TUM School of Computation, Information and Technology
Technical University of Munich

Master's Thesis in Robotics, Cognition, Intelligence

Object Relations for 3D Object Detection

Objektbeziehungen für 3D-Objekterkennung

Supervisor Prof. Dr.-Ing. habil. Alois C. Knoll

Advisor Mingyu Liu, Bare Luka Zagar

Author Marc Brede

Date November 15, 2023 in Munich

Disclaimer

I confirm that this Master's Thesis is my own work and I have documented all sources and material used.

Munich, November 15, 2023

(Marc Brede)

Abstract

This thesis introduces an approach to enhancing 3D object detection in autonomous driving systems, focusing on the role of object relations in two-stage detection pipelines. The core contribution is the development of "PVRCNN-Relation", a model using a Graph Neural Network to model object relations. This model demonstrates superior performance over its baseline PVRCNN. Additionally, this work identifies four key motivations for incorporating object relations: increase in receptive field, exploiting patterns, detecting occlusions, and proposal consensus. These motivations are critical in addressing the current challenges in autonomous driving scenarios. This work carries out extensive experiments to assess the general applicability of object relations to a variety of object detection architectures. Additionally, it pioneers a transfer learning approach, making the integration of object relations into existing architecture more efficient. Finally, this thesis presents a general framework for integrating relations between objects into existing detection pipelines. This framework is designed to be adaptable, allowing for easy incorporation of relation-based modules into different existing architectures. The corresponding code can be found at https://github.com/MarcBrede/OpenPCDet.

Zusammenfassung

Diese Thesis führt einen Ansatz zur Verbesserung der 3D-Objekterkennung in autonomen Fahrsystemen ein, wobei sie sich auf die Rolle von Objektbeziehungen in zweistufigen Erkennungspipelines konzentriert. Der Hauptbeitrag ist die Entwicklung von "PVRCNN-Relation", einem Modell, das ein Graph-Neuronales Netzwerk verwendet, um Objektbeziehungen zu modellieren. Dieses Modell zeigt eine überlegene Leistung gegenüber seinem Basismodell PVRCNN. Zusätzlich identifiziert diese Arbeit vier Schlüsselmotivationen für die Einbeziehung von Objektbeziehungen. Diese Motivationen sind entscheidend, um die aktuellen Herausforderungen in Szenarien des autonomen Fahrens anzugehen. Diese Arbeit führt umfangreiche Experimente durch, um die allgemeine Anwendbarkeit von Objektbeziehungen auf eine Vielzahl von Objekterkennungsarchitekturen zu bewerten. Darüber hinaus leitet sie einen Ansatz des Transferlernens ein, wodurch die Integration von Objektbeziehungen in bestehende Architekturen effizienter wird. Abschließend präsentiert diese Arbeit einen allgemeinen Rahmen für die Integration von Beziehungen zwischen Objekten in bestehende Erkennungspipelines. Der zugehörige Code ist unter https://github.com/MarcBrede/OpenPCDet zu finden.

Contents

Chapter 1

Introduction

Autonomous driving is the concept of vehicles operating without any human intervention. It will fundamentally transform the way we view and interact with vehicles, roads, and cities in the future [Yur+20; Par+22; Hu+23]. The roots of this transformative technology can be traced back to initiatives like the DARPA Grand Challenge, which started early advancements in autonomous vehicle technology [Cav+19]. Today, autonomous driving has moved from experimental stages to real-world testing and limited commercial deployment. Companies like Waymo, Tesla, and Cruise are pushing the boundaries and are already operating autonomous taxi services in select areas. To enable this, autonomous vehicles rely on a deep stack of software that encompasses everything from low-level hardware interactions to high-level decision-making algorithms. The foundation of this technology is perceiving the environment. Modern autonomous vehicles achieve this through learning techniques, building on layers of innovations in machine learning (ML). Identifying and locating objects in the vehicle's surroundings is a key part of perception. This process entails identifying an object's position in 3D space and assigning it to specific categories, such as vehicles, pedestrians, or cyclists. Typically, sensors capture 3D data of the surrounding environment, which is then analyzed to detect and locate relevant objects. This task is widely referred to as 3D object detection. Achieving high accuracy in this task is essential for the effectiveness of subsequent planning algorithms responsible for safe driving. The objective of this work is to enhance autonomous vehicles' capabilities in 3D object detection. The upcoming sections of this chapter will dive into the importance of advancing autonomous vehicle technology and will outline the contributions and structure of this work.

1.1 Motivation

The development and implementation of autonomous driving promises to bring profound changes in the domain of transportation and beyond. A primary motivation for advancing this technology is road safety. Human-driven vehicles are often prone to errors due to distractions, fatigue, and other factors. In contrast, autonomous vehicles operate based on algorithms and sensors, which are not influenced by such human limitations, potentially reducing the number of road accidents [PPS22]. Furthermore, the widespread adoption of autonomous vehicles can provide increased mobility to various segments of the population. For instance, the elderly and individuals with disabilities, who might face challenges with conventional driving, could find greater independence [Kas+21]. Additionally, from a broader societal perspective, the efficient movement of autonomous vehicles could lower traffic congestion, leading to more efficient transport. This efficiency could also contribute to reducing fuel consumption, supporting sustainability goals [Ona+23]. In essence, the push for au-

tonomous driving is not merely about technological innovation but encompasses a vision for a safer, more inclusive, and sustainable future.

Robust perception, specifically for 3D object detection, is a critical step towards safe and reliable autonomous vehicles. However, state-of-the-art methods fail to achieve the required precision. Especially when the input sensors only capture sparse data due to occlusions, today's perception systems struggle to detect objects precisely. They fail to incorporate the necessary global context when predicting objects. To this end, we propose a module that explicitly models object relations to improve the performance of 3D object detection pipelines. By using a module that exchanges information between objects in a scene, the model can form a global understanding of the scene and subsequently use this information to reliably predict objects. With this module, this work aims to make existing object detection models more robust and ultimately contribute to the safety of autonomous vehicles.

1.2 Contribution

This work focuses on exploring the impact of modeling object relation in two-stage object detection pipelines, aiming to enhance their detection performance. To do so it introduces PVRCNN-Relation, an implementation that includes a module to relate objects (*Object Relation Module*) using a Graph Neural Network (GNN) with edge convolution message passing. This module is integrated into the PVRCNN baseline, a recent architecture with high rankings across various benchmarks. The primary contributions of this work are:

1. Establishing four key motivations for integrating object relation in 3D object detection models: increasing the receptive field, using proposal consensus, detecting occlusions, and exploiting traffic patterns.

2. Demonstrating the superiority of PVRCNN-Relation over the standard PVRCNN baseline, showcasing consistent improvements across various classes and difficulty levels.

3. Introducing a novel approach utilizing transfer learning and showing its capability to achieve comparable results to end-to-end trained models while being significantly more efficient.

4. Releasing comprehensive code as part of the OpenPCDet repository, featuring a framework for the *Object Relation Module* that can be integrated into any two-stage detection pipeline. This framework includes implementations for the baselines PVRCNN, PVRCNN++, PartA2, and Voxel RCNN extended with a module that relates objects.

By extending existing research on object relation in 3D object detection, this work not only improves detection performance of PVRCNN but also tests the general applicability of an *Object Relation Module* across multiple baseline detectors. The published code lays a solid foundation for future advancements in object relations, potentially enhancing various detection pipelines and contributing to the efficiency and safety of autonomous driving.

1.3 Structure

The structure of this work consists of six chapters, beginning with a foundational explanation of the theoretical background. This chapter will familiarize the reader with different sensors that are being used in autonomous vehicles to perceive the environment. It will also discuss

Deep Learning (DL) methodologies like Convolutional Neural Networks (CNNs) and GNNs. Finally, it will present specific ML models that are applied to the task of object detection. Chapter 3 reviews related work, presenting various approaches for 3D object detection, and includes literature reviews on object refinement and object relations for 3D object detection. Chapter 4 on methodology articulates this work's motivation in depth, shows shortcomings in related work, and details this work's implementation. Chapter 5 presents the results, offering both quantitative and qualitative insights. Finally, this work ends in chapter 6 by giving future directions and a conclusion.

Chapter 2

Theoretical Background

3D object detection for autonomous driving is an active research field that builds on top of a long chain of inventions both in the hardware and software domains. For the sake of understanding subsequent chapters, this chapter is dedicated to giving the reader the necessary theoretical background. It starts by explaining the hardware side, i.e. sensors that are being used for perceiving the environment. It then introduces software concepts applied to this task, starting with the general domain of DL and finally with the more specific task of object detection.

2.1 Hardware for Perception

Sensors are hardware devices that can detect events and changes in the environment and provide quantitative measurements for further tasks [Yeo+21]. One can differentiate between proprioceptive sensors, which capture information about the internal state of a system, and exteroceptive sensors, which gather data from the environment. Proprioceptive sensors may include encoders for wheel position, or force sensors in robotic arms for detecting applied pressure. On the other hand, exteroceptive sensors could involve Light Detection and Ranging (LiDAR) sensors to measure object distance, temperature sensors for environmental conditions, or cameras for visual data collection [Yeo+21]. Sensors can additionally be classified as passive sensors, which receive energy from the surroundings to produce an output, or active sensors. Passive sensors include cameras that capture light that is reflected from objects in the environment. In contrast, active sensors like LiDAR emit energy, such as light, to interact with the environment and measure properties like distance [Yeo+21]. Autonomous vehicles typically utilize proprioceptive sensors like Global Navigation Satellite Systems, the inertial measurement unit, and other vehicle odometry sensors to determine the absolute and relative position of the vehicle [STC19]. This section will specifically discuss the exteroceptive sensors used in autonomous driving in more depth, i.e. camera, radar, and LiDAR.

Camera

A camera can detect light emitted by the environment through a lens, producing a clear image of the environment [Cam+18]. The output of a camera is a dense representation of the surroundings that captures color and textual information to the extent that road signs, traffic lights, text, and lane markings can be detected [Cam+18; Jog+11]. Cameras are widely available, inexpensive, and provide a wide field of view [QLL22a]. A well-known drawback of them is that they cannot directly perceive depth. To overcome this issue, camera systems

of autonomous vehicles are often using several cameras [QLL22a]. The idea is to install two or more cameras at slightly different positions on the vehicle. The distance between the cameras, called baseline, leads to slightly different images being obtained from the cameras. An algorithm matching points between the images can be used to recover depth information about each point [QLL22a; Jog+11]. Essentially it uses the same mechanism that animals with two eyes are using [Jog+11]. Finally, another significant disadvantage of cameras is their sensitivity to illumination. At night, in rain or during heavy snowfall their perception capability is insufficient [Yan+18].

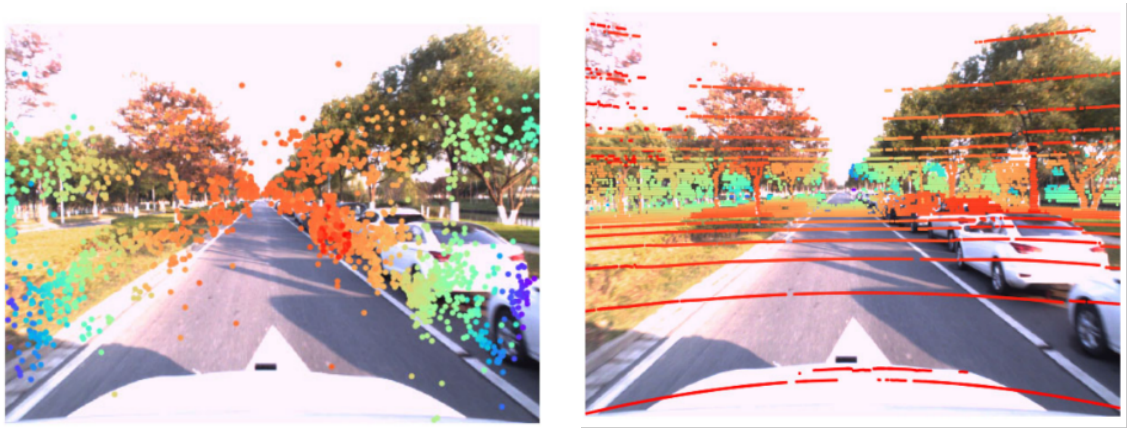
Radar

Radar is an active exteroceptive sensor that operates on the principle of radiating electromagnetic (EM) waves within the area of interest. Objects reflect these waves, they are received by the sensor and further processed to perceive the environment [Zim+22]. Radar can establish range information and using the Doppler property of EM waves it is the only sensor of the three that is capable of detecting the relative velocity of objects [STC19]. Unlike a camera, radar is not capable of perceiving text on road signs or lane markings [WWN19]. Radar additionally suffers from many false detections, especially around metal objects. Initial measurements are noisy and have to be filtered before they are usable for subsequent tasks [WWN19]. Radar can penetrate certain non-metallic materials, allowing it to have a long scanning range to detect objects far away [Zim+22]. Finally, the propagation of EM waves is independent of adverse weather conditions, hence, they can operate day, and night, in foggy, snowy, or cloudy conditions [STC19].

LiDAR

Instead of emitting EM waves, LiDAR operates by emitting pulses of infrared beams or laser light which reflect off target objects. Again, these reflections are captured and the time interval it took to receive them can be used to calculate the distance of objects in the environment. The result is a 3D representation of the surroundings in the form of a point cloud. Compared to radar the point cloud obtained from a LiDAR sensor has more precise depth information [Cam+18]. At present, either spinning or solid-state LiDAR is being used in autonomous vehicles. Spinning LiDAR involves a sensor mounted on a rotating base, which spins the sensor 360 degrees to collect data from all directions. On the other hand, solid-state LiDAR uses electronic beam steering rather than mechanical parts to direct the laser pulses [RCG21]. LiDAR is invariant to illumination meaning it works day and night but like radar, it is not capable of perceiving color or text [HKW22].

Figure 2.1 shows a comparison between a point cloud obtained from radar and LiDAR. Both point clouds have structural differences. As mentioned, LiDAR systems commonly use a rotating mechanism to send and receive light pulses. As the LiDAR unit rotates, it sends out laser pulses in a scanning motion, often in a horizontal plane. This results in the scan-like pattern seen in figure 2.1b. We can observe the capability of radar to detect points in the far distance, as seen in the middle of figure 2.1a. LiDAR technology has seen a high demand lately due to its precision which is crucial for commercial use cases such as autonomous driving, perception for drones, or humanoid robots. The demand is mainly driven by various private companies that were founded. Hence, the LiDAR market is forecasted to grow on average by 28% between 2019 and 2025 [LI20]. On the other hand, some car manufacturers are pursuing a camera-first approach and in fact, are replacing sensors with cameras. A third approach is to combine camera and LiDAR or radar information which is called fusion.



(a) Radar point cloud **(b)** LiDAR point cloud

Figure 2.1: Radar and LiDAR comparison, source: [Zhe+22]

It leverages the strength of each sensor to enhance the perception system. Various fusion strategies exist, including early [Bre+20], late [BSU22], or tight fusion [DV20], each with its trade-offs regarding system complexity, performance, and robustness. Additionally, technologies like RGB-D cameras, which combine color imaging with depth sensing, are being used because they offer a compact solution to capture both color and geometric information simultaneously [LFU13]. Camera, radar, and LiDAR all inherently suffer from occlusion issues, detecting primarily the first border of an object that intersects with their sensing rays. Be it a laser pulse from a LiDAR sensor, a radio wave from a radar sensor, or a light ray captured by a camera, the sensing mechanism can only register the initial surface it encounters, leaving anything behind that surface undetected and occluded [ZJ+22].

2.2 Deep Learning

ML has become the de facto standard when it comes to different perception tasks like object detection, segmentation, or optical flow [Zou+23]. The basic building block of an ML system, a learnable single-layer perceptron, is theoretically capable of approximating any continuous function to any desired degree, as proven by the Universal Approximation Theorem [Cyb89]. However, in practice stacking perception layers has empirically shown better performance than just using a single-layer [LBH15]. A model with many stacked layers is called a multi-layer perceptron (MLP) and is the basis for the field of DL. The key part of any ML system is that unlike a rule-based system, it learns from data rather than being explicitly programmed. This is achieved by iteratively updating the internal parameters of the model to minimize a loss function that quantifies prediction errors. Given the model's current prediction and the respective labels, the loss can be used to calculate the gradients of the model's parameters. These gradients are then used to update the parameters in a way that reduces the loss. This process is called backpropagation and is the engine of ML [Bis06]. Overall, the concept of learning complex functions through optimization has significantly advanced the field of computer vision. But MLPs are not the only building block of a DL system. Other blocks are designed for specific tasks. This section will introduce two such blocks, CNNs and GNNs.

2.2.1 Convolutional Neural Network

CNNs have emerged as the best architecture for visually perceiving the world. This is due to their ability to understand spatial hierarchies and local contexts in their inputs [Zou+23]. The important property of locality is built right into convolutions as they are essentially learnable filters that are being applied to specific regions. Initially, the field experimented with fixed kernels, such as the Sobel operator which was used for edge detection. However, these kernels had to be carefully designed by humans and failed to have the generality to be applicable to different tasks and inputs [Lin+20]. The field, therefore, shifted towards learnable kernels [LBH15]. Learning these kernel weights allowed models to be end-to-end learnable from data thus increasing the adaptability and performance on a wide range of applications [Zha+23].

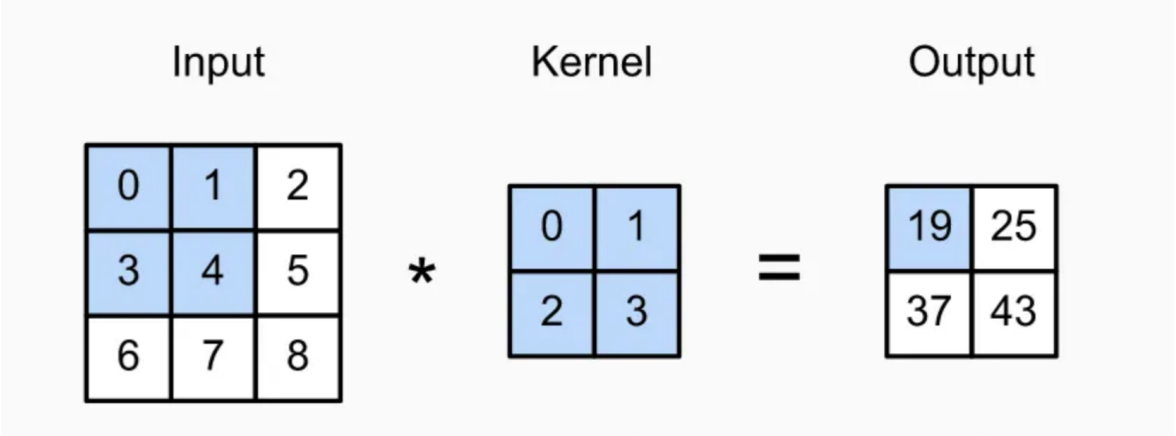


Figure 2.2: Kernel applied to input to obtain output, source: [Zha+23]

$$h_{i,j}^{(l)} = \sum_{m,n} W_{m,n}^{(l)} \cdot h_{i+m,j+n}^{(l-1)} \tag{2.1}$$

Figure 2.2 shows how a convolutional kernel is applied to an input to obtain an output. The kernel is fixed to one position of the input, then the kernel weights are multiplied with the respective input values, and all values are summed up and forwarded to the output. The kernel is subsequently moved to the next position and the process is repeated. In figure 2.2 we can see that the output value 19 is influenced by four values in the upper-left corner of the input, making it clear that locality is built into convolutions. The range of input values influencing each output value is termed the receptive field of the convolution. Equation 2.1 shows the formal representation of the convolution operation. It shows how the neuron value $h_{i,j}$ in layer l is calculated by summing up the products of the input values $h_{i+m,j+n}$ of layer $l-1$ and the kernel weights $w_{m,n}$.

In a typical modern CNN architecture, several layers of convolutions are stacked and connected with non-linear activation functions. The first layers learn low-level features like edges and corners, while the deeper layers learn more complex features like shapes and objects. The final layers are fully connected layers that use the learned features to compute the final output. The LeNet5 architecture [LeC+98] was one of the first CNNs to achieve state-of-the-art performance on the MNIST dataset for handwritten digit recognition.

The convolution operation in each layer is defined by the kernel size, the number of kernels, and the stride. The number of kernels defines how many kernels are applied to

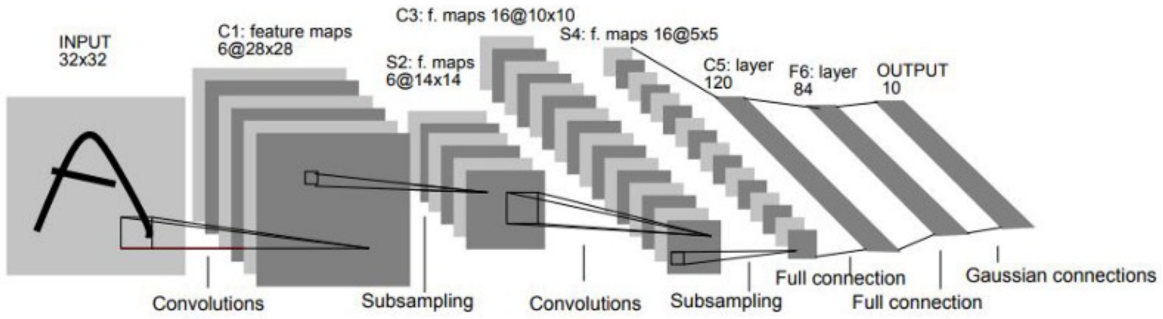


Figure 2.3: LeNet5 architecture, source: [LeC+98]

the input and the stride defines the number of pixels the kernel is moved between each convolution. Each of these values is a hyperparameter that can be tuned to achieve the best performance.

3D Convolutions

3D convolutions are an extension of 2D convolutions to handle 3D data. Here, the kernel is applied to voxels instead of pixels. Similar to 2.1, 2.2 shows the formal representation of 3D convolutions with the kernel size $M \times N \times P$ and three-dimensional input. Sparse convolutions are an extension of normal convolutions that are adapted to the sparsity of 3D data. Sparse convolutions aim to optimize computational efficiency by focusing only on the non-zero regions of the input volume. This is especially useful for 3D data, where entire regions can be empty or irrelevant, thus saving computational resources. Instead of applying the kernel to every position, it is applied only where it matters. This can lead to faster training and inference times while maintaining the same level of accuracy. Sparse convolutions are often used in applications like 3D object detection and medical image analysis, where data sparsity is a common issue [CGS19]. In equation 2.3, $\mathcal{S}(i,j,k)$ is the set of non-empty inputs. This effectively ignores the zero regions, making the convolution sparse. The rest of the equation is similar to the standard 3D convolution.

$$h_{i,j,k}^{(l)} = \sum_{m,n,p} W_{m,n,p}^{(l)} \cdot h_{i+m,j+n,k+p}^{(l-1)} \tag{2.2}$$

$$h_{i,j,k}^{(l)} = \sum_{(m,n,p)\in\mathcal{S}(i,j,k)} W_{m,n,p}^{(l)} \cdot h_{i+m,j+n,k+p}^{(l-1)} \tag{2.3}$$

Receptive Field

The concept of the receptive field is crucial for understanding the behavior of convolutional layers in a CNN. The receptive field refers to the spatial extent of the input that influences a particular output neuron. As mentioned before, for a single convolutional layer, the size of the receptive field is equal to the size of the convolutional kernel. However, as we stack multiple convolutional layers, the receptive field for neurons in deeper layers grows with the stride of the convolutions, encompassing a larger portion of the input image.

$$RF_{\text{new}} = RF_{\text{prev}} + (K-1) \times S_{\text{prev}} \tag{2.4}$$

In Equation 2.4, RF_{new} represents the receptive field of the current layer, RF_{prev} is the receptive field of the previous layer, K is the kernel size, and S_{prev} is the stride of the previous

layer. This equation shows how the receptive field increases as we go deeper into the network. It is important to consider the receptive field size during the design of a CNN, especially for tasks that require understanding the context of larger regions in the input.

2.2.2 Graph Neural Network

GNNs have emerged as a powerful tool for learning from graph-structured data, extending traditional neural network architectures to handle irregular data formats. GNNs operate by iteratively aggregating information from a node's local neighborhood, allowing for the incorporation of both node and edge attributes in the learning process. This enables GNNs to perform well in tasks such as node classification [Zho+19], link prediction [ZC18], and graph classification [Err+19]. GNNs have shown promise across various domains, including social network analysis, bioinformatics, and computer vision, demonstrating their utility in capturing complex relationships within data [Wu+22b].

Formally, a GNN operates on a Graph $G = (V, E)$ that is defined by its vertices V and edges E. Equation 2.5 shows how the hidden representation of vertex v_i in layer l is computed. The hidden representation of a vertex is computed by aggregating the hidden representations of its neighbors in the previous layer. The function $\mathcal{N}(i)$ returns the set of neighbors of vertex v_i, which defines what vertices are exchanging features. Given this neighborhood, the learnable weights $W_{j,i}$ are multiplied with the hidden representation of the neighbor $h_j^{(l-1)}$ and aggregated via the function $\oplus$. This function can be implemented as any permutation invariant function, such as max or sum.

$$h_i^{(l)} = \oplus_{j \in \mathcal{N}(i)} \left(W_{j,i}^{(l)} \cdot h_j^{(l-1)} \right) \tag{2.5}$$

Upon close inspection of the equation in 2.5 there is a notable similarity with the equation of convolutions given in 2.1. In fact, convolutions can be seen as a special case of GNNs, where the graph is a grid and the neighborhood of each vertex is defined by the kernel size and stride. Choosing $\sum$ as the permutation invariant function $\oplus$ and defining the neighborhood $\mathcal{N}$ with the distance of the pixels in the image plane, the convolution operation is obtained. This makes GNNs a generalization of convolutions, allowing for the application of GNNs to a wider range of problems.

2.3 Object Detection

To gain an understanding of the environment it is not sufficient to classify objects in the vicinity, but one should also be able to locate them. The task of classifying and locating objects is referred to as object detection; it is a fundamental problem of computer vision [Fel+09]. Obviously, both sub-tasks are crucial for many real-world applications. A perception system of an autonomous vehicle that can classify pedestrians in its vicinity but is unable to exactly locate them won't result in safe driving.

	2D	3D
Device	Camera	LiDAR, Radar, RGB-D Camera
Inputs	Image $\in \mathbb{R}^{h \times w \times c}$	Point Cloud $\in \mathbb{R}^{h \times w \times d \times c}$
Output	$[N \times 4], [N \times C]$	$[N \times 7], [N \times C]$

Table 2.1: Comparison of the inputs and outputs of 2D and 3D object detection.

Object detectors work on representations of the environment, such as images from cameras or point clouds from radar or LiDAR sensors. Table 2.1 outlines the inputs and outputs for 2D and 3D object detection systems. 2D detectors focus on a grid of pixels arranged in a two-dimensional plane, or simply, an image. Each pixel can have either a single value for its grayscale intensity or multiple values for its color components, e.g. RGB values. For depth information, radar, or LiDAR sensors come into play, adding a third dimension to the point cloud and possibly a value for reflectivity. The output for both 2D and 3D systems consists of a set of detected objects, including bounding boxes and classification scores. Class assignment is treated as a classification problem, providing a confidence score for each of the C possible classes. Determining the bounding boxes is generally approached as a regression task where a 2D bounding box uses four parameters, while a 3D one requires seven [Zha+19].

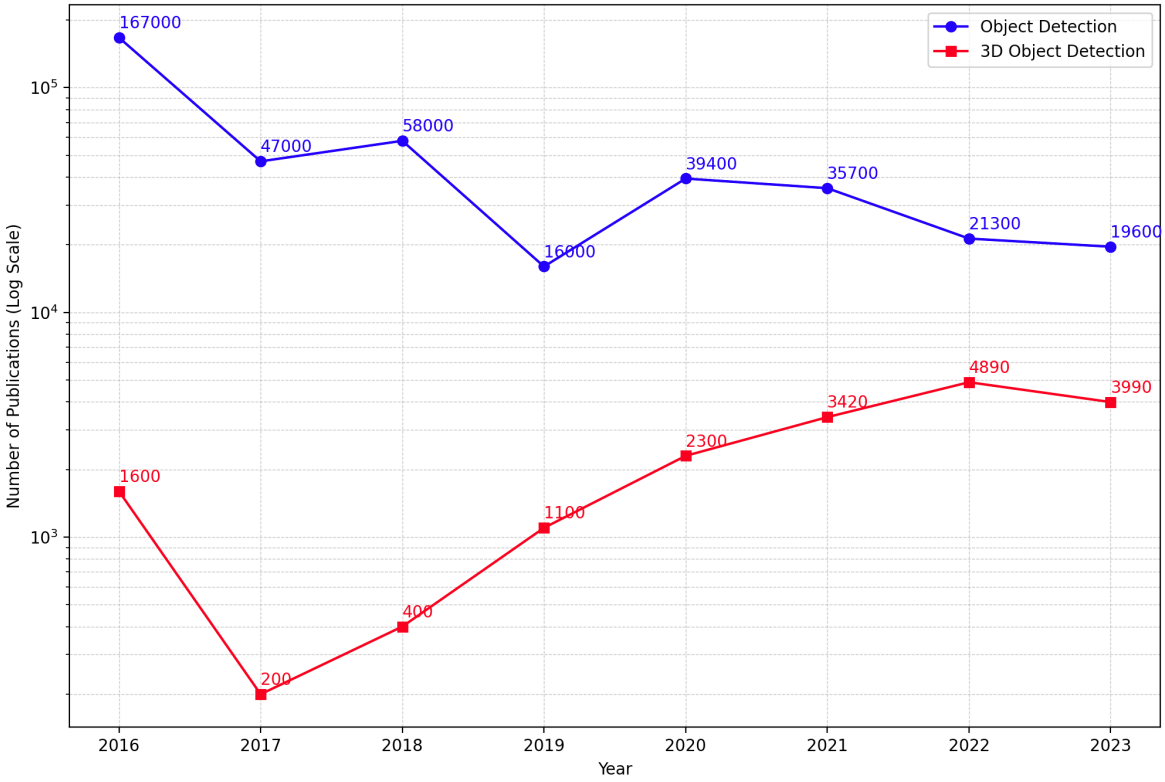


Figure 2.4: Publications on Google Scholar including 'Object Detection' or '3D Object Detection'. The figure shows data up until 21.09.2023.

In the development of object detection, there has been a shift from 2D to 3D techniques [Yan+22b]. Figure 2.4 displays the amount of papers focused on 2D versus 3D object detection. Since 2016, 3D-related publications have been increasing, while 2D ones are diminishing. This trend is driven by the richer information 3D detection offers, such as object orientation and size in a scene, which is crucial for areas like autonomous driving and virtual reality (VR) [Yan+22b]. Thanks to advancements in Graphics Processing Units (GPUs), handling the extra computational load for 3D data is now practical [Hua+21]. Lastly, interest in 3D perception is growing, not just in academic circles but also among private firms, which contribute significantly through publications, open-source code, or datasets [Sun+20].

A key component of academic research in object detection is the use of standardized dataset benchmarks like COCO [Lin+14] and Open Images [Kuz+20] for 2D detection, SUN RGB-D [SLX15] for 3D indoor detection, and KITTI [Gei+13] and Waymo Open Dataset [Sun+20] for 3D detection specifically for autonomous driving. These datasets feature a

set of inputs (images and/or point clouds) along with corresponding labels, such as ground-truth bounding boxes and class labels. This allows the training of learnable object detectors using the provided labels. Moreover, the availability of published benchmarks enables direct comparison between different model architectures.

2.3.1 Evaluation

Evaluating object detectors typically involves assessing the quality of the generated bounding boxes and class labels. Detected objects are categorized as true positives (TP) if they match a ground-truth bounding box, or as false positives (FP) if they don't. Ground-truth bounding boxes with no corresponding predicted box are termed false negatives (FN). This matching is often done using the Intersection over Union (IoU) metric. Equation 2.6 shows how IoU quantifies the overlap between two bounding boxes. A match between a predicted and a ground-truth bounding box is considered a TP if it surpasses a certain threshold, usually 0.5 or 0.7 [Gei+13]. Using the counts of FP, TP, and FN, one can calculate precision and recall as per equations 2.7 and 2.8.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{2.6}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2.7}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{2.8}$$

In object detection, there is a trade-off between precision and recall. Precision measures how many of the predicted objects are actual objects, while recall evaluates how many actual objects the model predicted. Object detectors often output a confidence score to indicate their certainty that a proposed object is genuine. Setting a high threshold usually results in high precision but lower recall, whereas a lower threshold tends to produce lower precision and higher recall. Rather than assessing object detectors at a single, arbitrary threshold, the metric of mean average precision (mAP) considers a range of thresholds. Figure 2.5 shows the precision-recall curve for three different object detectors at 20 different thresholds. The area under the curve is the mAP score. The mAP score is the most common metric for evaluating object detectors. Detector A in figure 2.5 achieves an mAP score of 0.34, detector B of 0.6, and detector C of 0.8. The closer the mAP value is to 1, the better the detector performs, meaning C has the best performance of the three.

2.3.2 Architectures

Broadly, 2D object detection architectures can be categorized into two major types: one-stage and two-stage detectors. One-stage detectors prioritize speed, making them ideal for real-time applications. On the other hand, two-stage detectors focus on achieving higher accuracy. This section will discuss the most common architectures for both types.

One-Stage Detectors

One-stage detectors are aiming for real-time performance while still achieving competitive performance. An early architecture was You Only Look Once (YOLO) [Red+16], which separated itself from the two-stage approach and processed the entire image in a single pass

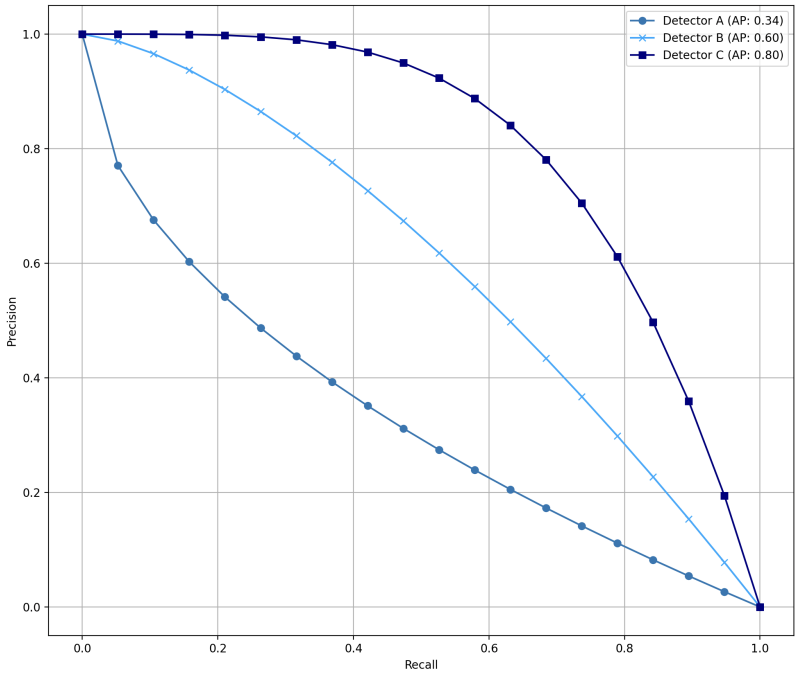


Figure 2.5: Recall-Precision curve for three detectors

to both localize and classify objects. This was achieved by dividing the image into a grid and predicting bounding boxes and class probabilities for each grid cell. YOLO was able to achieve real-time performance. The success of YOLO was followed by the developments of YOLOv2 [RF17] and YOLOv3 [RF18], each bringing refinements in terms of accuracy and speed. Another popular one-stage detector is Single Shot Detector (SSD) [Liu+16]. This architecture utilizes a similar approach to YOLO, dividing the image into a grid and predicting bounding boxes and class probabilities for each grid cell. However, instead of predicting the bounding box coordinates directly, SSD predicts offsets to default bounding boxes of different aspect ratios and scales. This allows SSD to handle objects of different sizes and aspect ratios more effectively. SSD also uses a feature pyramid network to extract features from different scales, allowing it to detect objects of different sizes. Figure 2.6 shows the architecture of YOLO and SSD.

Two-Stage Detectors

The advancements of two-stage detectors represent a continuous effort to meet the challenges posed by object detection in computer vision. It began with R-CNN [Gir+14], which introduces the principle of having two separate steps in the detection pipeline. In the first step, region proposals are generated via selective search. In the second step, the proposed regions are classified with a CNN. Figure 2.7 shows the pipeline of R-CNN. This was a significant move towards precise object detection, but R-CNN had a slow processing speed due to the fact that the CNN had to be applied to each proposed region separately. Identifying the inefficiency in R-CNN, Fast R-CNN [Gir15] was designed to address this by introducing a technique known as region of interest (RoI) pooling. This allowed sharing the computation over the whole image, significantly reducing the time needed for processing an image. However, Fast R-CNN was still using the selective search, which continued to be a bottleneck in terms of speed. The Faster R-CNN [Ren+15] architecture was introduced to solve this issue. Here, a Region Proposal Network (RPN) was integrated to replace the selective

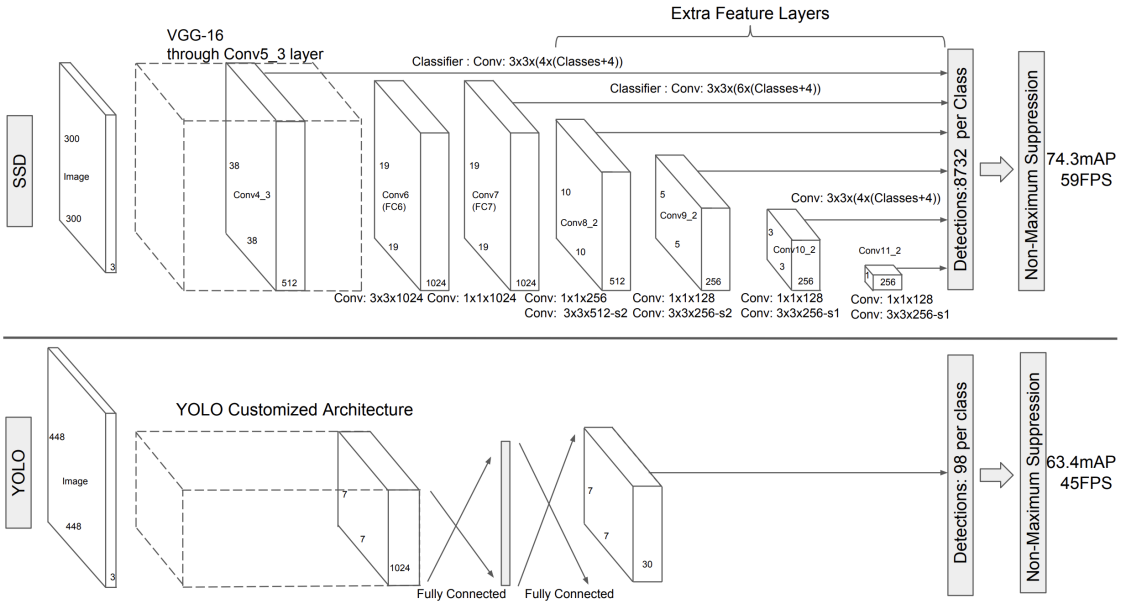


Figure 2.6: YOLO and SSD architecture, source: [Liu+16]

search for generating region proposals. The RPN and the detection network share the convolutional features, accelerating the processing. This was a major step towards real-time object detection.

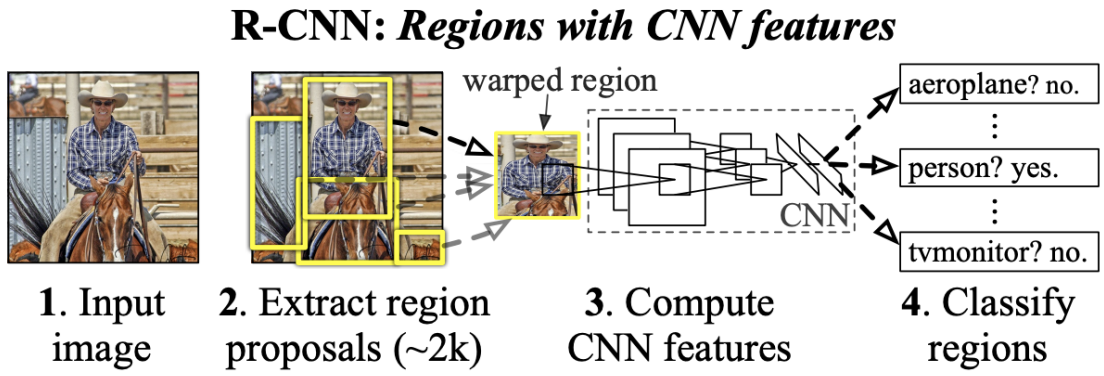


Figure 2.7: R-CNN pipeline, source: [Gir+14]

Chapter 3

Related Work

This chapter discusses related research that provides a foundation for this work. The aim is to give the reader an idea of how this work is embedded into a broader research field and how other work is applying similar ideas. First, the field of 3D object detection is introduced. Then, this chapter discusses the refinement step of two-stage 3D object detectors. Finally, other papers that are concerned with modeling object relations are presented.

3.1 3D Object Detection

As chapter 2.3 mentioned, 3D object detection has seen an increase in interest lately because it provides richer information about the detected objects. It was also already mentioned that 3D object detection aims to find a set of 3D bounding boxes encoded with seven values as well as class probabilities encoded with one value per class. Similarly to 2D object detectors, 3D detectors can be differentiated between one-stage and two-stage architectures with the same trade-offs between precision and efficiency. Examples of one-stage detectors include YOLO3D [Ali+18] or PointPillars [Lan+19], which perform classification and bounding box regression in a single shot. On the other hand, two-stage detectors like Part-A2 [Shi+19] or PVRCNN [Shi+23b] have a separate refinement stage. However, not all concepts from 2D object detection can be applied to 3D perception, as this might lead to inferior performance. The main challenge of 3D data is its sparsity and irregularity. It is an open question which model architecture is best capable of extracting discriminative features [Shi+19]. In chapter 2.1 this work already discussed sensors being used for autonomous driving. This chapter will focus on architectures for LiDAR 3D object detection, however, some of the mentioned architectures are also capable of processing data from different sensors.

3.1.1 Point-Based Approaches

Point-based approaches are essentially adapted to work with the unstructured data and unordered nature of point clouds, which are a collection of points in a three-dimensional coordinate system [Yan+20]. A first example of this kind of approach was PointNet [Qi+17a], which demonstrated that processing raw point clouds directly was feasible. Following, PointNet++ [Qi+17b] refined the concept, introducing a hierarchical neural network to capture fine-grained structure, alongside global context. These architectures work by extracting features from individual points and their neighborhood, progressively obtaining a global feature descriptor that encapsulates information about the entire point cloud. The key to point-based methods is their ability to keep information about the initial point cloud, making them capable of obtaining precise detections [Yan+20]. A shortcoming of point-based methods is the

computational intensity required to process many points, alongside managing the irregularity of point clouds [Yan+20]. Figure 3.1 shows the architecture of PointNet. One can see that each point is processed individually up until the max pooling layer which forms the global feature vector. Initially processing points individually is typical for point-based approaches.

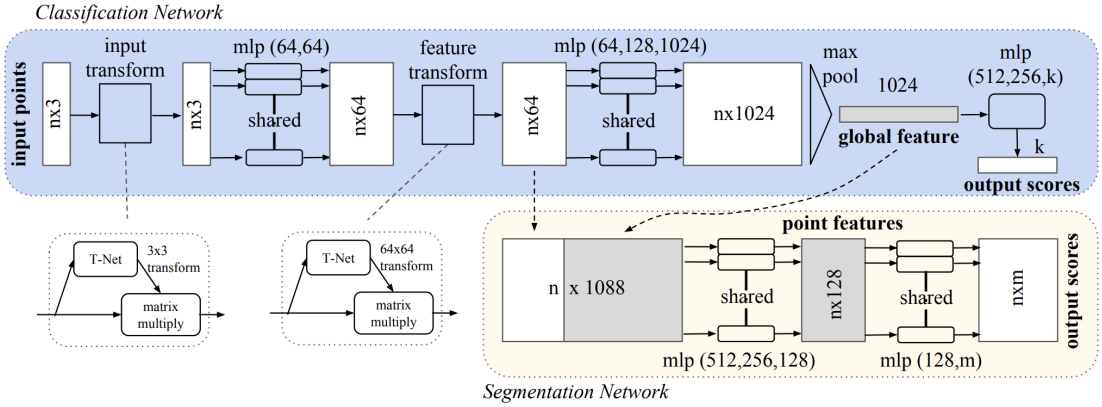


Figure 3.1: Architecture of PointNet, source: [Qi+17a]

3.1.2 Voxel-Based Approaches

Voxel-based approaches stand in contrast to point-based approaches as they aim to convert the unstructured nature of point clouds into a structured format. To achieve this, the point cloud is divided into equally spaced 3D voxels. Points located inside a voxel are used by a voxel feature encoding (VFE) layer to obtain a feature vector per voxel. The VFE layer aims to discretize the point cloud while keeping low-level information about it [ZT18]. By aggregating points within each voxel, these methods achieve a reduction in data complexity which is important for computational efficiency. Additionally, converting an unstructured point cloud into a structured voxel representation makes it possible to apply 3D convolutions to the data. The highly successful 2D convolutions can, therefore, easily be extended to 3D perception [He+21]. Section 2.2.1 explained the concept of sparse convolutions which were an adaptation to deal with the sparsity of 3D space. VoxelNet [ZT18] was an early architecture using several layers of 3D convolutions and achieving competitive results. While voxel-based methods lower the computational strain by structuring the point cloud data, they often neglect fine-grained information. The process of discretization can sometimes oversimplify the data, losing nuanced details crucial for precise detection [Den+21]. Figure 3.2 shows the architecture of VoxelNet. The first step in this architecture is taking the unstructured data of the point cloud and converting it into discrete voxels. These voxes are subsequently processed using convolutions.

3.1.3 Pillar-Based Approaches

Pillar-based approaches aim to find a middle ground between the unstructured nature of point clouds in point-based methods and the structured representations in voxel-based methods. This is done by dividing the point cloud into vertical columns, or 'pillars', each containing points along the z-axis [Lan+19]. PointPillars [Lan+19] was a pioneering work using this approach. Here, each pillar is processed to obtain a feature vector, usually by using PointNet-like architectures. These features are arranged into a pseudo-image to which 2D

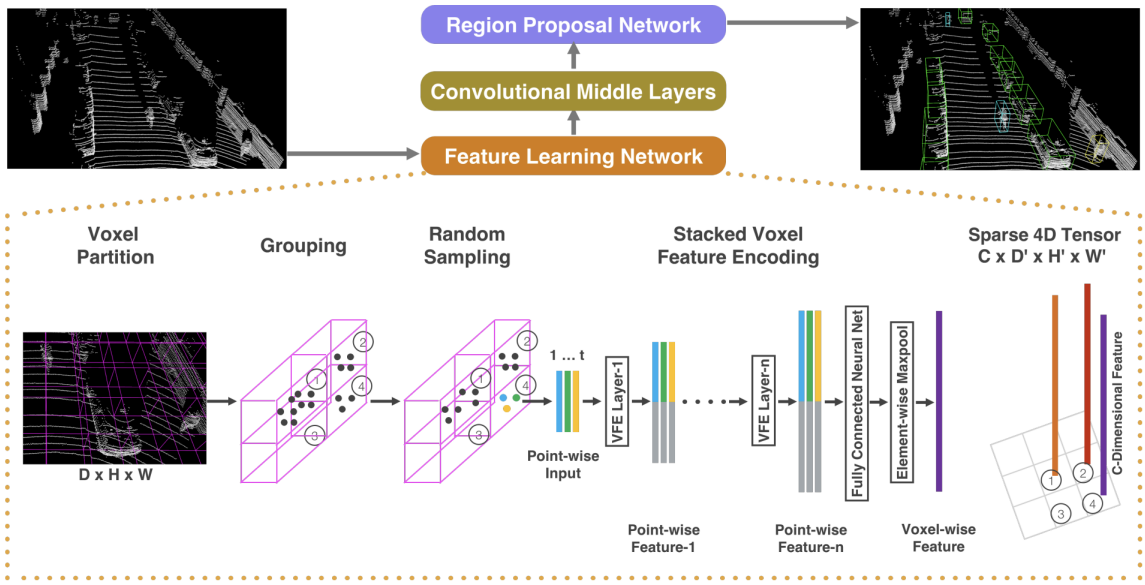


Figure 3.2: Architecture of VoxelNet, source: [ZT18]

convolutions can be applied. The pseudo-image format allows the model to capture both local features within each pillar and global contextual information. Pillar-based methods allow for efficient computation while still retaining fine-grained information [LLY23]. Specifically in applications where the vertical dimension does not contain much critical information, such as autonomous driving, pillar-based methods offer a good trade-off between efficiency and precision. Figure 3.3 shows the architectuer of PointPillars. One can see how the pseudo-image is formed by arranging the pillar features into a 2D grid. This grid is then processed by classical 2D convolutions.

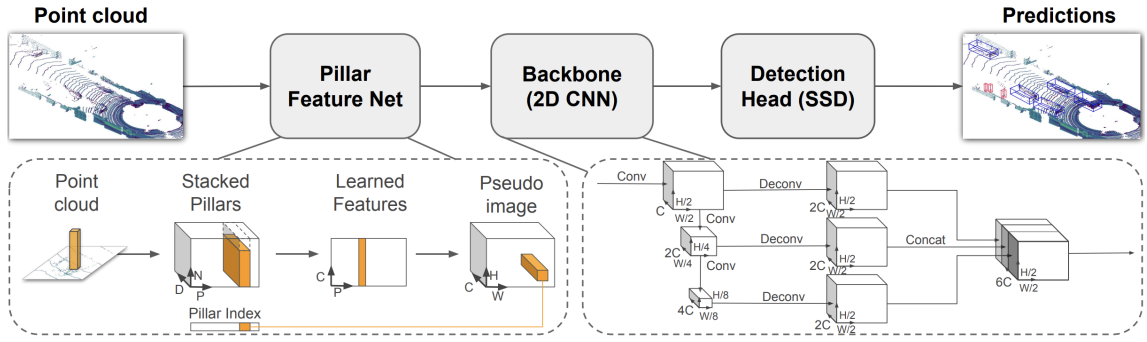


Figure 3.3: Architecture of PointPillars, source: [Lan+19]

3.1.4 Point-Voxel-Based Approaches

Point-voxel-based approaches aim to combine the strengths of both point-based and voxel-based approaches for 3D object detection. The goal is to handle the sparsity and irregularity of point clouds while maintaining computational efficiency [NLH21]. Similarly to voxel-based methods, in this approach, the data is initially segmented into voxels. However, instead of simply aggregating the points in each voxel to obtain a feature vector per voxel, point-based methods like PointNet are applied to points within each voxel. Processing the points in

this way allows for more nuanced feature extraction, maintaining fine-grained information [Li+21a]. The voxelized structure allows the use of 3D convolutions, just like in pure voxel-based approaches. Examples of architectures that use point-voxel-based approaches include PVF-Net [CZ20] and Hvpr [NLH21]. These architectures demonstrate improved performance over purely point-based or voxel-based methods, particularly in environments with varying point cloud density and complexity [NLH21]. Figure 3.4 shows the architecture of HVPR [NLH21]. This model makes use of point-based methods as well as voxel-based methods to predict the bounding boxes of objects.

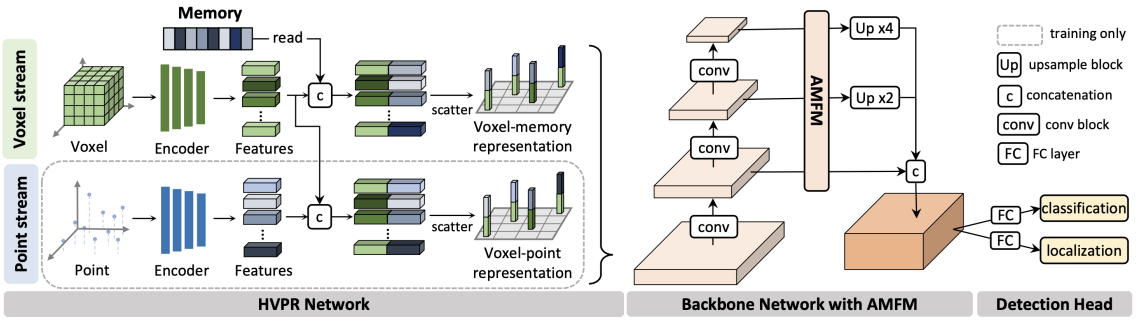


Figure 3.4: Architecture of HVPR, source: [NLH21]

3.2 Object Refinement

Similar to two-stage 2D object detectors, the detection pipeline for two-stage 3D architectures consists of a first stage identifying proposals, i.e. object candidates or RoI, and a second stage that predicts refinements relative to these proposals. Compared to one-stage detectors they usually achieve higher accuracy. This is achieved with the selective processing approach, i.e. focusing the computational effort of the second stage on regions that were already identified as interesting. This more efficient way of processing detections allows the model to incorporate more context which yields better detections [DZW20]. Figure 3.5 shows the typical pipeline of a two-stage 3D detector. Typically, during the first stage, point, voxel, or pillar-based architectures are used to obtain a feature map. Using this feature map, the location of a set of object proposals including their 3D bounding boxes is predicted. Each proposal is forwarded to the second stage, where RoI pooling is used to extract discriminative features from the respective locations in the raw point cloud, from the 3D backbone features, or potentially other sources. Subsequently, all features are processed and used by the detection head to predict an offset to the bounding box of the proposal. It is important to note that the predicted offset is relative to the proposal obtained from the first stage. This means that the second stage does not predict the absolute position of the bounding box. The final prediction can be obtained by adding the predicted offset to the proposal. There are differences in how exactly the refinement step is implemented, this chapter aims to give an overview of which methods are being used, specifically in the domain of autonomous driving. To achieve this, a literature review was conducted identifying recent papers that propose novel methods for the refinement step. Then, these papers were categorized based on the methods they used. An overview of this can be found in table A.1 and a visualization in figure 3.6. The following sections will discuss the different methods in more detail.

Once a set of features for each proposal is obtained, an important step in the refinement stage is the further processing of it. Initially, each feature vector individually describes just

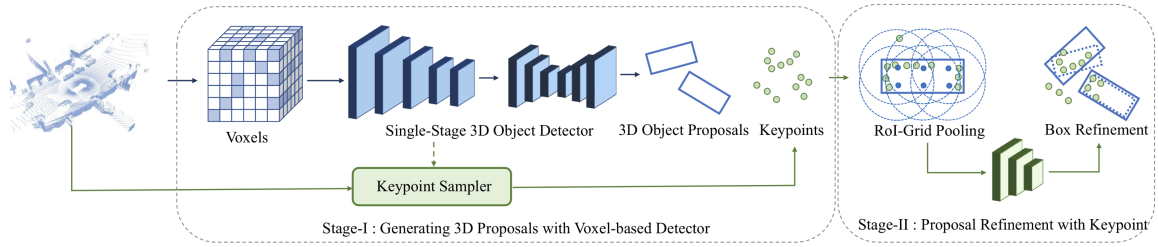


Figure 3.5: Typical architecture of a two-stage 3D detector, source: [Mao+23]

its neighborhood. It is crucial that the model gains a representation of the whole proposal to subsequently be able to precisely predict its bounding box. The authors aim to obtain this global context using different methods. Inspired by the transformer [Vas+17], the work of [She+21; DHF22; MZ22] utilizes an encoder-decoder architecture to obtain global representations of proposals. [She+21] uses channel-wise context aggregation for point features within each proposal consisting of a decoder and encoder. The channel-wise reweighting, as an update to the standard transformer architecture, emphasizes different features that are captured per channel. [DHF22] uses Vector Attention that similarly assigns different weights to different channels within the features. Several RoI encoders transform feature maps produced by the backbone into a global representation of the proposal. Finally, [MZ22] uses a traditional transformer architecture. The refinement stage in [Ma+; Yan+22a] makes use of graph structures to model the context between the features obtained in the pooling phase. [Yan+22a] builds local graphs to mine point relations through iterative message passing. This helps to alleviate occluded regions in the proposals because it models contextual information through message passing. This is achieved by aggregating messages from neighbors in a k-NN graph. [Yan+22a] uses graphs at two different scales. First, a local point-graph pooling module computes point-to-point correlations in a local region and aggregates features from neighboring points. Then a long-range point-aware module aims to integrate the local features with global information.

The first stage of a two-stage detector predicts RoI for potential objects that are to be detected. The second stage takes these regions and aims to predict exact bounding boxes. To achieve this, features are pooled from the respective locations. The architectures of [Ren+23; Ai+23; Gua+22; HZ22; HJR23; Shi+23b] innovate with the idea of pooling features from different sources at different scales. This yields a more robust detection pipeline that can handle objects at different scales. [Ren+23] pools raw points and introduces a proposal-aware fusion module that integrates positional and geometrical information through the positional relationship to their respective proposal location. Using raw points has the advantage of not losing any information which allows for precise detections. Another work uses the features obtained from a transformer backbone, the raw point cloud, and features used to predict the 3D bounding box in the first stage in the pooling step [Ai+23]. M3DETR [Gua+22] similarly uses its transformer and convolutional backbone features at different scales to refine the proposals. Besides raw points, SC-RCNN [Wan+23] and PVRCNN [Shi+23b] use voxel and bird-eye view (BEV) features pooled from the respective locations of the proposals. Additionally, SC-RCNN uses a pyramid candidate box augmentation algorithm that uses a pyramidal paradigm, adaptively capturing more multi-scale features outside the candidate proposals while maintaining fine-grained geometric details. Pyramid R-CNN [HZ22] is a similar work that proposes a voxel feature pyramid, a method that aggregates features from the convolutional backbone based on the size and sparsity of the respective proposal. If the proposal is sparse, more context is aggregated by pooling from lower convolutional layers, effectively increasing the receptive field of the proposal and including more context about

the surroundings.

Typically, when raw points or features from the backbone are pooled to refine a proposal, not all points are used; instead, a subset is sampled. Some models [Gao+23a; Yan+22a; MZ22; HKW22] come up with innovative sampling strategies. [Gao+23a] aims to sample points from the outer shape of the object proposal. The Graph R-CNN model [Yan+22a] proposes the patch search to quickly search points in a local region for each proposal. Then dynamic farthest voxel sampling is applied to evenly sample points. This sampling strategy preserves the object structure better than other works, where the sampling strategy is not adapted to the sparsity of some proposals. In [HKW22] voxel features are localized via voxel point centroids and then aggregated through density-aware RoI grid pooling.

Due to the nature of 3D data, some proposals will suffer from sparsity and occlusions. The work of [Yan+22a; XZN22; HKW22] follows the idea of predicting the complete shape of objects based on sparse proposals in the refinement step. [Yan+22a] fuses semantic features from the respective image region to the proposal features to deal with LiDAR occlusions. This is achieved by decorating local graphs with image features by bilinear interpolation. [XZN22] uses shape priors to estimate the occupancy of complete object shapes in regions impacted by occlusions. SIENet [Li+21b] also takes occluded proposals and predicts dense shapes. This is done by taking the sparse points describing the object and completing its shape by adding virtual points until a fixed number of points inside the proposal is reached. The completed proposal is forwarded to predict the bounding box refinement. Lastly, [SZL22] deals with the problem of incomplete proposals when the proposal is not correctly aligned with the object. The simple solution is to add a set of shifted proposals around the initial one. In that way, at least one accurately localized proposal can be guaranteed for the refinement stage.

The concept of receptive fields can be applied to proposal refinement. Based on the features pooled for each proposal, the receptive field describes the radius around the proposal from which information is used to refine the bounding box. The larger the receptive field, the more context about the scene is included to predict the object's position. [Wan+23; Gao+23b] introduce specific methods to increase the receptive field of the proposals. Both architectures generate a series of grid points around proposals and explicitly extract voxel features beyond the proposal boundary.

PartA2 [Shi+20] uses an intra-proposal segmentation method to improve the expressiveness of proposal features. For this, accurate intra-object parts are derived from the 3D ground truth boxes. With these labels, the model is trained to predict the object part to which each point belongs. This segmentation information is appended to the initial proposal features and all features are used to predict the bounding box refinement.

Finally, Voxel-RCNN [Den+21] is focusing its refinement step on using only voxel features. It can achieve high accuracy while its computational cost is much lower compared to other methods.

3.3 Object Relation

The previous chapter explained the typical mechanisms of the refinement stage for a two-stage 3D object detector. It is important to note that every object proposal obtained by the first stage is individually forwarded to the second stage. For a given proposal the refinement stage usually pools features from different sources. The radius from which features are pooled around the proposals defines the receptive field of the proposal. The receptive field of a proposal includes all information that is being used to predict its bounding box. Building on this insight, authors realized that current detectors lack sufficient context for making accurate predictions. A naive solution to simply expand the receptive field, as done by [Gao+23b;

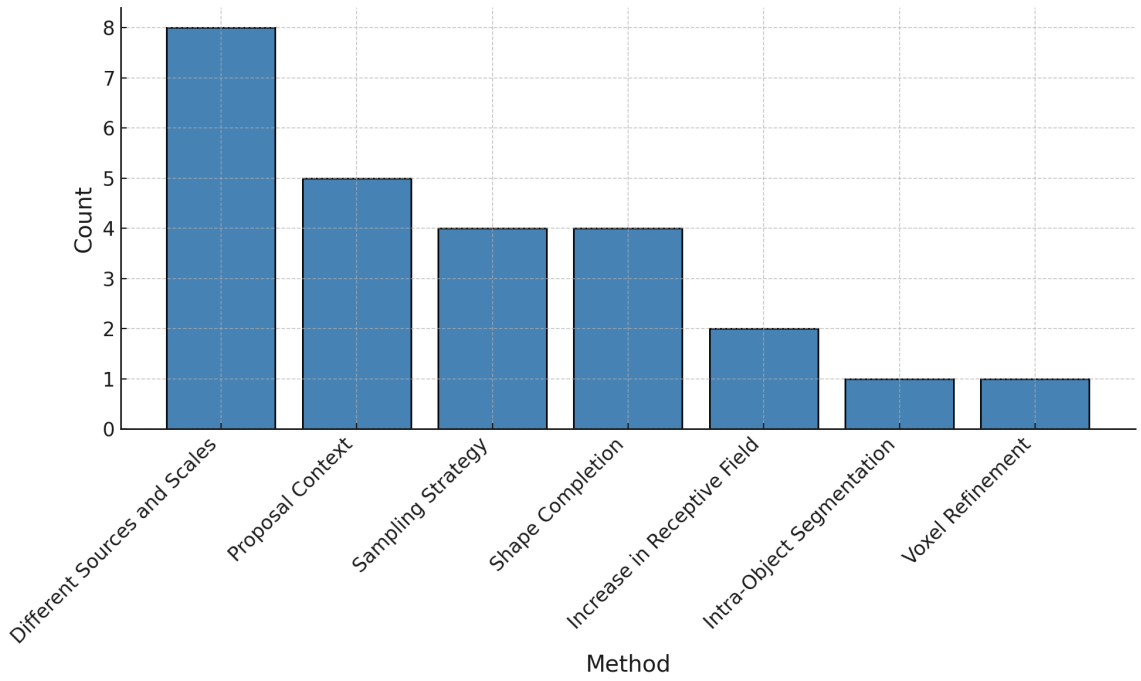


Figure 3.6: Refinement methods used in the literature

Wan+23], increases the context, however, it is not computationally feasible to infuse proposals with global information about the scene. A more efficient way to incorporate context and global information is by facilitating the sharing of proposal features. This tweak means that the set of proposals forwarded to the second stage are no longer processed in isolation but are instead related to each other, potentially enhancing the accuracy and robustness of the object detection process. This chapter provides an overview of object relation methods and the motivation behind them in indoor and outdoor detection.

3.3.1 Indoor

Object relation is widely used for indoor 3D object detection [Xie+20; Fen+20; Li+20; Lan+21; Lan+22]. MLCVNet [Xie+20] identifies the lack of context when object proposals are processed individually by the second stage. It argues that representations of scanned point sets can be ambiguous when looked at individually. In some instances, indoor depth scans are so occluded that context could even play a more important role in recognizing objects than the point cloud data itself. It, therefore, introduces modules to include context at three levels: patch-level, object-level, and global-level. Specifically, for the object relation context it uses a self-attention module before the features are used to classify and locate the proposed object. Figure 3.7 depicts a point cloud scan of an indoor scene, showing a partially obscured chair. Infusing the chair's proposal feature with the global context - that the point cloud is a scan of a dining room - increases the detector's confidence in identifying it as a chair. Finally, by exchanging features of nearby objects, the detector can hypothesize that the chair could be part of a set encircling a table. The orientation of the table and adjacent chairs might further refine the predicted orientation of the chair's bounding box.

The work [Fen+20; Li+20] makes use of the intuition that close objects typically influence each other more than objects that are further away. Based on the spatial distance between proposals, [Fen+20] defines a threshold to cut any object relations connecting objects

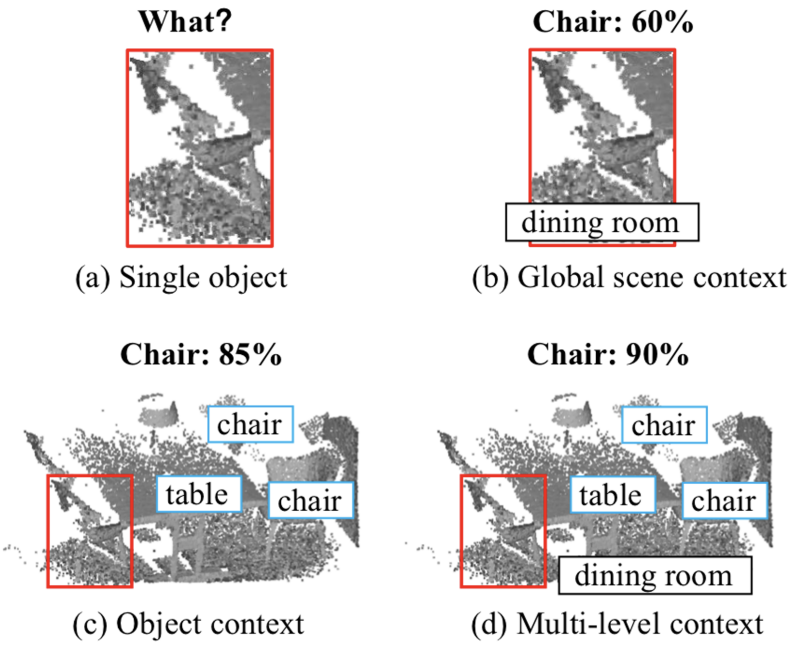


Figure 3.7: Object Relation in an indoor scene, source: [Xie+20]

that are far away from each other. Similarly, [Li+20] only considers the n nearest neighboring proposals for each proposal to leverage their relation features. ARM3D [Lan+22] and 3DRM [Lan+21] introduce object relation on a pair-wise level. ARM3D [Lan+22] encompasses an object-aware relation reasoning to extract pair-wise relation contexts among qualified proposals. 3DRM [Lan+21] aims to predict the type of relation between objects. There are four types of relation: group, same as, support and hang on. This information is infused into the feature of the proposal to subsequently predict its class and bounding box.

3.3.2 Outdoor

The concept of relating features of proposals is less explored for outdoor 3D object detection in the domain of autonomous driving. Object DGCNN [WS21] models 3D object detection as message passing on a dynamic graph. To detect objects, this work uses queries that attend to certain areas in the scene that potentially contain objects. To relate these queries to each other a graph is constructed connecting the nearest neighbors. Here, distance is not measured in $\mathbb{R}^3$ space but in feature space, therefore, objects with similar features are connected. Each query is connected to its 16 nearest neighbors. A feature vector is learned per edge and then aggregated back to the vertices to produce a new set of object queries that now contain the object relation information.

BADet [QLL22b] represents each proposal as a node in a graph. Within a given cut-off threshold proposals are connected, and their features are related to each other. Unlike other approaches, it focuses on relating overlapping proposals for the same object, rather than linking proposals for different objects. This work identifies that challenges arise when the proposals are not perfectly aligned with the respective objects. Figure 3.8 displays clusters of object proposals surrounding actual objects. Subfigure (b) illustrates the uncommon scenario where proposals already align well with the object. In contrast, subfigures (a), (c), and (d) represent the more typical case where some proposals are better aligned than others. Since individual proposals often lack sufficient information for precise object localization,

this work introduces a boundary-aware GNN. Here, sufficiently close proposals can exchange their features. The final object prediction is essentially a consensus derived from all nearby proposals.

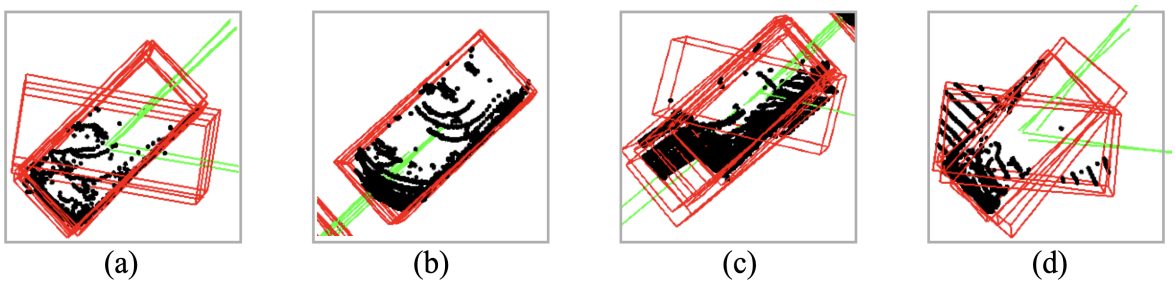


Figure 3.8: Proposal cluster and ground truth alignment, source: [QLL22b]

Ret3D [Wu+22a] in contrast tries to exploit relations between proposals for different objects. It claims that locations and geometric features of 3D objects can provide rich context and structural information for scene understanding and precise object detection. It uses CenterPoint's [YZK21] first stage to obtain object proposals. In the refinement stage, the basic features of each proposal are concatenated with map-view features pooled from the backbone at the respective locations. The feature vectors for each proposal are connected in a graph and related to each other via message passing. Then they are forwarded to the heads in which refined locations and labels are predicted.

All work presented on object relations until now modeled the relationship between proposals with learnable methods and trained a detection model in an end-to-end way. GACE [Sch+23] follows a different approach. It models the relationships between objects without any knowledge about the internals of the object detection pipeline. To achieve this, detections from some baseline detectors are obtained and a model is trained to predict whether a detection is a false or true positive. In the first step, information about a single detection is obtained. Equation 3.1 shows what information is used: b bounding box values, α angle of the detection, $||c||$ distance to the detection, and χ values containing information about the subset of the point cloud that is located inside the bounding box. These features are embedded by an MLP H_i to obtain the embedding f^I.

$$f^I = H_i\left([b, \alpha, ||c||, \chi_b, \chi_b^{\text{mean}}, \chi_b^{\text{std}}, \chi_b^{\text{min}}, \chi_b^{\text{max}}]^T\right) \tag{3.1}$$

Similar information is captured from surrounding detections, embeddings are obtained from an MLP, and a pooling operation is applied to obtain the feature vector f^C containing information about all detections in the near environment. Finally, f^I and f^C are combined to predict a confidence score for the object being a true positive. Figure 3.9 shows an example of how this confidence score is computed for the pedestrian with the orange bounding box which is related to two other pedestrians.

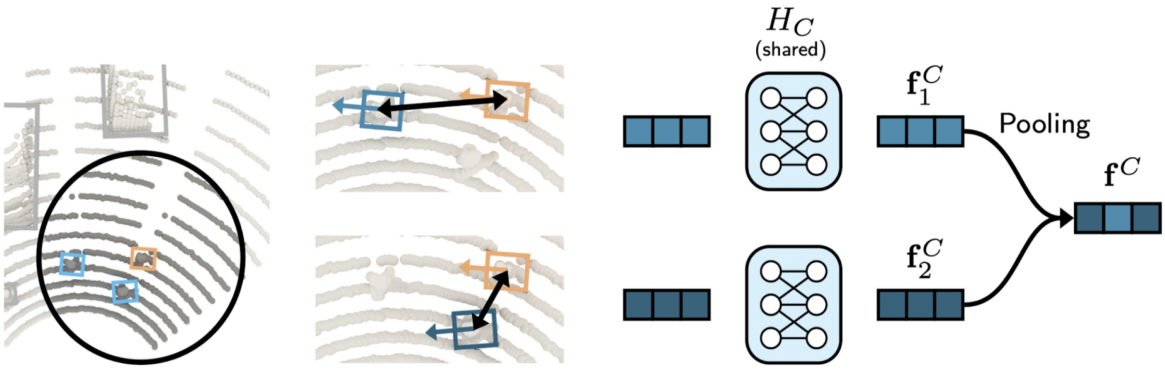


Figure 3.9: Object relation for pedestrians, source: [Sch+23]

Chapter 4

Methodology

This work aims to improve 3D object detection for autonomous driving. Specifically, it will focus on perception using only LiDAR sensors. LiDAR is capable of directly sensing depth and is already widely used in autonomous driving. The idea is to improve LiDAR-based detection by modeling object relations in a learnable way. As earlier discussed, the traditional refinement stage of two-stage detectors processes proposals individually. This work introduces an *Object Relation Module*, which relates proposals in a scene with each other. This module can be integrated into any two-stage detector. The hypothesis of doing this is that object relations can be exploited. Explicitly modeling object relations should improve the performance of detection pipelines. The *Object Relation Module* is integrated into the already existing code base of OpenPCDet. The following sections will first motivate the concept of modeling object relations. Then, it will justify this work by highlighting shortcomings in the related work. It will present the architectures of the baselines to which the *Object Relation Module* is applied. Finally, it will present the implementation of the *Object Relation Module*.

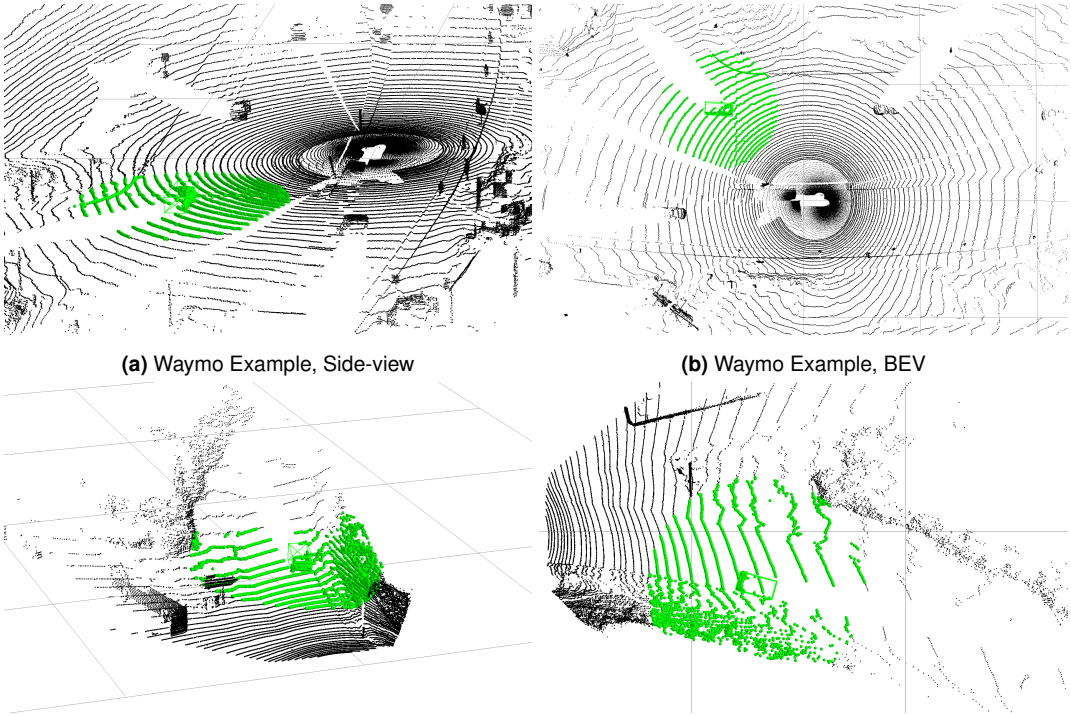
4.1 Motivation for Object Relation

Explicitly modeling object relations in two-stage object detectors has various motivations. Section 3.3 already discussed some of these motivations that other works have referred to. However, the motivations previously mentioned are not exhaustive. Therefore, this section will provide the reader with a clear overview.

Increasing the Receptive Field

The receptive field is the radius around a proposal that is being used to predict the bounding box offset. Only points of the input point cloud that lie inside this radius are being used in the second stage for each given proposal. The size of this radius is a product of the architectural design of the model. It depends on the receptive field of the convolutional backbone and on how the pooling operations are implemented. Typically, the receptive field is large enough to capture the whole object and some of its surroundings, however, it will not capture information about the scene at a global level. Context about the scene at a global level or at an extended regional level might be valuable information to make precise predictions that current object detection pipelines are missing. The direction of the street can function as a prior to the model when predicting the angle of vehicles. Knowing whether a scene is in the city or the countryside can provide the model with information about what patterns of objects to expect. Explicitly modeling object relation increases the receptive field of a proposal to a shared receptive field with all the proposals it is connected to. It can be

seen as an efficient way to increase the receptive field and incorporate context into each proposal. Figure 4.1 shows the receptive field of the PVRCNN model for the Waymo and KITTI datasets. Looking at this figure, it becomes clear that the second stage fails to capture information about the scene on a global level.



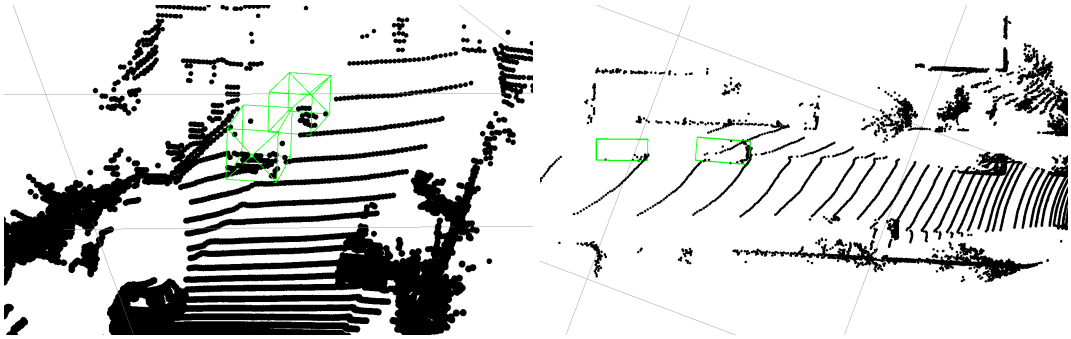
(a) Waymo Example, Side-view **(b)** Waymo Example, BEV

(c) KITTI Example, Side-view **(d)** KITTI Example, BEV

Figure 4.1: Receptive field of the PVRCNN detector for a vehicle proposal in the Waymo and KITTI dataset. Based on the different configurations used in the datasets, the receptive field can be computed as 11.95m for Waymo and 10.175m for KITTI. For details see A.4. All points that are inside the receptive field of the proposal are colored in green.

Detecting occluded Objects

For safe driving, robust detection of all occluded objects is important, making object occlusion a major challenge in autonomous driving. Occluded objects often have point clouds that differ significantly from those of fully visible objects, sometimes appearing so sparse that they are indistinguishable from noise without context. Context can for example be knowing whether an object is occluded. When object proposals are processed in isolation, they can't determine their own occlusion status. However, modeling object relations allows this context to be integrated into the feature set of each proposal. Consider figure 4.2, which illustrates an occluded vehicle and the vehicle causing the occlusion. If the occluded object is evaluated in isolation, it is hard to say if it is noise or an actual object. But if the model can understand the relationship between both proposals, it can inform the occluded object about its occlusion status. This is achieved by making both proposals aware of their relative positions to the observer. If one is behind the other, it is likely to be occluded. A model can account for this by predicting a vehicle more confidently, even if its point cloud looks different from a usual vehicle.

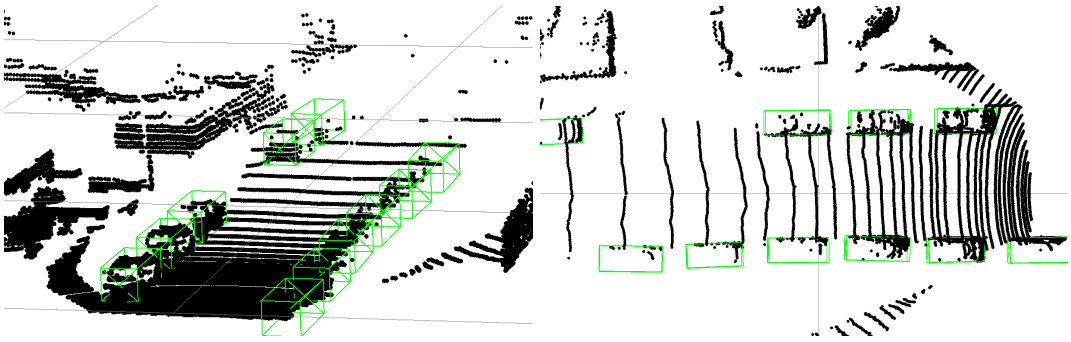


(a) Side-view **(b)** BEV

Figure 4.2: A vehicle occluding another vehicle leading to a highly sparse point cloud for the occluded vehicle. The figure shows two vehicles indicated with green bounding boxes.

Exploiting Patterns

Objects in everyday traffic scenes often occur in patterns. Vehicles might be parked orderly in a parking lot or one after the other on the side of the road. Modeling object relations allows the model to exploit these patterns which serve as a prior. Additionally, specific classes of objects don't usually come alone but often appear in pairs or groups. The detection of a traffic light makes the presence of another traffic light more likely, and a detection model should use this information. On the other hand, the object detector might detect a set of vehicles that are unorderly located in an area with highly different angles. As such anti-patterns usually do not occur in traffic scenes, object relation again serves as a prior reducing the likelihood of such scenes. Figure 4.3 shows a scene with vehicles parked orderly in a line at the side of the road. The BEV shows how each object's point cloud gets sparser the further back it is in the line of parked vehicles. Without exploiting this scene's pattern, it might not be possible to detect all vehicles precisely.



(a) Side-view **(b)** BEV

Figure 4.3: Scene with Objects forming a Pattern that can be exploited. The figure shows a set of vehicles indicated with green bounding boxes.

Proposal Consensus

All previous motivations were about modeling the relations between object proposals that belong to different objects. The first stage of detection pipelines usually leads to clusters of proposals in the vicinity of potential objects. The refinement stage will then try to align the proposed bounding boxes with the actual objects. If subsequently more than one bounding box belongs to one object, non-maximum-suppression (NMS) will be used to filter out proposals with low confidence values. Different proposals might, therefore, describe the same

object. However, these proposals might have different views on the object and capture different data from the point cloud. When features between these proposals are shared, they can learn about their different views on the same object. Object relations might lead to better detections, even if relations are only being shared between proposals of the same object. Finally, the predicted bounding box of one proposal can be understood as a consensus prediction of all the proposals belonging to the same object. Figure 4.4 shows such proposal clusters around potential objects. We can see that proposals of the same object are not identical and offer different views on the object.

Figure 4.4: Cluster of proposals for objects. Each proposal is visualized with a green bounding box.

4.2 Analysis of Related Work

The last section gave a clear motivation to explicitly model object relations in object detectors. Some of these motivations were already identified by the research field and methods were introduced to improve object detectors with object relations as seen in 3.3. Given the efforts that were already made, this section provides the reader with a justification for this work. To do so, it will highlight shortcomings in the general field as well as shortcomings in the specific related works presented earlier.

While modeling object relations for indoor datasets has been studied in-depth with many publications in the last few years, applying these relations to outdoor scenes is much more uncommon. Object relations in indoor scenes are important because objects like chairs relate in a specific way to other chairs or a table. However, the same kind of relationships can be found in traffic scenes, e.g. between vehicles parked in a parking lot. Modeling object relations for outdoor scenes can, therefore, be identified as an under-studied area of 3D object detection. Additionally, existing works on object relations for outdoor scenes are often not applied to state-of-the-art detectors and, therefore, don't lead to the best results. Ret3D [Wu+22a] applies object relations to CenterPoint [YZK21] and BADet [QLL22b] together with its custom backbone. Object relations haven't been applied to the top-performing models yet. Moreover, in many works, a module to relate objects is combined with other innovations, e.g. an update to the backbone of the model. This makes it difficult to study the effect of object relations in isolation. Finally, object relations have been identified to capture context about a scene and lead to an increase in the receptive field of the proposals. However, publications often fail to report what the receptive field of the baseline architecture is. Explicitly reporting the receptive field can often make it obvious that more context is needed to robustly detect all

objects.

BADet [QLL22b] focuses on capturing relations between proposals belonging to the same object. Therefore, it only captures the motivation of *Proposal Consensus* from the previous section. It uses a radius graph to relate all proposals that are sufficiently close to each other. Here, the threshold of this radius graph is chosen to be a small value that effectively only connects proposal clusters that belong to the same object. Rather than modeling object relations in an actual sense, this work exploits a shortcoming of two-stage detectors.

GACE [Sch+23] can be seen as a different type of work. Although it achieves impressive results on a variety of baseline architectures by modeling object relations, it is not end-to-end learned. While training such an extra model on top of a detection pipeline improves the detection results, it is inherently non-optimal. Important to note is that most of the method's improvements are due to capturing instance-specific properties about the detected objects rather than by modeling object relations. Additionally, each object gets processed individually. While each object uses information about surrounding objects, it does not use the result of these objects made by the model (whether they are true or false positives). This means that if an object gets classified as a false positive, the surrounding object will not be aware of this. If a potential object is in the path from the observer to another object, the object relation model will expect to find an occluded object further back. However, if later the model classifies the object in-between as a false positive, this would change the expected occlusion status of the potential second object. Sequentially classifying objects as true and false negatives is, therefore, also not optimal.

4.3 Baselines

The *Object Relation Module* is not standalone; it must be integrated into a base detector. It can be applied to any two-stage 3D detector. This chapter will introduce four baselines that are used in this work: PVRCNN, PVRCNN++, PartA2, and Voxel-RCNN.

4.3.1 PVRCNN

PVRCNN was identified in section 3.2 as a two-stage detector with a refinement stage pooling from different sources: raw points, voxel-features, and BEV-features. The main idea of this model is to combine the strengths of voxel-based and point-based approaches. This section should give the reader an in-depth explanation of its architecture.

To apply 3D convolutions, the first step is to divide the input points P into small voxels with the spatial resolution of $L \times W \times H$. Choosing L, W, and H is a hyperparameter decision and should be adapted according to the dataset. The model utilizes a series of 3D sparse convolutions to obtain feature volumes with down-sampled sizes. The final 3D feature volumes are stacked along the z-axis to obtain a 2D BEV features map. This format allows the network to make use of existing 2D detection heads. The SSD detector is used to obtain a set of high-quality 3D proposals for each pixel. Here, the pixel features are forwarded to a classification head to predict a class probability as well as to a regression head to predict the 3D bounding box values. The result is a set of proposals obtained from the first stage of the model that is subsequently forwarded to the refinement stage. Besides utilizing both voxel features and BEV features, the model also incorporates raw points during the refinement stage. As mentioned in section 3.1.1, this strategy preserves detailed information that could otherwise be lost in voxel-based methods. Initially, the point cloud is down-sampled to n points via a farthest point sampling algorithm, where n serves as a hyperparameter specific

to the dataset in use. These remaining n points are named keypoints $\mathcal{K}$. The subsequent *Voxel Set Abstraction* module aims to aggregate multi-scale semantic features from the 3D backbone onto these keypoints. For each keypoint $p_i \in \mathcal{K}$, a set of neighboring voxel-wise feature vectors are retrieved by pooling them at the k-th level within a radius r_k. The architecture aggregates voxel features from four different levels $\{f_i^{pv_k}\}_{k=1}^{4}$. Besides, it interpolates the raw point features f_i^{raw} and the BEV features f_i^{bev} at the keypoint location. The resulting feature vector is then concatenated to form the final feature vector $f_i^{(p)}$ for each keypoint p_i. The equation 4.1 shows the concatenation of the different features.

$$f_i^{(p)} = \text{Concat}\left(\{f_i^{(pv_k)}\}_{k=1}^{4}, f_i^{(raw)}, f_i^{(bev)}\right) \tag{4.1}$$

Points corresponding to foreground objects should have a greater influence on proposal refinement than keypoints from the background. To achieve this, the model introduces a *Predicted Keypoint Weighting* module for re-weighting the keypoint features. After processing through this module, the weighted keypoint features $\mathcal{F} = \{f_i^{(p)}\}_{i=1}^{n}$ not only include multi-scale semantic features from the 3D voxels but also have precise location data via their 3D keypoint coordinates. This provides the model with the ability to preserve the 3D structural information of the entire scene. Thus, the keypoints make use of a point-voxel approach. The *RoI-grid pooling* module infuses proposals with information from corresponding keypoints based on their positions. For a given 3D proposal, the model uniformly samples $6 \times 6 \times 6$ *grid points* within the proposal. To aggregate the features from keypoints to these *grid points*, the module first determines the neighboring keypoints for each *grid point*. Keypoints within a distance of $r^{(g)}$ are considered. Notably, a single keypoint might contribute to multiple *grid points* because the neighboring balls around *grid points* can overlap. Equation 4.2 illustrates the operation of the *RoI-grid pooling* module. For a specific *grid point* g_i, keypoints within $r^{(g)}$ are taken into account, and both their features and relative positions to the grid point are utilized.

$$\Psi = \left\{\left(f_j^{(p)}, p_j - g_i\right) \mid \left\|p_j - g_i\right\| < r^{(g)}\right\} \tag{4.2}$$

Given the RoI-aligned features, two sibling subnetworks are used to predict a confidence score and a proposal refinement. The confidence score is used to filter out low-quality proposals. The proposal refinement is used to adjust the proposal location and size. The final output of the model is a set of refined proposals with their corresponding confidence scores. The model is illustrated in figure 4.5.

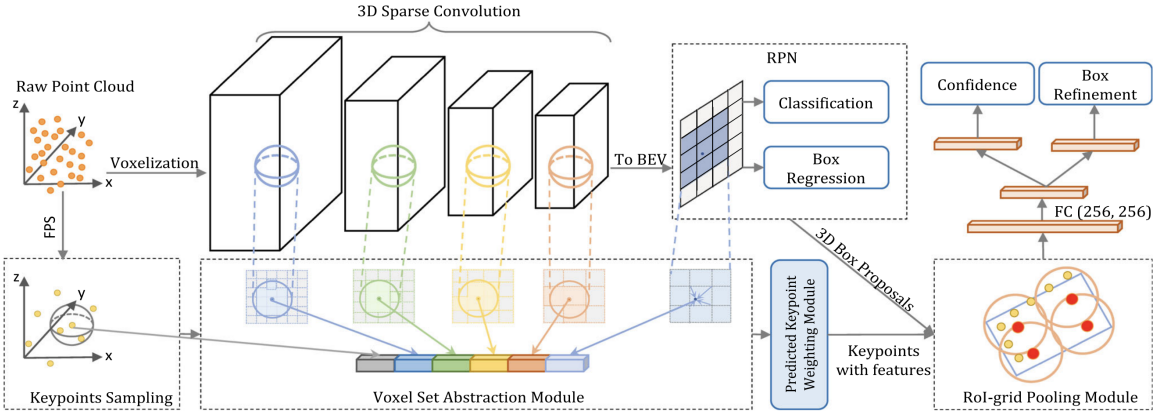


Figure 4.5: Architecture of PVRCNN, source: [Shi+23b]

4.3.2 PVRCNN++

PVRCNN++, building on the architecture of PVRCNN, introduces enhancements aimed at improving efficiency. A key advancement is the optimized Voxel Set Abstraction module, which reduces computational complexity while maintaining robust feature representation. This allows for more efficient processing of the proposals in the second stage. Additionally, PVRCNN++ refines the keypoint selection and weighting process, focusing on keypoints in critical areas with dense point clouds. The RoI-grid pooling module in PVRCNN++ is also improved. An adaptive sampling strategy that concentrates on areas of higher informational content is applied to pool the appropriate information from different sources. This not only refines the feature pooling step but also reduces computational overhead, making PVRCNN++ a more robust and efficient model for processing complex 3D data. In conclusion, PVRCNN++ can be seen as mainly an efficiency improvement over PVRCNN but it also comes with a slightly better performance.

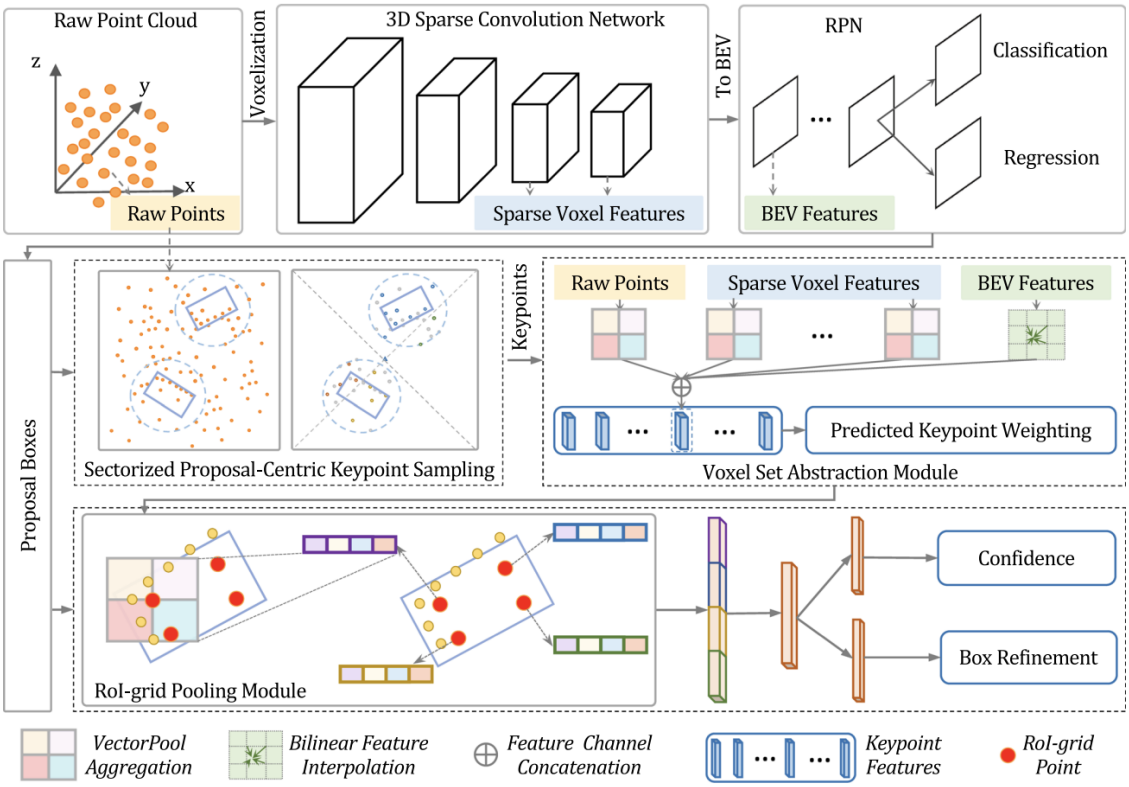


Figure 4.6: Architecture of PVRCNN++ with its updated Keypoint Sampling strategy and Voxel Set Abstraction module compared to PVRCNN, source: [Shi+23b]

4.3.3 PartA2

The PartA2 architecture represents another significant leap in the field of 3D object detection. PartA2, standing for Part-Aware and Aggregation, introduces a novel approach to the processing and interpretation of 3D point cloud data. At its core, PartA2 architecture utilizes a unique part-aware sampling technique. This technique intelligently selects points from the point cloud, focusing on parts of objects that are most informative for detection tasks. Unlike the keypoint sampling in PVRCNN and PVRCNN++, which selects points based on spatial

distribution or feature richness, PartA2 emphasizes the semantic significance of each point, leading to a more effective representation of the object's structure. In a sub-step the architecture predicts to which part of the object each point belongs to, see figure 4.7. This information is in the subsequent model used to make more precise predictions. The architecture of PartA2 is designed to be both computationally efficient and highly effective in detecting objects.

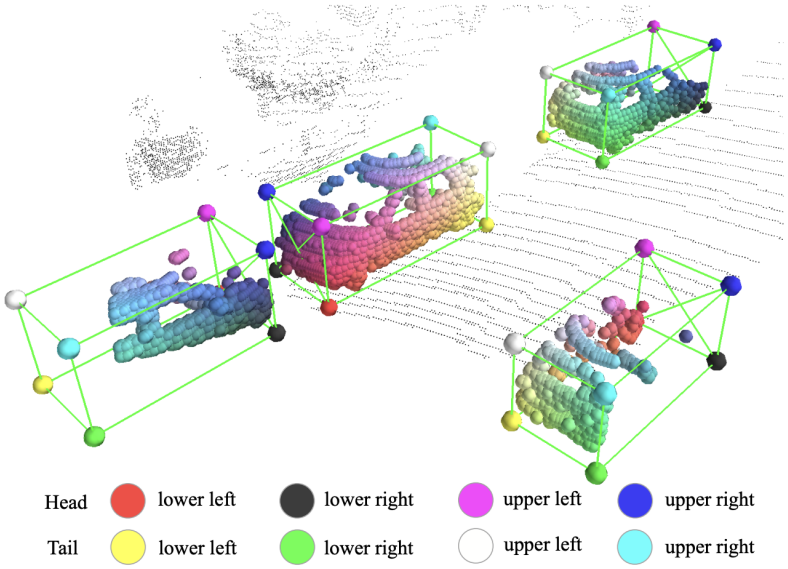


Figure 4.7: PartA2 classifies each point in a proposal based on which part of the object they belong to, source: [Shi+19]

4.3.4 Voxel-RCNN

The Voxel R-CNN framework introduces a novel approach to 3D object detection, primarily focusing on enhancing the representation and processing of voxelized point cloud data. Building upon the concepts of R-CNN architectures, Voxel R-CNN leverages the strengths of both 3D and 2D feature learning, as can be seen in figure 4.8. At the initial stage, Voxel R-CNN takes a point cloud input and voxelizes it into a structured format suitable for convolutional processing. This voxelization step is crucial as it transforms the irregular, sparse point cloud into a regular grid, enabling the application of efficient 3D convolutions. The 3D backbone network processes these voxelized inputs to extract sparse 3D features, capturing the spatial hierarchy and geometric detail within the data. The subsequent stage involves the 2D backbone network and RPN, which operate on the BEV representations converted from the 3D feature maps. This representation allows the model to predict precise object proposal candidates. The main innovation of Voxel R-CNN is its Voxel RoI pooling module, which efficiently aggregates voxel features within the RoI. For each proposal, a set of grid points is uniformly sampled, and the corresponding voxel features are pooled to form a RoI feature vector. This vector includes the contextual information necessary for accurate object detection. Finally, the detection head, composed of fully connected layers, takes the aggregated RoI feature vector and outputs the final detection results.

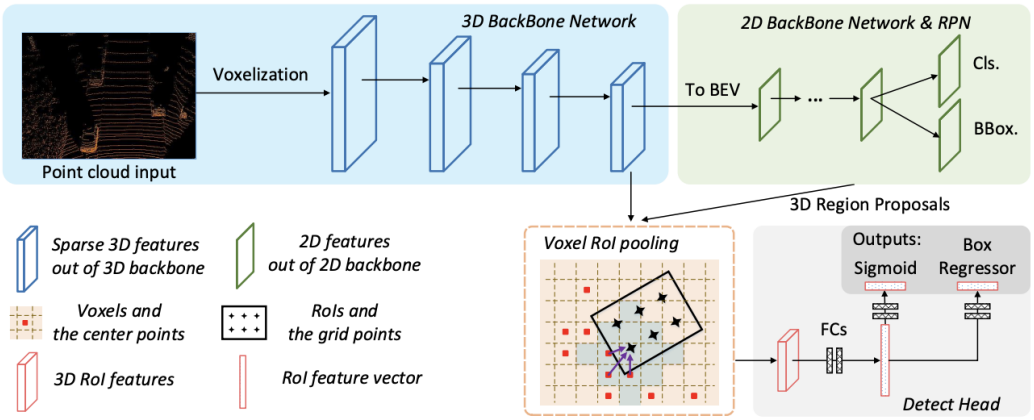


Figure 4.8: Architecture of Voxel-RCNN, source: [Den+21]

4.4 Object Relation Module

This section focuses on the implementation of this project. The goal of this work is to create a modular *Object Relation Module* compatible with any two-stage detectors. Experiments are conducted using both baseline architectures and those extended with the *Object Relation Module*. The baseline code was pre-existing, while the *Object Relation Module* is newly developed for this work. It enriches the detection pipeline by relating object proposals instead of treating them individually. The open-source OpenPCDet repository serves as the code foundation. It offers various 3D detection architectures consisting of sub-modules, making it easy to add new modules. The *Object Relation Module* is built into this codebase and can be applied to any architecture. Typically, the refinement stage in an object detection pipeline operates on object proposals in isolation. However, the *Object Relation Module* deviates from this convention by explicitly modeling inter-proposal relationships. As this module relies on identified proposals, it has to be part of the refinement stage in the object detection pipeline. The *Object Relation Module* takes as input both the classification and regression results from the first stage, as well as refined features from the second stage. Structurally, the module consists of two key sub-modules: the *Graph Constructor* and the *Relation Module*. These serve as generic interfaces that can be instantiated in various ways. The *Graph Constructor* takes the classification and regression values to form a graph by establishing edges between proposals that are to be related. Subsequently, the *Relation Module* uses these edges and the refined features to relate proposals based on the constructed graph. The end result is a tensor, referred to as *Related Features*, wherein each proposal feature is infused with contextual information drawn from other proposals. The *Related Features* are then forwarded to the classification and regression heads to obtain the final predictions. This high-level overview of the *Object Relation Module* is agnostic to the two-stage detector it is being integrated into. The *Object Relation Module* is illustrated in figure 4.9.

Formally, the *Object Relation Module* takes three inputs: 1. Classification values $\boldsymbol{C} \in \mathbb{R}^{n \times 1}$ 2. Box regression values $\boldsymbol{B} \in \mathbb{R}^{n \times 7}$ and 3. Refined features $\boldsymbol{F} \in \mathbb{R}^{n \times d}$. It outputs the *Related Features* $\boldsymbol{F}' \in \mathbb{R}^{n \times d'}$ as table 4.1 shows. The remaining part of this section will explain the two sub-modules in detail and will present the PVRCNN-Relation instantiation of the *Object Relation Module*.

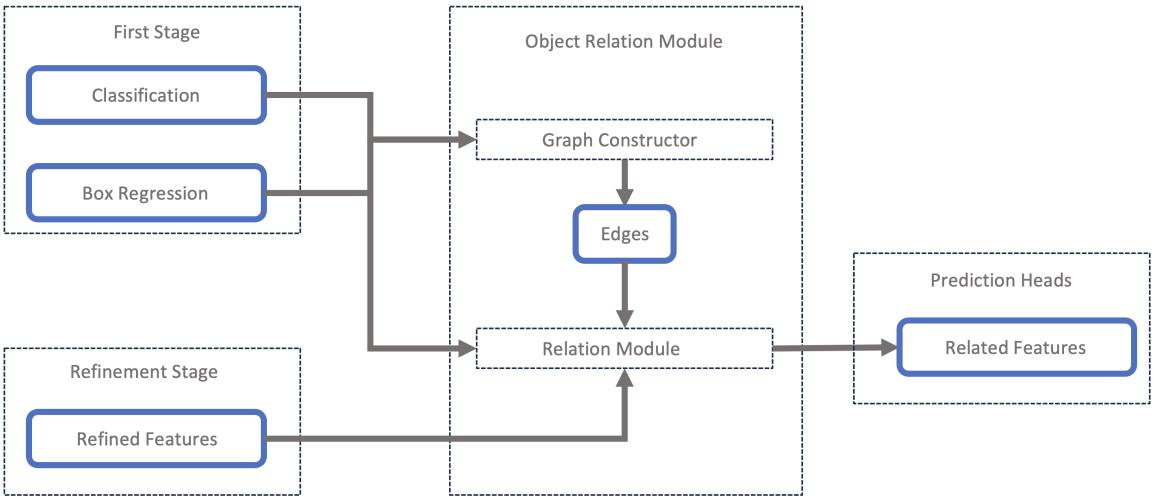


Figure 4.9: Diagram of the *Object Relation Module*

Input	
Classification	$\boldsymbol{C} \in \mathbb{R}^{n \times 1}$
Box Regression	$\boldsymbol{B} \in \mathbb{R}^{n \times 7}$
Refined Features	$\boldsymbol{F} \in \mathbb{R}^{n \times d}$

Output	
Related Features	$\boldsymbol{F}' \in \mathbb{R}^{n \times d'}$

Table 4.1: Inputs and outputs of the *Object Relation Module*

4.4.1 Graph Constructor

The primary task of the *Graph Constructor* is to decide whether object proposals are to be related or not. As outlined in Section 3.3, there are various methods for establishing connections between proposals. One simple way is to create a fully connected graph among all proposals, as is done in MLCVNet [Xie+20]. In this scenario, the *Graph Constructor* would produce edges that form a fully connected graph. However, this approach has performance implications. In general, the runtime complexity of the subsequent *Relation Module* scales linearly with the number of edges. In a fully connected graph, the number of edges grows quadratically with the number of proposals. Therefore, the runtime complexity for the *Relation Module* in a fully connected setup scales as $O(n^2)$, where n represents the number of proposals. This quadratic complexity can be avoided by using only a subset of all possible edges. The metric used to decide whether two given object proposals are connected is usually a distance metric. There are works that measure this distance in feature space [WS21] or more commonly in $\mathbb{R}^3$ space [YZK21; Sch+23; QLL22b]. This comes down to the assumption that either similar proposals or neighboring proposals should have influence on each other. To exploit traffic patterns as section 4.1 mentioned, it is usually sufficient to relate objects in the vicinity to each other. Regarding choosing the distance threshold it is important to note that picking a low distance threshold will result in primarily modeling relations for proposals belonging to the same object while a larger threshold will also model relations for different objects. Taking the classification result from the first stage into consideration, the *Graph Constructor* can be designed to only establish edges between proposals of the same class. This can help to explicitly exploit the traffic patterns of a specific class, e.g. vehicles. The implementation of this work includes a k-NN mechanism, connecting a fixed number of proposals

with each other, and a radius graph, connecting all proposals within a certain distance. These mechanisms can additionally be configured to only connect object proposals within the same class. Algorithm 1 shows the pseudo-code for the *Graph Constructor*.

Formally, the *Graph Constructor* accepts two inputs: 1. Edge-generating values, denoted as $\boldsymbol{D}$, used for distance measurement. 2. Optional classification values $\boldsymbol{C} \in \mathbb{R}^{n\times 1}$. The distance can be calculated in feature space ($\boldsymbol{D} = \boldsymbol{F}$) or in $\mathbb{R}^3$ space ($\boldsymbol{D} = \boldsymbol{B'}$), where $\boldsymbol{B'}$ is a subset of $\boldsymbol{B}$ representing bounding box positions. It outputs a set m of edges $\boldsymbol{E} \in \mathbb{R}^{2\times m}$ that connect the objects based on the chosen metric.

Algorithm 1 Pseudo-code for the *Graph Constructor*: Edges connect either all or just objects of the same class. Function f computes the edges and can be implemented as a radius graph or k-NN graph. It expects one argument a being the radius or k-value.

procedure GET_EDGES($d\_tensor, class\_labels$)
  $batch\_vector \leftarrow [1, 1, ..., 2, ..., B, B]$
  if $connect\_same\_class\_only$ **then**
    $edges \leftarrow []$
    for $class$ from 1 to $number\_classes$ **do**
      $class\_indices \leftarrow$ get_indices_where($class\_labels = class$)
      $filtered\_d\_tensor \leftarrow$ filter_tensor_by($class\_indices$)
      $filtered\_batch\_vector \leftarrow$ filter_vector_by($class\_indices$)
      $current\_edges \leftarrow f(filtered\_d\_tensor, a, filtered\_batch\_vector)$
      update $current\_edges$ using $class\_indices$
      append $current\_edges$ to $edges$
    end for
  else
    $edges \leftarrow f(edge\_generating\_tensor, a, batch\_vector)$
  end if
  return $edges$
end procedure

4.4.2 Relation Module

The *Relation Module* is responsible for modeling the relation between proposals. It takes the refined features from the second stage of the object detector and using the edges obtained from the *Graph Constructor* relates them to each other. As mentioned in section 3.3 such a mechanism can be implemented with various types of learnable architectures. The code provided with this work contains several implementations of the *Relation Module*:

- Implementation based on the compact generalized non-local network [Yue+18] that operates on a fully connected graph.
- Implementation based on the BADet architecture [QLL22b].
- Implementation based on a GNN using a graph attention mechanism [Vel+17].
- Implementation based on a GNN using an edge convolution mechanism [Wan+19]

Inspired by OpenPCDet, all these implementations are easily configurable with files written in YAML format. Here, the high-level architecture that should be used is specified. Additionally, exact implementation details such as the number of layers, skip connection, drop-out

values, or batch norm can be configured in one place. During the experimentation phase of this work, this allowed to easily iterate on architectural designs and make them comparable. For future users of this code base, this provides a clear and easily extendable framework to continue the work on object relations for 3D object detection.

4.4.3 PVRCNN-Relation

PVRCNN-Relation is a specific implementation of the *Object Relation Module* applied to the PVRCNN architecture presented earlier. As most tests were carried out for this specific implementation, this chapter provides an in-depth view of its implementation.

First, it is important to understand the difference between the predictions of the first stage in the global frame of reference and the predictions of the second stage in a local frame of reference. This means that the first stage predicts the bounding box position of the proposals in the reference of the point cloud observer. Then, the second stage takes these proposals and predicts an offset relative to the proposal's position. A proposal will lose the information about its global position in the scene when it is processed in the second stage. This makes the refinement stage invariant to the proposal positions. Two identical point clouds belonging to proposals at different positions in the scene will lead to identical predictions in the second stage. Formally, let $L(p)$ be the function of the second stage that takes a proposal p and predicts a relative offset Δb. Let p_1 and p_2 be two proposals at different positions in the scene. If the point clouds inside their respective receptive fields (as determined by the architecture of L) are identical, then $L(p_1) = L(p_2)$. However, to model object relations between objects the *Object Relation Module* has to break this invariance. Two proposals that are to be related to each other must have information about how their location relates to each other. Breaking this invariance might have negative performance implications that have to be offset by modeling object relations for a successful architecture.

The *Object Relation Module* is integrated into the second stage of PVRCNN; the updated model is named PVRCNN-Relation. The baseline is preserved as is until the refinement stage pools all features to obtain Ψ. Then, a fully connected layer is used to obtain the *Refined Features*, denoted as $\boldsymbol{F} \in \mathbb{R}^{n \times 256}$. The *Graph Constructor* takes the regression values $\boldsymbol{B}$ and uses the subset $\boldsymbol{B'}$ that contains the bounding box positions of the first stage. The distance between the positions of the proposals is being used to construct the graph. The k-NN mechanism is used to connect each proposal to its n closest neighbors. The *Graph Constructor* G takes the positions $\boldsymbol{B'}$ and outputs edges $\boldsymbol{M}$ according to the k-NN metric, $G(\boldsymbol{B'}) = \boldsymbol{M} \in \mathbb{R}^{2 \times m}$. Both the *Refined Features* $\boldsymbol{F}$ as well as the edges $\boldsymbol{M}$ are being used by the *Relation Module*. Let $\boldsymbol{f}_i^0$ and $\boldsymbol{b}_i^0$ be the *Refined Features* and the bounding box results for proposal i. Additionally, let $\mathcal{N}(i)$ be the neighborhood of i constructed by the edges $\boldsymbol{M}$. Equation 4.3 shows the message-passing equation used to relate surrounding proposals to proposal i in layer k of the GNN.

$$\boldsymbol{f}_i^k \leftarrow \max_{\forall j \in \mathcal{N}(i)} \boldsymbol{W}^k \cdot \mathrm{CAT}\left(\{\boldsymbol{f}_i^{k-1}, \boldsymbol{f}_j^{k-1} - \boldsymbol{f}_i^{k-1}, b_j - b_i\}\right) \tag{4.3}$$

The local features f are iteratively updated while the global bounding box values b stay constant. The *Relation Module* is implemented as a GNN with l layers. Equation 4.4 shows how the *Related Features* $f_i' \in \boldsymbol{F'}$ are obtained by concatenating all hidden features obtained from the GNN.

$$\boldsymbol{f}_i' \leftarrow \cdot\mathrm{CAT}\left(\{\boldsymbol{f}_i^0, ..., \boldsymbol{f}_i^l\}\right) \tag{4.4}$$

Similar to the baseline, the *Related Features* are forwarded to the classification and regression heads to obtain the final predictions. Figure 4.10 shows the architecture of the *Relation*

Module. The *Refined Features* function as the node features that are iteratively updated in each layer of the GNN according to the message passing implementation. The graph uses directed edges between these nodes according to the output of the *Graph Constructor*. The vector from one proposal to another in the bounding box space functions as the edge features. These edge features incorporate information about how two proposals relate to each other.

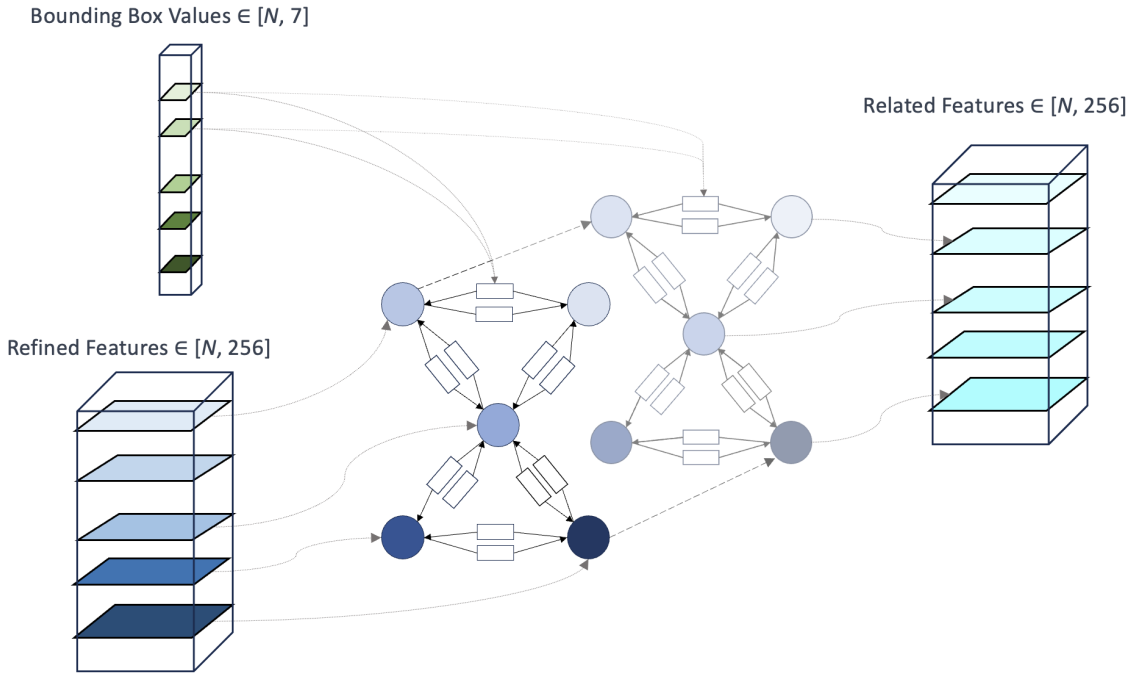


Figure 4.10: Architecture of the *Relation Module* implemented with a GNN

Chapter 5

Experiments and Results

The last chapter introduced the implementation of several baselines and the *Object Relation Module*. Based on these implementations, experiments are conducted. These experiments are aimed to test the hypothesis of this work: Can modeling object relation improve the performance of two-stage 3D object detectors? This chapter will summarize the results obtained during this work. First, it will introduce the KITTI and Waymo datasets that are being used in the experiments. After, it will describe the experimental setup. Then, it will present the quantitative results and, finally, show some qualitative results.

5.1 Datasets

There are many datasets openly available in the domain of autonomous driving that make training architectures easier. Annotating 3D scenes with bounding boxes and classes for each object requires a lot of effort by many people. The fast-paced environment of autonomous driving would not be possible if each researcher had to create a custom dataset to train on. Having common datasets that are shared by many people allows for efficient and faster iterations on new architectures. They are a major factor in the path towards autonomous driving. Additionally, if different methods are trained on the same data, architectures are comparable. Typically, datasets will make some of its data openly available for training and evaluation. However, the annotations for a part of the data will be kept by the dataset provider. Researchers have the possibility to send promising results from new architectures to the dataset provider who will evaluate them. The corresponding results will be published in a ranking. Published work, therefore, becomes comparable and it is easy to verify if a new method keeps up to its promises. Among the most common datasets are KITTI [GLU12], Waymo [Sun+20], Nuscenes [Cae+20], and Argoverse [Cha+19]. In this work, both the KITTI and Waymo datasets are being used. KITTI is an early dataset and is relatively small which allows for faster training of models, and fast iterations on architectures. Waymo dataset, in contrast, is bigger but considered state-of-the-art and widely used in research.

KITTI

KITTI provides benchmarks for the tasks of stereo, optical flow, visual odometry-SLAM, and 3D object detection. It consists of more than 200k 3D object annotations captured in cluttered scenarios with up to 15 vehicles in one scene. It was created with the aim to be highly realistic for autonomous driving and tried to close the gap between models performing well on earlier datasets but subsequently bad in real-world scenarios. KITTI is evaluated on the mAP of 40 thresholds (R40) dynamically chosen based on the model's detection. It comes with three classes (car, pedestrian, cyclist) and three difficulties (easy, moderate, hard).

Waymo

The Waymo dataset is a much larger dataset that tries to provide diverse data from a variation of environments. It consists of 1150 scenes that each span 20 seconds, consisting of high-quality LiDAR and camera outputs captured across a range of urban and suburban geographies. It provides three classes and is evaluated with mAP, mAPH (mean Average Precision High) measuring the performance on larger objects closer to the observer and mAPL (mean Average Precision Low) measuring objects further away. The objects are categorized into two classes LEVEL_1 the easy detections and LEVEL_2 the more difficult detections.

5.2 Experimental Setup

All experiments conducted during this work aimed at finding the best architecture to model object relations for 3D object detection pipelines. The first step in conducting experiments is choosing an appropriate baseline to which the *Object Relation Module* can be applied. It was already mentioned that the PVRCNN architecture was chosen. PVRCNN is a relatively new architecture that is already well-cited and has shown good detection performance on a variety of benchmarks. Additionally, this architecture is implemented in OpenPCDet, making PVRCNN a good baseline. Most experiments were conducted on a RTX3090 GPU and some on a V100 GPU. This hardware allowed for the training of PVRCNN models on KITTI in around 18 hours, whereas training on the full dataset of Waymo was not feasible. Therefore, a subset (20 percent) of Waymo data was used to train a model in 120 hours. The initial architecture of the *Object Relation Module* was designed based on previous work. Then, the architecture was iteratively updated; see section 5.3.4 for details. This process resulted in the implementation of PVRCNN-Relation presented earlier.

5.3 Quantitative Results

This section will go into detail on how the training of models was conducted and will show quantitative results. First, it will present the results of the fully end-to-end trained PVRCNN-Relation architecture. Then, results using a transfer learning-based approach, results of the *Object Relation Module* applied to a variety of baselines and finally it will show the ablation of PVRCNN-Relation.

5.3.1 PVRCNN-Relation

This architecture is the main contribution of this work. The implementation details were already presented in 4.3.1. This architecture is the result of iterations of architectural changes that are ablated in section 5.3.4. While the first iterations were conducted on all classes of the KITTI dataset, later models were only trained and evaluated on the car class. It was found that training the PVRCNN-Relation or the baseline model on the KITTI dataset comes with fluctuations in performance. Meaning that the same model trained twice will lead to slightly different outcomes when evaluated. This makes iterating on architectural changes difficult. It might be necessary to conduct experiments more than once to verify the results. Especially the results for the pedestrian and cyclist classes have a high variance. This might be due to the fact that these classes are underrepresented; the KITTI dataset comes with many more detected vehicles compared to pedestrians and cyclists. Therefore at some point, it was

Model	Easy R11 / R40	Moderate R11 / R40	Hard R11 / R40	All R11 / R40
PVRCNN	89.39 / 92.02	83.63 / 84.80	78.86 / 82.58	83.91 / 86.45
PVRCNN-Relation	89.59 / 92.53	84.56 / 85.22	79.04 / 82.99	84.35 / 86.90
Improvement	**+0.20 / +0.51**	**+0.93 / +0.42**	**+0.18 / +0.41**	**+0.44 / +0.45**

Table 5.1: Comparison of PVRCNN and PVRCNN-Relation on KITTI validation set. Trained and evaluated only on the car class. All models were trained for 80 epochs and the best-performing epoch per model and metric was chosen. **Both models were trained three times** and the average is reported. The *Improvement* row represents the difference in mAP between the two models. See table A.2 for full results.

decided to train and evaluate only the vehicle class. Both models, PVRCNN and PVRCNN-Relation, were trained for 80 epochs and with a batch size of 2. The results of both models trained on the car class of KITTI can be seen in table 5.1.

The full quantitative results of A.2 show that the updated architecture using the *Object Relation Module* consistently outperforms the PVRCNN baseline. The improvements are equally distributed over all the difficulty classes. The most important difficulty class Moderate has a mAP improvement of 0.42% when using the standard R40 approach (evaluating the precision-recall-curve at 40 different thresholds that are dynamically chosen depending on the model's predictions). This rather small but consistent improvement, therefore, shows that modeling object relations is able to improve the performance of the PVRCNN baseline for the car class. Additionally, table 5.2 shows a performance comparison of PVRCNN and PVRCNN-Relation when trained on the whole KITTI dataset with all classes. Here, the car class shows similar improvements with an improvement for all difficulties. Additionally, the cyclist class is also improved for all difficulties with the main improvement coming from the Easy class. Finally, the pedestrian class yields mixed results. There is an improvement for the Easy and Moderate difficulties, however, for the Hard difficulty, PVRCNN-Relation performs slightly worse. Besides the pedestrian Hard class, the PVRCNN-Relation architecture shows an improvement when trained in all classes too.

As previously mentioned, training on the KITTI dataset came with problems of fluctuations when evaluating different architectures. KITTI is a relatively small dataset when compared to more recent datasets like Nuscenes or Waymo. One can assume that the larger the dataset is, the more consistent the model evaluation. Therefore, there is a shift in the research field away from using KITTI. Models are typically trained on the Waymo dataset. However, training a model on a large-scale dataset like Waymo comes with high computational costs. Even when trained on 20% of the Waymo data, the PVRCNN model took 120 hours to train, as previously mentioned. It was still decided to train both PVRCNN and PVRCNN-Relation on 20% of the Waymo data. The results can be found in appendix A.3. Surprisingly, there are no mAP improvements for PVRCNN-Relation when trained on Waymo data. In fact, the results show a high decrease in performance so that a configuration or implementation error cannot be ruled out. Due to time and computational constraints, the results could not be carried out again.

5.3.2 Transfer Learning

Transfer learning has emerged as an important technique in computer vision, particularly given the complexity and resource intensity of training DL models from scratch. It involves applying knowledge gained from solving one problem to a different but related problem. In computer vision, this typically means using a pre-trained model, often developed on large-scale datasets like ImageNet, as a starting point. This approach can be significantly more

	Car			
	Easy R11 / R40	Moderate R11 / R40	Hard R11 / R40	All R11 / R40
PVRCNN	89.33 / 92.22	83.69 / 84.60	78.77 / 82.50	83.84 / 86.37
PVRCNN-Relation	89.40 / 92.40	84.68 / 85.09	78.88 / 82.86	84.23 / 86.72
Improvement	**+0.07 / +0.18**	**+0.99 / +0.49**	**+0.11 / +0.36**	**+0.39 / +0.35**
	Pedestrian			
	Easy R11 / R40	Moderate R11 / R40	Hard R11 / R40	All R11 / R40
PVRCNN	66.27 / 66.48	58.84 / 58.70	55.26 / 54.42	59.93 / 59.79
PVRCNN-Relation	66.52 / 67.07	59.37 / 58.67	54.07 / 53.15	59.82 / 59.51
Improvement	**+0.25 / +0.59**	**+0.53 / -0.03**	**-1.19 / -1.27**	**-0.11 / -0.28**
	Cyclist			
	Easy R11 / R40	Moderate R11 / R40	Hard R11 / R40	All R11 / R40
PVRCNN	87.65 / 91.53	72.44 / 73.76	69.29 / 69.59	76.11 / 78.05
PVRCNN-Relation	89.48 / 93.08	73.15 / 73.63	69.37 / 69.68	77.16 / 78.50
Improvement	**+1.83 / +1.55**	**+0.71 / -0.13**	**+0.08 / +0.09**	**+1.05 / +0.45**

Table 5.2: Comparison of PVRCNN and PVRCNN-Relation on KITTI validation set. Trained and evaluated in all classes. All models were trained for 80 epochs and the best performing epoch per model and metric was chosen. The Improvement row represents the difference in mAP between the two models.

efficient. The effectiveness of transfer learning in computer vision is attributed to the generalizability of learned features from specific datasets. This methodology has accelerated advancements in various applications, such as image recognition or object detection.

Transfer learning encompasses also the idea of fine-tuning. Here, the weights of a pretrained model are used to train a new model, e.g. with a different classification head. This idea can also be applied to 3D object detection. In this work, the weights from an already fully trained PVRCNN model are used to train a new model with a different head. All weights up until the *Object Relation Module* are taken from the pretrained model and kept frozen in the subsequent process. Then, the weights for the head including the *Object Relation Module* from the new model are initialized randomly. Finally, the head is trained with KITTI data for a few epochs while keeping all other weights but the head frozen. This approach allows training PVRCNN-Relation in a matter of minutes instead of hours. Transfer learning, if successful, leads to a significant efficiency improvement.

To see whether such an approach works, the weights of a fully end-to-end trained PVRCNN model are used. Using these weights a new model is trained (Pretrained-PVRCNN). This model reuses the weights of the PVRCNN model up until the classification and regression head. The weights of the heads are initialized randomly. All reused weights are kept frozen and this new model is trained on KITTI data for a few epochs. It is important to note, that this model, Pretrained-PVRCNN, has an identical architecture to the model, PVRCNN, from which it took the weights. The results previously obtained from the end-to-end trained PVRCNN model, as well as the results of the Pretrained-PVRCNN model, can be seen in the table 5.3. One can see that the performance of these two models is very similar. For some classes and difficulties, the Pretrained-PVRCNN model even outperforms the PVRCNN model (see Pretrained-PVRCNN trained for 10 epochs and evaluated on Easy for the car class). This shows that the transfer learning approach works for models with identical architectures.

However, the transfer learning approach should be applied to models with different ar-

	Car		
	Easy	Moderate	Hard
	3 ep. / 5 ep. / 10 ep.	3 ep. / 5 ep. / 10 ep.	3 ep. / 5 ep. / 10 ep.
PVRCNN	92.22	84.53	82.57
Pret.-PVRCNN	92.18 / 92.06 / 92.26	84.55 / 84.57 / 84.66	82.45 / 82.52 / 82.90
Pret.-PVRCNN-Relation	92.03 / 92.03 / 92.26	83.08 / 83.11 / 83.50	82.46 / 82.60 / 82.90
Improvement	**-0.15 / -0.03 / +0.00**	**-1.47 / -1.46 / -1.16**	**+0.01 / +0.08 / +0.00**
	Pedestrian		
	Easy	Moderate	Hard
	3 ep. / 5 ep. / 10 ep.	3 ep. / 5 ep. / 10 ep.	3 ep. / 5 ep. / 10 ep.
PVRCNN	64.24	55.49	51.20
Pret.-PVRCNN	64.79 / 63.38 / 63.68	56.12 / 54.87 / 55.07	51.45 / 50.70 / 50.95
Pret.-PVRCNN-Relation	63.74 / 63.83 / 62.79	56.10 / 56.16 / 54.72	51.31 / 51.32 / 49.95
Improvement	**-1.05 / +0.45 / -0.89**	**-0.02 / +1.29 / -0.35**	**-0.14 / +0.62 / -1.00**
	Cyclist		
	Easy	Moderate	Hard
	3 ep. / 5 ep. / 10 ep.	3 ep. / 5 ep. / 10 ep.	3 ep. / 5 ep. / 10 ep.
PVRCNN	90.80	73.62	68.87
Pret.-PVRCNN	90.43 / 90.82 / 90.61	72.87 / 73.41 / 73.17	68.32 / 68.71 / 68.36
Pret.-PVRCNN-Relation	90.73 / 90.65 / 90.34	72.41 / 72.35 / 72.04	67.55 / 68.61 / 67.45
Improvement	**+0.30 / -0.17 / -0.27**	**-0.46 / -1.06 / -1.13**	**-0.77 / -0.10 / -0.91**

Table 5.3: Results from transfer learning. The table shows the results of PVRCNN, Pretrained-PVRCNN, and Pretrained-PVRCNN-Relation. The improvement row shows the difference in mAP between the Pretrained-PVRCNN and Pretrained-PVRCNN-Relation models.

chitectures, e.g. different regression and classification heads. In the next step, therefore, another model was trained (Pretrained-PVRCNN-Relation). Similar to Pretrained-PVRCNN, this model takes the pretrained weights of the PVRCNN model up until the head. Here, however, an architectural change is introduced. Pretrained-PVRCNN-Relation includes the *Object Relation Module* in the second stage. During training, all reused weights are kept frozen while the weights in the *Object Relation Module* and in the subsequent heads are changed. The results of Pretrained-PVRCNN-Relation can also be seen in table 5.3. The performance from Pretrained-PVRCNN and Pretrained-PVRCNN-Relation are mixed. When both models are evaluated on the pedestrian class for difficulty Moderate, the model with the additional relation module yields better performance. However, on car Moderate the Pretrained-PVRCNN model achieves better results. To sum it up, these results show that an *Object Relation Module* can be applied to an already trained model. While these results fail to improve all detection classes, the fact that these models can be trained in much less time makes this approach worth to further investigate.

5.3.3 Other Baselines

After the promising results of the PVRCNN-Relation model, in the next step, the *Object Relation Module* is applied to other baselines. Other baselines that are being used are PVRCNN++ [Shi+23b], an extension to PVRCNN, PartA2 [Shi+19], and Voxel-RCNN [Den+21]. Implementation details of these models were already introduced in section 4.3. The *Object Relation Module* was slightly adapted to each baseline. The standard model as well as the model extended with the *Object Relation Module* was trained for 80 epochs exclusively on the car class. The results can be found in table 5.4. The results of PVRCNN++-Relation are similar to its baseline but fail to consistently improve the baseline. For the PartA2 and Voxel-RCNN model, the *Object Relation Module* does not seem to be suited. Both models extended with a module

Model	Easy R11 / R40	Moderate R11 / R40	Hard R11 / R40	All R11 / R40
PartA2	89.42 / 92.28	83.01 / 83.85	78.80 / 82.25	83.54 / 85.84
PartA2-Relation	89.57 / 92.82	79.06 / 83.36	78.57 / 81.01	82.34 / 85.66
Improvement	**+0.15 / +0.54**	**-3.95 / -0.49**	**-0.23 / -1.24**	**-1.2 / -0.18**
PVRCNN-++	89.40 / 91.88	81.45 / 84.37	81.08 / 82.94	83.98 / 86.40
PVRCNN-++-Relation	89.58 / 91.82	81.36 / 84.35	80.97 / 82.37	83.92 / 86.07
Improvement	**+0.18 / -0.06**	**-0.09 / -0.02**	**-0.11 / -0.57**	**-0.06 / -0.33**
Voxel-RCNN	89.66 / 92.65	84.79 / 85.42	78.99 / 82.94	84.41 / 86.98
Voxel-RCNN-Relation	89.31 / 92.58	79.21 / 83.33	78.51 / 80.86	82.32 / 85.59
Improvement	**-0.35 / -0.07**	**-5.58 / -2.09**	**-0.48 / -2.08**	**-2.09 / -1.39**

Table 5.4: Comparison of different architectures with and without the *Object Relation Module*. All models were trained for 80 epochs and the best-performing epoch per model and metric was chosen. The Improvement row represents the difference in mAP between each two.

for object relation perform worse.

The iterative process of improving the *Object Relation Module*, which was carried out on the PVRCNN baseline, leads to an implementation that is tailored towards a specific architecture. This implementation fails to generalize to other architectures. Only for the PVRCNN++ model, which is very similar to the PVRCNN model, does the performance not drop significantly when the *Object Relation Module* is applied.

5.3.4 Ablation Study

This section provides an ablation for the PVRCNN-Relation architecture. The reader is advised to see the corresponding results in table A.4. The table shows five experiments that were carried out sequentially. In each experiment something about the architecture was changed and evaluated. Therefore, the architecture was iteratively improved. The better-performing architecture is always depicted at the bottom of each experiment and the *Improvement* row shows the difference between both models. While both models for each experiment are trained with the same hyperparameter configuration, to ensure they are comparable, models from different experiments might not be comparable.

Global Information

An initial *Object Relation Module* integrated into PVRCNN was trained without incorporating global information. As earlier mentioned, the refinement stage loses information about the global location of proposals. Without global information, two proposals that are related to each other are not aware of how they relate to each other in $\mathbb{R}^3$ space. In an experiment, the bounding box values of the first stage (which include the proposals' global positions) are appended to the *Refined Features* before the *Object Relation Module* is applied. The results show that appending this global information to each proposal leads to a better result.

Radius and kNN Graph

It was already discussed that proposals that are close to each other should be related to each other. However, the *Graph Constructor* can be implemented in different ways. An experiment was conducted to compare a radius and a k-nearest-neighbors (kNN) graph. For the radius

graph, a radius of $r = 6m$ was chosen and the kNN graph used a k value of $k = 16$. Both lead to roughly the same number of connections over all scenes, therefore, both methods are comparable. Qualitative results for both implementations of the *Graph Constructor* can be found in appendix A.2. It is interesting to note, that using a radius graph leads to a graph connecting many proposals belonging to the same object. One can, therefore, say that the radius graph focuses on the *Proposal Consensus* motivation discussed in section 4.1. The results, however, show that the kNN graph, which focuses on relating different objects, leads to better performance.

Edge Features

The next experiment was focused on improving how the global information is introduced to the proposals. Instead of simply appending the global information (in form of the bounding box values) to the *Refined Features*, in an updated architecture, they are introduced by using edge features. Here, the bounding box values of two connected proposals are subtracted from each other for a given connection, leading to a graph with directed edges. The information on how two proposals globally relate to each other is, therefore, iteratively infused into each node feature during the message passing of the GNN. The results show that this way of introducing global information leads to a better performance.

Batchnorm and Dropout

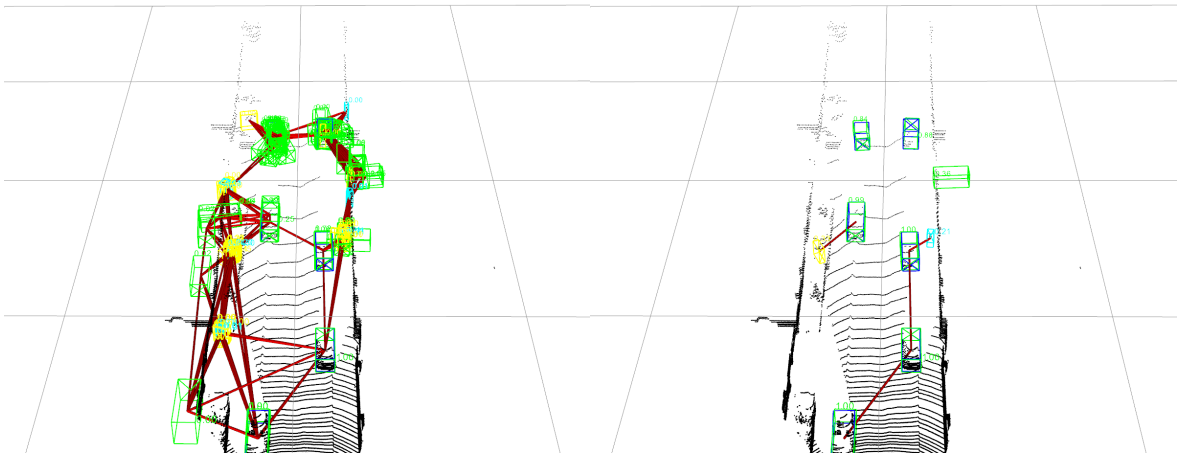
Batchnorm and Dropout can improve the learning capability of models for a variety of architectures and problems. An experiment was conducted to see if these methods can also be applied to the PVRCNN-Relation model. The linear layer that is being used during the message passing was extended with a Batchnorm and Dropout layer. The results show that extending the model with these two layers leads to a better performance.

Appended GNN Features

Finally, it was tested whether appending the GNN features after the *Relation Module*, as equation 4.4 shows, leads to better performance. Appending the features allows the final regression and classification head to make use of all features that are created during the message passing instead of just using the last one. The results show that using all hidden states of the GNN leads to better results.

5.4 Qualitative Results

The last step of many two-stage detectors, including all baselines used in this work, is a post-processing step. Here, NMS is applied to all bounding boxes that have an overlap larger than a threshold. The aim is to have only one bounding box per object. Additionally, bounding boxes that have a confidence value lower than a threshold will be filtered out. The *Object Relation Module* operates on the set of proposals before this post-processing step. This allows it to make use of relations between proposals describing the same object and it, therefore, captures the *Proposal Consensus* motivation. Figure 5.1 shows an example of PVRCNN-Relation's predictions on KITTI data. The figure on the left depicts the graph formed by the *Graph Constructor* before the post-processing step. The figure on the right shows the scene after this step. It is important to note that this step is not learned but implemented with an algorithm. Both figures show the connections formed by the *Graph Constructor* in red. While the figure



(a) Before Post-Processing **(b)** After Post-Processing

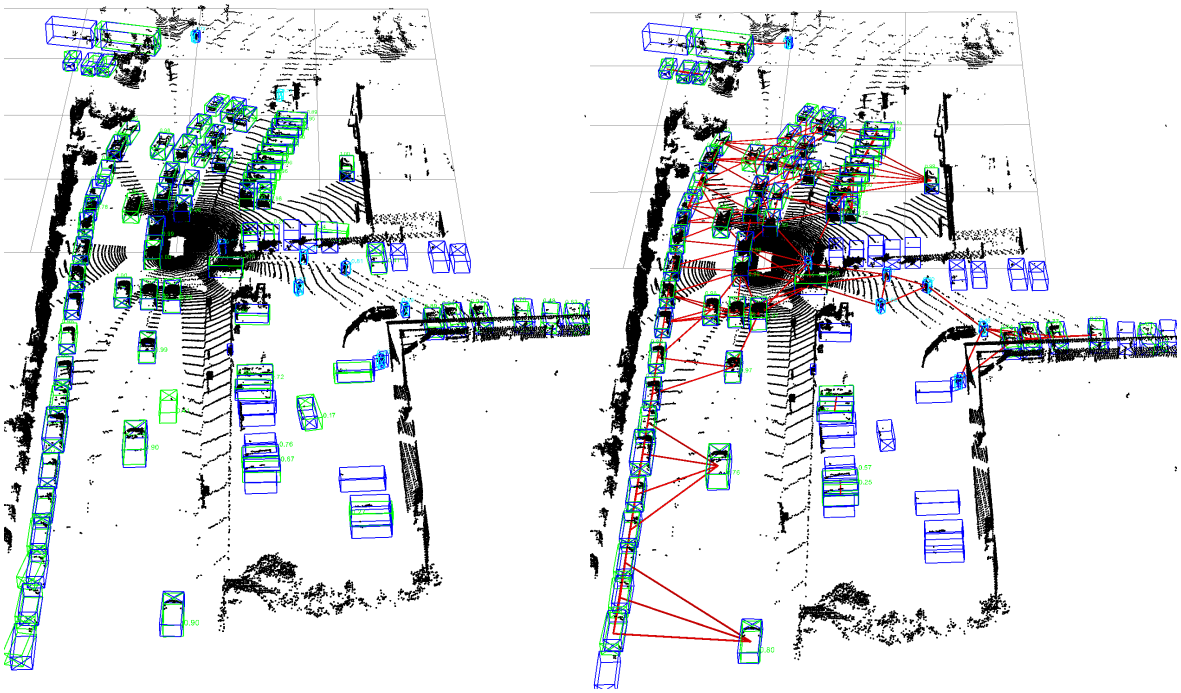
Figure 5.1: Figure showing the graph before and after post-processing. Since a kNN graph (k=16) was used, every proposal is connected to its 16 closest neighbors. The figure shows the bounding boxes of predicted vehicles in green, cyclists in yellow, and pedestrians in light blue. Ground truth labels are visualized in dark blue. Edges connecting proposals are visualized in red.

before the post-processing step shows all edges, the other figure shows the edges between proposals only if both are not filtered by the post-processing step. In the following all qualitative results will show the models' predictions after this post-processing step. Additionally, predictions for vehicles will have green, cyclists yellow, and pedestrians light blue bounding boxes. Ground truth labels are visualized in dark blue.

To give the reader an idea of how edges are formed by the *Graph Constructor* (using the kNN graph (k=16) of PVRCNN-Relation), two large scenes from Waymo are shown in the figures 5.2 and 5.3. The figures additionally show the predictions of the PVRCNN baseline.

The remainder of this section will focus on qualitative results of PVRCNN-Relation on KITTI. The goal is to identify examples in which the *Object Relation Module* likely improved the predictions of the baseline by modeling object relations. All figures will show the predictions of the baseline on the left and predictions of the baseline extended with the *Object Relation Module* on the right. Additionally, the edges connecting proposals are visualized for the figures on the right. It is important to note that the message passing of the *Relation Module* is applied several times, therefore, proposals that are not directly connected can still exchange information.

Figure 5.4 shows two scenes of PVRCNN and PVRCNN-Relation. Both sets of predicted objects are similar. However, in the center of both scenes, the baseline predicts two objects that are really close to each other. In fact, these predicted objects have some overlap (but not enough overlap to be captured by the NMS), making such a pattern unlikely to occur in a traffic scene. If proposals are processed individually they will not know about each other and the model is not able to suppress one of them. On the other hand, in PVRCNN-Relation both proposals are aware of the other. This allows the model to make use of the prior that such scenes usually do not happen. The result of modeling object relation can be seen in the right figure. Here, only one of the two proposals is predicted to be an actual object. Note that both models use the same first stage (which produces the proposals), therefore, it is likely that the PVRCNN-Relation model similar to the PVRCNN model had these two proposals before the post-processing step. Allowing these proposals to share information with each other likely



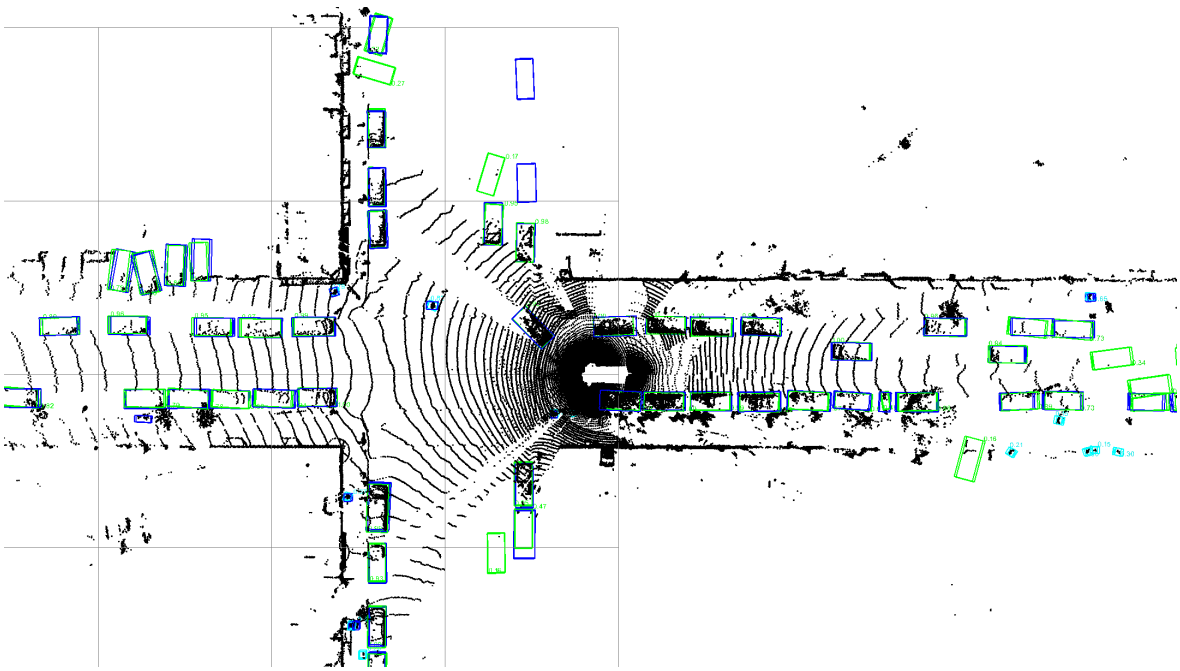
(a) PVRCNN **(b)** PVRCNN-Relation

Figure 5.2: Scene from Waymo with predictions from PVRCNN and PVRCNN-Relation. The figure shows the bounding boxes of predicted vehicles in green, cyclists in yellow, and pedestrians in light blue. Ground truth labels are visualized in dark blue. Edges connecting proposals are visualized in red.

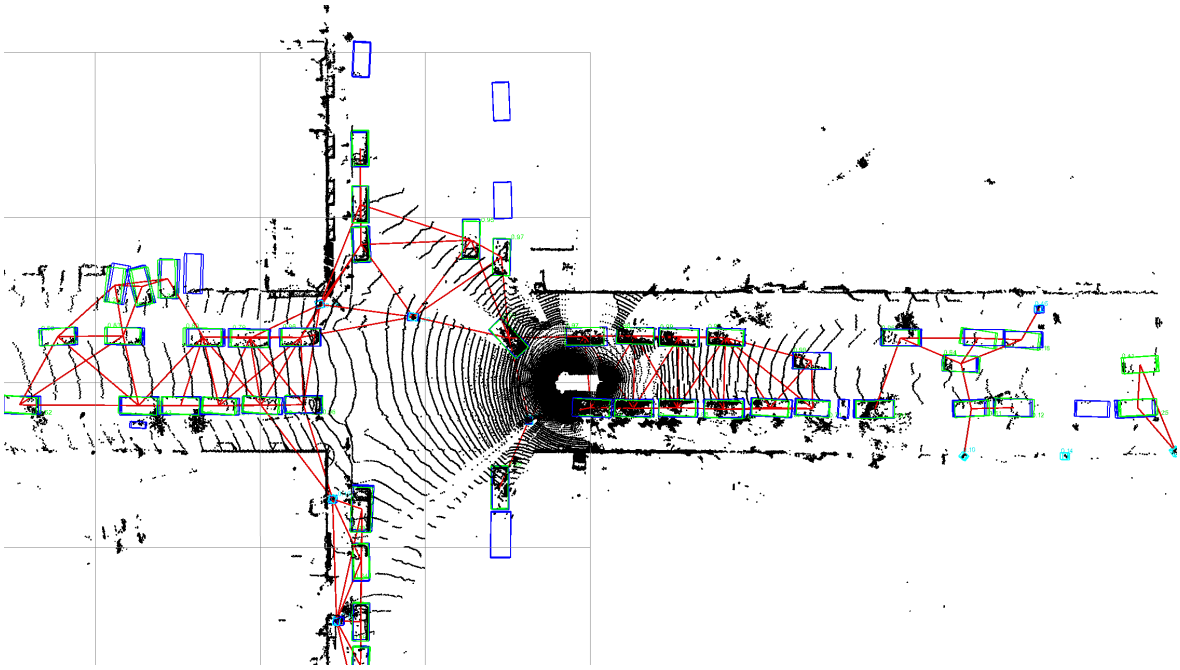
led to predicting a low confidence value for one of them which subsequently allowed the post-processing step to filter it.

Figure 5.5 shows another quantitative comparison between PVRCNN and PVRCNN-Relation. In the center-left of the scene, there is a vehicle with a ground truth annotation having a horizontal angle. Between this vehicle and the observer, there is a line of parked vehicles on the side of the road. Therefore, the corresponding point cloud for that vehicle is heavily occluded. Looking closely at the point cloud corresponding to that object, one can see that only half of the vehicle is visible. Given such a point cloud, processing it individually without any global context about the scene makes it impossible to decide whether it has a horizontal or vertical angle even for a human. PVRCNN, therefore, falsely predicts a vertical angle. On the other hand, while PVRCNN-Relation is processing this object it is able to detect that the corresponding point cloud is likely occluded. Additionally, it is able to infuse its *Related Features* with context. In this case, the context is provided by the other vehicles parked with a similar horizontal angle. The model is able to use this information and correctly predicts a horizontal angle for that vehicle.

Figure 5.6 shows a comparison of PVRCNN and PVRCNN-Relation for three different scenes. All three scenes show that the predictions of PVRCNN-Relation lead to a globally more consistent scene. The model gets rid of a number of false positives that are not consistent with the rest of the scene. In the section 4.1 the potential of the *Object Relation Module* to detect patterns that are unlikely to occur was already discussed. This ability of the *Object Relation Module* can be seen in the depicted figures. These qualitative results show that PVRCNN-Relation leads to more realistic and globally consistent scenes.

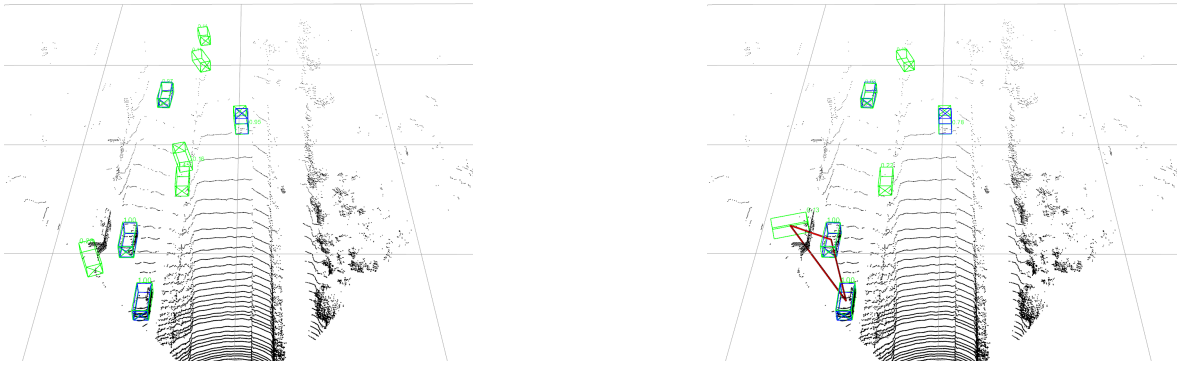


(a) PVRCNN

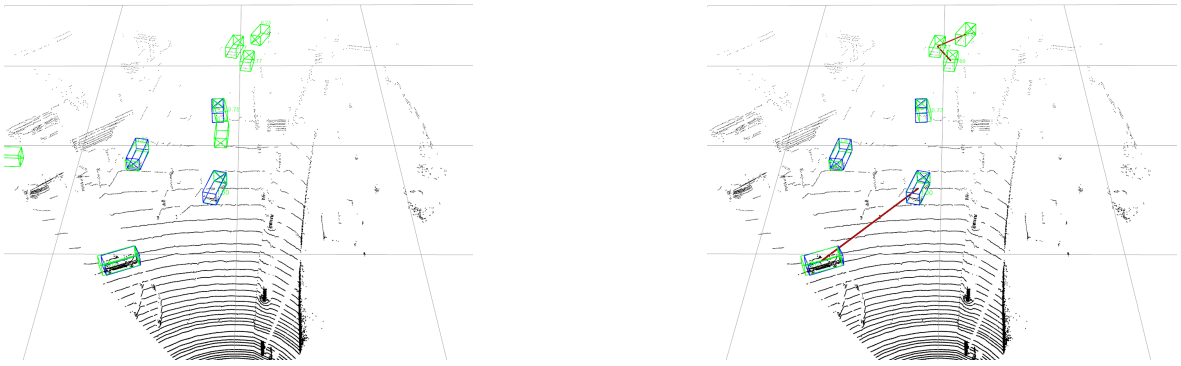


(b) PVRCNN-Relation

Figure 5.3: Scene from Waymo with predictions from PVRCNN and PVRCNN-Relation. The figure shows the bounding boxes of predicted vehicles in green, cyclists in yellow, and pedestrians in light blue. Ground truth labels are visualized in dark blue. Edges connecting proposals are visualized in red.

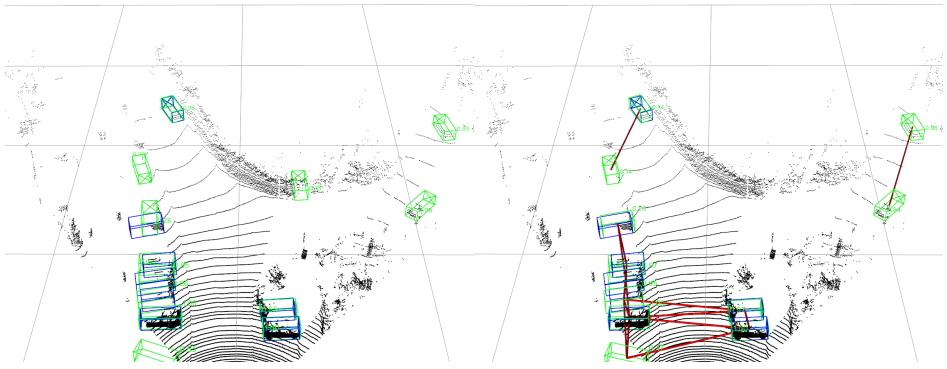


(a) PVRCNN **(b)** PVRCNN-Relation



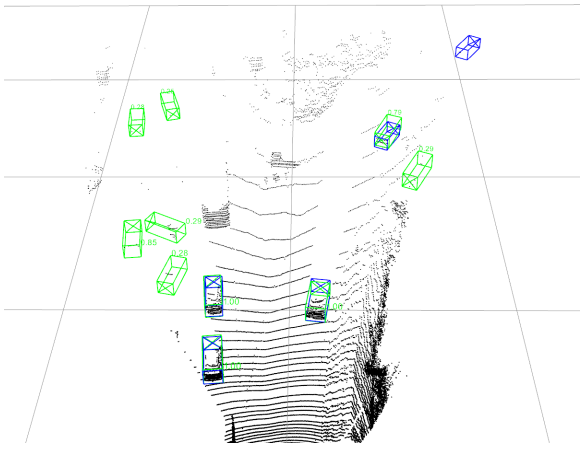
(c) PVRCNN **(d)** PVRCNN-Relation

Figure 5.4: Comparison of PVRCNN and PVRCNN-Relation on a KITTI scene. PVRCNN predicts two proposals that have some overlap. PVRCNN-Relation on the other hand only predicts one proposal. The figure shows the bounding boxes of predicted vehicles in green. Ground truth labels are visualized in dark blue. Edges connecting proposals are visualized in red.

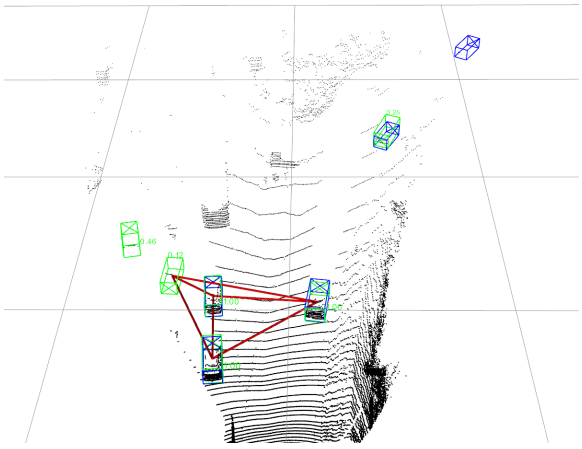


(a) PVRCNN **(b)** PVRCNN-Relation

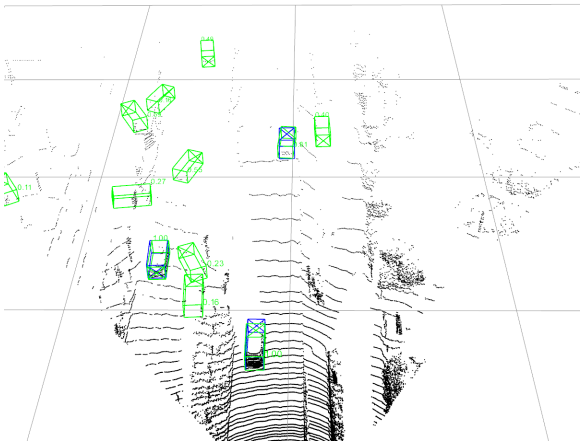
Figure 5.5: Comparison of PVRCNN and PVRCNN-Relation on a KITTI scene. The vehicle of interest is the object with a ground truth annotation in the center-left of the scene. PVRCNN predicts a vertical angle and PVRCNN-Relation a horizontal angle for this vehicle. The figure shows the bounding boxes of predicted vehicles in green. Ground truth labels are visualized in dark blue. Edges connecting proposals are visualized in red.



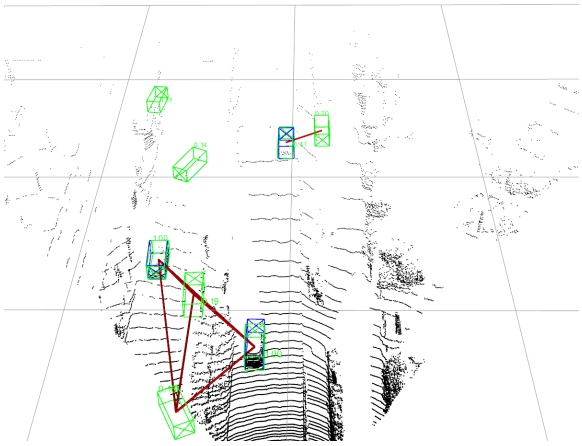
(a) PVRCNN



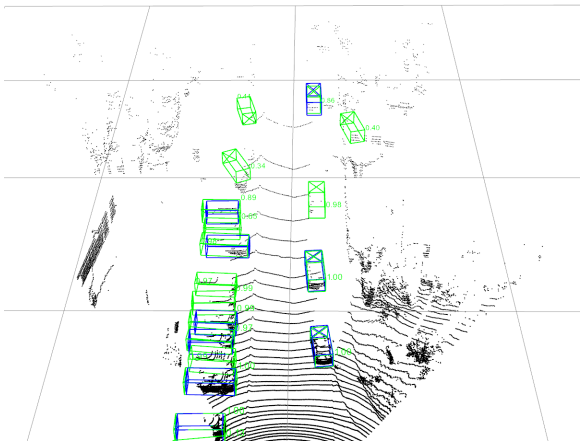
(b) PVRCNN-Relation



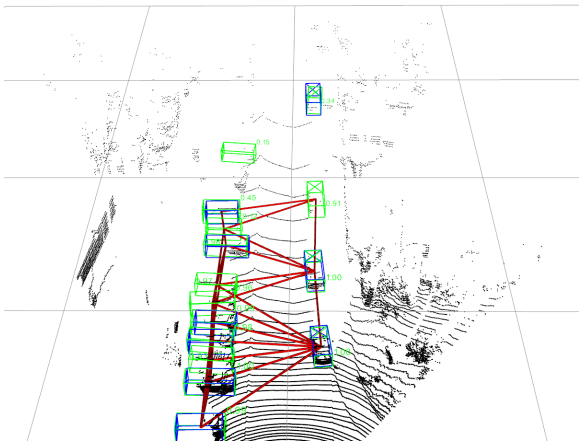
(c) PVRCNN



(d) PVRCNN-Relation



(e) PVRCNN



(f) PVRCNN-Relation

Figure 5.6: Comparison of PVRCNN and PVRCNN-Relation on a KITTI scene. PVRCNN-Relation predicts objects that are more realistic in a global sense. The figure shows the bounding boxes of predicted vehicles in green. Ground truth labels are visualized in dark blue. Edges connecting proposals are visualized in red.

Chapter 6

Future Work and Conclusion

The final chapter of this work will give an outlook on future work and will end with a conclusion.

6.1 Future Work

This section gives an outlook into potential future work. That includes ideas and areas that were not successful or there were not enough resources available to try them. First, the result of the *Object Relation Module* implementation PVRCNN-Relation is a successful example of how a two-stage detector can be improved by modeling object relation. However, it is unlikely to be the best architecture. There were not enough computing resources available to fully ablate all parameters of the model. Future work might want to investigate the optimal k values for the kNN graph, the number of iterations of the GNN, hidden dimensions of the linear layers, and other implementation details. Additionally, more fundamental changes to the *Object Relation Module* such as different message passing methods, fully connected graphs, or modeling object relation with attention can be tried. One shortcoming of this work is that this kind of iterative work on architectural design choices is heavily dependent on the availability of compute resources. To make progress, usually, two models must be fully end-to-end trained and evaluated to decide what architecture leads to better results. Additionally, one must deal with fluctuations in evaluation results that make this comparison difficult. These constraints made it impossible to carry out all experiments that might be desirable. Moreover, training PVRCNN-Relation on the Waymo dataset led to undesirable results. The computational constraints did not allow iterating on these results to improve them. Future work could investigate why PVRCNN-Relation did not improve results for this dataset. Another shortcoming of the *Object Relation Module* is that it breaks the positional invariance of the baseline as discussed in 4.4.3. Other two-stage detectors might not make this invariance assumption and could, therefore, better be suited to be extended with an *Obeject Relation Module*. Finally, computationally efficient ways to model object relations can be explored in the future. This work presented the transfer learning approach with mixed results that others can build upon. Using such an approach allows working with large-scale datasets such as Waymo even with constrained computing capabilities. Another approach that avoids being constrained by the available compute could be inspired by the GACE [Sch+23] approach presented in section 3.3. Here, the baseline model is not altered, and a model is independently trained to predict whether a proposal is a false or true positive. The *Object Relation Module* of PVRCNN-Relation could be applied in the same way.

6.2 Conclusion

A contribution of this work is giving four clear motivations why modeling object relations in two-stage object detection pipelines can improve performance. Increasing the receptive field, modeling proposal consensus, detecting occlusions, and exploiting patterns. Additionally, this work presented qualitative results that validated these motivations. PVRCNN-Relation, an implementation of the *Object Relation Module* using a GNN with edge convolution message passing applied to the PVRCNN baseline showed consistent improvements in performance. PVRCNN is a recent architecture that achieves high rankings on different benchmarks. PVRCNN-Relation consistently outperforms this baseline for almost any class and difficulty and is, therefore, another major contribution of this work. Additionally, this work introduced the idea of transfer learning and showed that such an approach can achieve similar results to models that are trained end-to-end. Finally, this work publishes code that is integrated into the OpenPCDet repository. It includes the framework *Object Relation Module* that can be implemented in various ways. It encompasses the implementation of PVRCNN-Relation, PVRCNN++-Relation, PartA2-Relation, and Voxel RCNN-Relation. Additionally, it enables training each of these models with the transfer learning approach.

This work extends the existing work on object relations for 3D object detection. PVRCNN-Relation is an example of successfully exploiting object relations to improve detection performance. This work is first at applying a module to relate objects to a range of baseline detectors testing the generalizability of such a module when trained end-to-end. The code published with this work provides a foundation to continue the work on object relations. Object relations have the potential to improve a wide range of different detection pipelines and ultimately make autonomous driving more efficient and safe.

Appendix A

Appendix

A.1 Object Refinement

Title	Year	Method	Description
Part A2 Net: Point Cloud with Part-aware and Part-aggregation Network [Shi+19]	2021	Intra-Object Segmentation	All the points inside a proposal are classified as to which part of the vehicle they belong to. This information is then used as part of the pooled information.
Improving 3D Object Detection with Channel-wise Transformer [She+21]	2021	Proposal Context	Transformer architecture is used to model channel-wise context aggregation for the point features within each proposal.
Real-Time Point Cloud Object Detection via Voxel-Point Geometry Abstraction [Shi+23a]	2023	Different Sources and Scales	Is using coarse voxel representation to accelerate proposal generation while using precise point representation to facilitate proposal refinement.
Dynamic graph transformer for 3D object detection [Ren+23]	2023	Different Sources and Scales	The proposals are refined using the proposal-aware spatial information and point-wise semantic features from the first stage.
Real-Time Stereo 3D Car Detection With Shape-Aware Non-Uniform Sampling [Gao+23a]	2023	Sampling Strategy	The refinement stage aims to sample outer points from the proposed object.
Spatio-Temporal Contextual Learning for Single Object Tracking on Point Clouds [Gao+23b]	2023	Increase in Receptive Field	Is concerned with single object tracking. The tracked template is generated by including surroundings outside the target box to include context.
MVTr: multi-feature voxel transformer for 3D object detection [Ai+23]	2023	Different Sources and Scales	Takes into account different scales of the backbone features for the pooling operations.
PVRCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection [Shi+23b]	2023	Different Sources and Scales	Uses raw points, different scales of features from the backbone and BEV features in the pooling for each proposal.

Graph R-CNN: Towards Accurate 3D Object Detection with Semantic-Decorated Local Graph [Yan+22a]	2023	Sampling Strategy, Proposal Context, Shape Completion	For each proposal, dynamic farthest voxel sampling is applied to evenly sample points. A local graph is used on the sampled points to better model contextual information within the features of the proposal. Fusion is applied to each proposal to cope with the potential sparsity of the proposed object.
Attention-based Proposals Refinement for 3D Object Detection [DHF22]	2022	Proposal Context	Refinement using Vector Attention, it assigns different weights to different channels within a point feature.
Keypoints Representation of Density-aware and the Spatial-Channel-wise Decoder for 3D Object Detection [MZ22]	2022	Proposal Context, Sampling Strategy	A certain number of key points is sampled per proposal, and self-attention is applied to model the context between them.
M3DETR: Multi-Representation, Multi-Scale, Mutual-Relation 3D Object Detection With Transformers [Gua+22]	2021	Different Sources and Scales	The refinement uses different point cloud representations (raw, voxels, bird-eye view) with different feature scales, as well as modeled relationships using transformers.
Behind the Curtain: Learning Occluded Shapes for 3D Object Detection [XZN22]	2021	Shape Completion	Occluded regions inside the proposed object are identified and filled with a probability map that is integrated with the obtained features.
SIENet: Spatial information enhancement network for 3D object detection from point cloud [Li+21b]	2021	Shape Completion	A module is used to predict the spatial shapes of the proposal. This shape is used to complete the proposed object.
SC-RCNN: LiDAR-only 3D object detection based on spatial context [Wan+23]	2023	Increase in Receptive Field, Different Sources and Scales	Local grid point pooling is used to extract features and spatial context around a proposed region. The refinement uses a pyramid candidate box augmentation to obtain multi-scale features.
Deformable Pyramid R-CNN for 3D object detection (ChinaMM2022) [HZ22]	2022	Different Sources and Scales	Voxel feature pyramid dynamically selects multi-layer 3D features based on the sparsity of the proposed object.
Point Density-Aware Voxels for LiDAR 3D Object Detection [HKW22]	2022	Sampling Strategy	The voxel features are aggregated through a density-aware RoI grid pooling module.
Structure Guided Proposal Completion for 3D Object Detection [SZL22]	2022	Shape Completion	The detected proposal is matched with a seen structure and based on that matching the proposal is completed.
NV2P-RCNN: Feature Aggregation Based on s Neighborhood for 3D Object Detection [HJR23]	2023	Different Sources and Scales	In the refinement stage, the features of the raw points are extracted and fused with the voxel features. The voxel coordinates are used to fuse the voxel-point features.
Lgnet: Local and Global Point Dependency Network for 3d Object Detection [Ma+]	2022	Proposal Context	Point-to-point correlations in local regions are aggregated from neighboring points.
Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection [Den+21]	2022	Voxel Refinement	A voxel RoI pooling is devised to extract RoI features directly from the voxel features for further refinement.

Table A.1: Overview of methods used in the refinement step of two-stage detectors

A.2 Quantitative Results

Model	Easy R11 / R40	Moderate R11 / R40	Hard R11 / R40	All R11 / R40
PVRCNN-Relation	**89.80** / **92.81**	84.71 / **85.52**	**79.21** / **83.21**	**84.53** / **87.15**
PVRCNN-Relation	89.50 / 92.58	**84.81** / 85.09	78.97 / 82.95	84.39 / 86.86
PVRCNN-Relation	89.48 / 92.20	84.17 / 85.06	78.94 / 82.83	84.14 / 86.69
PVRCNN	89.37 / 92.12	83.62 / 84.88	78.93 / 82.71	83.97 / 86.57
PVRCNN	89.38 / 92.08	83.80 / 84.78	78.92 / 82.63	83.99 / 86.44
PVRCNN	89.43 / 91.85	83.47 / 84.73	78.73 / 82.41	83.77 / 86.33

Table A.2: Full results for the KITTI validation set. All models were trained for 80 epochs and the best-performing epoch per model and metric was chosen. Results include the R40 results used on the KITTI benchmark as well as the R11 results. The models were trained several times to deal with training fluctuations.

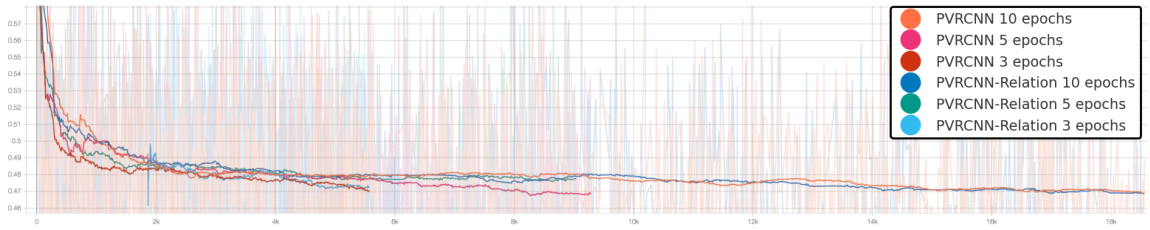


Figure A.1: Loss for PVRCNN and PVRCNN-Relation model for different epochs

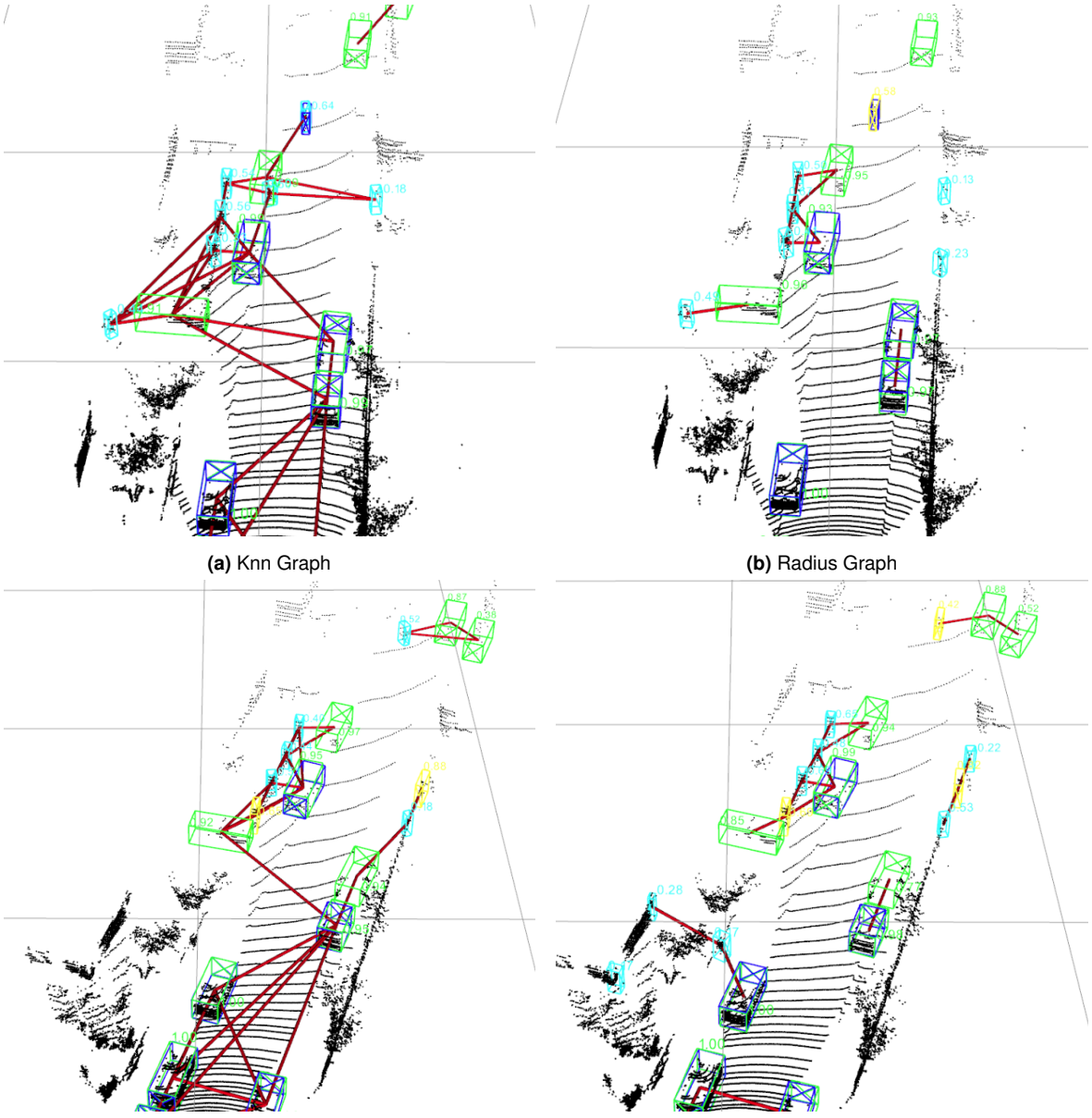
	Vehicle			
	Level 1 mAP	Level 1 mAPH	Level 2 mAP	Level 2 mAPH
PVRCNN	75.19	74.53	66.59	65.99
PVRCNN-Relation	70.28	69.54	61.60	60.94
Improvement	**-4.91**	**-4.99**	**-4.99**	**-5.05**
	Pedestrian			
	Level 1 mAP	Level 1 mAPH	Level 2 mAP	Level 2 mAPH
PVRCNN	71.46	34.88	62.51	30.53
PVRCNN-Relation	61.87	31.53	53.14	27.09
Improvement	**-9.59**	**-3.35**	**-9.37**	**-3.44**
	Cyclist			
	Level 1 mAP	Level 1 mAPH	Level 2 mAP	Level 2 mAPH
PVRCNN	50.54	30.46	48.61	29.30
PVRCNN-Relation	33.66	14.80	32.37	14.24
Improvement	**-16.88**	**-15.66**	**-16.24**	**-15.06**

Table A.3: Comparison of PVRCNN and PVRCNN-Relation on Waymo validation set trained and evaluated on all classes. Both models were trained with 20% of the Waymo data for 30 epochs. The Improvement row represents the difference in mAP between the two models.

	Car			
	Easy R11 / R40	Moderate R11 / R40	Hard R11 / R40	All R11 / R40
Without global information	87.79 / 89.79	81.79 / 82.91	77.96 / 81.28	82.28 / 84.54
With global information	87.78 / 90.24	82.35 / 83.25	78.17 / 81.54	82.55 / 84.79
Improvement	**-0.01 / +0.45**	**+0.56 / +0.34**	**+0.21 / +0.26**	**+0.27 / +0.25**
Radius graph	87.78 / 90.24	82.35 / 83.25	78.17 / 81.54	82.55 / 84.79
kNN graph	89.38 / 92.05	82.62 / 84.40	78.80 / 82.20	83.54 / 86.16
Improvement	**+1.60 / +1.81**	**+0.27 / +1.15**	**+0.63 / +0.66**	**+0.99 / +1.37**
Without Edge Features	88.62 / 91.24	83.47 / 84.07	78.07 / 81.95	83.27 / 85.69
With Edge Features	89.34 / 92.21	84.19 / 84.86	78.76 / 82.68	84.05 / 86.58
Improvement	**+0.72 / +0.97**	**+0.72 / +0.79**	**+0.69 / +0.73**	**+0.78 / +0.89**
Without BN + Dropout	89.41 / 91.86	79.15 / 82.43	78.61 / 81.60	82.35 / 85.21
With BN + Dropout	89.12 / 92.09	83.07 / 84.24	78.67 / 81.98	83.58 / 85.93
Improvement	**-0.29 / +0.23**	**+3.92 / +1.81**	**+0.06 / +0.38**	**+1.23 / +0.72**
Without GNN F. appended	89.26 / 91.74	83.45 / 84.33	78.71 / 82.21	83.76 / 86.09
With GNN F. appended	89.52 / 92.67	85.36 / 85.36	78.87 / 83.00	84.53 / 86.94
Improvement	**+0.26 / +0.93**	**+1.91 / +1.03**	**+0.16 / +0.79**	**+0.77 / +0.85**
	Pedestrian			
	Easy R11 / R40	Moderate R11 / R40	Hard R11 / R40	All R11 / R40
Without global information	66.06 / 66.78	59.39 / 59.94	55.30 / 54.62	60.14 / 60.45
With global information	66.12 / 67.18	59.68 / 59.28	55.63 / 55.15	60.48 / 60.53
Improvement	**+0.06 / +0.40**	**+0.29 / -0.66**	**+0.33 / +0.53**	**+0.34 / +0.08**
Radius graph	66.12 / 67.18	59.68 / 59.28	55.63 / 55.15	60.48 / 60.53
kNN graph	67.10 / 68.06	60.15 / 60.32	55.76 / 54.75	61.00 / 61.04
Improvement	**+0.98 / +0.88**	**+0.47 / +1.04**	**+0.13 / -0.40**	**+0.52 / +0.51**
Without edge Features	66.69 / 67.53	59.96 / 59.14	55.35 / 54.58	60.55 / 60.35
With edge Features	68.82 / 69.76	62.13 / 61.70	56.87 / 56.06	62.61 / 62.51
Improvement	**+2.13 / +2.23**	**+2.17 / +2.56**	**+1.52 / +1.48**	**+2.06 / +2.16**
Without BN + Dropout	67.06 / 67.08	60.43 / 59.91	55.62 / 54.50	60.97 / 60.43
With BN + Dropout	66.28 / 66.49	59.11 / 59.39	55.35 / 54.63	60.25 / 60.17
Improvement	**+0.78 / +0.59**	**+1.32 / +0.52**	**+0.27 / -0.13**	**+0.72 / +0.26**
Without GNN F. appended	66.11 / 66.00	59.49 / 58.79	54.33 / 53.60	59.98 / 59.46
With GNN F. appended	66.62 / 67.70	60.16 / 59.13	54.38 / 53.12	60.39 / 59.95
Improvement	**+0.51 / +1.70**	**+0.67 / +0.34**	**+0.05 / -0.48**	**+0.41 / +0.49**
	Cyclist			
	Easy R11 / R40	Moderate R11 / R40	Hard R11 / R40	All R11 / R40
Without global information	86.47 / 90.75	72.00 / 72.42	68.65 / 68.31	75.31 / 77.06
With global information	90.48 / 92.70	73.41 / 74.29	69.80 / 69.80	77.82 / 78.93
Improvement	**+4.01 / +1.95**	**+1.41 / +1.87**	**+1.15 / +1.49**	**+2.51 / +1.87**
Radius graph	90.48 / 92.70	73.41 / 74.29	69.80 / 69.80	77.82 / 78.93
kNN graph	89.65 / 91.50	72.17 / 72.84	69.12 / 68.79	75.89 / 77.02
Improvement	**-0.83 / -1.20**	**-1.24 / -1.45**	**-0.68 / -1.01**	**-1.93 / -1.91**
Without Edge Features	86.27 / 90.99	70.99 / 71.88	67.48 / 67.24	74.64 / 76.41
Edge Features	92.44 / 93.74	73.21 / 74.12	70.24 / 69.77	78.36 / 78.82
Improvement	**+6.17 / +2.75**	**+2.22 / +2.24**	**+2.76 / +2.53**	**+3.72 / +2.41**
Without BN + Dropout	91.50 / 92.95	73.76 / 73.97	69.57 / 69.35	77.61 / 78.73
With BN + Dropout	91.92 / 93.47	73.94 / 74.46	68.94 / 70.04	78.12 / 79.14
Improvement	**+0.42 / +0.52**	**+0.18 / +0.49**	**-0.63 / +0.69**	**+0.51 / +0.41**
Without GNN F. appended	87.32 / 92.40	73.35 / 73.56	69.08 / 69.81	76.42 / 78.01
With GNN F. appended	92.46 / 93.75	71.97 / 72.69	66.34 / 68.19	76.67 / 77.75
Improvement	**+5.14 / +1.35**	**-1.38 / -0.87**	**-2.74 / -1.62**	**+0.25 / -0.26**

Table A.4: Ablation of PVRCNN-Relation carried out on KITTI dataset.

A.3 Qualitative Results



(a) Knn Graph

(b) Radius Graph

(c) KNN Graph

(d) Radius Graph

Figure A.2: Comparison of a kNN and radius implementation of the *Graph Constructor* on KITTI data. Both methods lead to roughly the same number of connections between proposals. However, the radius graph connects many of the proposals that belong to the same objects. After NMS is applied, many connections vanish because they connected proposals from which at least one was suppressed. The figure shows the bounding boxes of predicted vehicles in green, cyclists in yellow, and pedestrians in light blue. Ground truth labels are visualized in dark blue. Edges connecting proposals are visualized in red.

A.4 Receptive Field of PVRCNN

The 3D convolutional backbone of PVRCNN consists of 12 sparse convolutional layers. Given equation A.1 the receptive field of each layer was computed according to table A.5. As section 4.3.1 described the refinement stage of PVRCNN uses *grid points* that are infused with

features from *keypoints* which are themselves infused with features from the 3D convolutional backbone, BEV features as well as raw point features. The convolutional features of layer 11 provide the keypoints with the largest receptive field, as they are pooled with the largest radius. To compute the receptive field of one proposal we need to look at the radius r_k used to pool voxel features and at the radius r_g used to pool keypoints to the *grid points*. Finally, together with the proposal's bounding box size, the receptive field of the 3D convolutional backbone, and the voxel size, we can compute an upper bound for the receptive field of a proposal. Equation A.2 shows how this is done for a proposal p. For Waymo $\max(W, L) = 0.1$ and for KITTI $\max(W, L) = 0.05$ to get an upper bound on the receptive field we can assume $\max(p_{\text{length}}, p_{\text{width}}) = 4$. The radii r_k and r_g are 4.8 and 1.6 respectively for both datasets. This yields a theoretical upper limit for the receptive field of **11.95m** for Waymo and **10.175m** for KITTI.

$$[H]\mathrm{RF}_l = \mathrm{RF}_{l-1} + (\mathrm{K}_l - 1) \cdot \prod_{i=1}^{l-1} \mathrm{S}_i \tag{A.1}$$

Layer l	RF_{l-1}	K_l	S_l	$\prod_{i=1}^{l-1} S_i$	RF_l
1	1	3	1	1	$1 + (3-1) \cdot 1 = 3$
2	3	3	1	1	$3 + (3-1) \cdot 1 = 5$
3	5	3	2	1	$5 + (3-1) \cdot 1 = 7$
4	7	3	1	2	$7 + (3-1) \cdot 2 = 11$
5	11	3	1	2	$11 + (3-1) \cdot 2 = 15$
6	15	3	2	2	$15 + (3-1) \cdot 2 = 19$
7	19	3	1	4	$19 + (3-1) \cdot 4 = 27$
8	27	3	1	4	$27 + (3-1) \cdot 4 = 35$
9	35	3	2	4	$35 + (3-1) \cdot 4 = 43$
10	43	3	1	8	$43 + (3-1) \cdot 8 = 59$
11	59	3	1	8	$59 + (3-1) \cdot 8 = 75$
12	75	3	2	8	$75 + (3-1) \cdot 8 = 91$

Table A.5: Receptive Field of 3D Convolutional Layers in PVRCNN

$$RF(p) \leq \frac{RF_{l=11}}{2} \cdot \max(W, L) + r_g + r_k + \frac{\max(p_{\text{length}}, p_{\text{width}})}{2} \tag{A.2}$$

List of Figures

List of Tables

Bibliography

[Ai+23] Ai, L., Xie, Z., Yao, R., and Yang, M. “MVTr: multi-feature voxel transformer for 3D object detection”. In: *The Visual Computer* (2023), pp. 1–14.

[Ali+18] Ali, W., Abdelkarim, S., Zidan, M., Zahran, M., and El Sallab, A. “Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud”. In: *Proceedings of the European conference on computer vision (ECCV) workshops*. 2018, pp. 0–0.

[BSU22] Baloch, Z., Shaikh, F. K., and Unar, M. A. “CNN-LSTM-Based Late Sensor Fusion for Human Activity Recognition in Big Data Networks”. In: *Wireless Communications and Mobile Computing* 2022 (2022).

[Bis06] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN: 978-0387310732.

[Bre+20] Brena, R. F., Aguileta, A. A., Trejo, L. A., Molino-Minero-Re, E., and Mayora, O. “Choosing the best sensor fusion method: A machine-learning approach”. In: *Sensors* 20.8 (2020), p. 2350.

[Cae+20] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. “nuscenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[Cam+18] Campbell, S., O’Mahony, N., Krpalcova, L., Riordan, D., Walsh, J., Murphy, A., and Ryan, C. “Sensor technology in autonomous vehicles: A review”. In: *2018 29th Irish Signals and Systems Conference (ISSC)*. IEEE. 2018, pp. 1–4.

[Cav+19] Cavazza, B. H., Gandia, R. M., Antonialli, F., Zambalde, A. L., Nicolai, I., Sugano, J. Y., and Neto, A. D. M. “Management and business of autonomous vehicles: a systematic integrative bibliographic review”. In: *International Journal of Automotive Technology and Management* 19.1-2 (2019), pp. 31–54.

[Cha+19] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al. “Argoverse: 3d tracking and forecasting with rich maps”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 8748–8757.

[CGS19] Choy, C., Gwak, J., and Savarese, S. “4d spatio-temporal convnets: Minkowski convolutional neural networks”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3075–3084.

[CZ20] Cui, Z. and Zhang, Z. “Pvf-net: Point & voxel fusion 3d object detection framework for point cloud”. In: *2020 17th Conference on Computer and Robot Vision (CRV)*. IEEE. 2020, pp. 125–133.

[Cyb89] Cybenko, G. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[DHF22] Dao, M.-Q., Héry, E., and Frémont, V. “Attention-based proposals refinement for 3D object detection”. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2022, pp. 197–205.

[DV20] Debeunne, C. and Vivet, D. “A review of visual-LiDAR fusion based simultaneous localization and mapping”. In: *Sensors* 20.7 (2020), p. 2068.

[Den+21] Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., and Li, H. “Voxel r-cnn: Towards high performance voxel-based 3d object detection”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 2. 2021, pp. 1201–1209.

[DZW20] Du, L., Zhang, R., and Wang, X. “Overview of two-stage object detection algorithms”. In: *Journal of Physics: Conference Series*. Vol. 1544. 1. IOP Publishing. 2020, p. 012033.

[Err+19] Errica, F., Podda, M., Bacciu, D., and Micheli, A. “A fair comparison of graph neural networks for graph classification”. In: *arXiv preprint arXiv:1912.09893* (2019).

[Fel+09] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. “Object detection with discriminatively trained part-based models”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.9 (2009), pp. 1627–1645.

[Fen+20] Feng, M., Gilani, S. Z., Wang, Y., Zhang, L., and Mian, A. “Relation graph network for 3D object detection in point clouds”. In: *IEEE Transactions on Image Processing* 30 (2020), pp. 92–107.

[Gao+23a] Gao, A., Cao, J., Pang, Y., and Li, X. “Real-Time Stereo 3D Car Detection With Shape-Aware Non-Uniform Sampling”. In: *IEEE Transactions on Intelligent Transportation Systems* 24.4 (2023), pp. 4027–4037.

[Gao+23b] Gao, J., Yan, X., Zhao, W., Lyu, Z., Liao, Y., and Zheng, C. “Spatio-Temporal Contextual Learning for Single Object Tracking on Point Clouds”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[Gei+13] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. “Vision meets robotics: The kitti dataset”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[GLU12] Geiger, A., Lenz, P., and Urtasun, R. “Are we ready for autonomous driving? the kitti vision benchmark suite”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3354–3361.

[Gir15] Girshick, R. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[Gir+14] Girshick, R., Donahue, J., Darrell, T., and Malik, J. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[Gua+22] Guan, T., Wang, J., Lan, S., Chandra, R., Wu, Z., Davis, L., and Manocha, D. “M3detr: Multi-representation, multi-scale, mutual-relation 3d object detection with transformers”. In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2022, pp. 772–782.

[He+21] He, Y., Xia, G., Luo, Y., Su, L., Zhang, Z., Li, W., and Wang, P. “DVFENet: Dual-branch voxel feature extraction network for 3D object detection”. In: *Neurocomputing* 459 (2021), pp. 201–211.

[HZ22] Hou, Y. and Zhang, X. “Deformable Pyramid R-CNN for 3D object detection (ChinaMM2022)”. In: *Displays* 75 (2022), p. 102322.

[HKW22] Hu, J. S., Kuai, T., and Waslander, S. L. "Point density-aware voxels for lidar 3d object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8469–8478.

[Hu+23] Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., et al. "Planning-oriented autonomous driving". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 17853–17862.

[Hua+21] Huang, J., Huang, G., Zhu, Z., Ye, Y., and Du, D. "Bevdet: High-performance multi-camera 3d object detection in bird-eye-view". In: *arXiv preprint arXiv:2112.11790* (2021).

[HJR23] Huo, W., Jing, T., and Ren, S. "NV2P-RCNN: Feature Aggregation Based on Voxel Neighborhood for 3D Object Detection". In: *Neural Processing Letters* (2023), pp. 1–21.

[Jog+11] Joglekar, A., Joshi, D., Khemani, R., Nair, S., and Sahare, S. "Depth estimation using monocular camera". In: *International journal of computer science and information technologies* 2.4 (2011), pp. 1758–1763.

[Kas+21] Kassens-Noor, E., Cai, M., Kotval-Karamchandani, Z., and Decaminada, T. "Autonomous vehicles and mobility for people with special needs". In: *Transportation research part A: policy and practice* 150 (2021), pp. 385–397.

[Kuz+20] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., et al. "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale". In: *International Journal of Computer Vision* 128.7 (2020), pp. 1956–1981.

[Lan+22] Lan, Y., Duan, Y., Liu, C., Zhu, C., Xiong, Y., Huang, H., and Xu, K. "ARM3D: Attention-based relation module for indoor 3D object detection". In: *Computational Visual Media* 8.3 (2022), pp. 395–414.

[Lan+21] Lan, Y., Duan, Y., Shi, Y., Huang, H., and Xu, K. "3DRM: Pair-wise relation module for 3D object detection". In: *Computers & Graphics* 98 (2021), pp. 58–70.

[Lan+19] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. "Pointpillars: Fast encoders for object detection from point clouds". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 12697–12705.

[LBH15] LeCun, Y., Bengio, Y., and Hinton, G. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[LeC+98] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[Li+21a] Li, J., Sun, Y., Luo, S., Zhu, Z., Dai, H., Krylov, A. S., Ding, Y., and Shao, L. "P2v-rcnn: Point to voxel feature learning for 3d object detection from point clouds". In: *IEEE Access* 9 (2021), pp. 98249–98260.

[LLY23] Li, J., Luo, C., and Yang, X. "PillarNeXt: Rethinking network designs for 3D object detection in LiDAR point clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 17567–17576.

[Li+20] Li, Y., Ma, L., Tan, W., Sun, C., Cao, D., and Li, J. “GRNet: Geometric relation network for 3D object detection from point clouds”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 165 (2020), pp. 43–53.

[LI20] Li, Y. and Ibanez-Guzman, J. “Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems”. In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 50–61.

[Li+21b] Li, Z., Yao, Y., Quan, Z., Yang, W., and Xie, J. “Sienet: Spatial information enhancement network for 3d object detection from point cloud”. In: *arXiv preprint arXiv:2103.15396* (2021).

[LFU13] Lin, D., Fidler, S., and Urtasun, R. “Holistic scene understanding for 3d object detection with rgbd cameras”. In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1417–1424.

[Lin+14] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. “Microsoft coco: Common objects in context”. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[Lin+20] Lin, W., Hasenstab, K., Moura Cunha, G., and Schwartzman, A. “Comparison of handcrafted features and convolutional neural networks for liver MR image adequacy assessment”. In: *Scientific Reports* 10.1 (2020), p. 20336.

[Liu+16] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. “Ssd: Single shot multibox detector”. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer. 2016, pp. 21–37.

[Ma+] Ma, J., Huang, Y., Qian, C., Kang, J., Liu, J., Zhang, H., and Hong, W. “Lgnet: Local and Global Point Dependency Network for 3d Object Detection”. In: *Available at SSRN 4264538* ().

[MZ22] Ma, Q. and Zhu, Y. “Keypoints Representation of Density-aware and the Spatial-Channel-wise Decoder for 3D Object Detection”. In: *2022 15th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE. 2022, pp. 1–6.

[Mao+23] Mao, J., Shi, S., Wang, X., and Li, H. “3D object detection for autonomous driving: A comprehensive survey”. In: *International Journal of Computer Vision* (2023), pp. 1–55.

[NLH21] Noh, J., Lee, S., and Ham, B. “Hvpr: Hybrid voxel-point representation for single-stage 3d object detection”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 14605–14614.

[Ona+23] Onat, N. C., Mandouri, J., Kucukvar, M., Sen, B., Abbasi, S. A., Alhajyaseen, W., Kutty, A. A., Jabbar, R., Contestabile, M., and Hamouda, A. M. “Rebound effects undermine carbon footprint reduction potential of autonomous electric vehicles”. In: *Nature Communications* 14.1 (2023), p. 6258.

[Par+22] Parekh, D., Poddar, N., Rajpurkar, A., Chahal, M., Kumar, N., Joshi, G. P., and Cho, W. “A review on autonomous vehicles: Progress, methods and challenges”. In: *Electronics* 11.14 (2022), p. 2162.

[PPS22] Pauwels, A., Pourmohammad-Zia, N., and Schulte, F. “Safety and Sustainable Development of Automated Driving in Mixed-Traffic Urban Areas—Considering Vulnerable Road Users and Network Efficiency”. In: *Sustainability* 14.20 (2022), p. 13486.

[Qi+17a] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[Qi+17b] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in neural information processing systems* 30 (2017).

[QLL22a] Qian, R., Lai, X., and Li, X. "3D object detection for autonomous driving: A survey". In: *Pattern Recognition* 130 (2022), p. 108796.

[QLL22b] Qian, R., Lai, X., and Li, X. "BADet: Boundary-aware 3D object detection from point clouds". In: *Pattern Recognition* 125 (2022), p. 108524.

[Red+16] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[RF17] Redmon, J. and Farhadi, A. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[RF18] Redmon, J. and Farhadi, A. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[Ren+15] Ren, S., He, K., Girshick, R., and Sun, J. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems* 28 (2015).

[Ren+23] Ren, S., Pan, X., Zhao, W., Nie, B., and Han, B. "Dynamic graph transformer for 3D object detection". In: *Knowledge-Based Systems* 259 (2023), p. 110085.

[RCG21] Roriz, R., Cabral, J., and Gomes, T. "Automotive LiDAR technology: A survey". In: *IEEE Transactions on Intelligent Transportation Systems* 23.7 (2021), pp. 6282–6297.

[Sch+23] Schinagl, D., Krispel, G., Fruhwirth-Reisinger, C., Possegger, H., and Bischof, H. "GACE: Geometry Aware Confidence Enhancement for Black-Box 3D Object Detectors on LiDAR-Data". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 6566–6576.

[STC19] Shahian Jahromi, B., Tulabandhula, T., and Cetin, S. "Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles". In: *Sensors* 19.20 (2019), p. 4357.

[She+21] Sheng, H., Cai, S., Liu, Y., Deng, B., Huang, J., Hua, X.-S., and Zhao, M.-J. "Improving 3d object detection with channel-wise transformer". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2743–2752.

[SZL22] Shi, C., Zhang, C., and Luo, Y. "Structure Guided Proposal Completion for 3D Object Detection". In: *Proceedings of the Asian Conference on Computer Vision*. 2022, pp. 4462–4478.

[Shi+23a] Shi, G., Wang, K., Li, R., and Ma, C. "Real-Time Point Cloud Object Detection via Voxel-Point Geometry Abstraction". In: *IEEE Transactions on Intelligent Transportation Systems* (2023).

[Shi+23b] Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., Wang, X., and Li, H. "PV-RCNN++: Point-voxel feature set abstraction with local vector representation for 3D object detection". In: *International Journal of Computer Vision* 131.2 (2023), pp. 531–551.

[Shi+20] Shi, S., Wang, Z., Shi, J., Wang, X., and Li, H. "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network". In: *IEEE transactions on pattern analysis and machine intelligence* 43.8 (2020), pp. 2647–2664.

[Shi+19] Shi, S., Wang, Z., Wang, X., and Li, H. "Part-a 2 net: 3d part-aware and aggregation neural network for object detection from point cloud". In: *arXiv preprint arXiv:1907.03670* 2.3 (2019).

[SLX15] Song, S., Lichtenberg, S. P., and Xiao, J. "Sun rgb-d: A rgb-d scene understanding benchmark suite". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 567–576.

[Sun+20] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al. "Scalability in perception for autonomous driving: Waymo open dataset". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2446–2454.

[Vas+17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[Vel+17] Velivckovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (2017).

[Wan+23] Wang, Q., Li, Z., Zhu, D., and Yang, W. "LiDAR-only 3D object detection based on spatial context". In: *Journal of Visual Communication and Image Representation* 93 (2023), p. 103805.

[WS21] Wang, Y. and Solomon, J. M. "Object dgcnn: 3d object detection using dynamic graphs". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20745–20758.

[Wan+19] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. "Dynamic graph cnn for learning on point clouds". In: *ACM Transactions on Graphics (tog)* 38.5 (2019), pp. 1–12.

[WWN19] Wang, Z., Wu, Y., and Niu, Q. "Multi-sensor fusion in automated driving: A survey". In: *Ieee Access* 8 (2019), pp. 2847–2868.

[Wu+22a] Wu, Y.-H., Zhang, D., Zhang, L., Zhan, X., Dai, D., Liu, Y., and Cheng, M.-M. "Ret3D: Rethinking Object Relations for Efficient 3D Object Detection in Driving Scenes". In: *arXiv preprint arXiv:2208.08621* (2022).

[Wu+22b] Wu, S., Sun, F., Zhang, W., Xie, X., and Cui, B. "Graph neural networks in recommender systems: a survey". In: *ACM Computing Surveys* 55.5 (2022), pp. 1–37.

[Xie+20] Xie, Q., Lai, Y.-K., Wu, J., Wang, Z., Zhang, Y., Xu, K., and Wang, J. "Mlcvnet: Multi-level context votenet for 3d object detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10447–10456.

[XZN22] Xu, Q., Zhong, Y., and Neumann, U. "Behind the curtain: Learning occluded shapes for 3d object detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 3. 2022, pp. 2893–2901.

[Yan+22a] Yang, H., Liu, Z., Wu, X., Wang, W., Qian, W., He, X., and Cai, D. "Graph r-cnn: Towards accurate 3d object detection with semantic-decorated local graph". In: *European Conference on Computer Vision*. Springer. 2022, pp. 662–679.

[Yan+22b] Yang, J., Wang, T., Ge, Z., Mao, W., Li, X., and Zhang, X. "Towards 3D Object Detection with 2D Supervision". In: *arXiv preprint arXiv:2211.08287* (2022).

[Yan+18] Yang, S., Xiang, Z., Wu, J., Wang, X., Sun, H., Xin, J., and Zheng, N. "Efficient rectangle fitting of sparse laser data for robust on-road obiect detection". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1–8.

[Yan+20] Yang, Z., Sun, Y., Liu, S., and Jia, J. "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.

[Yeo+21] Yeong, D. J., Velasco-Hernandez, G., Barry, J., and Walsh, J. "Sensor and sensor fusion technology in autonomous vehicles: A review". In: *Sensors* 21.6 (2021), p. 2140.

[YZK21] Yin, T., Zhou, X., and Krahenbuhl, P. "Center-based 3d object detection and tracking". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 11784–11793.

[Yue+18] Yue, K., Sun, M., Yuan, Y., Zhou, F., Ding, E., and Xu, F. "Compact generalized non-local network". In: *Advances in neural information processing systems* 31 (2018).

[Yur+20] Yurtsever, E., Lambert, J., Carballo, A., and Takeda, K. "A survey of autonomous driving: Common practices and emerging technologies". In: *IEEE access* 8 (2020), pp. 58443–58469.

[Zha+23] Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. *Hands-on Deep Learning*. Cambridge University Press, 2023. ISBN: 978-1009389433.

[ZC18] Zhang, M. and Chen, Y. "Link prediction based on graph neural networks". In: *Advances in neural information processing systems* 31 (2018).

[ZJ+22] Zhang, T., Jin, P. J., et al. "Roadside lidar vehicle detection and tracking using range and intensity background subtraction". In: *Journal of advanced transportation* 2022 (2022).

[Zha+19] Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. "Object detection with deep learning: A review". In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3212–3232.

[Zhe+22] Zheng, L., Ma, Z., Zhu, X., Tan, B., Li, S., Long, K., Sun, W., Chen, S., Zhang, L., Wan, M., et al. "Tj4dradset: A 4d radar dataset for autonomous driving". In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2022, pp. 493–498.

[Zho+19] Zhou, F., Cao, C., Zhang, K., Trajcevski, G., Zhong, T., and Geng, J. "Meta-gnn: On few-shot node classification in graph meta-learning". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, pp. 2357–2360.

[ZT18] Zhou, Y. and Tuzel, O. "Voxelnet: End-to-end learning for point cloud based 3d object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4490–4499.

[Zim+22] Zimmer, W., Ercelik, E., Zhou, X., Ortiz, X. J. D., and Knoll, A. “A survey of robust 3d object detection methods in point clouds”. In: *arXiv preprint arXiv:2204.00106* (2022).

[Zou+23] Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. “Object detection in 20 years: A survey”. In: *Proceedings of the IEEE* (2023).