Journal of Innovations in Computer Science and Engineering
Shahid Beheshti Unversity

May 2024, Volume 2, Issue 1

Adaptive Weighted Knowledge Distillation for 3D Object Detection in Self-Driving Cars Using Point Cloud

Samira Tajik ORCID: 0009-0004-9390-3491,
Armin Salimi-Badr✉, **ORCID: 0000-0001-6613-7921,**
Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran, sam.tajik@mail.sbu.ac.ir.
Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran, a_salimibadr@sbu.ac.ir.

ABSTRACT

3D object detection for self-driving cars has emerged as a critical challenge, largely due to the constraints of computing resources and the requirement for real-time processing. For evaluation, we utilize the KITTI benchmark dataset, which is widely used for self-driving cars and 3D object detection research. In this paper, an adaptive Weighted knowledge distillation approach is proposed to enhance detection accuracy while improving computational efficiency. Based on the performance of the teacher model, point clouds are divided into TPS (Teacher Performs Strongly) and TPW (Teacher Performs Weakly). The student model adapts its learning strategy dynamically: for TPS point clouds, it closely imitates the teacher by increasing the distillation weight, whereas for TPW point clouds, it prioritizes learning from raw data, reducing reliance on the teacher's guidance. Additionally, a data pruning mechanism creates a smaller dataset from the KITTI benchmark based on the teacher model's performance, while maintaining the TPS-TPW ratio. Experimental results indicate that the student model achieves comparable, and in some cases superior, performance to the teacher model. Specifically, it enhances recall by up to 1.5% and precision by up to 2.2% in complex scenarios. The student model is a lightweight network that learns from the teacher through knowledge distillation [28] and is solely used during execution. This design reduces execution time by nearly 50% while maintaining high detection accuracy .These findings emphasize the effectiveness of the proposed framework in real-time 3D object detection, making it well-suited for deployment in resource-constrained environments such as self-driving cars.

KEYWORDS

3D Object Detection, Self-driving Cars, Point Cloud, Knowledge Distillation, Computational Efficiency

1. INTRODUCTION

Comprehending the 3D environment is crucial for robotic perception. A point cloud, which is created by 3D sensors like LiDAR, is a common way to capture the shape of the environment. A point cloud provides accurate spatial information, making it invaluable for tasks including navigation, mapping, and autonomous systems. Convolutional neural networks (CNNs) are efficient for processing grid-like data, such as images. However, they require data to be arranged on a regular grid, which can lead to potential information loss or wasted computations in irregular structures, such as point clouds. In contrast, graph neural networks (GNNs) treat individual points as graph nodes and connect neighboring points to capture spatial relationships, effectively handling the sparse structure of point clouds. This allows GNNs to extract features efficiently and improve 3D object detection accuracy, particularly in applications like autonomous driving, where real-time performance is essential.

Recent developments in 3D object detection have addressed the specific challenges associated with point cloud data. Early methods like PointNet [4] and PointNet++ [5], introduced efficient frameworks to process raw point cloud data and learn both local and global features. Later advancements, including PointRCNN [6], VoteNet [7], and 3DSSD [8], detection by focusing on region-based refinement, voting mechanisms, and eliminating voxelization to optimize performance. Despite these improvements, challenges such as occlusion and sparse data remain significant. Techniques like AGO-Net [11]and ODSPC [12]combine real-world and

Submit Date: 2024-12-04
Revise Date: 2025-01-30
Accept Date: 2025-02-23
✉ Corresponding author

augmented features or integrate semantic information to enhance detection under occlusion. Additionally, graph neural networks (GNNs), such as Point-GNN [17] and dynamic graph CNNs [20], have shown promise in processing the irregular structure of point clouds. Knowledge distillation has further advanced field by transferring knowledge from large teacher models to lightweight student models, as demonstrated in methods like itKD [23] and CaKDP [25].

Misclassification or failure to detect objects in self-driving cars can have serious consequences, highlighting the importance of efficient and accurate 3D object detection. A major challenge lies in managing the large volumes of sensor data while maintaining high accuracy, particularly in dynamic and diverse environments. Autonomous vehicles must process this data quickly to ensure safe and reliable operation. Without optimized computational and memory resource management, these systems risk delays and reduced detection performance, posing risks to real-time functionality. To address the challenges of high computational costs substantial memory consumption in 3D object detection, knowledge distillation [28] as an effective solution. In knowledge distillation [28], a large teacher model is used to extract essential features and make accurate predictions. By transferring this knowledge to a lightweight student model, the student is able to achieve comparable performance while requiring fewer computational resources. This makes the student model highly suitable for real-time applications and deployment in environments with limited computational capacity.

In this paper, we propose an adaptive weighted knowledge distillation approach combined with a data pruning mechanism to address tackle the challenges of high computational demands and enhance execution time. Our method evaluates the teacher model's performance on each point cloud, divided data into TPS (Teacher Performs Strongly) and TPW (Teacher Performs Weakly). For TPS point clouds, the student model relies heavily on the teacher's knowledge to quickly learn key patterns and information, while for TPW point clouds, the student focuses on learning directly from the ground truth and its own loss function, fostering independent refinement of predictions. The teacher's guidance is still partially retained for TPW cases, ensuring the student can still benefit from partial guidance. This combination of teacher-driven learning for TPS and student-driven learning for TPW enables the model to develop a more comprehensive understanding. As a result, it achieves robust and generalized performance. It is worth mentioning that our approach involves a teacher model and a student model during training, where the student adaptively learning through knowledge distillation. The teacher provides guidance to the student during the offline training phase, but only the lightweight student model is used during execution. This design minimizes computational costs while maintaining performance close to the teacher, making it an efficient and practical solution for deployment. This paper introduces innovative adaptive weighted knowledge distillation and data pruning mechanisms, along with a lightweight student model, designed to enhance computational efficiency and improve real-time 3D object detection performance.

The rest of this paper is organized as follows: In Section 2, we review related work, focusing on recent progress in 3D object detection, graph neural networks, and knowledge distillation approaches. Section 3 introduces our proposed adaptive weighted knowledge distillation approach, including its integration with a data pruning mechanism and an adaptive training strategy. In Section 4, we describe the experimental setup, including datasets, evaluation metrics, and implementation details. Section 5 discusses the results and analysis, comparing the performance of our approach with an existing method. Finally, Section 6 concludes the paper by summarizing the main contributions and suggesting potential future research directions.

2. RELATED WORK

Image-based 2D object recognition methods have achieved remarkable success. These methods provide valuable insights. For example, MFIL-FCOS [1] introduces a multi-scale interactive and hybrid learning framework that effectively combines features across scales and not only enhances 2D recognition but also makes it robust to object size variations. Comparative studies such as SIFT, SURF, and ORB descriptors [2] provide a fundamental understanding of feature extraction methods, highlighting the trade-offs in accuracy and computational efficiency for 2D recognition tasks. TricubeNet [3] proposes a kernel-based representation for weakly occluded object recognition, which provides a novel approach to partial occlusion. Despite these advances in 2D object recognition, 3D object recognition based on LiDAR point clouds faces several challenges that various approaches have attempted to address. In this section, several different approaches to 3D object recognition are reviewed.

PointNet-based methods have made great progress in 3D object recognition in recent years. These methods directly process raw point cloud data and provide an efficient alternative to more complex processes such as voxelization. The original PointNet model[4] improved the classification and segmentation of 3D data by processing each point independently and combining features through aggregation layers. This basic model has paved the way for the development of numerous models. For example, PointNet++ [5] introduced multi-level feature learning, which is able to detect local and global structures in the point cloud, increasing its power in processing complex geometries. The PointRCNN model [6] extracts high-quality 3D propositions from raw point cloud data using PointNet layers and then improves them with a region-based process. VoteNet [7]uses a voting mechanism to accurately locate object centers and generate bounding boxes. Furthermore, 3DSSD [8] eliminates the voxelization process, combines the PointNet++ model into a single-step recognition framework, and offers higher accuracy and efficiency in 3D object recognition. Finally, STD [9] moves from sparse to dense representations and extracts better and more accurate features for recognition by using spherical anchors and voxelization.

Occlusion in point clouds is one of the main challenges in 3D object recognition, which occurs when objects are partially or completely hidden. This problem arises due to the lack of complete features of hidden objects and the increased complexity in

recognizing different objects. In this situation, it becomes more difficult to distinguish between different objects and the recognition accuracy decreases. LiDAR-based automated driving systems face more difficulties in low-density points in distant objects, especially on highways. To address this issue, Wang et al. [10] introduced a domain matching method that aligns the features of distant objects with near objects and improves recognition, although it is involved in various forms. AGO-Net [11] enhances detection under occlusion by combining incomplete perceptual features from real-world data with complete conceptual features from augmented 3D models. It uses a perceptual encoder, a conceptual generator, a feature association module, and a reweighting module to improve classification and localization accuracy. Other methods offer different solutions to this problem. ODSPC [12] combines semantic and point cloud information to improve recognition in obstructed scenarios. The monocular detection approach [13] increases visibility and accuracy using a single camera. DR.CPO [14] trains models with realistic obstruction patterns to make them more robust. MOSE [15] uses scene cues to improve detection of roadside objects even when objects are partially hidden. X-ray distillation [16] transfers knowledge to enhance detection under obstruction. These methods provide significant advances for improving 3D object recognition in challenging conditions such as obstruction, especially for autonomous driving.

Graph neural networks are able to improve the recognition accuracy of 3D objects by handling the irregular and disordered structure of point clouds. Unlike convolutional neural networks that face challenges with irregular data, graph neural networks represent 3D points as graph nodes and model their relationships through edges, enabling them to adapt to 3D space and improve recognition accuracy. The Point-GNN method [17] uses a single-stage graph-based model to recognize objects in point clouds, using fixed-radius connections to capture local features, and using GNN to process these features for classification and bounding box generation. Another approach [18] combines GNN feature learning with angular features, leading to improved object recognition accuracy in point cloud data. A semi-supervised approach [19] uses temporal graph neural networks to recognize 3D objects, reducing the need for labeled data and improving recognition. Dynamic graph CNN [20] introduces a novel approach to dynamically build and update graph structures based on local geometric relationships in point clouds, which improves feature learning and 3D object recognition accuracy.

In recent years, knowledge distillation has been widely used in 3D object recognition in autonomous vehicles. In knowledge distillation, a teacher model, usually a large and complex network, provides predictions or representations that guide the student, usually a small network, during training. Thus, a lightweight student model can perform similarly or closely to the teacher by imitating a large teacher model. The framework proposed in [21] consists of two teacher networks, the first teacher network is trained in a traditional way and provides guidance for intermediate representations and uses adversarial learning for alignment. The second teacher network guides the output probability distribution to the student network, but may be inefficient in providing accurate intermediate representations. To better transfer the intermediate knowledge, a discriminator is used, whose task is to predict the source of the intermediate representations as well as the class labels. Expanding on this theme, [22] presents knowledge distillation with a focus on useful information compression and noise removal. The teacher model transfers its knowledge to the student model to extract useful information and increase learning accuracy. In this method, suggested regions are selected from images or 3D point clouds that represent the target objects or background. For each suggested region in the teacher model, a similar region is identified in the student model and weighted with a distillation mask. This mask highlights more important regions for better learning, and the ultimate goal is to increase the mutual information between the features and the real labels to train the student model more accurately.

In recent works, new knowledge distillation techniques have been applied to 3D object recognition. The itKD framework [23] uses a trade-off-based approach that uses compression and channel reconstruction to efficiently transfer knowledge for 3D object recognition. Pre-pruned distillation[24] combines network pruning with knowledge distillation to increase the efficiency of knowledge transfer in 3D object recognition. The CaKDP framework [25] incorporates cluster-aware knowledge distillation and pruning strategies and enables lightweight 3D recognition models while maintaining high accuracy. Also, Efficient 3D object recognition with knowledge distillation [26] focuses on using distillation techniques to develop efficient voxel- and column-based detectors. Finally, Structured Knowledge Distillation for Multi-View Recognition [27] provides a structured approach to distillation, optimizing multi-view 3D recognition and significantly improving model performance in complex scenarios.

Although previous methods have achieved significant progress in addressing challenges as occlusion and computational efficiency, they often fail to balance real-time execution and high accuracy in resource-constrained environments. To address these issues, this paper introduces the following contributions:

- The proposed method divided point cloud data into two groups, TPS (Teacher Performs Strongly) and TPW (Teacher Performs Weakly), according to the performance of the teacher model. The student model applies adaptive weighted knowledge distillation by dynamically modifying its learning depending on whether the point cloud belongs to TPS or TPW, optimizing its ability to generalize. In TPS cases, the student model relies more on the teacher's knowledge, assigning a

higher weight to teacher's knowledge, whereas in TPW cases, the student prioritizes its own learning by reducing dependence on the teacher.

- A data pruning mechanism is implemented. This process creates a smaller and optimized version of the KITTI dataset based on the performance of the teacher model, while maintaining the proportion of TPS and TPW point cloud. This optimization enhances computational efficiency while maintaining detection performance.
- A lightweight student model is designed to be more computationally efficient and resource-conscious than the teacher model. This design enables the model to perform well in real-time applications within resource-limited environments while maintaining high detection accuracy.

3. PROPOSED METHOD

In this section, we describe the proposed approach for 3D object detection from point clouds. As shown in Figure 1, the framework consists of three main steps: graph creation, Teacher-Student Adaptive Weighted Knowledge Distillation Module, and bounding box optimization. Unlike structured image data, point clouds are sparse and unevenly distributed, making convolutional neural networks (CNNs) unsuitable for direct processing. Some studies have employed neural networks that process unordered point sets to extract features without mapping them to a structured grid. However, these methods often require repeated sampling and grouping, which can be computationally expensive for large-scale point clouds. In our method, we employ a Graph Neural Network (GNN) that naturally adapts to the irregular and sparse structure of point clouds. In the graph creation phase, points are encoded as vertices of the graph, and edges of the graph connect neighboring points that are within a fixed radius to allow information to flow between neighbors. The graph neural network used takes a point as input and outputs the classification and bounding boxes of the objects to which each vertex belongs.

In the Teacher-Student Adaptive Weighted Knowledge Distillation phase, we introduce an adaptive teacher-student learning strategy. The goal is to transfer the knowledge embedded within a large, complex teacher model to a smaller, lightweight student model. This is critical for achieving high accuracy while maintaining the computational efficiency required for real-time applications, such as self- driving cars. Offline distillation [28] is employed to achieve this transfer, as it does not require simultaneous training of the teacher and student models, significantly reducing resource demands. During the training phase, the proposed method involves two networks: the teacher and the student. Knowledge is distilled from the teacher to the student, allowing the student to learn efficiently while minimizing dependency on the larger teacher model. The teacher model evaluates each point cloud and groups it as TPS (Teacher Performs Strongly) or TPW (Teacher Performs Weakly) based on its recall, which reflects the teacher model's confidence in its predictions. This grouping dynamically adjusts the weight of knowledge transfer, allowing the student model to either rely more on the teacher's predictions for TPS or prioritize independent learning from raw data for TPW.

The bounding box refinement process improves object localization by refining the student model's predictions. Since the neural network may generate multiple bounding boxes for the same object, Non-Maximum Suppression (NMS) is commonly applied to merge overlapping boxes and assign confidence scores. NMS selects the bounding box with the highest classification score while suppressing the others. However, relying solely on classification scores can lead to inaccuracies, especially for partially occluded objects. To enhance localization accuracy, a refinement step is applied to merge bounding boxes effectively, ensuring that detected objects are well-defined and robust to variations in the input data. The bounding box formation process and NMS algorithm are detailed in Appendices A and B. As shown in Figure 2, during the execution phase, only the lightweight student model is utilized for inference. The student independently processes point cloud data, generating object proposals and refining bounding boxes to enhance detection accuracy. This design ensures computational and memory efficiency, making the student model well-suited for real-world deployment. Its efficiency is particularly advantageous in resource-limited environments, such as self-driving cars, where optimized resource usage and real-time performance are essential for safe and reliable operation.

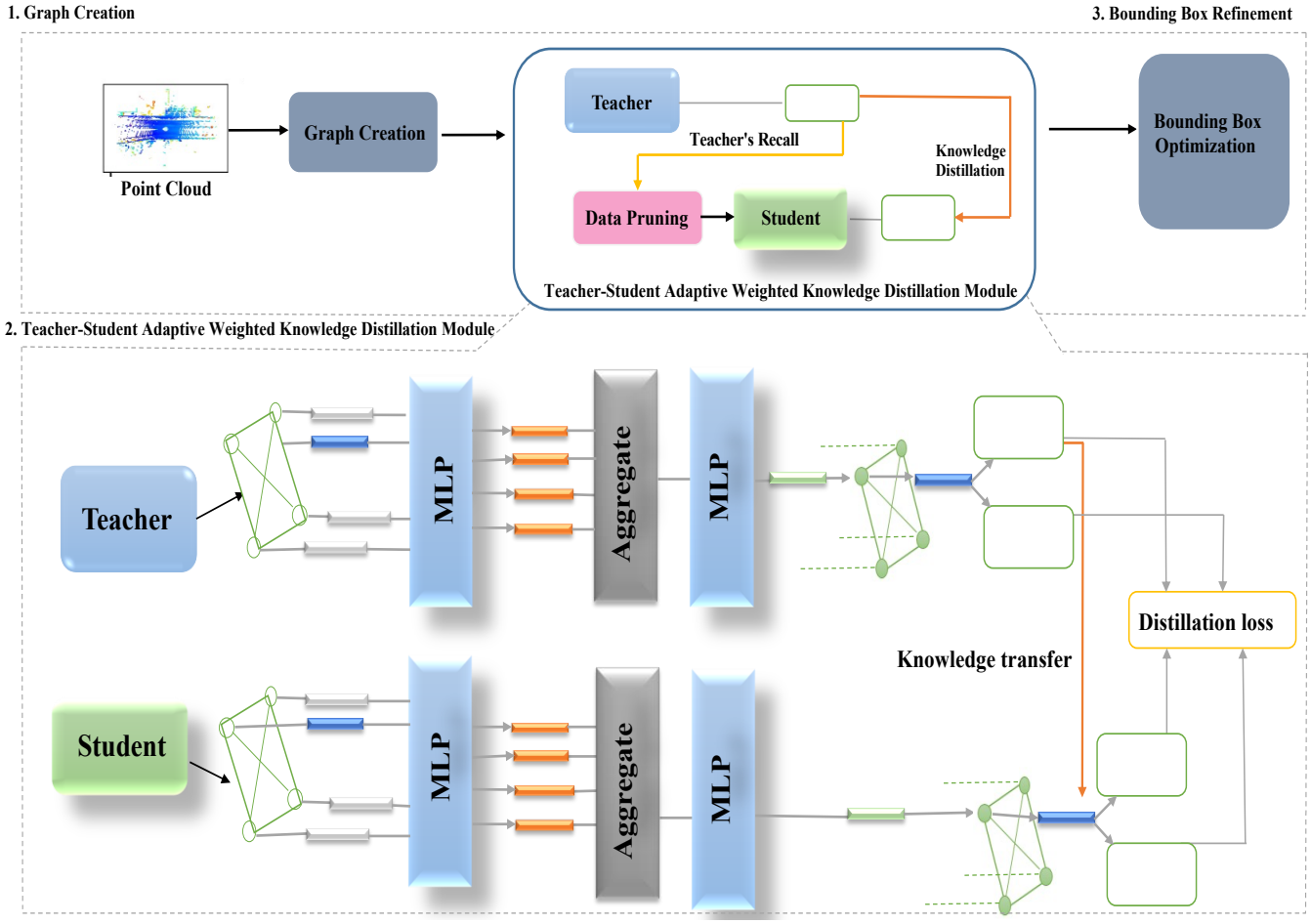

Figure 1: Overview of the training phase: (1) Graph Creation, where point clouds are processed to construct graph representations; (2) Teacher-Student Adaptive weighted Knowledge Distillation Module, where the teacher model transfers knowledge to the student model through adaptive distillation; and (3) Bounding Box Optimization, where bounding boxes are merged and refined for accurate object detection.

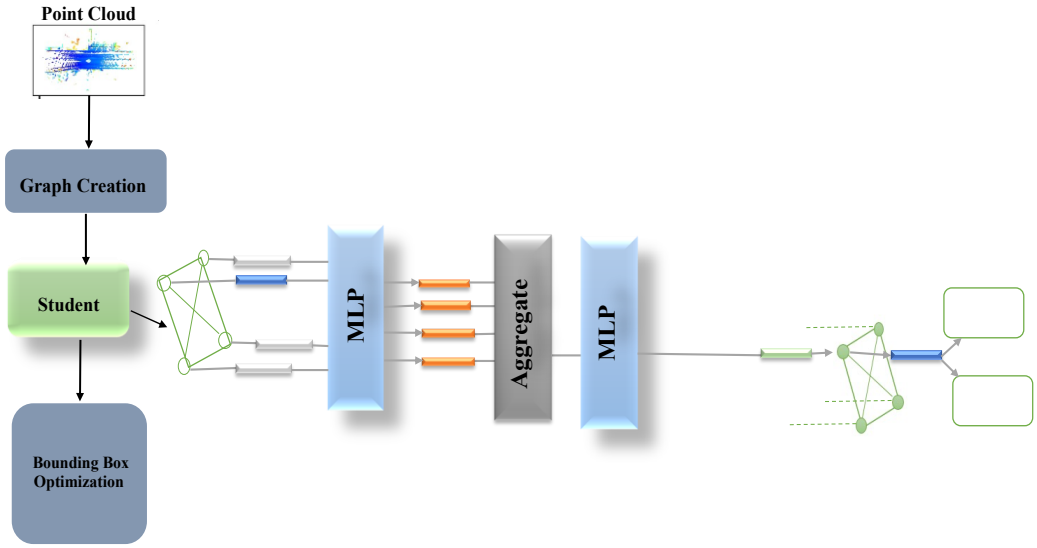

Figure 2: Execution Phase: The student model independently processes point clouds

3.1 Building a new dataset for the student model

To address the challenge of large computational requirements, we first created a small dataset from KITTI train based on the performance of the teacher model. In applications such as 3D object detection in self-driving cars, safety is of great importance, so high recall may be prioritized to ensure that all potential hazards are detected, even if it means having more false positives.

3.1.1 Examining the Performance of the Teacher Model

In 3D object detection, evaluation metrics such as Recall is used to evaluate the performance of the model. Recall measures the proportion of true positives that the model correctly identifies. A high value of Recall indicates that the model is relevant in recognizing all objects but may also include more false positives. The formula for Recall is given in Equation (1):

$$Recall = \frac{True\ positive}{True\ positive + False\ Negative} = \frac{TP}{TP+FN} \quad (1)$$

To evaluate the performance of the teacher model on the point cloud, the Recall metric is computed for each point cloud across all four classes. Vehicles are divided into two categories based on viewing angle: vehicle side view and vehicle front view, this division is done based on the yaw angle (the angle of rotation of the vehicle in the horizontal plane). Yaw is the angle that indicates the orientation of the vehicle relative to the camera or sensor. Side view vehicle: Vehicles whose yaw angle is in the range $[-\pi/4, \pi/4]$ are considered as side view vehicles. That is, if the vehicle is viewed from an angle between -45 degrees and +45 degrees relative to the sensor, it is classified as side view. Front view vehicle: Vehicles whose yaw angle is in the range $[\pi/4, 3\pi/4]$ are considered as front view vehicles. That is, if the vehicle is seen from an angle between +45 degrees and +135 degrees to the sensor, it is classified as front view. Each point is assigned a class based on its spatial location: if it falls inside an object's bounding box, it is classified as the corresponding object class; if it lies outside any bounding box, it is assigned as background. Additionally, points that do not belong to any bounding box and do not contribute to object detection are labeled as 'Do not care' and excluded from loss computation. The recall thresholds of 0.68 and 0.98 were determined based on an empirical analysis of the teacher model's performance on all point clouds. Recall values below 0.68 reflected weak performance due to a high rate of false negatives, making it the lower limit for TPW point clouds. The upper threshold of 0.98 was selected because all "Don't care" point clouds consistently exceeded this value, with no other classes reaching it. These thresholds were validated through iterative testing to ensure effective categorization and alignment with the objectives of the adaptive weighted knowledge distillation method. The grouping of point clouds based on these recall thresholds is formulated as follows:

$$\text{Point Cloud Grouping} = \begin{cases} \text{TPS (Teacher Performs Strongly)} \ \ if\ 0.68 \leq Recall < 0.98 \\ \text{TPW (Teacher Performs Weakly)} \ \ if Recall < 0.68 \end{cases} \quad (2)$$

After analyzing the teacher model's Recall performance for the three classes, the number of classes in which the teacher performed well (i.e., Recall within the range $0.68 \leq \text{Recall} < 0.98$**)** is counted. Based on this count, the point clouds are divided into two groups:

1. TPS (Teacher Performs Strongly):
   If the teacher model performs well in two or all three classes (i.e., Recall for two or three classes falls in the range $0.68 \leq$ Recall < 0.98), the corresponding point cloud is considered as TPS. This indicates that the teacher model has provided reliable predictions for the majority of classes in this point cloud.
2. TPW (Teacher Performs Weakly):
   If the teacher model does not perform well in two or all three classes (i.e., Recall for two or three classes falls below 0.68), the corresponding point cloud is considered as TPW. This indicates that the teacher model struggled to generate accurate predictions for most classes in this point cloud.

This framework allows for adaptive evaluation of the teacher's performance across multiple classes, enabling the student model to apply adaptive weighted learning strategies tailored to TPS and TPW.

Algorithm 1: Organizing Point Cloud Based on Teacher's Recall

Input: Point cloud data;
Teacher model's recall;
Recall thresholds: $T_{low} = 0.68$, $T_{high} = 0.98$;

Output: Organized point cloud into two groups:;
- TPS (Teacher Performs Strongly): Point clouds where the teacher achieves high recall, indicating strong performance.;
- TPW (Teacher Performs Weakly): Point clouds where the teacher achieves lower recall, indicating weak performance.;

foreach *point cloud* **do**
  Initialize $class\_performance\_TPS = 0$;
  Initialize $class\_performance\_TPW = 0$;
  foreach *class (except 'Don't care')* **do**
    Compute $Recall = \frac{TP}{TP+FN}$;
    if $T_{low} \leq Recall < T_{high}$ **then**
      $class\_performance\_TPS$ += 1;
    else if $Recall < T_{low}$ **then**
      $class\_performance\_TPW$ += 1;
  if $class\_performance\_TPS \geq 2$ **then**
    Assign point cloud to **TPS** (high-confidence samples);
  else if $class\_performance\_TPW \geq 2$ **then**
    Assign point cloud to **TPW** (low-confidence samples);
return *TPS, TPW*

3.1.2 Knowledge Distillation Based on Teacher Model Performance

In this paper, we propose a new approach to knowledge distillation that dynamically adapts the learning process based on the teacher model's performance. By dividing data into two groups TPS and TPW the student model can effectively balance learning from the teacher with refining its own predictions, resulting in greater efficiency and accuracy in 3D object detection. The categorization into TPS and TPW is determined by the Recall of the teacher model. The weights of the loss components are then adjusted based on whether the data is divided as TPS or TPW.

For TPS point clouds, where the teacher provides accurate and reliable predictions, we place greater emphasis on the distillation loss. By assigning higher weight to the distillation loss and reducing the weight of the student's initial loss, the student model can better mimic the teacher's knowledge and quickly learn key patterns. In contrast, for TPW point clouds, where the teacher struggles to make accurate predictions, the student model relies more on its own learning. In such cases, the weight of the student's initial loss is increased, and the reliance on distillation loss is minimized. By focusing more on the ground truth data, the student model independently refines its predictions while still partially utilizing the teacher's knowledge. This adaptive strategy ensures that the student focuses on imitating the teacher for TPS point clouds, where the teacher's performance is strong, and prioritizes independent learning for TPW point clouds, where the teacher's guidance is less reliable. This approach improves the student's overall learning process based on the teacher model's performance. This process is mathematically expressed in Equation 3. In this equation, l_{total_Std} represents the total loss for the student model, combining two weighted components: l_{Std}, which is the loss from the student model's independent learning based on ground truth data, and $l_{totall\_Distill}$, which represents the knowledge distillation loss. A more detailed explanation about Equation 3 and assigning w_1 and w_2 can be found in Section 3.2.

$$l_{total\_Std} = w_1(\, l_{Std}) + \, w_2\big(\, l_{totall\_Distill}\,\big) \tag{3}$$

3.1.3 Soft Targets

Soft targets use logits, which represent the inputs of the last softmax, to learn the student's model, and the contribution of each soft goal to knowledge distillation is modulated using the temperature parameter. Knowledge distillation from the teacher model to the student model is performed using a distillation cost function, the distillation function uses soft objectives to minimize the squared difference between the logits produced by the teacher model and the logits produced by the student model[28].

Response-based knowledge distillation uses the response of the last layer of the teacher model, and its main idea is to directly imitate the final prediction of the teacher model. Response-based knowledge distillation is simple yet effective for model compression, and therefore has been widely used in various tasks and applications. If we consider the output of the last fully connected layer of a deep model as the z-logit, the loss function in response-based knowledge distillation can be formulated as follows[28].

$$L_{ResD}(z_t\,, z_s\,) = \mathcal{L}_R\,(z_t\,, z_s\,) \tag{4}$$

In Equation 4, $\mathcal{L}_R(.)$ is the logit divergence loss, and z_t and z_s are the teacher and student logits, respectively. The most popular response-based knowledge for image classification is known as soft targets. Soft targets are probabilities that the input belongs to the classes and can be estimated by a softmax function.

$$p(z_i, \mathrm{T}) = \frac{\exp(z_i/\mathrm{T})}{\sum_j \exp(z_i/\mathrm{T})} \quad (5)$$

In Equation 5, z_i is the logit for class i, and a temperature coefficient T is introduced to control the importance of each soft target. The soft targets contain dark knowledge information from the teacher model. The distillation loss for soft logits can be rewritten as follows:

$$L_{ResD}(p(z_t, \mathrm{T}), p(z_s, \mathrm{T})) = \mathcal{L}_R(p(z_t, \mathrm{T}), p(z_s, \mathrm{T})) \quad (6)$$

3.2 Student's Loss Function

To classify an object in the student model, we compute a multiclass probability distribution $\{p_{c1}, \ldots, p_{cM}\}$ for each vertex. If a vertex is in an object bounding box, we assign it the object class, and if a vertex is outside the bounding box, we assign it the background class. For the classification loss, we use the average cross-entropy loss as shown is Eq. (7):

$$l_{clsStd} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{c_j}^i \log(p_{c_j}^i) \quad (7)$$

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

In Eq. (7), p_c^i represents the predicted probability of vertex i belonging to class c, where $c \in M\{1,2,\ldots,M\}$ and M is the total number of classes. Similarly, y_c^i represents the one hot class label for the $i-th$ vertex.

If a vertex is in a bounding box, we calculate the Huber loss between the ground truth and our prediction, and if a vertex is outside the bounding boxes or belongs to a class that we do not need to locate, we set its location loss to zero. To prevent overfitting, we add L1 regularization to each MLP. Kullback-Leibler (KL) divergence is used to calculate the divergence between the smoothed probabilities of the teacher and the student ($l_{distill}$), which measures the extent to which the student's predictions differ from the teacher's predictions. Huber loss is used to minimize the difference between the predictions of the student and teacher's bounding boxes ($l_{locDistl}$). The final loss for the student is represented as l_{total_Std}, as shown in Equation (9):

$$l_{total\_Std} = w_1(l_{clsStd} + l_{locStd} + l_{regStd}) + w_2(l_{distill} + l_{locDistl}) \quad (9)$$

To enhance the adaptability of the student model, we dynamically adjust the weight coefficients w_1 and w_2 based on the TPS and TPW for each point cloud. If a point cloud is considered as TPS (Teacher Performs Strongly), w_2 is assigned a higher value than w_1 , prioritizing knowledge distillation from the teacher model while still allowing the student to contribute with a lower weight. This approach ensures that the student benefits from the teacher's high-confidence predictions while simultaneously refining its own learning process. By doing so, the student gains structured guidance from the teacher but also develops flexibility, which helps it adapt to new, unseen scenarios where the teacher's rigid patterns may not fully generalize. On the other hand, if a point cloud is considered as TPW (Teacher Performs Weakly), w_1 is assigned greater importance than w_2 , emphasizing direct learning from raw data. However, the teacher's knowledge is not entirely ignored; instead, it is used with a lower weight to provide subtle guidance rather than direct supervision. This ensures that the student does not completely rely on potentially unreliable teacher predictions but still benefits from any valuable insights. This balanced strategy strengthens the student's generalization ability, making it more robust in handling complex and ambiguous cases where the teacher's performance is less reliable. By striking this balance, the student develops stronger generalization abilities and robustness, making it capable of handling complex and ambiguous cases where the teacher's performance is less reliable.

3.3 Preprocessing

A point cloud usually contains a large number of points, and constructing a graph with all points as vertices imposes a significant computational burden on the network. In 3D object detection, a voxel is the 3D equivalent of a pixel in 2D images. We divide the 3D space containing the point cloud into a grid of these small cubes (voxels). For each voxel, only one representative point is kept, and all other points in that voxel are discarded. This process reduces the density of the point cloud by removing redundant points, while still retaining enough points to maintain the overall structure because points that are spatially close are grouped into a voxel, and one point is kept for that voxel. While voxel-wise sampling reduces the number of points, the rich information of the dense point cloud is not lost by encoding important information in the initial state of each vertex of the graph.

To preserve the information within the original point cloud, we encode the dense point cloud into the initial state value s_i of the vertex. We search the raw points within radius r_0 of each vertex and embed the LiDAR reflectance intensity and relative coordinates

using an MLP, then we aggregate these features to select the most salient features and use the obtained features as the initial state value s_i of the vertex. Each vertex of the graph has a state s_i that contains object information and is updated at each iteration. We consider the state s as the vertex features, which initially contain the vertex features but after processing by the graph neural network contain more complex information about the object class to which the point belongs as well as the information of the bounding box of the object.

3.4 Graph Creation

In a graph neural network, a point cloud consists of N points, each point containing three-dimensional coordinates and a point attribute. We define a point cloud consisting of N points as P = $\{P_1, ...,P_N\}$ where $P_i = (x_i , s_i)$ is a point with three-dimensional coordinates $x_i \in R^3$ and $s_i \in R^k$ is a vector of length k representing the point information.

Each point in the point cloud is considered as a vertex of the graph. And by connecting a point to its neighbors located within a fixed radius, the graph is formed. Examining the distance between each pair of points to determine their neighborhood can be computationally expensive. To speed up the process of finding neighboring points, we divide the 3D point cloud into a grid of small sections. Then, instead of comparing each point in the point cloud with every other point, we only need to examine the points in the small region (where the point is located) and the neighboring cells. This approach reduces the number of comparisons needed to find neighbors.

A graph neural network modifies the vertex feature by aggregating features along the edges. To do this, we extract all the features of a vertex and then extract the edge features. We aggregate the edge features and finally update the vertex feature.

$$V_i^{t+1} = g^t\left(p\left(\{ e_{ij}^t \mid (i,j) \in E\}\right), v_i^t\right) \quad (10)$$

$$e_{ij}^t = f^t(v_i^t, v_j^t) \quad (11)$$

e^t and v^t are the edge and vertex features from the t^{th} iteration. The function $f^t(.)$ ،computes the edge feature between vertex i and its neighbor vertex j at iteration t, ρ(.) sums all edge features for vertex i and combines the information of all neighboring vertices. $g^t(.)$ combines the summed edge features with the current feature of vertex i to produce the updated feature of vertex i at iteration t+1. This process is repeated multiple times for each vertex, helping the vertices to refine their understanding of the graph structure based on their neighboring vertices.

3.5 Translational Motion in the Point Cloud

If you use absolute coordinates $x_i$and x_j, it becomes problematic if the point cloud moves because the absolute positions of the points change and the model has to relearn these positions. Relative coordinates help the graph neural network understand how the points are spatially related. One of the main reasons for using relative coordinates is to achieve translation invariance, which means that the model can handle changes in the entire point cloud. For example, if the entire point cloud moves 5 meters to the right, the relative distance between the points remains the same.

$$s_i^{t+1} = g^t\left(\rho\left(\left\{f^t\left(x_j - x_i, s_j^t\right)\middle|(i,j) \in E\right\}\right), s_i^t\right) \quad (12)$$

$f^t\left(x_j - x_i, s_j^t\right)$ instead of using vertex features to calculate edge features, it uses the relative coordinates between two vertices and the s_j^t feature of the neighboring vertex. This helps the model calculate spatial relationships between points. The model looks at points in relation to their neighbors, which helps it focus on the shape and structure of the object rather than its absolute position. This makes the model invariant to translational motion, which means it can accurately detect objects near, far, or at any different location in the hypersphere.

Relative coordinates help the model to remain insensitive to large changes, but local transitions can still cause changes in the model input. When a vertex experiences a small local transition, for example, it moves slightly due to sensor noise, the relative coordinates between the vertex and its neighbors change, increasing the variability in the network inputs. To overcome this problem, the coordinates of neighboring vertices can be dynamically adjusted to account for small local transitions. A vertex that is to be updated has its own feature or state from the previous stage, represented as s_i^t. A graph neural network using s_i^t uses an offset for its neighboring vertices, which helps adjust the position of neighboring vertices relative to the vertex being updated.

$$\Delta x_t^i = h^t(s_i^t) \quad (13)$$

$$\Delta x_i^t = MLP_t^h(s_i^t) \quad (14)$$

$$e_{ij}^t = MLP_t^f\left(\left[x_j - x_i + \Delta x_i^t, s_j^t\right]\right) \quad (15)$$

$$s_i^{t+1} = MLP_t^g\left(Max\left(\left\{e_{ij}^t\middle|(i,j) \in E\right\}\right)\right) + s_i^t \quad (16)$$

In equation (14), MLP takes the current state (s_i^t)of the vertex and calculates an offset that adjusts the vertex position to reduce the effect of local translational motion. Equation (15) shows the calculation of the edge feature between vertex i and vertex j based on the relative coordinates and the neighbor feature. In equation (16), the Max function collects the edge features from all neighboring

vertices j connected to vertex i and collects the most important information from the neighbors. The state of vertex i in the next iteration is updated based on the important features collected from the neighbors and the previous state.

3.6 Teacher's Loss Function

For object classification, we compute a multiclass probability distribution $\{p_{c1}, \ldots, p_{cM}\}$ for each vertex. If a vertex is in an object bounding box, we assign it the object class, and if a vertex is outside the bounding box, we assign it the background class. For the classification loss, we use the average cross-entropy loss, where p_c^i is the predicted probability and y_c^i is the class label for vertex i.

$$l_{clss\text{Tchr}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{c_j}^i \log\left(p_{c_j}^i\right) \quad (17)$$

$$l_{loc\text{Tchr}} = \frac{1}{N}\sum_{i=1}^{N}(v_i \in b_{interest}) \sum_{\delta\in\delta_{b_i}} l_{huber}\left(\delta - \delta^{gt}\right) \quad (18)$$

If a vertex is in a bounding box, we calculate the Huber loss between the ground truth and our prediction, and if a vertex is outside the bounding boxes or belongs to a class that we do not need to locate, we set its location loss to zero, which is done in a similar way to [17], (see Appendix A). To avoid overfitting, we add L1 regularization to each MLP. The final loss for the teacher is represented as l_{total_Tchr}, as shown in Equation (19):

$$l_{total\_\text{Tchr}} = l_{cls\text{Tchr}} + l_{loc\text{Tchr}} + l_{reg\text{Tchr}} \quad (19)$$

3.7 Bounding Boxes

In point clouds, multiple vertices can belong to the same object. As a result, the neural network can generate multiple bounding boxes for the same object, which is problematic because we only want one accurate bounding box for each object. To overcome this problem, we need to merge the bounding boxes into a single box and assign it a confidence score that reflects both the classification and localization quality. The standard method to achieve this goal is called non-maximum suppression. In essence, we select the box with the highest classification score and remove all other boxes that overlap with it. To improve the localization accuracy, similar to the method in [17], we calculate the merged box by considering the entire cluster of overlapping boxes (see Appendix B).

4. EXPERIMENTAL RESULTS

4.1 Dataset

The KITTI 3D object detection dataset[29], is the most widely used benchmark for 3D object detection, consisting of a training image and a test image, as well as the corresponding point clouds. Although 8 different classes are labeled in the dataset. Only the car and pedestrian classes are evaluated in the benchmark dataset, the following are only those classes that are labeled sufficiently for a comprehensive evaluation. In this paper, the focus is on the car class.

4.2 New dataset for student

The details of constructing a new dataset based on the performance of the teacher model are described in Section 3.1. After identifying TPS and TPW point clouds based on the teacher model's performance, we examine the ratio of TPS and TPW point clouds in KITTI training set. Due to the severe resource limitation for conducting the research, we select 1000 data as the new dataset while maintaining the original ratio of TPS and TPW ratio to ensure that the student model encounters TPW point clouds in the same proportion as the teacher model. Finally, the new dataset consists of 900 TPS random point clouds and 100 TPW random point clouds.

4.3 Evaluation Metrics

In 3D object detection, evaluation metrics such as Recall, Precision, and Average Precision (AP) are used to evaluate the performance of the model. Recall, which measures the proportion of true positives correctly identified by the model, has been thoroughly explained in Section 3.1.1. In this section, we focus on other evaluation metrics such as Precision and AP. Precision measures the proportion of identified samples that are actually correct. A high Precision indicates that when the model recognizes an object, it is usually correct but may miss some objects. The Precision formula is given in Equation (20):

$$Precision = \frac{True\ positive}{True\ positive + False\ Positive} = \frac{TP}{TP+FP} \quad (20)$$

Average precision (AP) is the average of the precision values at different recall levels, providing a single number that summarizes the Precision-Recall curve. AP indicates the ability of the model to maintain high precision while increasing recall. Higher AP indicates a better balance between precision and recall at all threshold levels.

4.4 Details of the Teacher and Student Model

The teacher model is a Graph Neural Network (GNN) specifically designed to process point cloud data efficiently. It consists of four graph layers, each incorporating multiple MLPs to perform feature extraction, edge feature computation, and vertex updates. The model includes a primary MLP with dimensions (300,128,64,32) for extracting initial vertex features, followed by three additional MLP layers with dimensions (3,64), (300,300) for computing relationships between neighboring points, and (300,300) for updating vertex features. In comparison, the student model is a more lightweight version of the teacher model, designed with two fewer graph

layers to achieve a balance between accuracy and computational efficiency. Through adaptive weighted knowledge distillation, the student effectively learns from the teacher while requiring fewer parameters, enabling it to maintain similar performance. This reduction in complexity improves computational efficiency, making the student model well-suited for real-time applications.

4.5 Numbers of Teacher and Student Parameters

The details of the teacher and student model are given in Section 4-4. In this section, we will examine the number of parameters of the teacher and student model. The teacher with 4 graph layers has 940300 parameters. By removing the two graph layers, the student model has 549182 parameters. The student model, by removing two layers, has a smaller number of parameters than the teacher model.

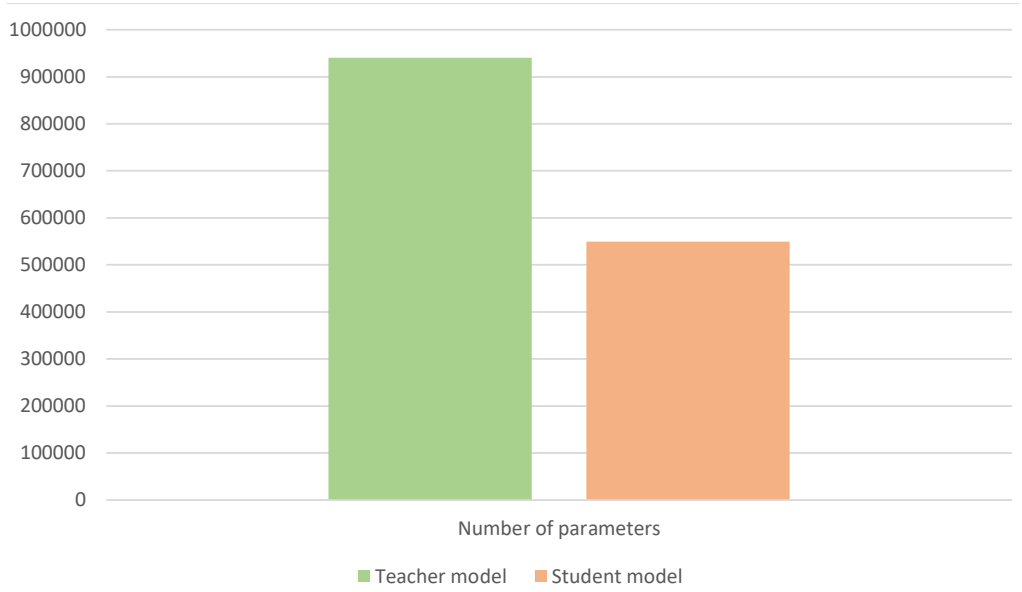

Figure 3: Comparison of the number of parameters in the teacher model and the student model.

4.6 Number of computations per layer

The teacher model is a Graph Neural Network (GNN) with four graph layers, while the student model is a more lightweight version with two fewer layers, designed to enhance computational efficiency. Each GNN layer comprises of two primary operations: vertex-wise feature transformations and edge-wise message passing. For vertex-wise computations, an MLP processes each vertex's features, resulting in a total computation of $V \times 524$, where V represents the number of vertices. The value 524 is derived from the total neurons across all MLP layers involved in vertex feature extraction, specifically 300, 128, 64, and 32 neurons, summing to 524. For edge-wise computations, interactions between connected vertices are calculated, resulting in a total computation of $E \times 600$, where E represents the number of edges. The value 600 is obtained from the neurons used in MLP layers for edge feature computations. Since the teacher model consists of four graph layers, its total computational cost is $(V \times 524 + E \times 600) \times 4$, while the student model, with two fewer graph layers, results in $(V \times 524 + E \times 600) \times 2$. By utilizing adaptive weighted knowledge distillation, the student model achieves comparable performance with significantly lower computational demands, making it more efficient for real-time applications.

4.7 Results

In this section, we analyze the results of our proposed method. The first part focuses on the comparison between the teacher and student models, examining their classification performance based on metrics such as Recall, Precision, and AP values. In the second part, the performance of the student model is compared with Point-GNN, a prominent baseline in 3D object detection, using the same metrics to highlight the effectiveness of our approach.

4.7.1 Comparison of Teacher and Student Models

In this subsection, we evaluate and compare the classification performance of the teacher and student models using metrics such as Recall, Precision, and AP values. Table 1 summarizes the results, and the following analysis highlights the strengths and weaknesses of each model across different classes. This comparison provides insights into the effectiveness of the knowledge distillation process applied to the student model.

Table 1: Comparison of Teacher and Student Results Based on Recall, Precision, and AP

Teacher Model				Student Model			
Class	Recall	Precision	AP	Class	Recall	Precision	AP
Do not care	0.989	0.982	0.997	Do not care	0.991	0.984	0.998
Car side-view	0.668	0.770	0.767	Car side-view	0.684	0.776	0.771
Car front-view	0.767	0.800	0.844	Car front-view	0.782	0.822	0.847
Background	0.501	0.635	0.565	Background	0.521	0.660	0.578

In the "Do not care" class, the two teacher and student models have almost similar results. The recall and precision of both models are close to 0.99, indicating that the data in this class is generally reliable and the teacher model has provided accurate predictions. In the "Car side-view" class, the student shows enhanced performance compared to the teacher. The recall of the teacher is 0.668, while the student model achieves a recall of 0.684. Additionally, the student's precision improves from 0.770 to 0.776, and the AP value is also higher than that of the teacher. For the "Car front-view" class, the results show that the teacher attains a recall of 0.767, demonstrating relatively strong performance. The student further enhances these results, achieving a higher precision of 0.822 compared to the teacher's 0.800 and an increased AP from 0.844 to 0.847. These improvements highlight the student model's ability to maintain and slightly enhance detection accuracy in this class. For the "Background" class, both the teacher and student models exhibit lower performance compared to other classes. The teacher model achieves a recall of 0.501, indicating difficulties in accurately predicting this class. The student model shows a slight improvement, achieving higher precision and AP values, though they remain lower than those of other classes.

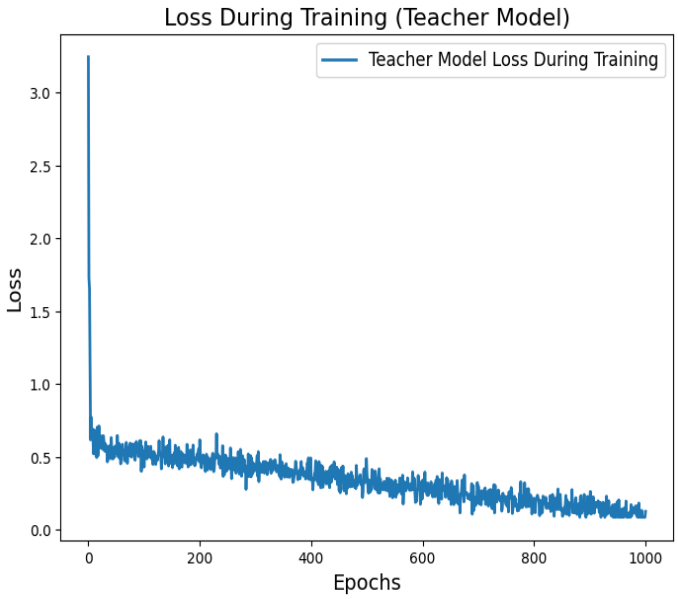

Figure 4: Training Loss Curve for the Teacher Model

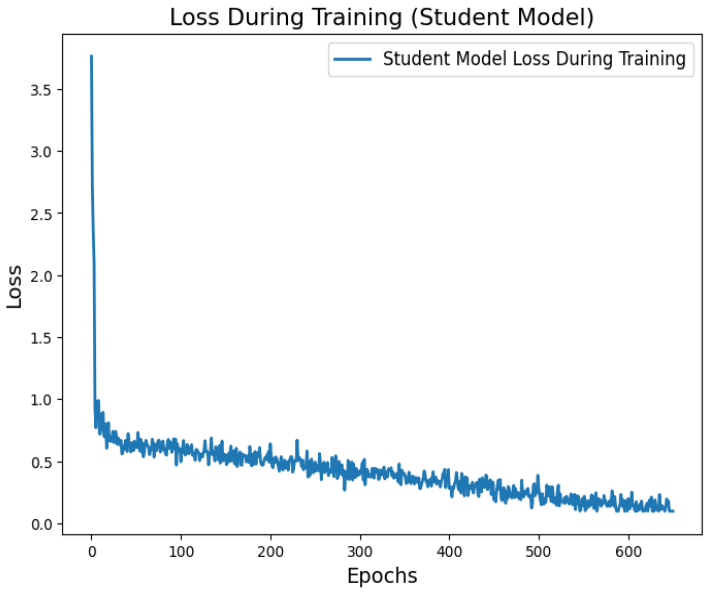

Figure ᵒ: Training Loss Curve for the Student Model

Figures 4 and 5 illustrate the training loss curves for the teacher and student models, respectively. Both models show a steep decline in loss during the initial training epochs, gradually stabilizing as training progresses. This demonstrates effective learning and convergence of both models. Notably, despite its reduced number of parameters, the student model achieves a loss reduction comparable to the teacher model, highlighting the effectiveness of the proposed knowledge distillation approach.

4.7.2 Comparison of Student Model with Point-GNN

This section compares the performance of the student model with Point-GNN [17], a widely recognized approach in 3D object detection. The comparison focuses on key evaluation metrics, including Recall**,** Precision**,** and AP, across multiple classes, as summarized in Table 2. By analyzing these metrics, we aim to highlight the advantages of the adaptive weighted knowledge distillation framework implemented in the student model, showcasing its improvements over Point-GNN[17].

Table 2: Comparison of Student Model with Point-GNN

Student Model				Point-GNN			
Class	Recall	Precision	AP	Class	Recall	Precision	AP
Do not care	0.991	0.984	0.998	Do not care	0.987	0.980	0.996
Car side-view	0.684	0.776	0.771	Car side-view	0.649	0.744	0.748
Car front-view	0.782	0.822	0.847	Car front-view	0.751	0.803	0.820
Background	0.521	0.660	0.578	Background	0.498	0.635	0.54٣

In the "Do not care" class, the student achieves a recall of 0.991, precision of 0.984, and AP of 0.998, slightly outperforming Point-GNN [17], which attains 0.987, 0.980, and 0.996, respectively. T These results indicate that both models maintain a high level of reliability in this class. For the "Car side-view" class, the student achieves a recall of 0.684, precision of 0.776, and AP of 0.771, compared to Point-GNN's recall of 0.649, precision of 0.744, and AP of 0.748. This shows that the student model enhances detection performance in this class. In the "Car front-view" class, the student attains a recall of 0.782, precision of 0.822, and AP of 0.847, outperforming Point-GNN [17], which achieves a recall of 0.751, precision of 0.803, and AP of 0.820. In the "Background" class, which presents more challenges due to data sparsity, the student achieves a recall of 0.521, precision of 0.660, and AP of 0.578, while Point-GNN [17] achieves lower values of 0.498, 0.635, and 0.543. Overall, the student model achieves higher recall, precision, and AP across all classes compared to Point-GNN [17]. These quantitative results highlight the effectiveness of the proposed approach in achieving better detection accuracy across different classes.

4.7.3 Localization Results of Teacher and Student Models

In this section, the results of the teacher-student model for positioning and estimating the dimensions of 3D objects in two classes of side view and front view of the vehicle are examined. These results include the positioning loss values and other geometric parameters such as x, y, z 3D coordinates of the center of the bounding box, l, h and w are the length, height, and width of the bounding box, respectively, and θ the angle of rotation of the bounding box in the horizontal plane, which tells the model how the vehicle or object is rotated in space.

Table 3: Comparison of Localization Results Between Teacher and Student Models

Teacher model								
class	x	y	z	l	h	w	θ	Localization loss
Car side-view	0.3810	0.1938	1.3103	0.4753	0.2016	0.1684	1.5462	0.6109
Car front-view	0.3423	0.4506	2.2580	0.9209	0.4691	0.4235	1.0462	0.8443
Student model								
class	x	y	z	l	h	w	θ	Localization loss
Car side-view	0.3594	0.1839	1.2367	0.4467	0.1919	0.1618	1.5363	0.5965
Car front-view	0.3336	0.4373	2.2314	0.8924	0.4558	0.4155	1.0329	0.8302

In the "Car side-view" class, the student achieves a lower localization loss (0.5965) compared to the teacher (0.6109). Additionally, the z-coordinate estimation improves from 1.3103 to 1.2367, indicating a reduction in depth estimation errors. The student also shows slight improvements in length, height, and width estimations, leading to more accurate bounding box localization. In the "Car front-view" class, the student also demonstrates better localization accuracy, achieving a lower localization loss of 0.8302 compared to 0.8443 in the teacher. The z-coordinate estimation improves from 2.2580 to 2.2314, along with minor refinements in bounding box dimensions. These results indicate that the student outperforms the teacher in localization accuracy, achieving lower localization loss and better depth estimation in both object classes.

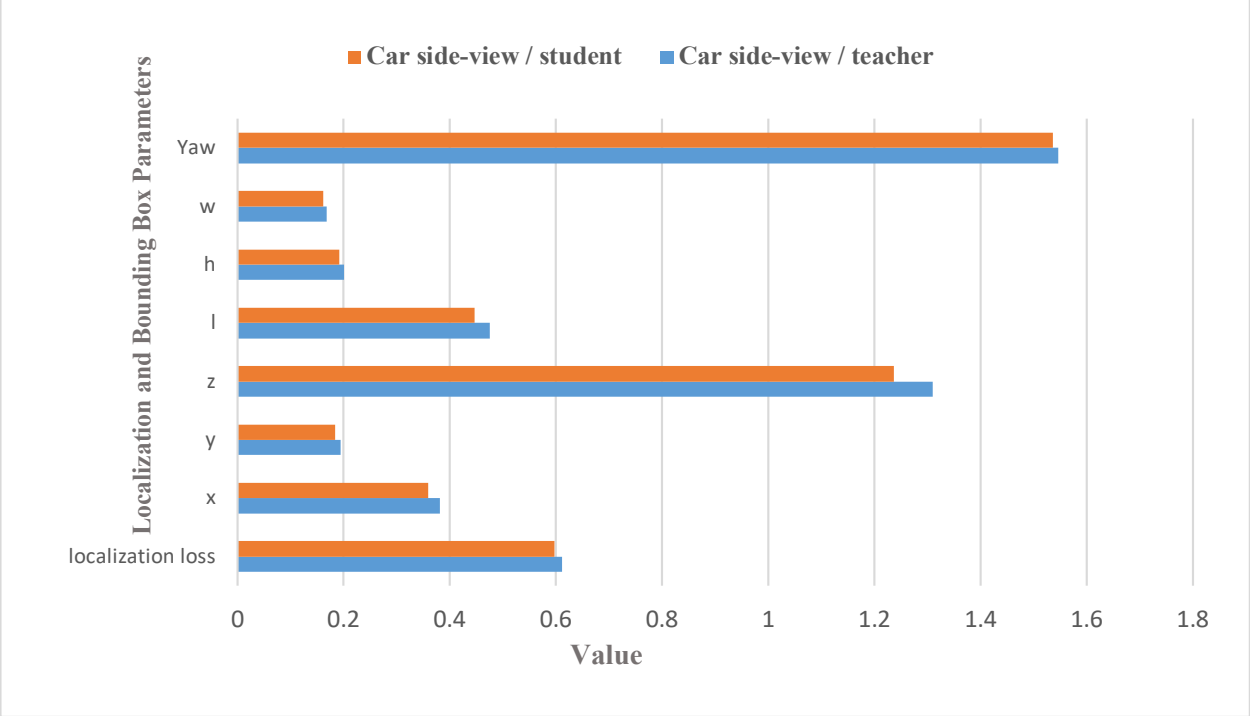

Figure 6*: Comparison of localization loss and bounding box parameters between teacher and student models for car side-view*

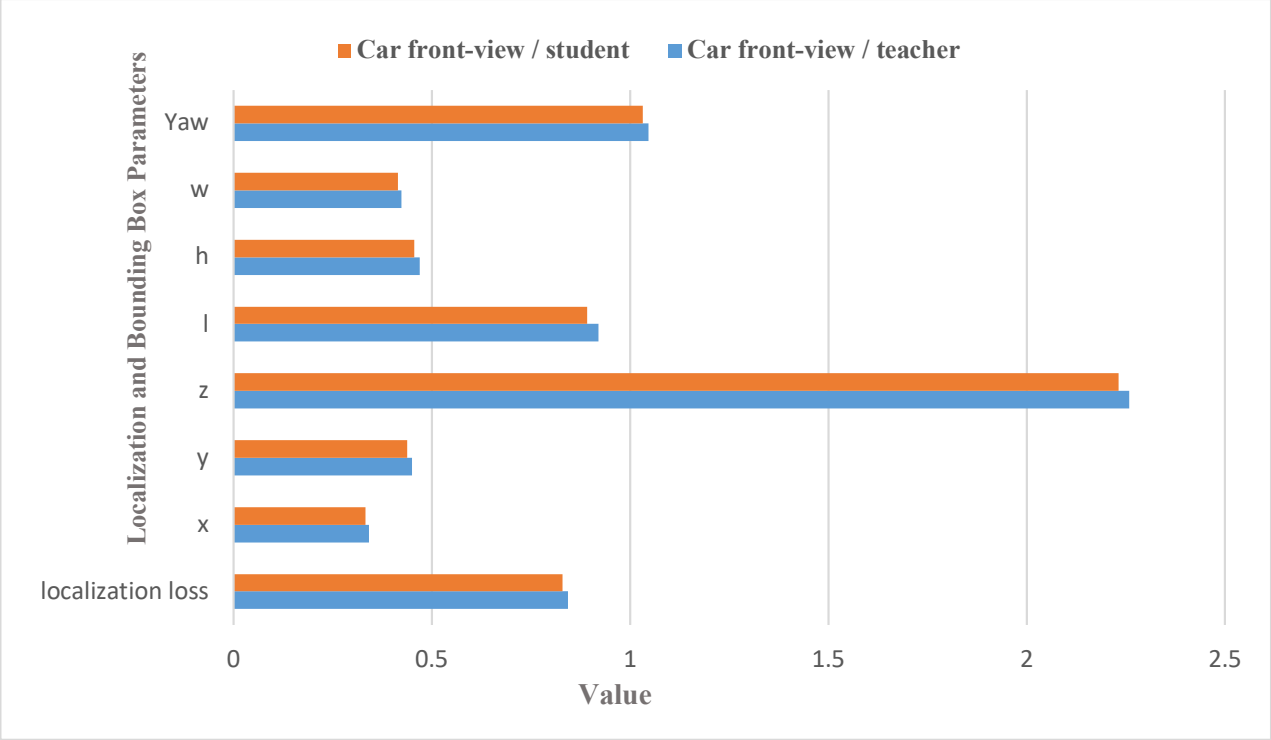

Figure 7: Comparison of localization loss and bounding box parameters between teacher and student models for car front-view

Figures 6 and 7 compare the localization loss and bounding box parameters between the teacher and student models for car side-view and front-view classes. The results illustrate the improvements in depth estimation and overall localization accuracy of the student model.

4.9 Ablation Study

This section presents an ablation study to investigate the influence of key components in the proposed model. The study assesses the contribution of adaptive weighted knowledge distillation mechanisms, the depth of the teacher model, and the number of neurons in the network in determining overall performance. The experiments include an analysis of the effects of removing the adaptive weighted knowledge distillation, which dynamically adjusts loss weights based on the teacher's performance. Additionally, we explored the effect of reducing the teacher model's depth from 4 to 3 layers, while maintaining the adaptive weighted knowledge distillation framework. Finally, we analyzed the performance by decreasing the number of neurons in both the teacher and student networks from 300 to 200. The results of these experiments, summarized in Table 3, provide valuable insights into how these changes affect recall, precision, and AP metrics across the four defined classes for both teacher and student models.

Table 4 **:**Ablation study analyzing the impact of key architectural and training modifications on model performance

Experiment	class	Model	Recall	Precision	AP
Removal of Adaptive Weighted Mechanism	Do not care	Teacher	0.989	0.982	0.997
		Student	0.988	0.983	0.997
	Car side-view	Teacher	0.668	0.770	0.767
		Student	0.670	0.773	0.766
	Car front-view	Teacher	0.767	0.800	0.844
		Student	0.774	0.815	0.846
	Background	Teacher	0.501	0.635	0.565
		Student	0.508	0.649	0.573
Teacher Depth Reduction: 4 to 3 Layers (with Adaptive Weighted Knowledge Distillation)	Do not care	Teacher	0.989	0.980	0.994
		Student	0.987	0.980	0.996
	Car side-view	Teacher	0.634	0.743	0.740
		Student	0.6٤8	0.752	0.753
	Car front-view	Teacher	0.729	0.779	0.817
		Student	0.734	0.797	0.829
	Background	Teacher	0.474	0.598	0.552
		Student	0.489	0.617	0.560
Decreasing Neurons Across Teacher and Student Networks: 300 to 200 (with Adaptive Weighted Knowledge Distillation)	Do not care	Teacher	0.987	0.982	0.995
		Student	0.986	0.984	0.996
	Car side-view	Teacher	0.637	0.753	0.750
		Student	0.648	0.772	0.765
	Car front-view	Teacher	0.743	0.778	0.827
		Student	0.757	0.802	0.833
	Background	Teacher	0.484	0.623	0.550
		Student	0.499	0.636	0.574

The ablation study examines the effects of key architectural and training modifications on model performance. Table 4 summarizes the results for three main experiments: the removal of adaptive weighted knowledge distillation, reducing the depth of the teacher model from 4 to 3 layers, and decreasing the number of neurons from 300 to 200. The results show that eliminating the adaptive weighted mechanism leads to a decline in recall, precision, and AP values across all classes, with a more noticeable drop in challenging classes such as "Car side-view" and "Car front-view." Reducing the teacher model's depth leads to a consistent drop in detection accuracy, affecting both teacher and student models. Additionally, decreasing the number of neurons results in a decline in performance, especially in complex classes where detailed feature extraction is critical. These results highlight the quantitative impact of each modification on detection performance, emphasizing the role of model complexity in maintaining accuracy.

5. CONCLUSION

In this paper, we proposed an Adaptive Weighted Knowledge Distillation framework for 3D object detection in self-driving cars using point cloud data. Our approach dynamically regulates the knowledge transfer process based on the teacher model's

performance, ddividing point clouds into TPS (Teacher Performs Strongly) and TPW (Teacher Performs Weakly). For TPS, a higher knowledge distillation weight and a lower student loss weight were assigned, which has allowed the student model to learn more accurate patterns than the teacher model. On the other hand, in TPW the student model was able to improve its results somewhat by increasing the initial loss weight and relying more on raw data. Additionally, a data pruning mechanism was introduced to optimize dataset size while maintaining the TPS-TPW ratio. Overall, this research investigates that using a smart approach in assigning different weights to the distillation and student losses can effectively help improve the performance of lighter models. This method not only keeps the model accuracy close to the teacher accuracy and even achieves better accuracy than the teacher model in some cases, but also avoids the consumption of large computational resources and provides a suitable solution for implementation in resource-constrained systems, such as self-driving cars. For future research, several directions can be explored:

1. Enhancing the Teacher Model: Future research could focus on employing state-of-the-art 3D object detection networks as teacher models Utilizing more advanced and well-optimized teacher models would strike a better balance between complexity and accuracy can be achieved, providing a stronger foundation for the knowledge distillation process. This could further enhance the performance of the student model while preserving computational efficiency.
2. Dataset Independence in Knowledge Distillation: The current approach relies on access to the original dataset for effectively training the student model. Future work could investigate data-free knowledge distillation methods, which generate synthetic data or transfer knowledge without requiring access to the real dataset. This would make the proposed framework more adaptable for privacy-sensitive environments or situations with limited availability of labeled data.
3. Dynamic Model Adaptation for Real-Time Scenarios: Future work could focus on designing a student model capable of dynamically adjusting its architecture or computational requirements based on the complexity of the input data. This approach would allow the model to conserve computational resources for simpler inputs while keeping high accuracy for more challenging cases This could ensure that the model keeps high accuracy for challenging cases while conserving computational resources for simpler inputs, making it even more suitable for real-world applications such as self-driving cars and robotics.

6. REFERENCES

[1] G. Zhang, W. Yu, and R. Hou, "MFIL-FCOS: A Multi-Scale Fusion and Interactive Learning Method for 2D Object Detection and Remote Sensing Image Detection," *Remote Sensing,* vol. 16, no. 6, p. 936, 2024.

[2] M. Bansal, M. Kumar, and M. Kumar, "2D object recognition: a comparative analysis of SIFT, SURF and ORB feature descriptors," *Multimedia Tools and Applications,* vol. 80, no. 12, pp. 18839-18857, 2021.

[3] B. Kim, J. Lee, S. Lee, D. Kim, and J. Kim, "TricubeNet: 2D kernel-based object representation for weakly-occluded oriented object detection," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2022, pp. 167-176.

[4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652-660.

[5] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems,* vol. 30, 2017.

[6] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770-779.

[7] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9277-9286.

[8] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11040-11048.

[9] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "Std: Sparse-to-dense 3d object detector for point cloud," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1951-1960.

[10] Z. Wang *et al.*, "Range adaptation for 3d object detection in lidar," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0-0.

[11] L. Du *et al.*, "Ago-net: Association-guided 3d point cloud object detection network," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 44, no. 11, pp. 8097-8109, 2021.

[12] S. Song, T. Huang, Q. Zhu, and H. Hu, "ODSPC: deep learning-based 3D object detection using semantic point cloud," *The Visual Computer,* vol. 40, no. 2, pp. 849-863, 2024.

[13] L. Fang, Y. Huang, Q. Zhao, and Y. Wan, "Monocular 3D Object Detection Based on Occlusion Optimization," in *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2022: IEEE, pp. 604-609.

[14] J. Shin, J. Kim, K. Lee, H. Cho, and W. Rhee, "Diversified and realistic 3D augmentation via iterative construction, random placement, and HPR occlusion," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023, vol. 37, no. 2, pp. 2282-2291.

[15] X. Chen, M. Chen, S. Tang, Y. Niu, and J. Zhu, "MOSE: Boosting Vision-based Roadside 3D Object Detection with Scene Cues," *arXiv preprint arXiv:2404.05280,* 2024.

[16] A. Gambashidze, A. Dadukin, M. Golyadkin, M. Razzhivina, and I. Makarov, "Weak-to-Strong 3D Object Detection with X-Ray Distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15055-15064.

[17] W. Shi and R. Rajkumar, "Point-gnn: Graph neural network for 3d object detection in a point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1711-1719.

[18] M. Afzal Ansari, M. Meraz, P. Chakraborty, and M. Javed, "Angle Based Feature Learning in GNN for 3D Object Detection using Point Cloud," *arXiv e-prints,* p. arXiv: 2108.00780, 2021.

[19] J. Wang, H. Gang, S. Ancha, Y.-T. Chen, and D. Held, "Semi-supervised 3D object detection via temporal graph neural networks," in *2021 International conference on 3D Vision (3DV)*, 2021: IEEE, pp. 413-422.

[20] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (tog),* vol. 38, no. 5, pp. 1-12, 2019.

[21] X. Chen, J. Su, and J. Zhang, "A two-teacher framework for knowledge distillation," in *Advances in Neural Networks–ISNN 2019: 16th International Symposium on Neural Networks, ISNN 2019, Moscow, Russia, July 10–12, 2019, Proceedings, Part I 16*, 2019: Springer, pp. 58-66.

[22] Y. Li, S. Xu, M. Lin, J. Yin, B. Zhang, and X. Cao, "Representation disparity-aware distillation for 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 6715-6724.

[23] H. Cho, J. Choi, G. Baek, and W. Hwang, "itkd: Interchange transfer-based knowledge distillation for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13540-13549.

[24] F. Li *et al.*, "Pre-pruned Distillation for Point Cloud-based 3D Object Detection," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, 2024: IEEE, pp. 3192-3198.

[25] H. Zhang, L. Liu, Y. Huang, Z. Yang, X. Lei, and B. Wen, "CaKDP: Category-Aware Knowledge Distillation and Pruning Framework for Lightweight 3D Object Detection," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024: IEEE, pp. 15331-15341.

[26] J. Yang, S. Shi, R. Ding, Z. Wang, and X. Qi, "Towards efficient 3d object detection with knowledge distillation," *Advances in Neural Information Processing Systems,* vol. 35, pp. 21300-21313, 2022.

[27] L. Zhang *et al.*, "Structured Knowledge Distillation Towards Efficient Multi-View 3D Object Detection," in *BMVC*, 2023, pp. 339-344.

[28] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision,* vol. 129, no. 6, pp. 1789-1819, 2021.

[29] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research,* vol. 32, no. 11, pp. 1231-1237, 2013.

APPENDIX A

The bounding box of the object is predicted as $b = (x, y, z, l, h, w, \theta)$where (x, y, z) is the center position of the bounding box, (l, h, w)represents the length, width and height of the bounding box respectively, and θ is the Yaw angle. We encode the bounding box with vertex coordinates $(x_{v,}\, y_{v,}, z_{v,})$as follows:

$$\delta_x = \frac{x - x_v}{l_m}\,, \delta_y = \frac{y - y_v}{h_m}\,, \delta_z = \frac{z - z_v}{w_m} \quad \text{(A-1)}$$

$$\delta_l = \log\left(\frac{l}{l_m}\right), \delta_h = \log\left(\frac{h}{h_m}\right), \delta_w = \log\left(\frac{w}{w_m}\right) \quad \text{(A-2)}\ \delta_\theta = \frac{\theta - \theta_0}{\theta_m} \quad \text{(A-3)}$$

In the above equations, $l_m, h_m, w_m, \theta_0, \theta_m$ are constant scale factors. The localization predicts the encoded bounding box $\delta_b = (\delta_x, \delta_y, \delta_z, \delta_l, \delta_h, \delta_w, \delta_\theta)$for each class. If a vertex is in a bounding box, we calculate the Huber loss between the ground truth and the prediction. If a vertex is outside the bounding box or belongs to a class that we do not need to locate, we set its localization loss to zero. Then the localization loss of all vertices is averaged:

$$l_{loc} = \frac{1}{N}\sum_{i=1}^{N}(v_i \in b_{interest}) \sum_{\delta \in \delta_{b_i}} l_{huber}\,(\delta - \delta^{gt}) \quad \text{(A-4)}$$

APPENDIX B

In localization, there may be multiple vertices on an object, and the neural network can output multiple bounding boxes of an object. These bounding boxes need to be merged into one and a confidence score assigned. Non-maximal suppression has been widely used for this purpose. The common approach is to select the box with the highest classification score and remove other overlapping boxes. However, the classification score does not always reflect the quality of localization. For example, a partially occluded object may have a strong clue indicating the type of object, but lack sufficient shape information. Standard non-maximal suppression may select an incorrect bounding box base based on classification alone.

To improve the localization accuracy[17], calculates the merged box by considering the entire cluster of overlapping boxes. The position and median size of the overlapping bounding boxes are considered and the confidence score is calculated as the sum of the classification scores weighted by the intersection of union (IoU) factor and an overlap factor. The overlap factor represents the proportion of the occupied volume. A box b_i has l_i ،w_i، ،h_i, as its length, width, and height, respectively, and v_i^l ، v_i^w ،v_i^h are unit vectors representing their directions, respectively. $x_j$is the coordinate of point p_j. The occlusion factor $o_i$is as follows:

$$o_i = \frac{1}{l_i\, w_i\, h_i} \prod_{v \in \{v_i^l\, v_i^w\, v_i^h\}} max_{p_j \in\, b_i}(v^T\, x_j) - min_{\,p_j \in\, b_i}(v^T\, x_j) \quad \text{(B-1)}$$

Algorithm 2: NMS with Box Merging and Scoring

Input: $\mathcal{B} = \{b_1, ..., b_n\}$, $\mathcal{S} = \{s_1, ..., s_n\}$, T_h
$\mathcal{B}$ represents the collection of identified bounding boxes.
$\mathcal{S}$ denotes the corresponding detection scores.
T_h is an overlapping threshold value.
The primary modifications are highlighted in red.

$\mathcal{M} \leftarrow \{\}$, $\mathcal{Z} \leftarrow \{\}$
while $\mathcal{B} \neq \emptyset$ **do**
  $i \leftarrow \arg\max \mathcal{S}$
  $\mathcal{L} \leftarrow \{\}$
  for $b_j \in \mathcal{B}$ **do**
    if $iou(b_i, b_j) > T_h$ **then**
      $\mathcal{L} \leftarrow \mathcal{L} \cup b_j$
      $\mathcal{B} \leftarrow \mathcal{B} - b_j$
      $\mathcal{S} \leftarrow \mathcal{S} - s_j$
      end
    end
  $m \leftarrow median(\mathcal{L})$
  $o \leftarrow occlusion(m)$
  $z \leftarrow (o+1) \sum_{b_k \in \mathcal{L}} IoU(m, b_k) s_k$
  $\mathcal{M} \leftarrow \mathcal{M} \cup m$, $\mathcal{Z} \leftarrow \mathcal{Z} \cup z$
end
return $\mathcal{M}, \mathcal{Z}$

The above algorithm returns the merged bounding boxes and their confidence score.

Samira Tajik received her M.Sc. degree in Computer Engineering with a specialization in Artificial Intelligence and Robotics from Shahid Beheshti University (SBU), Tehran, Iran. Her research interests include Computer Vision, Image Processing, Autonomous Systems and Robotics, and Graph Neural Networks.

Armin Salimi-Badr received the B.Sc., M.Sc. and PhD degrees in Computer Engineering, all from Amirkabir University of Technology, Tehran, Iran in 2010, 2012, and 2018 respectively. He also obtained a PhD degree in Neuroscience from University of Burgundy, Dijon, France in 2019, where he was researching on presenting a computational model of brain motor control in the Laboratory 1093 CAPS (Cognition, Action, et Plasticité Sensorimotrice) of the Institut National de la Santé et de la Recherche Médicale (INSERM). He was a Postdoctoral Research Fellow at Biocomputing lab of Amirkabir University of Technology from October 2019 to September 2020. Currently, he is an Assistant Professor at Faculty of Computer Science and Engineering of Shahid Beheshti University, Tehran, Iran and also the Head of Artificial Intelligence & Robotics & Cognitive Computing group in this faculty. He is also the founder and Chair of Robotics & Intelligent Autonomous Agents (RoIAA) Lab in Shahid Beheshti University. He is IEEE Senior Member and currently the Chair of Professional Activities Committee and a Board Member of Computer Society of IEEE Iran Section. Dr. Salimi-Badr was the recipient of the IEEE Young Investigator Award from the IEEE Iran Section in 2024. He is also the Distinguished Researcher of Shahid Beheshti University in the field of Computer Science and Engineering in 2025. His research interests include Computational Intelligence, Computational Neuroscience, and Robotics.