

With regard to the enhanced neural network architecture, the output of the BaseBEVBackbone is formulated as in

$$Output = f_n\left(f_{n-1}\left(\dots f_1(\phi(P))\right)\right) \tag{6}$$

where each layer f_i has the structure as in

$$f_i(x) = \sigma(W_i \cdot x + b_i) \tag{7}$$

Here, W_i is the weight matrix of the i-th layer, b_i is the bias, and σ is a non-linear activation function representing convolution operations.

For the BaseBEVResBackbone, skip connections are introduced, formulated as in

$$f_i(x) = \sigma(W_i \cdot x + b_i) = x \tag{8}$$

From the perspective of object prediction, the class probability for each region proposal r is calculated as in

$$p(c|r) = softmax(W_c \cdot f_r + b_c) \tag{9}$$

and the bounding box parameters are presented as in

$$box(r) = W_b \cdot f_r + b_b \tag{10}$$

Here, $p(c|r)$ is the class probability, $box(r)$ represents bounding box parameters (dx, dy, dz, dl, dw, dh, and $d\theta$), f_r is the region feature, and W_c, W_b, b_c, and b_b are learnable parameters.

The total loss function is defined as in

$$L = \lambda_1 L_{\text{class}} + \lambda_2 L_{\text{box}} + \lambda_3 L_{\text{dir}} \tag{11}$$

Here, L_{class} is the focal loss for classification, L_{box} is the smooth L1 loss for bounding box regression, and L_{dir} is the cross-entropy loss for orientation. The parameters λ_1, λ_2, and λ_3 are weighting hyperparameters.

The above formulas (equation 2 to 11) describe the data processing pipeline from raw point cloud data to final object predictions, with an emphasis on optimizing network architecture and voxel size to enhance object detection accuracy from the relatively sparse Velodyne VLP-16 LiDAR data. Each mathematical component is interlinked in the processing pipeline, where the output of one stage becomes the input for the next. Optimization is performed at every stage to improve the overall system performance.

Our approach focuses on integrating advanced feature extraction techniques and dynamic voxelization methods to improve the resolution and Accuracy of object detection. This method allows the autonomous vehicle to identify obstacles and understand their characteristics, such as size, shape, and motion patterns. By incorporating these enhancements, the vehicle can create detailed, annotated maps of its environment, predict the trajectories of moving objects, and plan safe, collision-free paths.

Several related works have also explored the development of PointPillars for various applications. For instance, in 2020, a work extended the PointPillars framework with "3DSSD: Point-based 3D Single Stage Object Detector." This effort enhanced object detection performance by focusing on point-wise feature aggregation and single-stage detection [13]. This approach improved detection Accuracy and computational efficiency, making it suitable for real-time applications in autonomous vehicles.

Moreover, the study in 2021 on "SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Cloud" introduced a self-ensembling mechanism to refine the predictions of the PointPillars model, achieving state-of-the-art results on several benchmark datasets [14]. Their work demonstrated the potential of self-ensembling techniques in improving the robustness and Accuracy of 3D object detection systems.

Based on previous research, it is clear that several limitations and challenges still need to be addressed in the context of 3D object detection for autonomous vehicles, especially in confined environments such as the one at KST Samaun Samadikun BRIN in Bandung. The potential benefits of this research are significant. First, most previous studies have used datasets such as KITTI or nuScenes, which only partially represent the characteristics of objects in BRIN-limited environments. This condition causes models trained with that dataset to be less effective when applied in different environments. Reference [15], developed in 2021, Reference [16], developed in 2022, and Reference [17], developed in 2023 show that reliance on a combination of multisensor data such as LiDAR, radar, and high-resolution cameras requires enormous costs and computation. This method is impractical for large-scale or resource-limited applications. Reference [8], discussing the original PointPillars in 2019, and Reference [18], introducing 'SECOND: Sparsely Embedded Convolutional Detection' in 2018, have proposed a more computationally efficient method. However, it still uses data from the KITTI dataset, which is less representative of the BRIN environment and often only detects particular objects, such as cars, pedestrians and cyclists. Therefore, this research seeks to develop a 3D object detection model that is more adaptive to the characteristics of objects in BRIN's limited environment by utilizing the PointPillars-based Deep Neural Network method. This method is expected to provide better efficiency and Accuracy without relying on high-resolution sensor data, thereby offering a more cost-effective and practical solution for 3D object detection in confined environments.

Our proposed method specifically targets optimizing the PointPillars framework to handle the unique challenges posed by sparse point cloud data generated by the Velodyne VLP-16 LiDAR. Unlike prior studies that have mainly focused on improving detection accuracy and processing speed, our approach prioritizes the effective management of data sparsity. By strategically adjusting voxel size and layer configurations, we aim to minimize the merging of different objects within a single voxel, thereby reducing classification errors and enhancing detection reliability. These modifications enable the framework to better differentiate objects, which is crucial in environments where data points are limited or scattered. The result is an improved version of PointPillars that significantly bolsters the LiDAR's ability to accurately detect and classify objects, even in complex and constrained spaces. This advancement is particularly beneficial for autonomous electric vehicles, allowing them to navigate intricate environments more effectively, anticipate and avoid potential obstacles, and optimize their path planning. Consequently, our method ensures safer and more efficient operation of these vehicles, promoting better route optimization and hazard prevention in limited and cluttered environments.

III. Research Method

Figure 1 outlines the research method employed in this study, which was conducted in three primary phases described as follows. The first phase involved creating a specific dataset from scratch. The raw dataset, captured by a Velodyne VLP-16 LiDAR, was in a point cloud format and saved as '.pcap' files. Thanks to VeloView, the application that had been provided freely to obtain the raw dataset. The critical information in the point cloud included the distance between the LiDAR and the points on the surfaces of detected objects, where the transmitted light from the LiDAR was reflected, as well as the three-dimensional coordinates (x, y, and z) of each detected point [19]-[21]. The raw dataset was then processed according to the KITTI standard (see [11], [22], [23]), 'bin', before being converted to a custom format named '.npy'. After that, hundreds of thousands of frames consisting of point cloud data in that format were selected to become 2,423 frames. This process could reduce the almost identical frames. About 90% of them were selected as a training dataset, while the rest were available as a testing dataset [24]. They were ready to be annotated using Supervisely (available at: https://supervisely.com/) during the following process. In the dataset annotation process, six classes of objects were to be annotated in the dataset. They were human, wall, car, cyclist, cart and tree. After the process of annotating these objects, the '.json' format, the dataset format given by

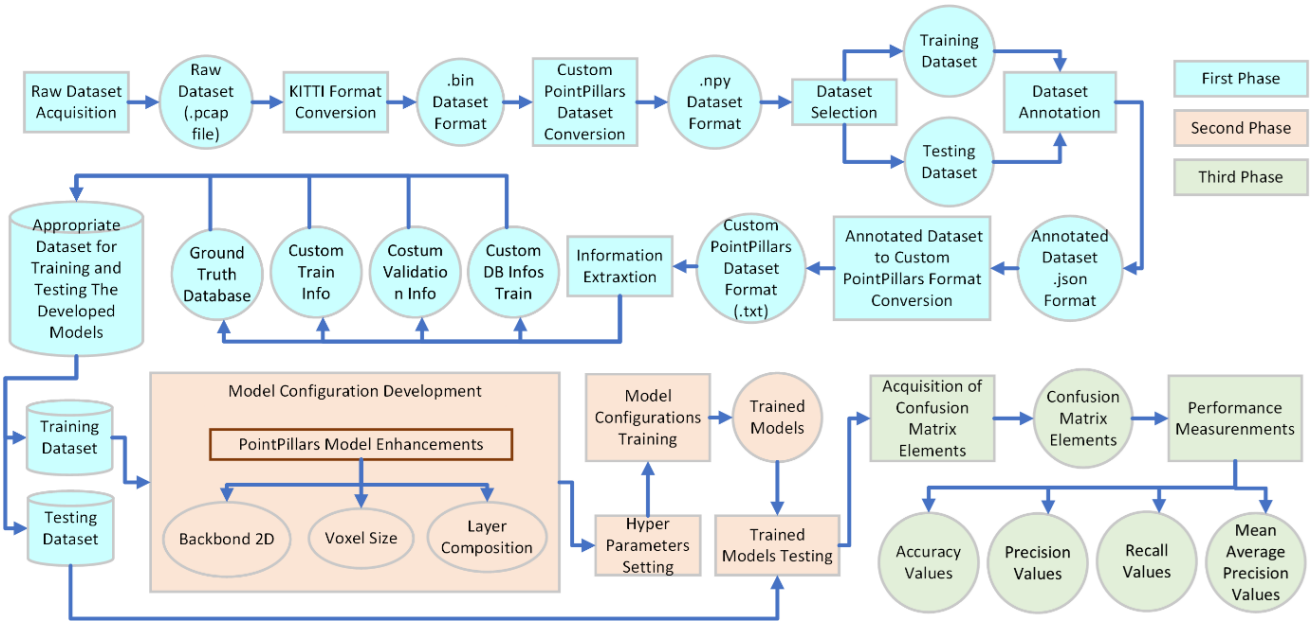


Fig. 1. The research process diagram.

Supervisely, was converted to a costume PointPillars dataset ('.txt'). This process aimed to enable ground truth database, custom train info, custom validation info, and custom database infos train through an information extraction process. Finally, the appropriate dataset for training and testing models developed in the further step, which was presented together with a complete set of the PointPillars format, could be established. This dataset, crucial for training and testing models, could be formed according to the train and test proportion.

In the second phase, as depicted in Figure 1, the focus shifted to the development of eight model configurations of PointPillars. These models, a key component of our study, were designed to optimize the performance of the neural network. The network configuration employed two distinct 2D backbone networks: BaseBEVBackbone and BaseBEVResBackbone. The BaseBEVBackbone, a standard component of our models, utilized convolutional blocks without residual connections. These blocks, consisting of layers including zero padding, convolution, batch normalization, and ReLU activation, were stacked multiple times to extract features from the input data. On the other hand, the BaseBEVResBackbone, a unique component of our models, leveraged residual blocks instead of standard convolutional blocks. These residual blocks, incorporating bypass connections, were instrumental in allowing gradients to flow directly through layers, thereby enhancing the network's learning capabilities. Subsequently, all model configurations needed to specify the number of layers and voxel size for the 2D backbones. The voxel size, a critical parameter, influenced the collection of points used for training, while the number of layers determined the structure and types of layers in a PointPillars model. The choice of these values was crucial as they significantly impacted the model's detection performance. The specific configurations of these values for each model are detailed in Table I. The first model configuration was promoted firstly by Lang et al., while the rest were our innovative contributions. After all model configurations had been established, each could be

TABLE I
CONFIGURATION OF THE DEVELOPED POINTPILLARS MODELS

Model Configuration	Backbone	Layer Composition	Voxel Size
1	BEV	3;5;5	16,000
2	BEV	3;5;5	8,000
3	BEV	3;5;5	32,000
4	BEV	4;6;6	16,000
5	BEVRes	2;3;3	16,000
6	BEV	3;5;5	64,000
7	BEVRes	2;3;3	8,000
8	BEVRes	2;3;3	64,000

trained using the same training dataset that had also been prepared at the final process of the first phase. However, before the training process began, hyperparameters, such as the type of optimizer, learning rate initiation point, number of epochs, and batch sizes, should be defined to minimize the training loss. Eight trained model configurations could be formed after the training process, and they were ready to be evaluated in the next phase.

Given that the performance of the trained models in 3D object detection and classification based on point clouds would be evaluated using Precision, Recall, F1-Score, Accuracy, and Mean Average Precision (mAP), a comprehensive set of confusion matrix elements needed to calculate these metrics had to be obtained at the start of the third phase. Reference [25] and [26] describe the confusion matrix as a table that assesses an object detection algorithm's performance, comparing predicted object classes with actual (ground truth) object classes. It contains values for True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). True Positives occur when the model correctly identifies and classifies an object, such as recognizing a car as a car. True Negatives

represent instances where the model correctly determines that no object is present or accurately labels a region as background, like recognizing an area without objects. False Positives happen when the model incorrectly detects an object or misclassifies a background area as an object, such as mistaking a background for a car or identifying a tree as a car. False Negatives occur when the model fails to detect an object or incorrectly classifies an object as background, such as missing a car in the point cloud data or labelling a car as background. True Negatives only influence the Accuracy value among these elements, while the other three elements affect all five-performance metrics.

Prior to determining the elements of the confusion matrix, it is essential to first calculate the Intersection over Union (IoU). IoU quantifies the degree of overlap between the predicted bounding box and the ground truth bounding box. It is defined as the ratio of the area of their intersection to the area of their union, providing a standardized measure of prediction accuracy. This metric is mathematically expressed as in

$$IoU = \frac{\text{the area of the intersection}}{\text{the area of the union of the predicted and ground truth bounding boxes}} \quad (12)$$

It is a critical performance metric that decides whether a predicted bounding box is a True Positive, False Positive, or False Negative. The IoU does not directly affect True Negative values since TNs pertain to background regions or non-object areas.

Since this research utilized six classes of objects, an illustration described in Table II can express a predefined process of finding a confusion matrix for this case.

Each value filled out from the observation of the testing dataset. For instance, when cars actually defined in the testing dataset were predicted appropriately by a model, the results can be stated cumulatively as the presence of car (a); otherwise, they could be avowed cumulatively as the presence of cyclist (b), human (c), wall (d), tree (e), or even cart (f). This mechanism applies to the rest of the classes. From then on, each element in Table II can be used to obtain the values of TP, FP, FN, and TN using appropriate formulation as described in Table III.

Finally, four other performance metrics above could be obtained, as follows. Precision measures the proportion of True Positive predictions out of all positive predictions made by the model, indicating how many detected objects are actually relevant. Precision can be calculated as in

$$Precision = \frac{TP}{TP+FP} \quad (13)$$

High Precision means a low False Positive rate. Recall, or sensitivity, measures the proportion of True Positive predictions out of all actual positive instances, indicating how well the model identifies all relevant objects. Recall can be calculated as in

$$Recall = \frac{TP}{TP+FN} \quad (14)$$

High Recall means a low False Negative rate. The F1-Score, the harmonic mean of Precision and Recall, provides a single metric balancing both concerns and is useful for imbalanced datasets. F1-Score can be calculated as in

$$F1 - Score = \frac{2*precision*recall}{precision+recall} \quad (15)$$

Accuracy measures the proportion of correct predictions (True Positives and True Negatives) out of the total predictions. It offers a general idea of performance but may not be the best metric for imbalanced datasets as it does not differentiate error types. Accuracy can be calculated as in

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (16)$$

Mean Average Precision (mAP) is the mean of the Average Precision (AP) values for each class. AP is calculated as the area under the Precision-Recall curve for a specific class. mAP provides a comprehensive measure of performance across all classes, combining Precision and Recall for each, making it essential for evaluating multi-class detection tasks by accounting for detection quality and the model's ability to handle different classes [27]. In this research, mAP was obtained by extending the mAP calculation to consider the top K predictions made by a model. It is beneficial in scenarios where we are interested in the model's ability to correctly rank the most relevant predictions among the top K results. This metric is valuable for assessing models in ranking-related tasks, such as recommender systems. It can be effectively adapted for 3D object detection and classification by considering the rank of the 3D predicted

TABLE II
A PREDEFINED PROCESS OF FINDING A CONFUSION MATRIX FOR SIX CLASSES OF OBJECT

		Predicted					
	Classes	car	cyclist	human	wall	tree	cart
Actual	car	*a*	*b*	*c*	*d*	*e*	*f*
	cyclist	*g*	*h*	*i*	*j*	*k*	*l*
	human	*m*	*n*	*o*	*p*	*q*	*r*
	wall	*s*	*t*	*u*	*v*	*w*	*x*
	tree	*y*	*z*	*aa*	*bb*	*cc*	*dd*
	cart	*ee*	*ff*	*gg*	*hh*	*ii*	*jj*

TABLE III
CONFUSION MATRIX FOR SIX CLASSES OF OBJECT

Classes	**TP**	**FP**	**FN**	**TN**
car	*a*	*g+m+s+y+ee*	*b+c+d+e+f*	*h+i+j+k+n+o+p+q+t+u+v+w+z+aa+bb+cc+ff+gg+hh+ii*
cyclist	*h*	*b+n+t+z+ff*	*g+i+j+k+l*	*a+c+d+e+f+m+o+p+q+r+s+u+v+w+x+y+aa+bb+cc+dd+ee+gg+hh+ii+jj*
human	*o*	*c+i+u+aa+gg*	*m+n+p+q+r*	*a+b+d+e+f+g+h+j+k+l+s+t+v+w+x+y+z+bb+cc+dd+ee+ff+hh+ii+jj*
wall	*v*	*d+j+p+bb+hh*	*s+t+u+w+x*	*a+b+c+e+f+g+h+i+k+l+m+n+o+q+r+y+z+aa+cc+dd+ee+ff+gg+ii+jj*
tree	*cc*	*e+k+q+w+ii*	*y+z+aa+bb+dd*	*a+b+c+d+f+g+h+i+j+l+m+n+o+p+r+s+t+u+v+x+ee+ff+gg+hh+jj*
cart	*jj*	*f+l+r+x+dd*	*ee+ff+gg+hh+ii*	*a+b+c+d+e+g+h+i+j+k+m+n+o+p+q+s+t+u+v+w+y+z+aa+bb+cc*

bounding boxes (TPs) in a list composed of predicted and unpredicted 3D bounding boxes (TPs and FNs). Obtaining mAP began with calculating AP for each class, as denoted in

$$AP_i = \frac{1}{n}\sum_{m=1}^{K} Precision@m * R(m) \quad (17)$$
$$R(m) = \begin{cases} 1, & if\ item\ at\ m_{th} rank\ is\ relevant \\ 0, & otherwise \end{cases}$$

where the items are relevant divided by the total number of relevant items (n) in the top K recommendations. By doing this, the maximum values of AP can be reached 1 when all relevant items stay in the first ranks, while the minimum can happen when all relevant items sit in the lower ranks. This research considered all relevant items on the lower ranks, avoiding a too-optimistic judgement. Ultimately, mAP could be calculated by averaging the AP scores across all object classes, as defined in

$$mAP = \frac{\sum_{i=1}^{Q} AP_i}{Q} \; ; Q = the\ total\ number\ of\ classes \quad (18)$$

IV. Results and Discussion

A. Dataset Production Results

To create the dataset, we embarked on our research journey by gathering a comprehensive LiDAR point cloud data from a specific area at KST Samaun Samadikun BRIN in Bandung, Indonesia. Figure 2 shows the road route indicated by the yellow line around the office building, where LiDAR recognizes essential objects constantly visible on the body or side of the road. The point cloud, a rich representation, contains six classes of objects found along the roads within that area. To capture this wealth of data, we utilized the Velodyne VLP-16 LiDAR device, capable of capturing up to 600,000 cloud data points per second. Figures 3.a and 3.b visualize the point cloud data,

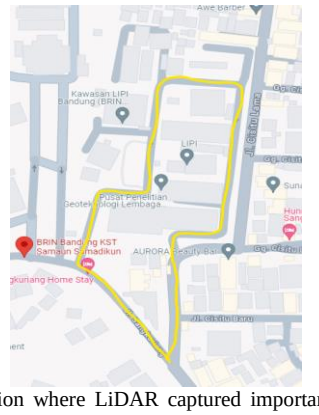
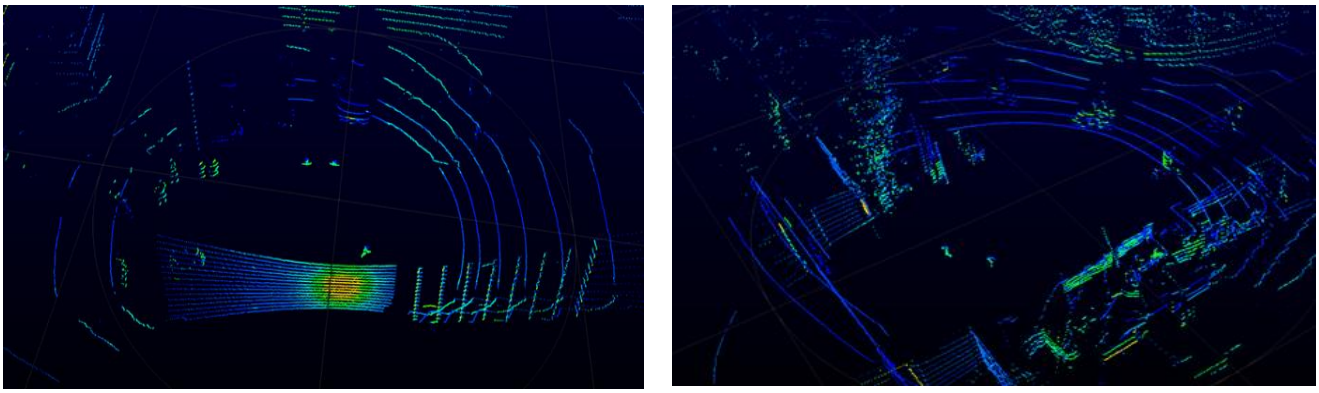


Fig. 2. The location where LiDAR captured important objects at KST Samaun Samadikun BRIN in Bandung, Indonesia.

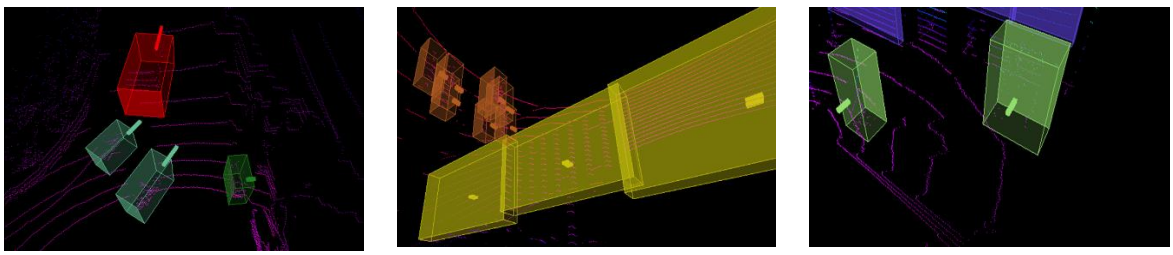
representing these objects in a single image. These objects were displayed in the bird's eye vision provided by the VelovView application. The point cloud tends to be sparse. This condition occurs because of the characteristics of the LiDAR Velodyne VLP-16, which is categorized as low-resolution LiDAR but has minor computing power compared to other types of Velodyne LiDAR. In both images, the solid points at the bottom represent the ground plane, while the sparse points at the top represent background objects or noise in the data. In Figure 3.a, it can be seen that several objects, such as humans and walls, were clearly visualized, while cars were pretty visible even though the point cloud was relatively sparse. In Figure 3.b, the cyclists and trees can be seen clearly, while the visible point cloud for the carts was relatively sparse. Finally, all information captured by LiDAR was saved into raw point cloud data in '.pcap' format.

The point cloud data, meticulously stored and processed, was transformed into a comprehensive dataset. This was achieved by annotating six object classes on each point cloud data frame using a web-based application called Supervisely. Figures 4.a, 4.b, and 4.c vividly depict the



(a) (b)

Fig. 3. Visual representation of objects in two LiDAR point cloud frames.



(a) (b) (c)

Fig. 4. Visualization of annotation outcomes on three different frames: car, cyclist, and human classes (a), human and wall classes (b), and tree class (c).

outcomes of this detailed annotation process on such object classes that occurred in three different frames. This research carried out an annotation process on 2423 frame point clouds. This crucial annotation process produced structured data, which was then meticulously formatted into a '.bin' file. Next, this '.bin' file was converted into '.npy' or numpy format, a data format widely recognized and used in machine learning and 3D object detection. The 2,438 datasets that had been converted were then separated into two parts, namely training data and testing data. A total of 2,180 datasets were allocated for training the PointPillars model with eight different configurations, while the remaining 243 datasets were used for testing the trained models.

B. Training Phase Results

Each of the eight PointPillars model configurations, rigorously assessed during the training phase using the identical dataset, demonstrated remarkable precision. The outcomes of this meticulous training process are detailed in Figure 5. Notably, every model configuration achieved its lowest training loss at a specific point during the training, with a minimum training loss of 0.09 serving as the threshold for a well-trained model. Figure 5 illustrates the efficiency of the training process, with configurations 1 and 8 converging at the minimum training loss value in the fewest number of training steps, around 7000000 steps (7,000,000/2,195 = 3,189 epochs). Configurations 5 and 3 followed suits, taking around 7,500,000 steps (7,500,000/2,195 = 3,416 epochs). Configuration 7 required around 8500000 steps (8,500,000/2,195 = 3,872 epochs), while model configurations 4 needed 9,200,000 steps (9,200,000/2,195 = 4,191 epochs). Model configuration 2 and 6, although requiring more steps, still demonstrated the efficiency of the process with 10,500,000 steps (10,500,000/2,195 = 4,783 epochs) and 11,000,000 steps (11,000,000/2,195 = 5,011 epochs) respectively.

In the training process of the eight model configurations, a comprehensive testing and optimization of hyperparameters was conducted to ensure the best model

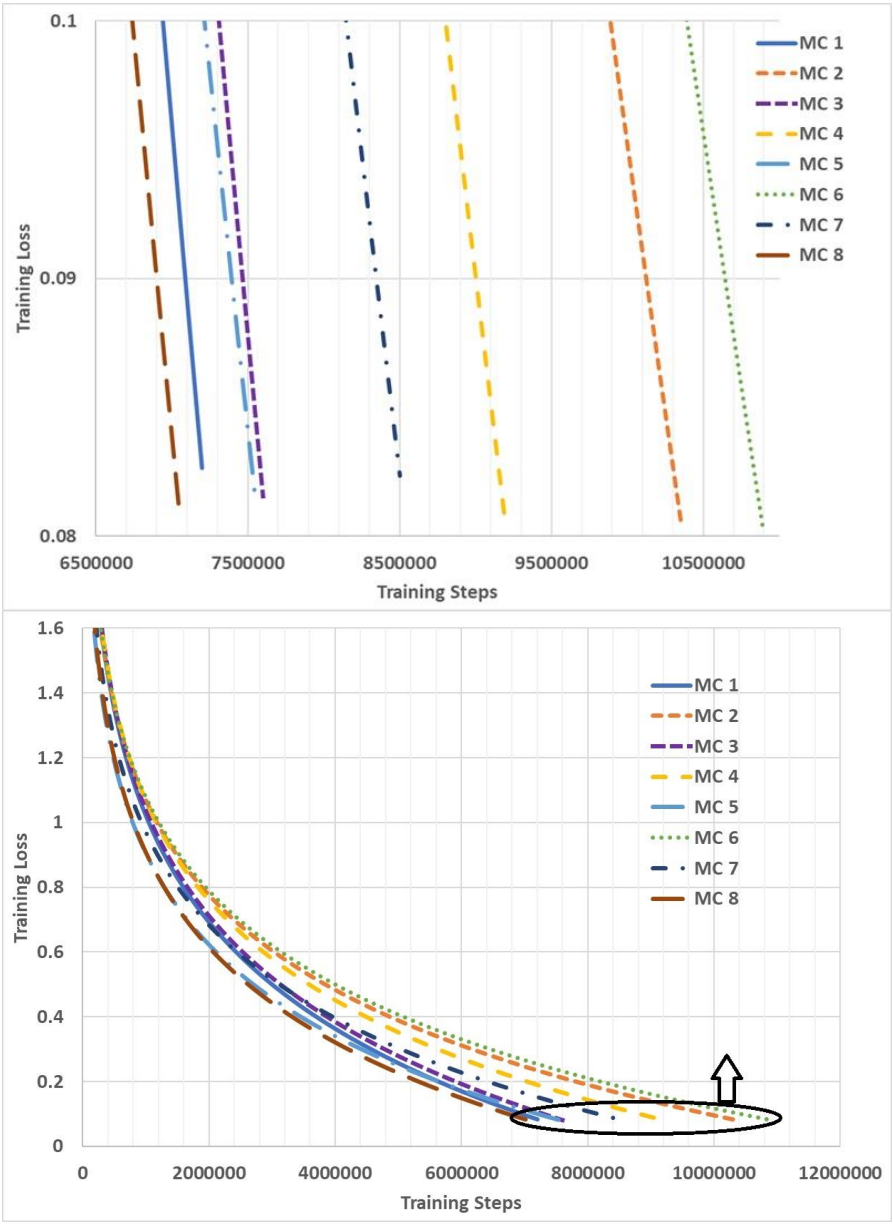


Fig. 5. Training loss results for each model. The lower figure displays graphs for all models across the entire range of obtained data, while the upper figure represents a magnified view of the area marked by a black circle in the lower figure, showing detailed graphs of all models within the training loss range of 0.08 to 0.1 and training steps range of 6,500,000 to 11,000,000.

performance. Among the essential hyperparameters was the Adaptive Moment Estimation with Weight Decay (AdamW) optimizer, a version that overcomes the shortcomings of its predecessor by incorporating a more effective way to implement weight decay. The BATCH_SIZE_PER_GPU was set at 1, corresponding to the number of GPUs used in this research. The initial learning rate was set at 0.003, weight decay at 0.01, momentum at 0.9, learning rate decay at 0.1, and LR_CLIP at 0.0000001. This LR_CLIP hyperparameter was used to prevent the learning rate from becoming too small and hindering training progress. The thoroughness of the hyperparameter testing is evident in the complete settings presented in Table IV. The number of parameters resulting from the eight model configurations can be seen in Table V, further demonstrating the robustness of the model. Tables IV and V offer crucial insights into the model configurations and the number of parameters each setup produced. Configurations 1, 2, 3, and 6 shared a total of 4,874,668 parameters, highlighting a consistent parameter count across these models. Configuration 4, with 5,651,508 parameters, had a more complex structure, while Configurations 5, 7, and 8 had the highest parameter count at 6,472,500, reflecting potentially more intricate processing capabilities.

TABLE IV
THE CONFIGURATION OF HYPERPARAMETERS

Hyperparameters	Numbers
BATCH_SIZE_PER_GPU	1
OPTIMIZER	AdamW
LEARNING RATE	0.003
WEIGHT_DECAY	0.01
MOMENTUM	0.9
LR_DECAY	0.1
LR_CLIP	0.0000001

TABLE V
THE NUMBER OF PARAMETERS PRODUCED BY EACH PARAMETER

Model Configuration	Parameters
1, 2, 3, 6	4,876,468
4	5,651,508
5, 7, 8	6,472,500

C. Testing Phase Results

In the testing process, each trained model configuration was tested using a 243-frame dataset. During this evaluation process, we meticulously assessed the performance of eight model configurations, aiming to identify their strengths and areas for improvement. The results presented in Table VI, VII, VIII, IX, X, XI, XII, and XIII are comprehensively discussed in the following paragraphs.

Model Configuration 1 informed in Table VI, the default version, showcased its potential with Accuracy values greater than 0.9 for each class. Precision, however, showed significant variation, ranging from a maximum of 1 for the car class to a minimum of 0.63 for the tree class. Recall values spanned from 0.56 for the cyclist class to 0.95 for the wall class. The F1-Score, a key metric, ranged from 0.63 for the cyclist to 0.94 for the human class, while AP values fluctuated between 0.35 for the cyclist and 0.86 for the wall class. Despite its decent performance, its mAP score was the lowest at 0.64, indicating the potential for further improvement.

Model Configuration 2 presented in Table VII showed an improvement, achieving Accuracy values above 0.94 for each class. Precision remained high, with a maximum of 1 for the car class and a minimum of 0.79 for the cyclist class. Recall values ranged from 0.85 for the cyclist class to 0.95 for the car class. The F1-Score varied from 0.82 for the cyclist to 0.97 for the car, and AP values ranged between 0.68 for the cyclist and 0.86 for the car. With a mAP score of 0.78, this configuration performed significantly better than the default.

Model Configuration 3 achieved an Accuracy of more than 0.92 for each class. Precision varied from 0.71 for the

TABLE VI
MODEL CONFIGURATION 1

Class	TP	FP	FN	TN	Accuracy	Precision	Recall	F1-Score	AP
car	176	0	32	1,571	0.982012366	1	0.846153846	0.916666667	0.662061733
cyclist	146	51	114	1,468	0.907251265	0.741116751	0.561538462	0.638949672	0.358144493
human	708	47	42	982	0.949971894	0.937748344	0.944	0.940863787	0.82967383
wall	187	39	8	1,545	0.973580663	0.827433628	0.958974359	0.888361045	0.865885776
tree	89	51	17	1,622	0.961776279	0.635714286	0.839622642	0.723577236	0.655069656
cart	192	93	68	1,426	0.909499719	0.673684211	0.738461538	0.704587156	0.526918032
Average									0.649625587

TABLE VII
MODEL CONFIGURATION 2

Class	TP	FP	FN	TN	Accuracy	Precision	Recall	F1-Score	AP
car	176	0	8	1,548	0.995381062	1	0.956521739	0.977777778	0.860135945
cyclist	215	55	36	1,426	0.947459584	0.796296296	0.856573705	0.825335893	0.682751267
human	643	22	63	1,004	0.950923788	0.966917293	0.910764873	0.938001459	0.763943698
wall	220	34	12	1,466	0.973441109	0.866141732	0.948275862	0.905349794	0.84056931
tree	109	13	9	1,601	0.987297921	0.893442623	0.923728814	0.908333333	0.791665412
cart	221	24	20	1,467	0.974595843	0.902040816	0.917012448	0.909465021	0.776801249
Average									0.785977814

tree class to 0.99 for the car class. Recall values ranged from 0.70 for the cart class to 0.94 for the car and tree classes. The F1-Score spanned from 0.75 for the cart class to 0.96 for the car class, while AP values ranged between 0.48 for the cart and 0.83 for the car class. This configuration had a mAP score of 0.72, indicating moderate performance. The performance results of Model Configuration 3 are shown in Table VIII.

Model Configuration 4 described in Table IX showed high Accuracy above 0.95 for each class. Precision varied from 0.84 for the cyclist class to 0.99 for the car class. Recall ranged from 0.82 for the cart class to 0.98 for the car and wall classes. The F1-Score ranged from 0.85 for the cart to 0.99 for the car, with AP values between 0.63 for the cart and 0.95 for the car and wall classes. This configuration had a high mAP score of over 0.8, demonstrating strong overall performance.

Model Configuration 5 indicated in Table X also achieved Accuracy values above 0.95 for each class. Precision varied from 0.83 for the cyclist class to 0.98 for the car class. Recall ranged from 0.83 for the car and cyclist classes to 0.99 for the wall class. The F1-Score spanned from 0.83 for the cyclist class to 0.96 for the wall class, with AP values ranging from 0.64 for the cyclist to 0.98 for the wall class. Its mAP score was 0.76, showing solid performance, particularly in detecting the wall class.

Model Configuration 6 (Table XI), while not the top scorer, still demonstrated a balanced performance, with an Accuracy above 0.95 for each class. Precision, a crucial metric, ranged from 0.80 for the cyclist class to 0.98 for the

TABLE VIII
MODEL CONFIGURATION 3

Class	TP	FP	FN	TN	Accuracy	Precision	Recall	F1-Score	AP
car	121	1	7	1,799	0.995850622	0.991803279	0.9453125	0.968	0.835685705
cyclist	215	59	40	1,614	0.948651452	0.784671533	0.843137255	0.812854442	0.657321735
human	727	51	84	1,066	0.929979253	0.934447301	0.896424168	0.915040906	0.738626307
wall	327	45	21	1,535	0.965767635	0.879032258	0.939655172	0.908333333	0.821114966
tree	111	44	7	1,766	0.973547718	0.716129032	0.940677966	0.813186813	0.825991701
cart	188	39	80	1,621	0.938278008	0.828193833	0.701492537	0.75959596	0.487409414
Average									0.727691638

TABLE IX
MODEL CONFIGURATION 4

Class	TP	FP	FN	TN	Accuracy	Precision	Recall	F1-Score	AP
car	172	1	2	1,520	0.998230088	0.994219653	0.988505747	0.991354467	0.950707654
cyclist	184	35	13	1,463	0.971681416	0.840182648	0.934010152	0.884615385	0.810452824
human	652	20	33	990	0.968731563	0.970238095	0.951824818	0.960943257	0.847219818
wall	261	16	3	1,415	0.98879056	0.942238267	0.988636364	0.964879852	0.950325009
tree	106	9	14	1,566	0.986430678	0.92173913	0.883333333	0.90212766	0.732592306
cart	211	28	44	1,412	0.957522124	0.882845188	0.82745098	0.854251012	0.635548038
Average									0.821140941

TABLE X
MODEL CONFIGURATION 5

Class	TP	FP	FN	TN	Accuracy	Precision	Recall	F1-Score	AP
car	183	3	35	1,587	0.978982301	0.983870968	0.839449541	0.905940594	0.652444046
cyclist	220	42	44	1,502	0.952433628	0.839694656	0.833333333	0.836501901	0.643533676
human	695	39	28	1,046	0.962942478	0.946866485	0.961272476	0.9540151	0.869703501
wall	272	19	1	1,516	0.988938053	0.934707904	0.996336996	0.964539007	0.980924567
tree	90	5	16	1,697	0.988384956	0.947368421	0.849056604	0.895522388	0.668509275
cart	207	33	17	1,551	0.972345133	0.8625	0.924107143	0.892241379	0.790453263
Average									0.767594721

TABLE XI
MODEL CONFIGURATION 6

Class	TP	FP	FN	TN	Accuracy	Precision	Recall	F1-Score	AP
car	185	16	31	1,611	0.974498101	0.92039801	0.856481481	0.887290168	0.67700314
cyclist	220	52	29	1,542	0.956049919	0.808823529	0.883534137	0.84452975	0.718565354
human	641	12	45	1,145	0.969072165	0.981623277	0.934402332	0.957430919	0.809478434
wall	288	40	19	1,496	0.967986978	0.87804878	0.938110749	0.907086614	0.818051808
tree	105	22	12	1,704	0.981551818	0.826771654	0.897435897	0.860655738	0.74394899
cart	234	28	34	1,547	0.966359197	0.893129771	0.873134328	0.883018868	0.701866907
Average									0.744819106

human class. Recall values varied between 0.85 for the car class and 0.93 for the human and wall classes. The F1-Score, a measure of balance between Precision and Recall, ranged from 0.84 for the cyclist to 0.95 for the human class, with AP values between 0.67 for the car and 0.81 for the wall class. The mAP score of 0.74, while slightly lower than the top performers, still reflects a balanced performance, indicating that the efforts put into this configuration were not in vain.

Model Configuration 7 achieved the highest Accuracy above 0.96 for each class. Precision ranged from 0.87 for the tree class to 0.98 for the car class. Recall values varied from 0.83 for the cyclist to 0.98 for the car class. The F1-Score ranged from 0.86 for the cyclist to 0.98 for the car class, with AP values between 0.63 for the cyclist and 0.95 for the car class. This configuration demonstrated the best overall performance with a mAP score of over 0.8. The results for this Model Configuration can be seen in Table XII.

Finally, in Table XIII, Model Configuration 8 also achieved an Accuracy above 0.96 for each class. Precision varied from 0.84 for the cyclist class to 0.98 for the car class. Recall ranged from 0.85 for the cyclist class to 0.96 for the human class. The F1-Score spanned from 0.84 for the cyclist to 0.96 for the human class, with AP values ranging from 0.67 for the cyclist to 0.87 for the human class. The mAP score was 0.76, indicating a strong performance similar to Model Configuration 5.

Figure 6 presents a visualization of the objects predicted by Model Configuration 4 on two examples from the testing dataset. In Figure 6.a, the model correctly identified two walls (indicated by blue boxes) and two humans (indicated by green boxes). However, it incorrectly predicted one car (indicated by yellow boxes) and two carts (indicated by purple boxes). In Figure 6.b, the model accurately predicted two cars (yellow boxes), two walls (blue boxes), two cyclists (red boxes), two humans (green boxes), one cart (purple box), and one tree (blue box). These visualizations highlight the model's Accuracy and areas for improvement in object detection.

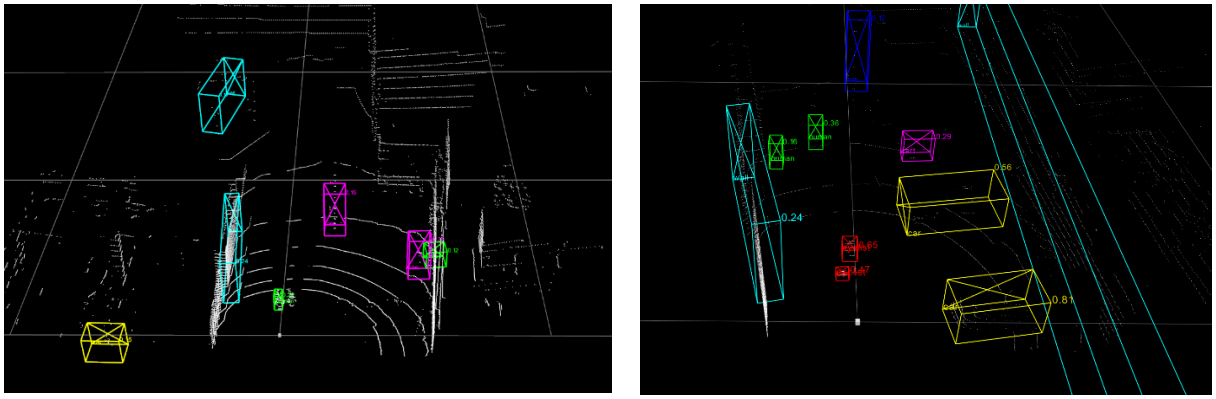
The analysis of the PointPillars model using the 2D BaseBEVBackbone configuration revealed that increasing the voxel size beyond 8,000 did not enhance the mean Average Precision (mAP) value. This observation held true across various model configurations: Model Configuration 2 with 8,000 voxels, Model Configuration 1 with 16,000 voxels, Model Configuration 3 with 32,000 voxels, and

TABLE XII
MODEL CONFIGURATION 7

Class	TP	FP	FN	TN	Accuracy	Precision	Recall	F1-Score	AP
car	188	3	2	1,471	0.996995192	0.984293194	0.989473684	0.98687664	0.953969479
cyclist	176	19	36	1,433	0.966947115	0.902564103	0.830188679	0.864864865	0.639672854
human	673	21	31	939	0.96875	0.969740634	0.955965909	0.962804006	0.856862964
wall	240	24	17	1,383	0.975360577	0.909090909	0.93385214	0.921305182	0.809551586
tree	94	14	9	1,547	0.986177885	0.87037037	0.912621359	0.890995261	0.771378657
cart	192	20	6	1,446	0.984375	0.905660377	0.96969697	0.936585366	0.855283925
Average									0.814453244

TABLE XIII
MODEL CONFIGURATION 8

Class	TP	FP	FN	TN	Accuracy	Precision	Recall	F1-Score	AP
car	179	2	20	1579	0.987640449	0.988950276	0.899497487	0.942105263	0.745777509
cyclist	197	36	34	1513	0.960674157	0.845493562	0.852813853	0.849137931	0.671462259
human	698	25	27	1030	0.970786517	0.965421853	0.962758621	0.964088398	0.87340868
wall	255	21	17	1487	0.978651685	0.923913043	0.9375	0.930656934	0.816979846
tree	103	18	16	1643	0.980898876	0.851239669	0.865546218	0.858333333	0.692457326
cart	217	29	17	1517	0.974157303	0.882113821	0.927350427	0.904166667	0.796695708
Average									0.766130221



(a) (b)

Fig. 6. Visualization of predicted objects from the testing phase.

Model Configuration 6 with 64,000 voxels. The underlying cause of this phenomenon was the nature of the point cloud data generated by the employed LiDAR, which produced sparse point clouds. When smaller voxels were used, the likelihood of combining distinct objects within a single voxel was minimized, thereby reducing classification confusion between different objects.

Additional layer configurations were necessary to improve the mAP value, as demonstrated by Model Configuration 4, which used a layer composition of 4;6;6 and 16,000 voxels. However, this enhancement came at the cost of increased training steps needed to achieve a training loss convergence value of 0.09, indicating a more extended training duration. Similar outcomes were noted in the configurations using a 2D BaseBEVRESBackbone with a layer composition of 2;3;3. For instance, Model Configuration 7 with 8,000 voxels achieved a better mAP value than configurations with 16,000 (Model Configuration 5) and 32,000 voxels.

Based on the configuration details provided for Model Configurations 1, 2, 3, 4, and 6 using the BEV backbone, an analysis of prediction errors reveals key insights into their performance differences due to voxel size and layer composition.

Model Configurations 1, 2, 3, and 6 share the same layer composition of 3;5;5 but differ in voxel size. Model Configuration 1 uses a voxel size of 16,000, resulting in strong car detection with 0 FP but encountering 32 FN, indicating challenges in detecting partially obscured cars. Cyclist detection shows significant issues with 51 FP and 114 FN, suggesting difficulties in distinguishing cyclists from similar objects, especially at a distance or when overlapping. Human detection records 47 FP and 42 FN, potentially due to insufficient detail in the point cloud data, while walls exhibit 39 FP and 8 FN, showing misclassification in identifying elongated structures. The tree class reports 51 FP and 17 FN, indicating challenges in capturing fine details. The cart class faces 93 FP and 68 FN, highlighting problems due to shape similarities with other objects.

Model Configuration 2, with a smaller voxel size of 8,000, shows improvement in detecting most classes, including a lower 8 FN for cars. However, cyclists still pose a challenge with 55 FP and 36 FN, and humans face 22 FP and 63 FN, suggesting that while a smaller voxel size may enhance some aspects of detail, certain objects remain difficult to detect due to overlap or complexity.

Model Configuration 3 uses a larger voxel size of 32,000, which impacts detection. This model records 1 FP and 7 FN for cars, indicating reliable performance with minimal misclassifications. However, cyclists report 59 FP and 40 FN, and humans show 51 FP and 84 FN, reflecting significant detection difficulties, particularly in low visibility conditions. Wall detection reports 45 FP and 21 FN, indicating misclassifications due to complex structures. The tree class has 44 FP and 7 FN, suggesting difficulty in differentiating objects with similar structures.

Model Configuration 6, with a voxel size of 64,000, shows increased detection challenges. Cars show 16 FP and 31 FN, indicating errors in distinguishing them from other objects. Cyclists have 52 FP and 29 FN, while human detection reports 12 FP and 45 FN, pointing to issues in complex conditions. Wall detection has 40 FP and 19 FN, and the tree class shows 22 FP and 12 FN, indicating consistent difficulties in classifying varied object shapes. The cart class records 28 FP and 34 FN, highlighting detection challenges due to shape similarities.

Model Configuration 4, with a unique layer composition of 4;6;6 and a voxel size of 16,000, demonstrates enhanced detection capabilities. It records 1 FP and 2 FN for cars, suggesting nearly flawless detection. Cyclist detection reports 35 FP and 13 FN, showing more accurate identification. Human detection records 20 FP and 33 FN, indicating reduced error rates compared to other models. Wall detection, with 16 FP and 3 FN, shows high accuracy. The tree class records 9 FP and 14 FN, indicating low detection errors. However, the cart class, with 28 FP and 44 FN, still experiences significant challenges.

In summary, the comparison highlights that Model Configurations 1, 2, 3, and 6, despite sharing the same layer composition (3;5;5), exhibit varying detection accuracy influenced by voxel size. Smaller voxel sizes, such as 8,000 in Model Configuration 2, generally improve detail capture but do not fully eliminate detection errors, particularly in complex classes like cyclists and humans. The unique structure of Model Configuration 4 (4;6;6) provides better overall performance, suggesting that increased layer depth contributes to enhanced detection accuracy, particularly for cars and walls.

Among these five model configurations, the object classes that frequently demonstrate poor detection accuracy are cyclists, humans, and carts. The high number of false positives (FP) and false negatives (FN) in these classes, as seen in all configurations, suggests that these models struggle with distinguishing these objects from others and accurately detecting them when they are overlapping or positioned in complex scenarios. This can be attributed to the complexity of shapes and their resemblance to other objects, making feature extraction challenging for the models, especially with varying voxel sizes that may either lack or over-simplify detail.

On the other hand, the object classes that consistently show good detection accuracy across these configurations are cars and walls. The relatively lower FP and FN counts indicate that these objects are more straightforward for the models to recognize. The consistent shapes and distinctive features of cars and walls make them easier to classify accurately, even with different voxel sizes and layer compositions. The use of a more detailed layer structure in Model Configuration 4 (4;6;6) further supports enhanced recognition, particularly for classes with clearly defined geometries such as walls and cars, due to the deeper feature representation and improved segmentation capabilities.

Based on the configuration details for Model Configurations 5, 7, and 8, which use the BEVRes backbone with a layer composition of 2;3;3 but differ in voxel sizes, the prediction error analysis highlights their comparative performance.

Model Configuration 5 uses a voxel size of 16,000 and demonstrates generally reliable detection across most classes, with 3 FP and 35 FN for cars, suggesting the model effectively identifies cars but may miss some due to

occlusions or data limitations. Cyclists pose more challenges, with 42 FP and 44 FN, indicating significant issues in distinguishing cyclists from other objects and detecting them accurately. Humans record 39 FP and 28 FN, highlighting errors particularly in low-visibility conditions. The wall class has 19 FP and 1 FN, showing excellent detection accuracy. Trees report 5 FP and 16 FN, indicating detection difficulties under specific conditions. The cart class records 33 FP and 17 FN, reflecting struggles in correctly identifying carts due to their complex structures.

Model Configuration 7, with a smaller voxel size of 8,000, shows enhanced detection for certain classes. For cars, the model reports 3 FP and 2 FN, demonstrating high accuracy in car detection. Cyclists, however, still present a challenge with 19 FP and 36 FN, highlighting some undetected cases. The human class has 21 FP and 31 FN, indicating moderate detection issues. Walls record 24 FP and 17 FN, suggesting some walls are missed, possibly due to variations in shape and position. Trees show 14 FP and 9 FN, reflecting reliable performance but not without errors. The cart class performs well with 20 FP and 6 FN, indicating effective detection overall.

Model Configuration 8 uses a larger voxel size of 64,000, resulting in minimal errors for cars with 2 FP and 20 FN, demonstrating strong car detection but occasional misses. Cyclists have 36 FP and 34 FN, continuing to show detection challenges. The human class records 25 FP and 27 FN, indicating some detection errors, although they are somewhat reduced. Walls have 21 FP and 17 FN, reflecting consistent detection difficulties. Trees report 18 FP and 16 FN, suggesting issues under certain conditions. The cart class records 29 FP and 17 FN, indicating struggles with cart identification.

Across the three configurations, the cyclist, human, and cart classes consistently show poor detection accuracy. This is evident in the high number of false positives (FP) and false negatives (FN), suggesting that these classes have complex shapes or frequently overlap with other objects, leading to consistent detection challenges regardless of voxel size. Cyclists often have high variability in appearance and may be mistaken for other moving objects, while humans can be difficult to detect accurately due to varied poses and partial occlusions. Carts, due to their diverse forms and potential resemblance to other structures, also show high FP and FN counts, indicating challenges in feature extraction and classification.

Conversely, the car and wall classes generally exhibit reliable detection across all configurations, as seen in their consistently lower FP and FN counts. The clear and well-defined structures of cars make them easier to differentiate from the background and other objects, contributing to higher detection accuracy. Walls, with their typically consistent shapes and larger presence in the data, are also detected with high reliability. The performance in these classes is aided by distinct features that make classification straightforward.

The differences in detection accuracy can also be attributed to the voxel size used in each configuration. The smaller voxel size in Model Configuration 7 (8,000) helps capture finer details, resulting in enhanced performance for most object classes due to the higher resolution of data points. Conversely, larger voxel sizes, such as 64,000 in Model Configuration 8, may simplify the data representation too much, leading to a loss of detail and a negative impact on the detection of objects with intricate structures.

Across both BEV and BEVRes backbones, car and wall detection exhibit consistently high accuracy, as indicated by lower counts of false positives (FP) and false negatives (FN). The well-defined shapes and relatively large sizes of cars and walls contribute to their reliable classification. This trend is particularly noticeable when smaller voxel sizes, such as 8,000 in Model Configurations 2 and 7, are employed, as they provide the finer detail necessary for enhanced detection accuracy.

The detection of cyclists, humans, and carts consistently shows lower accuracy in both the BEV and BEVRes backbones, evidenced by higher FP and FN counts. Cyclists often pose a challenge due to their variable and complex shapes, which can be mistaken for other moving objects. Human detection is similarly affected by dynamic poses and partial occlusions, leading to inaccuracies. The cart class struggles with accurate identification, as their forms can closely resemble other objects. Larger voxel sizes, such as 32,000 and 64,000 used in Model Configurations 3, 6, and 8, tend to oversimplify the representation of data, worsening detection difficulties for these intricate classes.

The use of smaller voxel sizes, like 8,000 in Model Configurations 2 and 7, typically supports more detailed feature extraction. This approach tends to improve detection performance for simpler and more defined classes such as cars and walls. However, these smaller voxel sizes do not entirely resolve the detection challenges faced with more complex objects like cyclists, humans, and carts. In contrast, larger voxel sizes, such as 64,000 in Model Configurations 6 and 8, result in a reduction of detail, leading to higher detection errors in classes with less distinct or more intricate features.

The deeper layer composition of Model Configuration 4, which has a 4;6;6 structure, enhances detection performance across most object classes. This suggests that a more complex feature extraction pipeline facilitates better differentiation of complex shapes and features. Even when paired with a standard voxel size of 16,000, this deeper layer configuration supports improved recognition and classification accuracy.

Overall, both the BEV and BEVRes backbones face recurring challenges in accurately detecting cyclists, humans, and carts due to their intricate and variable structures. However, car and wall classes generally show high detection accuracy across various configurations, benefiting from their distinct shapes. The choice of voxel size and the refinement of layer composition play critical roles in optimizing overall detection performance. Smaller voxel sizes enhance detail, aiding in the classification of simpler objects, while deeper layer structures, as seen in Model Configuration 4, offer improvements across most classes by supporting more nuanced feature extraction.

V. Conclusion

By identifying the most effective architecture configuration of this method's backbone, we have successfully addressed the challenge of detecting and

classifying six classes of objects within a limited area. The findings from eight model configurations have provided valuable insights into the diverse characteristics of this method's architecture configurations.

Among the model configurations, 4 and 7 emerged as the top performers, boasting mAP scores above 0.8 and demonstrating superior overall performance. Configuration 2 was a close competitor with a respectable mAP score of 0.78, while Configurations 5 and 8 held their ground with solid scores of 0.76. Configuration 6 showed a balanced performance with a mAP score of 0.74, and Configuration 3 had a moderate score of 0.72. Unfortunately, the default Model Configuration 1 fell behind with the lowest mAP score of 0.64, indicating the need for improvement.

Overall, the PointPillars model with a 2D BaseBEVRESBackbone generally outperformed the 2D BaseBEVBackbone at equivalent voxel values (8,000, 16,000, 64,000). Nevertheless, a model configuration using the 2D BaseBEVBackbone could slightly surpass the BaseBEVRESBackbone's performance if the layer composition had been increased, as evidenced by Model Configuration 4 (4;6;6). This condition suggests that while both backbone types have their strengths, the BaseBEVBackbone benefits more significantly from additional layers. In contrast, the BaseBEVRESBackbone performs optimally with smaller voxel sizes and fewer layers.

ACKNOWLEDGMENT

Special acknowledgement is given to the students from Universitas Pendidikan Indonesia, Universitas Nurtanio, Universitas Majalengka, and Universitas Garut, who have actively contributed to supporting this research.

REFERENCES

[1] J.W. Pyo, S.H. Bae, S.H. Joo, M.K. Lee, A. Ghosh, and T.Y. Kuc, “Development of an Autonomous Driving Vehicle for Garbage Collection in Residential Areas,” Sensors, vol. 22, no. 23, p. 9094, 2022.

[2] P. de S. Carneiro, A. R. Pereira, and M. Mira da Silva, “A Model for the Deployment of Shared Autonomous Vehicles in Urban Areas Based on the Research Literature,” International Journal of Transport Development and Integration, vol. 7, no. 3, pp. 199–213, 2023.

[3] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, “The ApolloScape Open Dataset for Autonomous Driving and Its Application,” IEEE Trans Pattern Anal Mach Intell, vol. 42, no. 10, pp. 2702–2719, 2020.

[4] P. Sun et al., “Scalability in Perception for Autonomous Driving: Waymo Open Dataset,” in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 2443–2451. 2020.

[5] A. Saha and B. C. Dhara, “3D LiDAR-based obstacle detection and tracking for autonomous navigation in dynamic environments,” Int J Intell Robot Appl, vol. 8, no. 1, pp. 39–60, 2024.

[6] V. Lehtola et al., “Comparison of the Selected State-Of-The-Art 3D Indoor Scanning and Point Cloud Generation Methods,” Remote Sens (Basel), vol. 9, no. 8, p. 796, 2017.

[7] S. Thrun et al., “Stanley: The robot that won the DARPA Grand Challenge,” J Field Robot, vol. 23, no. 9, pp. 661–692, 2006.

[8] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “PointPillars: Fast Encoders for Object Detection from Point Clouds,” in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 12697–12705, 2019.

[9] R. Girshick, “Fast R-CNN,” in 2015 IEEE International Conference on Computer Vision (ICCV), IEEE, pp. 1440–1448, 2015.

[10] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” IEEE Trans Pattern Anal Mach Intell, vol. 39, no. 6, pp. 1137–1149, 2017.

[11] Y. Zhou and O. Tuzel, “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection,” in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, pp. 4490–4499, 2018.

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 770–778, 2016.

[13] Z. Yang, Y. Sun, S. Liu, and J. Jia, “3DSSD: Point-Based 3D Single Stage Object Detector,” in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 11037–11045, 2020.

[14] W. Zheng, W. Tang, L. Jiang, and C.-W. Fu, “SE-SSD: Self-Ensembling Single-Stage Object Detector from Point Cloud,” in 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 14489–14498, 2021.

[15] B. Sun, W. Li, H. Liu, J. Yan, S. Gao, and P. Feng, "Obstacle Detection of Intelligent Vehicle Based on Fusion of Lidar and Machine Vision," Engineering Letters, vol. 29, no.2, pp722-730, 2021.

[16] X. Bai et al., “TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection with Transformers,” in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 1080–1089, 2022.

[17] X. Xu et al., “FusionRCNN: LiDAR-Camera Fusion for Two-Stage 3D Object Detection,” Remote Sens (Basel), vol. 15, no. 7, p. 1839, 2023.

[18] Y. Yan, Y. Mao, and B. Li, “SECOND: Sparsely Embedded Convolutional Detection,” Sensors, vol. 18, no. 10, p. 3337, 2018.

[19] M. Hollaus, W. Wagner, B. Maier, and K. Schadauer, “Airborne Laser Scanning of Forest Stem Volume in a Mountainous Environment,” Sensors, vol. 7, no. 8, pp. 1559–1577, 2007.

[20] B. Lohani and S. Ghosh, “Airborne LiDAR Technology: A Review of Data Collection and Processing Systems,” Proceedings of the National Academy of Sciences, India Section A: Physical Sciences, vol. 87, no. 4, pp. 567–579, 2017.

[21] Z. Wang and M. Menenti, “Challenges and Opportunities in Lidar Remote Sensing,” Frontiers in Remote Sensing, vol. 2, 2021.

[22] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, “A survey on deep learning techniques for image and video semantic segmentation,” Appl Soft Comput, vol. 70, pp. 41–65, 2018.

[23] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 3354–3361, 2012.

[24] N. Moustafa and J. Slay, “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in 2015 Military Communications and Information Systems Conference (MilCIS), IEEE, pp. 1–6, 2015.

[25] K. M. Ting, “Confusion Matrix,” in Encyclopedia of Machine Learning and Data Mining, Boston, MA: Springer US, pp. 260–260.

[26] T. Fawcett, “An introduction to ROC analysis,” Pattern Recognit Lett, vol. 27, no. 8, pp. 861–874, 2006.

[27] T.-Y. Lin et al., “Microsoft COCO: Common Objects in Context,” in ECCV 2014, pp. 740–755, 2014.

Alif Ilman Nafian received a Bachelor's degree from the Software Engineering Department at Universitas Pendidikan Indonesia in 2024. For the past two years (2022-2024), he has started an apprenticeship in BRIN's research and development of object detection for autonomous vehicles. His current research interests are machine learning, artificial intelligence, and LiDAR-based object detection.

Dian Anggraini is a lecturer in the Software Engineering Department at Universitas Pendidikan Indonesia. The author holds a Masters in Engineering (M.T) from the Department of Electrical Engineering at the Bandung Institute of Technology in Indonesia in 2018. Her areas of interest are artificial intelligence, database and software engineering.

Mochamad Iqbal Ardimansyah received his Master’s degree from Telkom University in Indonesia in 2017. He works as a lecturer in the Software Engineering Department at Universitas Pendidikan Indonesia. His research interests are machine learning, data mining, internet of things, and related about parallel and distributed computing.

Arief Suryadi Satyawan received his master's degree in Electrical Engineering from the Bandung Institute of Technology in Indonesia in 2007. In 2019, he received his Computer and Communication Engineering Doctorate from Waseda University, Japan. Since 1997, he has worked for the Indonesian Institute of Sciences, which was later merged into the National Research and Innovation Agency, Indonesia, in 2021. Since 2020, he has been researching the development of object detection and telecommunication for the application on autonomous vehicles operated in limited areas, which the Indonesia Endowment Fund for Education (Lembaga Pengelola Dana Pendidikan) and the National Research and Innovation Agency (BRIN), Indonesia, support. His research interests are applications of machine learning, artificial intelligence, LiDAR-based object detection, and visual communication based on machine learning.