Comparative Analysis of Point Sampling Strategies in Point-based 3D Object Detection

by

Rui Zhu

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Comparative Analysis of Point Sampling Strategies in Point-based 3D Object Detection

by

Rui Zhu

Supervisory Committee

Dr. Kin Fun Li, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Fayez Gebali, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Point-based 3D object detection has been receiving increasing attention as it can preserve the geometric information of a point cloud and avoid quantization errors or information loss caused by voxelization or projection. Point sampling plays an important role in point-based 3D detectors yet has not been thoroughly explored. In this research, we conduct a comparative analysis of three point sampling strategies to gain a deep understanding of the effect that each strategy imposes on the final performance and intermediate stages of the network. We introduce density-aware sampling and semantic-aware sampling strategies and fit them into the backbone of a lightweight and effective baseline model, aiming to reduce the density imbalance of the point cloud and better utilize semantic information. The density-aware strategy effectively balances the density but the inference time is not applicable for real-time application. Semantic-aware sampling biased on foreground points achieves a 0.19% improvement on the baseline. Analysis on statistics and visualization reveals future research direction. We build our models on MMDetection3D platform and evaluate the performance on KITTI dataset.

Contents

List of Tables

List of Figures

GLOSSARY

AP	Average Precision
ACC	Adaptive Cruise Control
ADAS	Advanced Driver-Assistance System
AEB	Automatic Emergency Braking
BSD	Blind Spot Detection
D-FPS	Farthest Point Sampling based on Euclidean distance, equal to FPS
DA-FPS	Density-Aware Farthest Point Sampling
FP	Feature Propagation
FPS	Farthest Point Sampling
F-FPS	Farthest Point Sampling based on Feature distance
HOG	Histogram of Oriented Gradients
IEA	International Energy Agency
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute of Chicago
LiDAR	Light Detection And Ranging
LKA	Lane Keeping Assit
mAP	mean Average Precision
NMS	Non-Maximum Suppression
R-CNN	Regional-based Convolutional Neural Networks
SA	Set Abstraction
SAE	Society of Automotive Engineers
SA-FPS	Semantic-Aware Farthest Point Sampling
SSD	Single Shot MultiBox Detector
ToF	Time of Flight
YOLO	You Look Only Once

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Kin Fun Li, for guiding me through my program. He has always been supportive of me both academically and professionally. His kindness and professionalism make my time at UVic a rewarding journey, which encouraged me to move forward to further my study in a doctoral program under his supervision.

Many thanks should also go to Chenchen Guo, Tim Hu, and Jinmin Huang from Akasha Technology Inc. A lot of inspiration has been gained from the cooperation with them. Their in-site point cloud data also offers me opportunities to verify my models.

I feel grateful to work with everyone in Dr. Kin's lab. It is a welcoming and positive team, a lot of happy and heartwarming clips with them have been saved in my mind. I wish everyone in our team big achievements in academia and profession.

Rui Zhu

Chapter 1

Introduction

Computer vision, starting out in the early 1970s, has been benefiting individuals, industries and our society in a wide range of applications such as medical imaging, machine inspection, warehouse logistics, self-driving vehicles, etc. [44] Among the fundamental tasks of computer vision, object detection aims to tell what and where are meaningful objects located in images and videos, which contributes a lot to other computer vision tasks, including instance segmentation, image description, object tracking, etc. [58] According to the latest IEA's annual Global Electric Vehicle Outlook [10], more than 10 million electric cars were sold worldwide in 2022 and sales are expected to grow by another 35% in 2023 to reach 14 million, which will take 18% sharing of the overall car market. Electric vehicle users are quite used to partial autonomous driving which relieves them from tiring driving. But few people know that it is the real-time and accurate object detection system that empowers the vehicle brain to understand the surrounding traffic and make decisions accordingly and promptly.

Object detection on images has been fast evolving in the past two decades. Before 2012, object detection was based on traditional methods that relied on hand-crafted features and classifiers. Classical detectors include Viola-Jones detector [46], Histogram of Oriented Gradients (HOG) detector [7], Deformable Par Models (DPM) detector [13], etc. With the introduction of AlexNet [21] in 2012, object detection entered a new era, which implements deep learning methods that learn features and classification from data. Some of the recognized detectors in this period are Region-based Convolutional Neural Networks (R-CNN) [16], Fast R-CNN [15], Faster R-CNN [37], You Only Look Once (YOLO) [36], Single Shot MultiBox Detector (SSD) [28], Mask R-CNN [17], etc. The achievement in object detection has rocketed in both

academia and industry. Large annotated image datasets boosted the research of detection models. Two popular datasets are PASCAL Visual Object Classes (VOC) Challenge [12] and Microsoft Common Objects in Context [27], both of which provide a standard dataset of images, annotation, and evaluation procedures. The development of GPU and parallel computing mechanisms make training and inference of big models much easier and faster than ever before. In the auto market, Tesla Autopilot claims to offer Full Self-Driving capability, which means the system can perform lane changing, acceleration, braking and parking tasks according to navigation and detected surrounding traffic through cameras and radars.

Since the invention of Velodyne's 64-laser LiDAR and its successful application in the DARPA Grand Challenge (2007) [38], object detection in the 3D space has drawn increasing attention and made huge progress. Information acquired by 2D cameras is affected by illumination, occlusion, and lack of depth. Depth collected from stereo cameras is inaccurate. Radar can provide the speed and distance of objects, but not their shape or appearance because of its low resolution [1]. Modern LiDARs, like Velodyne's, provide precise and high spatial resolution 3D point cloud data due to the directivity and energy concentration of the laser. Besides, since LiDAR actively emits laser and receives the echo, data quality is not hindered by light condition of the environment [49]. In addition to 3D imaging sensors, large 3D LiDAR datasets act as another strong force that drives 3D object detection to another level. The earliest and most popular 3D dataset is the KITTI dataset [14] published by Karlsruhe Institute of Technology and Toyota Technological Institute of Chicago in 2012. The dataset consists of 7481 training images and 7518 test images as well as the corresponding point clouds, comprising a total of 80,256 labeled objects [14]. Since its introduction, over 400 deep learning methods have been trained and evaluated on KITTI dataset, the latest top model has reached an average precision of 87.2% on moderate-level Car detection [48]. The nuScenes [2] and Waymo [43] datasets are later introduced that cover more scenes and provide better annotations. New evaluation metrics have also been proposed to complement the existing ones. The performance of 3D object detection has been unprecedentedly progressing with emerging deep learning technologies validated on the enlarging open datasets.

At the early stage of research on 3D object detection models, researchers naturally thought of applying 2D object detection methods to 3D object detection. The research in [4, 22] projects point cloud into Bird's Eye View (BEV) and front view, after which 2D CNN is implemented to extract feature maps and produce 3D pro-

posals. A fusion strategy is usually followed to merge the multi-view information. Partitioning a point cloud into voxels, from which features are extracted is another stream that the academia has been working on. With different voxelization and representation methods, the 2D detection paradigms or 3D convolution can be applied to generate 3D boxes. In 2017, Qi et al. [34, 35] introduced PointNet and its extension PointNet++. The latter can capture local and global features of individual points while considering their relationships in a larger context. The capability of providing informative point-wise features made PointNet++ a pioneering stone for point-based 3D object detection. Compared to projection-based or voxel-based methods, point-based methods can better preserve the geometric information of the points and avoid quantization errors or information loss caused by voxelization or projection [42].

Akin to object detection in images, there are two-stage and single-stage methods in 3D object detection. For two-stage methods, a region proposal network (RPN) is designed to produce coarse proposals. Then another module refines the bounding boxes and class labels in the second stage. A single-stage network, which is also called a single-shot detector, directly predicts class probabilities and regresses 3D bounding boxes. Two-stage methods were believed to produce more accurate predictions with more complicated and deliberately designed architectures. But emerging single-stage methods present comparable accuracy meanwhile consume fewer resources and run faster.

3DSSD [52] is the first lightweight and efficient point-based 3D single-stage object detection framework. This framework not only gets rid of the refinement stage but also abandons feature propagation (FP) layers which are applied for upsampling and broadcasting features to discarded points during downsampling. Although the network structure is much simpler, the performance of 3DSSD surpasses most state-of-the-art 3D object detectors on the Kitti dataset.

The simple but effective framework of 3DSSD became representative of the point-based models. The pipeline mainly consists of several set abstraction layers that downsample and encode the points with geometric and semantic features, a candidate layer that groups points into potential instances, and a prediction head that produces 3D bounding boxes and class scores. Follow-up researchers try different strategies in the main components of the framework to achieve better results or apply the framework to another scenario. Ning et al. [30] replace Farthest Point Sampling with Density-aware Farthest Point Sampling in order to sample more low-density points. A Center Density Attention module is proposed to model the relationship

between center point densities and feature vectors. Wang et al. [47] introduce a foreground-biased sampling strategy and ray-based feature grouping to learn better feature representation of the surface geometry of foreground objects. All of the subsequent studies are inspiring and show that point sampling strategies play an important role in the overall performance of a 3D object detection model.

It has been noticed that the point sampling strategies are not necessarily combined with the following layers in a model. But few researchers explored how different sampling strategies work in a single framework and whether the network can be more accurate or faster. This leads us to a question: which strategy is optimal? or which strategy serves better for certain circumstances? In this research, it is believed that a deep understanding of point sampling strategies helps researchers to design better point-based 3D object detection models under specific scenarios. While a strategy that achieves high detection accuracy within a short time frame is undoubtedly ideal, there are situations where a fast-running approach, even if it doesn't rank among the most accurate methods, can still be suitable for time-sensitive tasks that do not require top accuracy.

For this study, 3DSSD has been selected as the framework. Another two typical point sampling strategies are implemented and tested on this framework. For each strategy, various designs and parameters have been experimented and compared. The results are discussed in Chapter 4.

A paper [39] of our research has been accepted by the 5^{th} International Conference on Image, Video Processing and Artificial Intelligence. We list our contributions as follows:

1. We design the first SA layer with a density-aware sampling strategy in order to sample a point set with less density nonuniformity. Experiments show that the calculation of density is time-consuming and adds up more than ten times to the inference time. The sampling strategy also has the unintended consequence of introducing a high duplicate sampling rate in the subsequent SA layers which leads to degraded performance.

2. We append a segmentation head to the SA layers after the first layer to extract point-wise semantic information. Then use the foreground points as candidate points to generate proposals. With a fixed number of sampled points in each layer, various ratios between foreground and background points have been explored. A foreground-bias strategy achieves a 0.19% improvement on the

reproduced 3DSSD baseline.

3. The impacts of each sampling strategy on the intermediate layers have been showcased and analyzed through visualization and statistics methods, offering a deep understanding of how sampling strategies affect the performance and efficiency of the models in terms of both detection accuracy and computational speed. Future research directions are also proposed.

The rest of the thesis is organized as follows. Chapter 2 reviews the related literature on 3D object detection algorithms, sensors and open datasets. Considering LiDAR's wide application in 3D object detection, LiDAR-related methods and datasets are more focused on in this chapter. Chapter 3 introduces the fundamentals of point-based 3D object detection and the framework of our deep learning model. The design of backbones for 3 different point-sampling strategies is presented. Chapter 4 presents the setup and results of the experiments to explore the strategies. Further analysis and comparison are also included in this chapter. Chapter 5 concludes our findings and provides future work direction.

Chapter 2

Literature Review

This chapter presents a comprehensive literature review on 3D object detection, focusing on three key aspects that have significantly contributed to its progress. We begin by introducing the sensors applied in 3D object detection in Section 2.1. The continuous advancements in sensor technology have played a vital role in pushing the boundaries of 3D object detection towards higher resolution levels, enabling more accurate and detailed perception of objects in 3D space. Recognized products are introduced in this section. In section 2.2, we explore three prominent methodologies: project-based methods, voxel-based methods, and point-based methods. Each approach is critically analyzed, and prominent research on each approach is introduced. The last section 2.3 investigates the datasets used for training and evaluating deep learning models in the field of 3D object detection. These datasets play a crucial role in shaping the robustness and generalization capability of models.

2.1 3D Object Detection Sensors

Sensors have always been an empowering force for machines and computers to perceive the environment. The achievement of computer vision is closely linked with the usage of sensors such as cameras, radar, LiDAR, and sonar. 3D object detection, which adds a third dimension to conventional 2D images, was initially facilitated by stereo cameras which mimic the way humans perceive depth. However, the required computational power for the depth is substantial and the estimated depth is of low accuracy. The addition of a separate depth sensor greatly enhances perception capabilities. This led to the advent of RGB-D sensors, radar, and LiDAR, all of which

offer ranging capabilities. In this section, we introduce the evolution of these three sensors and typical products used in this field. Our focus is on 3D object detection on land so sensors like sonar which are mostly used underwater are not covered.

2.1.1 RGB-D Camera

RGB-D camera is the combination of a conventional color camera (RGB) with a depth sensor (D)[57]. An infrared projector is used to measure the depth of each pixel in the image. Based on the depth measurement principle, RGB-D cameras are divided into two categories: Triangulation-based and Time-of-Flight (ToF)-based. Triangulation approaches use the image under visible light to interpret the range based on the location of the pixel and the mirror angles. Two branches of triangulation-based sensing approaches exist: passive triangulation and active triangulation [11]. In the case of passive systems utilized in consumer-grade applications, stereo vision systems are commonly employed, requiring a pair of calibrated cameras. In active triangulation systems, the second camera is replaced by a calibrated light projector, which projects a specific light pattern to the scene. These active systems are also referred to as structured light systems. A Time-of-Flight (ToF) system determines the distance by calculating the round-trip time taken by light to travel to the surface, get reflected back, and reach the ToF system. This time delay can be directly estimated by utilizing a high-resolution clock to measure the time interval between the emission and reception of light pulses (referred to as pulse-based ToF). Alternatively, it can be indirectly determined by measuring the phase shift of an amplitude-modulated optical signal (known as phase-based ToF)[11]. Compared to ToF system, triangulation approaches reach a shorter detection range but offer higher scanning accuracy which can be used to generate CAD models and help reverse engineering.

Kinect, released by Microsoft in 2010, is the first consumer-grade RGB-D camera based on the structured light principle. It was originally designed for Xbox 360 as a motion controller peripheral with which gamers can play a game just with body movement. In 2013, Microsoft released a second generation of Kinect with improved tracking capabilities for Xbox One. Phase-based ToF was adapted for this version. In addition, Kinect for Windows SDK was also introduced for programmers to develop applications for Windows PCs. Kinect for Xbox One was terminated in 2017, but Microsoft continued to use Kinect technology in other products and applications such as computer vision and speech models.

In 2012, ASUS introduced Xtion, a similar device to Microsoft Kinect with a smaller size and lower power consumption. Like the first-generation Kinect, it also used the structured-light principle and had a depth range between 0.8m and 3.5m. ASUS Xtion was designed for developers to create various applications and games that use gesture and voice input, such as fitness, dance, sports, education, and entertainment.

Intel introduced its first-generation RGB-D camera RealSense F200 in 2014. This camera was based on the structured light principle and was mainly used for gesture recognition and face scanning. Realsense SR300 was an improved version of the F200 that had a higher resolution, frame rate, and accuracy. It was introduced 2 years after F200. The D400 series came out in 2018 and used two infrared cameras and an infrared projector to capture depth images. Different from the previous models, they implemented the active stereo vision principle and were designed for both indoor and outdoor conditions. In 2019, L515 was introduced and used a laser scanner and a global shutter sensor for depth measuring. It was based on the ToF principle and optimized for low power consumption and high accuracy.

Besides the brands mentioned above, Orbbec and Occipital are also key players in the RGB-D camera market. Table 2.1 [11] shows the commercial RGB-D cameras and their classification according to the measurement principle.

Like the first generation of Microsoft Kinect, the original applications of RGB-D cameras were gesture recognition and face scanning. Then the applications expanded to 3D reconstruction and mapping, 6D pose estimation, augmented reality, virtual reality, etc. The biggest difference between RGB-D cameras and LiDAR is the sensing distance range. A typical depth range of an RGB-D camera is 0.4-10m [11], while a LiDAR can detect objects hundreds of meters away. For autonomous vehicles, the depth range of RGB-D cameras is not sufficient for high-speed driving or long-distance sensing.

2.1.2 Radar

Radar is a sensing technology that utilizes radio waves to detect and locate objects in its vicinity. Traditionally, radar has been widely used for military and aviation purposes, such as detecting enemy aircrafts, ships, and missiles. Civil applications include air traffic control, weather forecasting, and navigation [31]. In recent years, there has been a growing interest in leveraging radar for 3D object detection, partic-

Measurement principle	Commercial RGB-D Cameras
Active stereo vision	Intel RealSense R200
	Intel RealSense D400 series
	Orbbec Astra Stereo
	Occipital Structure Core
Structured-light (temporal encoding)	Intel Realsense F200
	Intel Realsense SR300
	Intel Realsense SR305
Structured-light (spatial encoding)	Microsoft Kinect I
	Asus XTion 2
	Orbbec Astra
	Occipital Structure
Time-of-flight	Microsoft Kinect II
	Microsoft Kinect III
	Intel Realsense L515

Table 2.1: Classification of Commercial RGB-D Cameras [11]

ularly in the context of autonomous vehicles and advanced driver-assistance systems (ADAS).

As one of the world's leading auto parts manufacturers, BOSCH provides a radar product line for applications like driver assistance systems, surrounding sensing, and object detection. LRR4 is a long range radar sensor introduced by BOSCH in 2013. It can detect objects up to 250 meters ahead and covers an angle of ±18 degrees. Supported functions include adaptive cruise control, forward collision warning, and emergency brake assist. In 2016, a mid range radar sensor modeled MRR1plus was released. Compared to LRR4, MRR1plus has a shorter detection range of 160 meters but covers a bigger angle of ±45 degrees. it was designed for blind spot detection, lane change assist, and rear cross-traffic alert. A rear mid range radar sensor was rolled out in 2018. It can detect objects up to 80 meters behind and covers an angle of ±150 degrees. SRR3 was revealed in 2019 to complement the product line. It is a short range radar with a detection range of 30 meters around the vehicle and covers an angle of ±90 degrees.it supports functions such as parking assist, exit warning, and pre-crash sensing.

Continental AG is another well-known German company that produces radars. Its long range products include ARS4-A and ARS4-B. ARS4-A can detect objects up to 250 meters ahead and covers an angle of ±45 degrees. While ARS4-B is an upgraded version that can reach objects in 300 meters and covers an angle of ±45

degrees. ARS4-B has a bandwidth of 1 GHz, offering a higher range resolution than ARS4-A, the bandwidth of which is 0.5 GHz. SRR520 and SRR600 are Continental's surround radar. Both have a detection range of 200 meters. The angle coverage is ±150 degrees and ±175 degrees respectively. The latter offers a higher resolution and a more powerful processor.

Unlike BOSCH and Continental, Ainstein AI is a U.S. company that focuses on radar sensing and perception solutions, but the applications are not limited to automobiles. It also serves aerospace, industry, sports, as well as smart cities. The company integrates radar sensors with built-in, high performance radar processors and advanced radar processing algorithms, enabling reliable detection and tracking of vehicles and pedestrians under inclement conditions. WAYV Air is a product for industry and IoT applications, it is an indoor, short-range (6 meters) radar sensor designed for people tracking and occupancy detection. The azimuth and elevation angle are both 120 degrees. Applications include elder care, people counting, space utilization, etc. US-D1, LR-D1, LR-D1 pro are designed for aerospace. They offer different ranges of altitude measurement and precision that can be fit into various UAV and drone capabilities. O-79 and T-79 serve specialty vehicles. The detection range of O-79 is 20 meters while T-79 can reach 80 meters. But O-79 is of higher range resolution (0.2 meters compared to 0.78 meters). They both work well in all environments and detect both moving and stationary objects.

Arbe Robotics is a NASDAQ listed company that claims to be the world's 1st company to demonstrate ultra-high-resolution 4D imaging radar with post-processing and SLAM. Its competency originated from a holistic integration of patented processor chipset, long-range perception radar, surround imaging radar, and perception algorithms. The powerful chipset includes a processor chip that integrates radar processing unit (RPU) architecture with embedded radar signal processing algorithms, a Rx receiver chip, and a Tx transmitter chip. The 350-meter long range radar, named Phoenix, achieves a range resolution of 7.5 centimeters, an azimuth resolution of 1 degree, and an elevation resolution of 1.7 degrees. The surround imaging radar, named Lynx, complements Phoenix radar to deliver a 360-degree view around the vehicle. Lynx can detect up to 260 meters with a resolution of 10 centimeters. The azimuth and elevation resolution is 2.5 and 6.4 degrees respectively. AI-powered algorithms identify, classify, and track objects in 360 degrees, creating a full free space map around the vehicle, as well as an analysis of the evolving hazards sensed by the radars.

The number of radar producers is far larger than the above-mentioned four and is still growing. The increasing number is driven by the high demand from the explosive autonomous vehicle market [8]. Radar has been the default sensor for emergency braking and adaptive cruise control thanks to its capability of range detection and velocity estimation at a relatively low cost. Compared to LiDAR and Cameras, radar works more reliably under low visibility conditions such as snow, rain, dust, and fog. Progress has been made to implement radar in 3d object detection for autonomous vehicles or traffic monitoring. But due to the intrinsic measuring principle, the resolution of radar can not reach that of LiDAR, which can create a detailed 3D image of the environment.

2.1.3 LiDAR

Since the invention of laser in the early 1960s, LiDAR has been developing and applied to various applications. Wind sensing is one of the earliest applications starting in the 1960s. Then the military explored the huge potential of light detection and ranging technology. As rangefinders, LiDARs were used for proximity fuses and weapon guidance. When functioning as an altimeter, LiDAR was used to measure the surface topography of the moon. From 1979 until 1984, a DARPA program named the Autonomous Terminal Homing (ATH) used coherent LiDAR to navigate cruise missiles [29]. As mentioned in Chapter 1, All of the DARPA Grand Challenge (2007) finishers used at least one LiDAR on their autonomous vehicles. Since then, commercial LiDAR has become more prevalent, especially in the application of self-driving cars. According to industry estimates, around 70 companies worked on LiDAR products in 2018 [20].

Velodyne Lidar, merged with Ouster in February 2023, is one pioneering and leading company that produces a wide range of LiDAR products for applications such as autonomous vehicles, Drone/UAV, mapping, robotics, security, and smart cities. Puck LiDAR sensor (previously VLP-16), is a classical model on the market released in 2014. It is a cost-effective surround-view LiDAR sensor that provides rich computer perception data to enable autonomy and enhance safety. It has a range of 100 meters, a resolution of 0.2 degrees, a field of view of 360 degrees horizontal and 30 degrees vertical, and a data rate of 300,000 points per second. In 2016, the Puck family expanded to Puck LITE, Puck Hi-Res, and Ultra Puck. The Puck LITE is the lightest surround-view LiDAR sensor on the market that retains the performance

and reliability of the original Puck. The Puck Hi-Res provides higher resolution data in a narrow vertical field of view to enable object identification at longer ranges. It has a range of 100 meters, a resolution of 0.1 degrees horizontal and 0.3 degrees vertical. The vertical view decreased to 20 degrees. The Ultra Puck has a range of 200 meters, with other specifications consistent with Puck. Besides Puck family, Velodyne introduced Alpha Prime in 2019, which is a high-performance surround-view LiDAR that delivers unrivaled perception for safe and reliable autonomous mobility. It has a range of 300 meters, a resolution of 0.1 degrees, a field of view of 360 degrees horizontal and 40 degrees vertical. A solid-state LiDAR Velarray was released in 2017, It has a range of 200 meters, a resolution of 0.1 degrees horizontal and vertical, a field of view of 120 degrees horizontal and 16 degrees vertical. Along with hardware product, Velodyne also developed a software solution named Vella that enhances vehicle safety by enabling advanced driver assistance systems (ADAS) features such as automatic emergency braking (AEB), adaptive cruise control (ACC), lane keeping assist (LKA), blind spot detection (BSD).

Luminar is an autonomous vehicle sensor and software company with the vision to power every autonomous vehicle by delivering the only LiDAR capable of making them both safe and ubiquitous. Iris is Luminar's flagship LiDAR sensor that is designed for series production and highway autonomy. It has a range of 250 meters, a resolution of 300 points per square degree, a field of view of 120 degrees horizontal and 30 degrees vertical, and a data rate of up to 1.5 million points per second. Luminar offers Sentinel as its full-stack platform for safety and autonomy. It features Luminar's lidar, perception software, HD mapping technology, as well as control and planning software that enables proactive safety and highway autonomy. It accelerates time-to-market for automakers by providing a ready-to-deploy solution that integrates Luminar's hardware and software components.

Valeo is a French multinational company that provides automotive components and systems for various applications such as lighting, thermal management, comfort, and driving assistance. Valeo has developed SCALA®, the first mass-produced vehicle-grade LiDAR for advanced driver assistance systems (ADAS) and autonomous driving. The first generation of its LiDAR is SCALA® Gen1, released in 2017. It is a 4-channel mechanical scanning LiDAR that uses a rotating mirror to steer the laser beam across the scene. It has a range of 150 meters, a resolution of 0.25 degrees horizontal and 3.2 degrees vertical, a field of view of 145 degrees horizontal and 3.2 degrees vertical, and a data rate of 12,800 points per second. SCALA® Gen2

was introduced in 2020. It is an improved version of SCALA® Gen1 that uses a dual-axis MEMS mirror to steer the laser beam across the scene. It has a range of 200 meters, a resolution of 0.1 degrees horizontal and vertical, a field of view of 120 degrees horizontal and 16 degrees vertical, and a data rate of up to 1 million points per second. SCALA® Gen2 has been equiped on Honda Legend and Mercedes-Benz Class S, which both claimed to reach SAE level 3 automated driving.

AEye was founded in 2013 by Luis Dussan, a leading aerospace designer of targeting systems for fighter jets. He understood self-driving cars would face similar challenges: the need to see, classify, and respond to an object in real time. So he gathered an engineering group from NASA, Lockheed, the USAF, and DARPA. The team created AEye's patented 4Sight™ LiDAR platform for sensing applications for automotive, trucking, smart infrastructure, logistics, and rail. In 2020, AEye released 4Signt A and 4SigntS. 4Sight A provides advanced perception for ADAS and autonomous driving applications. It has a range of 300 meters, a resolution of 0.1 degrees horizontal and vertical, a field of view of 120 degrees horizontal and 45 degrees vertical. While 4Sight S is a solid-state LiDAR sensor that uses an optical phased array (OPA) to steer the laser beam across the scene without any moving parts. It has a shorter range of 200 meters, and the field of view and resolution is the same as 4Sight A. A high-resolution, ultra-long range version named 4Sight M is introduced in 2021. It can reach a customized range of 1000 meters, with a resolution of 0.1 degrees horizontal and vertical.

A large number of LiDAR producers also have been emerging in China. Hesai Technology, Robosense, Livox, Neuvition are the outstanding ones that received sufficient investment and earns a large share of the Chinese market. Technological innovation, economies of scale, and market competition have been driving LiDAR to a longer range, higher resolution, wider field of view, faster processing, meanwhile lower price. In 2016, the Puck LiDAR from Velodyne was over 8000 U.S. dollars, but in January 2023, Livox published the price of its Mid-360 at 600 U.S. dollars, less than one-tenth of Puck LiDAR. As pulsed lasers, LiDAR systems often have a minimum operational range. Objects that are too close to the sensor may not be detected accurately because the light pulses return too quickly for the sensor to process. The present LiDARs on the market can achieve range holes less than 0.5 meters. The long-range capability, high resolution, and active sensing enable LiDAR to detect small objects such as pedestrians in the distance even in the evening, which can be hardly achieved by cameras or radars. The further development of LiDAR along with

the 3D object detection models will benefit the future of transportation as well as various 3D sensing applications. Considering the popularity and promising potential of LiDAR sensors, we carry out the research on a dataset collected by LiDARs.

2.2 Deep Learning Models for 3D Object Detection

The capability to learn features from data and produce accurate object prediction makes deep learning models pervasive in computer vision tasks, both in 2D and 3D space. Compared to object detection from 2D images, 3D object detection poses unique and substantial challenges, involving complex classification and localization in 3D space. Some methods used in 2D space are migrated to work under 3D scenarios. While many researchers propose strategies explicitly for 3D tasks. This section investigates three fundamental branches of methods employed in 3D object detection. Notably, some models incorporate a fusion of different strategies, even if they are categorized under one primary branch based on the dominant technique they employ

2.2.1 Projection-based Models

At the early stage of research on 3D object detection, researchers tended to implement methods that had been proven successful on 2D object detection tasks. To realize this, projecting point cloud to 2D grids is a normal preprocessing approach. The projection can be either the bird's view or a frontal view. Researchers tend to implement a fusion of the two projection strategies.

Chen, el at. [4] proposed a Multi-View 3D object detection network (MV3D) that combines LiDAR Bird View, LiDAR Front view and Image (RGB) as input and predicts the full 3D extent of objects in 3D space. A multi-view encoding scheme to obtain a compact and effective representation for a sparse 3D point cloud was implemented first. Then the network is divided into two parts: a 3D Proposal Network and a Region-based Fusion Network. The 3D proposal network utilizes a bird's eye view representation of point cloud to generate highly accurate 3D candidate boxes through 2D CNN layers. The 3D object proposals can be projected to any view in 3D space. The multi-view fusion network extracts region-wise features by projecting 3D proposals to the feature maps from multiple views. Then region-based fusion networks combine features from multiple views, jointly classify object proposals, and

do oriented 3D box regression. The speed of MV3D is too slow (2.8 fps) for practical applications as it generates proposals for each view and implements a deep fusion strategy, which both demand a high level of computation.

Ku, el at. [22] introduced an Aggregate View Object Detection (AVOD) network for autonomous driving scenarios. Both LiDAR point clouds and RGB images have been used to maximize the performance. RGB images provide a frontal view input, while point clouds have been projected into a bird's view. 2D CNN is implemented to generate feature maps from the two views. A fusion of the two views of feature maps is implemented and predicts detected objects through fully connected layers. NMS is utilized to filter out the best ones. The AP of AVOD is 10% higher than MV3D. Meanwhile, by implementing Feature Pyramid fusion architecture, the network parameters are approximately 16% that of MV3D. The runtime of AVOD is 1/4 of MV3D. However, the fusion here happens at a coarse level, which may bring significant resolution loss.

Liang, el at. [26] also try to boost the detection performance from both LiDAR and cameras. They propose a 3D object detector that reasons in bird's eye view and fuses image features by learning to project them into BEV space. To achieve this, an end-to-end learnable architecture that exploits continuous convolutions to fuse image and LiDAR feature maps at different levels of resolution is introduced. The proposed continuous fusion layer is capable of encoding dense accurate geometric relationships between positions under the two modalities. Compared to AVOD, this method achieves higher 3D AP (Easy+8.95%, Moderate+0.22%, Hard+5.66%) with shorter inference time (-0.02s) on Kitti Vision Benchmark.

2.2.2 Voxel-based Models

Voxel-based models first transform unstructured raw points into regular voxel grids with diverse strategies. Early models use 3D fully convolutional networks to extract features from the grids, which suffer from inefficiency. Then researchers come up with varying methods to improve the performance and efficiency.

Li, et al. [24] simply encode point clouds into binary voxels. {0, 1} is used to present whether there is any point observed at the corresponding grid elements. The input point cloud is discretized into grids of 10cm edge length. Then 3D Fully Convolutional Network is implemented. 3 convolutional layers down-sample grids into feature maps. A deconvolutional layer output classification and bounding boxes.

The downside of this method is the demand of computational resources.

VoxelNet [56], introduced by Apple Inc., implemented a different way of voxelization. The size of voxels is pre-defined to 0.4×0.2×0.2 (D×H×W) meters, then a group of points within each voxel are transformed into a unified feature representation through the voxel feature encoding (VFE) layer, using PointNet[34]. 3D convolution further aggregates local voxel features, transforming the point cloud into a high-dimensional volumetric representation. Finally, a RPN consumes the volumetric representation and yields the detection result.

The computational cost of VoxelNet makes it difficult to use for real-time applications. To address the challenges in 3D convolution-based detections, Yan, el at. [51] introduced SECOND (Sparsely Embedded Convolutional Detection) which maximizes the use of rich 3D information present in point cloud data. Spatially sparse convolutional networks are introduced for LiDAR-based detection and are used to extract information from the z-axis before the 3D data are down-sampled to something akin to 2D image data. In comparison to a dense convolution network, their sparse-convolution-based detector achieves a factor-of-4 speed enhancement during training on the KITTI dataset and a factor-of-3 improvement in the speed of inference.

Lang, el at.[23] introduced PointPillars which divides point clouds into vertical columns (pillars) on x-y plane. PointNet is then utilized to learn the feature representation of each pillar. The point cloud is converted to a pseudo-image of size (C, H, W), where C is the dimension of the feature vectors and H, W indicates the original pillar positions. The author then implements 2D CNN on the pseudo-image and a followed SSD detection head detects and regresses 3D boxes. Experimentation shows that PointPillars outperforms previous encoders with respect to both speed and accuracy by a large margin. On the Kitti test 3D detection benchmark, PointPillars achieves a mAP of 59.20 at 62 Hz, while SECOND gets a mAP of 56.69 at 20Hz and VoxelNet only reaches a mAP of 49.05 at 4.4Hz.

Deng, el at. [9] believe that voxel-based methods can offer sufficient detection accuracy without requiring precise positioning of raw points. To achieve this, they designed a voxel-based two-stage framework named Voxel R-CNN. The paper proposes a novel voxel RoI pooling layer that extracts features from 3D regions of interest (RoIs) directly from the voxel features, without converting them to point features. This layer enables the use of a region proposal network (RPN) and a detection head that operate on 2D bird-eye-view (BEV) images, which are more efficient and accurate than 3D counterparts. The paper also introduces a 3D backbone network that

consists of 3D convolutional layers and residual blocks to extract voxel features from the point cloud. The model outperforms PointPillar, SECOND, and VoxelNet by a big margin, meanwhile runs at a real-time speed of 25 FPS.

Voxel-based methods demonstrate good detection performance with promising efficiency, especially with optimized 3D convolution strategies. It still suffers from quantization loss, but the emerging state-of-the-art voxel-based papers show that various point-based methods such as PointNet/PointNet++ have been implemented jointly to complement the deficiency and boost detection accuracy.

2.2.3 Point-based Models

Point-based Detection methods directly process point clouds for 3D object detection. Much attention has been drawn to point-based detection since the advent of PointNet[34] and its evolved version PointNet++[35], which output pointwise features or features of a group of points for object classification, part segmentation, instance segmentation, etc. PointNet/PointNet++ does not directly generate 3D object proposals, but the informative encoding of the points makes it the basis for a lot of point-based 3D object detectors. So in this section, PointNet/PointNet++ will be reviewed first followed by several well-known 3D object detection models.

PointNet was introduced by Charles R. Qi in 2017. The motivation behind this research is that when reasoning about 3D geometric data, the typical convolutional architectures require highly regular input data formats. But the point cloud is irregular and sparse, needing to be transformed into regular 3D voxel grids or collections of images before being fed into a deep net architecture. This transformation makes data voluminous, meanwhile introducing quantization artifacts that can obscure natural invariances of the data. PointNet takes raw point cloud data as input, then applies input and feature transformation through two symmetric functions and two following shared multi-layer perceptron. Pointwise features have been obtained after the feature transformation. Classification or Segmentation then can be carried out with local and global feature aggregation. The result shows that on tasks like object classification, part segmentation and semantic segmentation, PointNet obtain on par or better performance than state of the arts on a standard benchmark. The basic idea of PointNet is to learn a spatial encoding of each point and then aggregate all individual point feature to a global point cloud signature. However, PointNet does not capture local structures induced by the metric space points live in, limiting its

ability to recognize fine-grained patterns and generalizability to complex scenes.

Charles R. Qi proposed PointNet++ to evolve PointNet later in 2017. PointNet++ aims to address two issues: how to generate the partitioning of the point set, and how to abstract sets of points or local features through a local feature learner. The two issues are correlated because the partitioning of the point set has to produce common structures across partitions so that the weights of local feature learners can be shared. The second issue can be addressed by PointNet. To solve the first issue, PointNet++ introduces a sampling layer and a grouping layer. Farthest point sampling (FPS) is implemented to choose a subset of points. Compared with random sampling, it better covers the entire point set given the same number of centroids. With the sampled centroids, ball query works to find all points that are within a radius of the query point(centroids). A combination of sampling, grouping and PointNet make a set abstraction layer, which down-samples the points and abstracts local structure information into a smaller number of feature vectors. Experimentation shows that on ModelNet40 shape classification, PointNet++ achieves an accuracy of 91.9%, compared to 89.2% of PointNet.

PointRCNN [40] is one of the first models that directly generate 3D box proposals and detection results from raw point clouds. The network is composed of two stages: stage-1 for the bottom-up 3D proposal generation and stage-2 for refining proposals in the canonical coordinates to obtain the final detection results. Instead of generating proposals from RGB image or projecting point cloud to bird's view or voxels as previous methods do, PointRCNN's stage-1 sub-network directly generates a small number of high-quality 3D proposals from point cloud in a bottom-up manner via segmenting the point cloud of the whole scene into foreground points and background. The stage-2 sub-network transforms the pooled points of each proposal to canonical coordinates to learn better local spatial features, which is combined with global semantic features of each point learned in stage-1 for accurate box refinement and confidence prediction. Testing on the 3D detection benchmark of KITTI dataset shows that the model outperforms previous state-of-the-art methods by using only point cloud as input.

In 2019, The author of PointNet/PointNet++ proposed VoteNet [32], another influential network in 3D object detection. VoteNet is an end-to-end 3D object detection network based on a synergy of deep point set networks and Hough voting. A major challenge in directly predicting the bounding boxes from scene points is that a 3D object centroid can be far from any surface point thus hard to regress accurately in

one step. To address the challenge, the author implements Hough voting to generate new points that lie close to object centers, which are then grouped and aggregated to generate box proposals. The network is trained and tested on two indoor 3D object detection datasets: SUN RGB-D [41] and ScanNet [6]. The result is even better than the prior arts that use both RGB and geometry or even multi-view RGB images.

3DSSD is the first lightweight and efficient point-based 3D single stage object detection framework [52]. Observing that feature propagation (FP) layers and the refinement stage consume half of the inference time in point-based methods, the author proposes a novel sampling strategy based on feature distance, called F-FPS, which effectively preserves interior points of various instances. Points sampled by F-FPS are taken as representative points to generate candidate points through the voting strategy in VoteNet. Features are then extracted from clustered points around candidate points and fed into an anchor-free regression head to predict 3D bounding boxes. The evaluation on the KITTI dataset shows that 3DSSD outperforms all state-of-the-art voxel-based single stage method and achieve comparable performance to all two stage point-based methods at a much faster inference time.

Ning, el at. [30] notice that 3DSSD concentrated on the imbalance between the number of foreground and background points, but the nonuniform density has not been emphasized. The proposed DA-3DSSD replaces Farthest Point Sampling (FPS) with density-aware points sampling, in order to include the interior points in the low-density area. Then a center density attention module is added to enhance the classification ability of the network. The model achieved a comparable result on KITTI test set on car class.

RBGNet [47] considers previous point-based methods not leveraging the surface geometry of foreground objects to enhance grouping and 3D box generation. To address this, it proposes two novel strategies to boost the performance of 3D object detection by implicitly learning from foreground object features. Firstly, it proposes the ray-based feature grouping that could learn better feature representation of the surface geometry of foreground objects. Secondly, it implements a foreground-based feature extraction module to sample more points on the object surface while keeping the coverage rate for the whole scene. The network is evaluated on SUN RGB-D and ScanNet, achieving state-of-the-art performance.

IA-SSD [54] aims to improve the efficiency of point-based 3D object detection. It emphasizes the importance of foreground points for object detectors and exploits two learnable, task-oriented, instance-aware downsampling strategies to hierarchically

select the foreground points belonging to objects of interest. A contextual centroid perception module is also proposed to further estimate precise instance centers. The average precision of the network on KITTI benchmark is higher than 3DSSD but the speed is nearly 4 times faster.

Previous research shows that farthest point sampling based on Euclidean distance, farthest point sampling based on feature distance, density-based sampling, and semantic-based sampling are common sampling methods used in point-based 3D object detection. But how different sampling strategies work in a single framework and whether the network can be more accurate or faster is not obvious or deeply explored. An enhanced understanding of point sampling strategies helps researchers to design better point-based 3D object detection models under specific scenarios.

2.3 3D Object Detection Datasets

All the state-of-art 3D object detection algorithms nowadays are deep learning models that require a substantial amount of labeled training data to learn and generalize patterns effectively. Datasets provide the foundation for training these models by providing large-scale labeled examples. The following subsections introduce several widely used indoor and outdoor datasets that have served the 3D scene understanding and pattern recognition. Besides offering 3D bounding boxes and class labels for 3D object detection, these datasets also provide annotations for tasks such as surface reconstruction, semantic segmentation, odometry, 3D object tracking, etc.

2.3.1 SUN RGB-D Dataset

SUN RGB-D dataset is an indoor dataset introduced by Song, et al. [41] from Princeton University. The introduction was in 2015 and motivated by the strong demand for large-scale RGB-D data which not only offer annotation and evaluation metrics in 2D image domain but also in 3D space. The dataset contains 10,225 RGB-D images with dense annotations in both 2D and 3D, including objects and room layout. The objects are annotated with 146,617 2D polygons and 64,595 3D bouning boxes, 3D polygons are offered to label the room layout. There are 47 scene categories and approximately 800 object categories in total for SUN RGB-D dataset.

2.3.2 ScanNet

ScanNet [6] is another popular indoor dataset initially introduced in 2017. It provides much more scenes and images than previous datasets. 1513 scenes were scanned in 707 distinct spaces, providing 2.5M images annotated with 3D camera poses, surface reconstructions, and semantic segmentation. CAD model placements are also provided for a subset of the scans. In 2018, ScanNet v2 was presented with new semantic label and instance annotations. For all 1513 original scans, the updated annotations cover approximately 90% of the surface versus the previous 63%. Some segments annotated as "remove" no more exist as the meshes are more deliberately reconstructed.

2.3.3 The KITTI Dataset

The KITTI dataset [14] is one of the most widely used datasets for 3D object detection in autonomous driving. It was introduced in 2012 by Andreas Geiger et al., providing high-resolution RGB images and 3D point cloud data captured from a Velodyne LiDAR sensor, along with ground truth annotations for several object classes such as cars, pedestrians, and cyclists. There are 7481 images/point clouds for training, and 7518 images/point clouds for testing. KITTI offers annotations for different tasks, including object detection, tracking, and road segmentation, making it suitable for multiple research focus. Besides the labeled data, KITTI also provides the evaluation metric for performance comparison. For 3D object detection task, the evaluation metric is the average precision (AP) based on the 3D bounding box overlap ratio between the predicted and ground truth objects. The AP is computed for different classes (car, pedestrian, cyclist) and different difficulty levels (easy, moderate, hard) based on the object size, occlusion level, and truncation level. The AP is also computed for different IoU thresholds (0.7 for car and 0.5 for pedestrian and cyclist) to measure the localization accuracy. There are also metrics for visual odometry and object tracking tasks.

2.3.4 The ApolloScape Dataset

ApolloScape [19] was introduced by Baidu Inc. in 2018 for autonomous driving. Compared with the previous datasets, it offers a larger diversity of scenarios by collecting data from multiple cities, traffic conditions, and daytimes. The dataset

provides stereo driving videos (100+ hours), videos at the same site under different day times (morning, noon, night), dense per-pixel per-frame semantic labeling (35 classes, 144K+ images), semantic 2D segmentation (8 classes, 90K+ images), and 2D car key points and 3D car instance labeling (70K cars). The supported tasks include 3D LiDAR object detection and tracking, scene paring, lane segmentation, self-localization, trajectory prediction, etc. The dataset is mainly used by Chinese institutions.

2.3.5 The nuScenes Dataset

The nuScenes dataset [3] is a large-scale dataset designed for autonomous driving research. It contains a diverse collection of sensor data, including high-resolution camera images, LiDAR point clouds, radar data, and ego-motion information. The dataset covers various urban driving scenarios captured in different cities, offering a wide range of environmental conditions and traffic scenarios. nuScenes provides detailed annotations for object detection, tracking, and motion forecasting, including 3D bounding boxes, semantic segmentation, and instance segmentation. It includes various object classes like vehicles, pedestrians, bicycles, and more. The dataset also includes information about drivable areas, lanes, and traffic signs. nuScenes is notable for its comprehensive nature, large-scale data size, and extensive annotation information, making it suitable for various perception tasks and advanced research in autonomous driving.

2.3.6 The Waymo Open Dataset

The Waymo Open Dataset [43] is one of the largest datasets for machine perception and autonomous driving introduced in 2019. It provides two subsets: the perception dataset with high-resolution sensor data and labels for 2,030 segments; and the motion dataset with object trajectories and corresponding 3D maps for 103,354 segments. The perception dataset offers bounding box labels, 2D video panoptic segmentation labels, key point labels, 3D semantic segmentation labels, and association between 2D and 3D bounding boxes. As for bounding box labels, 4 object classes - vehicles, pedestrians, cyclists, and signs - are provided. All 12.6M 3D bounding boxes are associated with tracking IDs on lidar data. The data was collected by multiple sensors, including 1 mid-range LiDAR, 4 short-range LiDAR, and 5 cameras (front and sides). Since the introduction, over 120 3D object detection models have been trained and

evaluated on Waymo perception dataset.

Among all the datasets reviewed above, the KITTI dataset is the earliest and most widely used. The high-quality annotation and globally recognized evaluation protocols make it still work as the training and evaluating dataset for the latest SOTA algorithms. The subsequent datasets offer a wider variety of scenarios, annotations or applications. This research is focused on 3D objection detection, and the baseline model is trained and tested on KITTI dataset, so the KITTI dataset is chosen as the test bed for the comparison of different sampling strategies in this research.

Chapter 3

Methodology

Point-based methods in 3D object detection have made considerable progress since the advent of PointNet/PointNet++, which directly encode raw points into informative features for a range of recognition tasks. High detection accuracy has been achieved by the emerging point-based methods. But the computation on point-wise feature is quite demanding since the order of magnitude of points in the point cloud is tens of thousands, which can be a bottle neck for models to achieve real-time efficiency. To lighten the computation load, most point-based models implement downsampling to reduce the point number meanwhile maintain the accuracy. This research adopts a simple uniform framework and investigates how different point sampling strategies can influence the detection results and how much resource is needed for each specific strategy.

First, we noticed that besides farthest point sampling which is adopted by PointNet++, there exist other strategies influencing the detection result. We decide that the density imbalance of point distribution and semantic information of point-wise features are the most worth exploring.

We adopt 3DSSD [52] as the baseline and framework to compare and explore the researched sampling strategies.

Inspired by Ning, el at. [30], we calculate a normalized point-wise density based on the number of points around each point. Then we impose the density-aware factor derived from point-wise density to the calculation of point distance during the farthest point sampling. The distribution of sampled points is supposed to be controlled by the weight of the density-aware factor.

We design a prediction head in set abstraction layers to utilize the segmentation capability of PointNet and distinguish foreground points from background points.

Then sample the points from foreground and background points respectively for the subsequent layers of the model framework.

We explore different values of parameters that influence each strategy and train multiple models corresponding to each setting. The performance of each trained model is evaluated on KITTI evaluation metric for comparison. The inference times are also recorded.

To better understand how different sampling strategies work, we introduce statistics such as bounding box recall ratio, positive points capture rate, unique point ratio and voting effect ratio. We also visualize the point distribution of each layer to showcase how it is influenced by different strategies or settings.

This chapter reviews the theoretical background and describes how our study was conducted.

3.1 A Groundwork for Point-based 3D Object Detection

PointNet/PointNet++ laid a solid base for the point-based 3D object detection. PointNet designs a simple but effective architecture to produce point-wise and global features for a point cloud, then PointNet++ implements partitioning and grouping strategies to capture features of local structure, and progressively enlarge the receptive fields with a layered design that functions like a CNN. The informative features learned from PointNet++ enable direct 3D object detection proposals [40] [32] [52] [30] [47] [54], or work in combination with other methods [56] [51] [23] [9]. In traditional image processing, features are typically hand-crafted based on domain knowledge, common examples include edges, corners, texture descriptors (like SIFT, HOG), and color histograms. They are explicitly programmed and often linear or simple non-linear transformations of the raw data. However in deep learning models like PointNet, features are learned automatically from data using deep learning neural networks. They often represent more abstract and complex patterns compared to traditional features and are obtained through multiple layers of non-linear transformation. Features learned from deep learning models are higher-dimensional, capturing a broader range of data characteristics.

It is worth reviewing the design and theoretical base of these two deep learning networks for point-based 3D object detection.

3.1.1 PointNet

Point cloud data is sparse, unordered, unstructured, and non-uniform. Transforming such data to regular 3D grids or collections of images causes information loss and asks for computation resources. To consume raw point cloud without voxelization or projection, and carry out a number of 3D recognition tasks, a deep learning model need to be invariant to input permutation and certain geometric transformations. PointNet addresses these challenges with a simple and unified architecture. The proposed architecture is shown in Figure 3.1 [34].

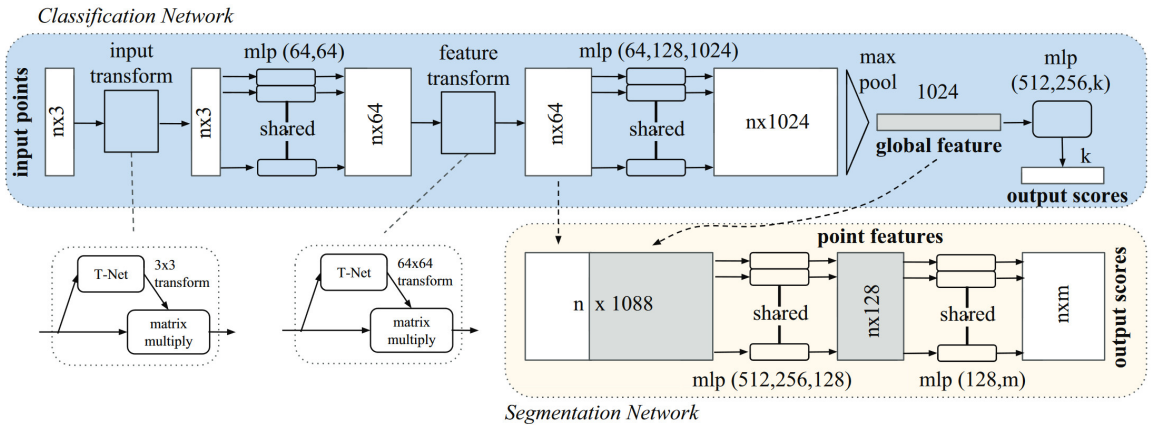


Figure 3.1: PointNet Architecture (Original diagram used under author's permission) [34]

The classification network takes n points as input. A point cloud is represented as a set of 3D points $\{P_i \mid I = 1, ..., n\}$, where each point P_i is a vector of its coordinate (x, y, z) plus extra feature channels such as reflection received by LiDAR. In PointNet, the (x, y, z) coordinate is used as 3 input channels. Input transformation and feature transformation are applied to make the point set invariant to possible geometric transformation. The learned point-wise features are aggregated by max pooling to form a global feature that is invariant to input permutation. A classification score for k classes of an object in the point cloud can be output by a 3-layer MLP which takes in the global feature.

The segmentation network is an addition to the classification network, outputting poise-wise scores for m categories. Point-wise features with 64 channels in the classification network are concatenated with the 1024-channel global feature. After going through several shared MLPs, the per-point scores are predicted.

A mini-network (T-Net in Figure 3.1) predicts an affine transformation matrix.

By applying matrix multiplication on the affine transformation matrix, the input points and features become invariant to possible geometric transformation. The mini-network is composed of basic modules of point independent feature extraction, max pooling, and fully connected layers, resembling the big network of PointNet.

PointNet achieved state-of-the-art classification results on ModelNet40 [50] and segmentation results on ShapeNet part dataset [53]. Besides, the network is robust to various kinds of input corruption. Experiments show that even 50% points are missing, the accuracy only drops by 2.4% and 3.8% when using furthest and random input sampling respectively. As to outliers, the network has more than 80% accuracy even when 20% of the points are outliers.

PointNet learns a spatial encoding of each point and then aggregates all individual point features to a global point cloud signature. The design does not capture the local structure induced by the metric. However, the success of convolutional architectures has proven that the capability of learning the feature of local structures is important. A layered feature representation of local structures progressively increases the receptive field, leading to a hierarchical resolution for the success of various recognition tasks.

3.1.2 PointNet++

PointNet++ tries to realize the hierarchical feature representation of a point cloud by addressing two issues: how to generate the partitioning of the point set and how to abstract sets of points or local features through a local feature learner. Obviously, PointNet is an ideal feature learner for the second issue. So the remaining task is to generate overlapping partitioning of a point set. An intuitive solution is to select a set of centroids evenly from the point set, then group the points in a radius surrounding the centroids into clusters for further feature learning. Classification or segmentation is followed and carried out on the learned features. The architecture of PointNet++ is shown in Figure 3.2. The architecture mainly includes three parts: **Hierarchical feature learning**, **Segmentation** and **Classification**. Hierarchical feature learning will be focused here since it is the core of the network, the following applications are not limited to segmentation or classification.

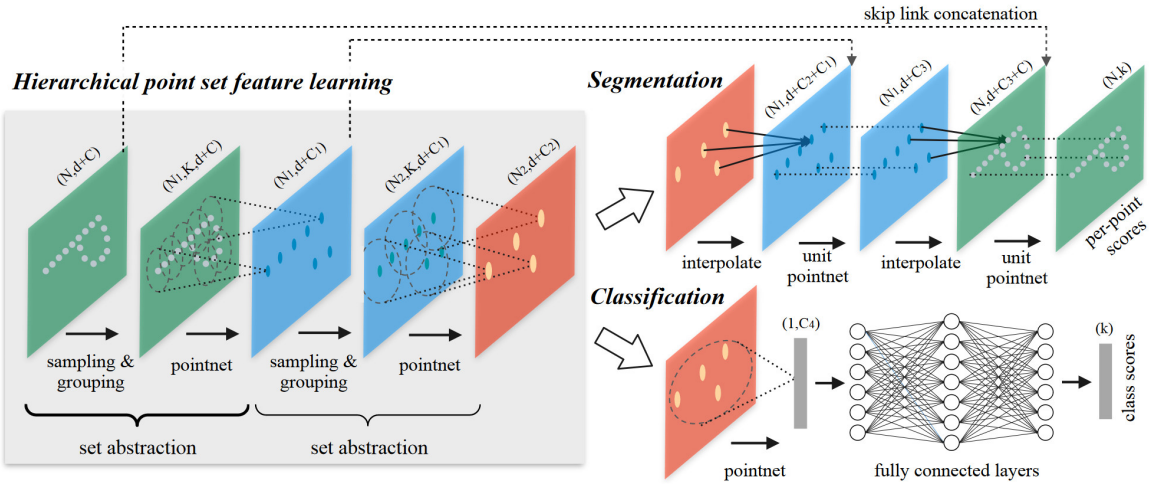


Figure 3.2: PointNet++ Architecture (Original diagram used under author's permission) [35]

Hierarchical Point Set Feature Learning

The hierarchical structure is composed of several **set abstraction** layers. At each level, a set of points is processed and abstracted to produce a new set with fewer elements. The set abstraction level is made of three key layers: **Sampling layer**, **Grouping layer** and **PointNet layer**. The Sampling layer selects a set of points from input points, which defines the centroids of local regions. Grouping layer then constructs local region sets by finding "neighboring" points around the centroids. PointNet layer uses a mini-PointNet to encode local region patterns into feature vectors.

Each set abstraction level takes an $N \times (d + C)$ matrix as input that is from N points with d-dim coordinates and C-dim point feature. It outputs an $N' \times (d + C')$ matrix of N' subsampled points with d-dim coordinates and new C'-dim feature vectors summarizing local context. The following paragraphs elaborate on the three layers of a set abstraction level.

Sampling layer Farthest point sampling (FPS) strategy is implemented to choose a subset of points $\{x_{i_1}, x_{i_2}, ..., x_{i_m}\}$ from the input points $\{x_1, x_2, ..., x_n\}$, such that x_{i_j} is the most distant point from the set $\{x_{i_1}, x_{i_2}, ..., x_{i_{j-1}}\}$ with regard to the rest points. Random sampling is an alternative, but the coverage of the entire point set is not as good as FPS given the same number of centroids.

Grouping layer The input to this layer is a point set of size $N \times (d + C)$ and the coordinates of a set of centroids of size $N' \times d$. The output are groups of point sets of size $N' \times K \times (d + C)$, where each group corresponds to a local region and K is the number of points in the neighborhood of centroid points. Ball query is implemented to find all points that are within a radius to the query point (K is set as the upper limit).

PointNet layer In this layer, the input are N' local regions of points with data size $N' \times K \times (d + C)$. Each local region in the output is abstracted by its centroid and local feature that encodes the centroid's neighborhood. Output data size is $N' \times (d + C)$. The coordinates of points in a local region are first translated into a local frame relative to the centroid point: $x_i^{(j)} = x_i^j - \hat{x}^j$ for $i = 1, 2, ..., K$ and $j = 1, 2, ..., d$ where $\hat{x}$ is the coordinate of the centroid. By using relative coordinates together with point features we can capture point-to-point relations in the local region.

The architecture of PointNet++ introduced above implements a single-scale grouping strategy. This design limits the generalization of the model between point sets with various sparsity. A model trained for sparse point cloud may not recognize fine-grained local structures. To make models adaptive to non-uniformity situations. PointNet++ introduces two grouping strategies: **multi-scale grouping (MSG)** and **multi-resolution grouping**.

MSG simply applies grouping layers with different scales (i.e. different radii) followed by according PointNets to extract features of each scale. Features at different scales are concatenated to form a multi-scale feature, as shown in Figure 3.3 (a). This method is computationally expensive since it runs local PointNet at large scale neighborhoods for every centroid point.

As shown in Figure 3.3 (b), MRG obtains features of a certain region with different resolutions. Features of a region at some level L_i is a concatenation of two vectors. One vector (left in the figure) is obtained by summarizing the features at each subregion from the lower level L_{i-1} using the set abstraction level. The other vector (right) is the feature that is obtained by directly processing all raw points in the local region using a single PointNet.

Table 3.1 shows the classification accuracy of PointNet++ with previous state-of-the-art methods.

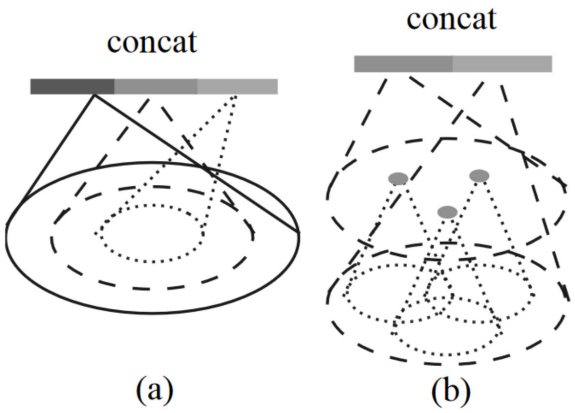


Figure 3.3: (a) Multi-scale grouping (MSG); (b) Multi-resolution grouping (MRG). [35]

Method	Input	Accuracy(%)
Subvolume	voxel	89.2
MVCNN	image	90.1
PointNet	point cloud	89.2
PointNet++	point cloud	**91.9**

Table 3.1: Classification Result on ModelNet40 [35]

3.2 Proposed Model Architecture

We adopt the framework of 3DSSD [52] as the test bed of different sampling strategies. The framework can be divided into three components: **Backbone**, **Candidate Generation Layer**, and **Prediction Head**, as shown in Figure 3.4. The backbone mainly takes in raw point cloud data, processes the points through several set abstraction (SA) layers and outputs downsampled points plus learned point features encoding valuable information. This is where we investigate different point sampling strategies. The candidate generation layer deals with downsampled points and groups them into different clusters as instance candidates. The prediction head generates 3D bounding boxes and categories sores for each candidate. In this section, the detailed design of each component is elaborated. The loss function corresponding to the network design is covered following the subsection of prediction head.

3.2.1 Backbone Design

The backbone consists of several set abstraction layers (SA) that progressively downsample and encode points into implicit feature vectors, as shown in Figure 3.4. In

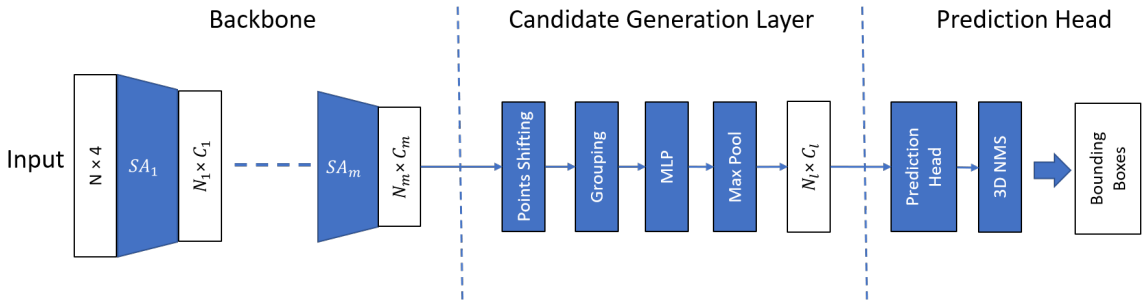


Figure 3.4: The framework of test bed model

each SA layer, the input points plus features sequentially go through a downsampling layer, grouping layer, and PointNet layer. Different sampling strategies have been proposed by researchers to downsample points in various models. In this research, we first introduce the 3 sampling strategies from the original source, then design the backbones based on the strategies.

D-FPS and F-FPS

The authors of 3DSSD observed that points sampled by the furthest points sampling based on 3D Euclidean space (D-FPS) are mostly background points, some instances with few interior points tend to be missed during downsampling. To keep the interior points of various instances, the furthest point sampling based on feature distance (F-FPS) is proposed.

Given a point set $P = \{p_i | i = 1, 2, ..., n\}$, $p_i = (x_i, y_i, z_i)$, where (x_i, y_i, z_i) represents the coordinate of a point p_i in the geometric space. The Euclidean distance between two points can be mathematically annotated in Equation 3.1.

$$euclidean\_dist(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{3.1}$$

Suppose $f_i = (v_1^i, v_2^i, ..., v_c^i)$ is the feature vector assciated with point p_i, where c is the dimension of the vector f_i. Then feature distance can be annotated in Equation 3.2.

$$feature\_dist(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 + (v_1^i - v_1^j)^2 + ... + (v_c^i - v_c^j)^2} \tag{3.2}$$

The implementation of furthest point sampling can be carried out with Equation

3.3. If N points are to be sampled from the point set P, then the following equation should be applied N times iteratively with updated sampled set S and unsampled set U. The distance criterion adopted will decide whether D-FPS or F-FPS sampling strategy is utilized.

$$s = \arg\max_{p_s \in U} \left(\min_{p_s \in S} dist(p_u, p_s) \right) \tag{3.3}$$

where

s is the point to be sampled from the unsampled set U,

S is the set of sampled points,

U is the set of unsampled points,

p_u is an unsampled point that is being considered for selection,

p_s is a point in the sampled set S.

$dist(p_u, p_s)$ is the distance calculation criterion between two points, the metric can be $euclidean\_dist(p_u, p_s)$ or $feature\_dist(p_u, p_s)$.

The above equations indicate that F-FPS demands more computation during the feature distance calculation. But as the learned feature from PointNet reserves implicit semantic information, which is distinct between different objects, the F-FPS keeps a high percentage of positive points. Experiments show that even downsampled from over 100,000 to 512 points, a point cloud can still preserve 68.4% of the positive points through F-FPS [52]. A large number of positive points helps the localization of bounding boxes, but the negative points, i.e. background points, contribute to the classification of the instances. So a balanced number between positive and negative points would benefit object detection most. 3DSSD adopts a fusion sampling strategy that uses both D-FPS and F-FPS in designated layers for the best accuracy. Figure 3.5 shows the design of the backbone from 3DSSD.

The backbone in Figure 3.5 consists of 3 SA layers downsampling the point cloud from the original 16384 points to 4096, 1024, and finally 512. The first SA layer only applies D-FPS to evenly downsample the input points and a multi-scale grouping strategy to learn the features. The reason for utilizing D-FPS only in the first SA layer is that there is no learned feature for F-FPS before a first SA layer is employed. The following SA2 and SA3 layers implement a fusion sampling that samples the same number of points from D-FPS and F-FPS. For the last SA layer, i.e. SA3, the points sampled by D-FPS and F-FPS are kept separate for the candidate generation layer.

In every SA layer, multi-scale grouping is used to obtain features of local structures in 3 different scales, which are decided by the radii of ball query for grouping. The radii used for each SA layer are marked with $r_i, i = \{1, 2, 3\}$ in the figure. $c_i, i = \{1, 2, 3\}$ corresponding to r_i represents the number of channels of multi-layer perceptron for local feature extraction of points in radius r_i. The output of the backbone network is 512 points encoded with 256-channel features. Half of the points are sampled through D-FPS, and the other half through F-FPS.

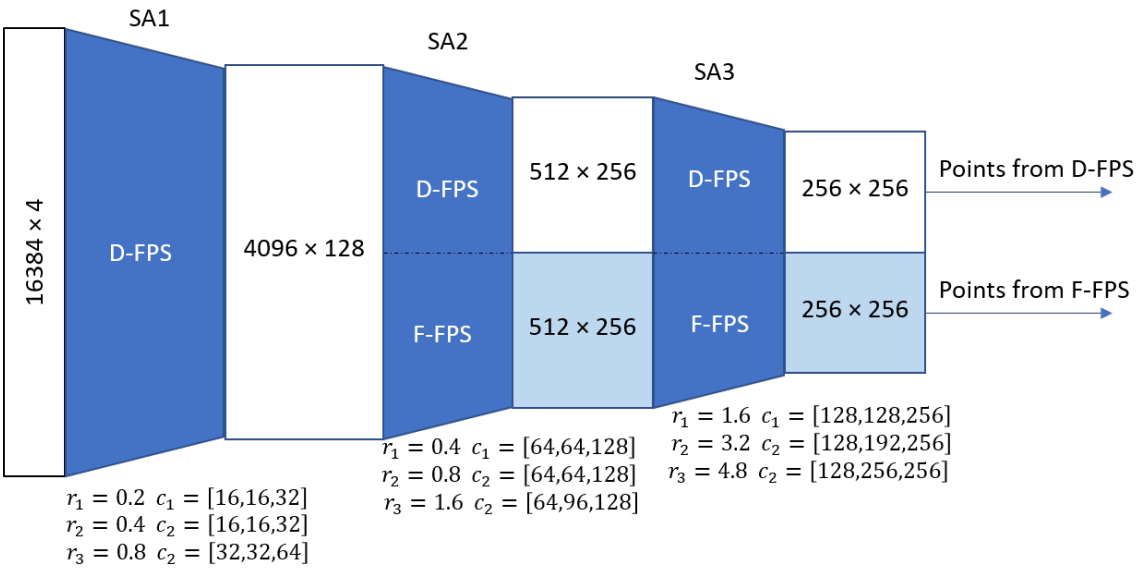


Figure 3.5: The backbone based on D-FPS and F-FPS

Density-aware Sampling

As the laser beams diverge over the distance, objects located far from the LiDAR return fewer points than those located closer. This means points in the distance of a point cloud are of low density. If few internal points are reserved after sampling, the objects are hard or impossible to be detected. Several researchers [30, 25, 18] proposed different strategies to deal with the density non-uniformity. Among them, Ning, et al. [30] introduced a density-aware sampling strategy that tries to keep more low-density points in the sampling process. In this research, we fit this sampling strategy into our framework to see how it works.

The number of points within a fixed distance around a point is used to describe the density of the point, i.e., the number of points in a specific radius of the point. However, for a point cloud, the number of points within a radius can range from a

thousand to only one, which leads to numerical instability and imbalanced feature importance in deep learning. In this case, normalization will be a solution. Suppose the point set of a point cloud is $P = \{p_i | i = 1, 2, ..., N\}$. A normalized calculation of the density of p_i is shown in Equation 3.4.

$$d_{p_i} = \frac{min(ct_{p_i}, ct_{max}) - \min\limits_{p_j \in P} ct_{p_j}}{min(ct_{max}, \max\limits_{p_j \in P} ct_{p_j}) - \min\limits_{p_j \in P} ct_{p_j}} \tag{3.4}$$

where
ct_{p_i} is the number of points within a radius of point p_i,
ct_{max} is the limitation number of maximum points number.

With the calculation of point density, the next step is associating the sampling strategy with the density information to reserve more points in the sparse region of a point cloud. Inspired by [55], a density aware factor $k_{p_i} = \left(\frac{1}{d_{p_i}}\right)^{\lambda}$ is introduced. λ is used to balance the influence of the density. As a hyperparameter, the effect of λ imposed on the sampling will be explored in the experiment section. The coordinates of point p_i in Equation 3.1 are multiplied by corresponding k_{p_i} to formulate the new equation to calculate the density aware distance between two points, which is shown in Equation 3.5.

$$density\_dist(p_i, p_j) = \sqrt{(k_{p_i}x_i - k_{p_j}x_j)^2 + (k_{p_i}y_i - k_{p_j}y_j)^2 + (k_{p_i}z_i - k_{p_j}z_j)^2} \tag{3.5}$$

From Equation 3.5 and the definition of k_{p_i}, a low-density point p_i has a big value of k_{p_i}, which upscales the density aware distance between two points. By implementing the furthest point sampling on the density aware distance, a point with low density will have more chances to be sampled compared to that based on Euclidean distance. So instances in the sparse distance have a greater likelihood to reserve their internal points. We call this sampling strategy Density Aware Furthest Point Sampling (DA-FPS).

The proposed backbone that implements DA-FPS sampling is shown in Figure 3.6. Compared to the backbone applying D-FPS and F-FPS, we replace D-FPS with DA-FPS in the first SA layer. The idea behind this strategy is that we consider the sampled points would tend to be uniformly distributed after applying DA-FPS in the first SA layer, so the need for implementing DA-FPS in the following SA layers is

weak based on this assumption. Another reason is that calculating the density for every point in a point cloud is time-consuming, which at least doubles the time of inference. More details about how the hyperparameter and network design influence the accuracy and efficiency will be discussed in the experiment chapter.

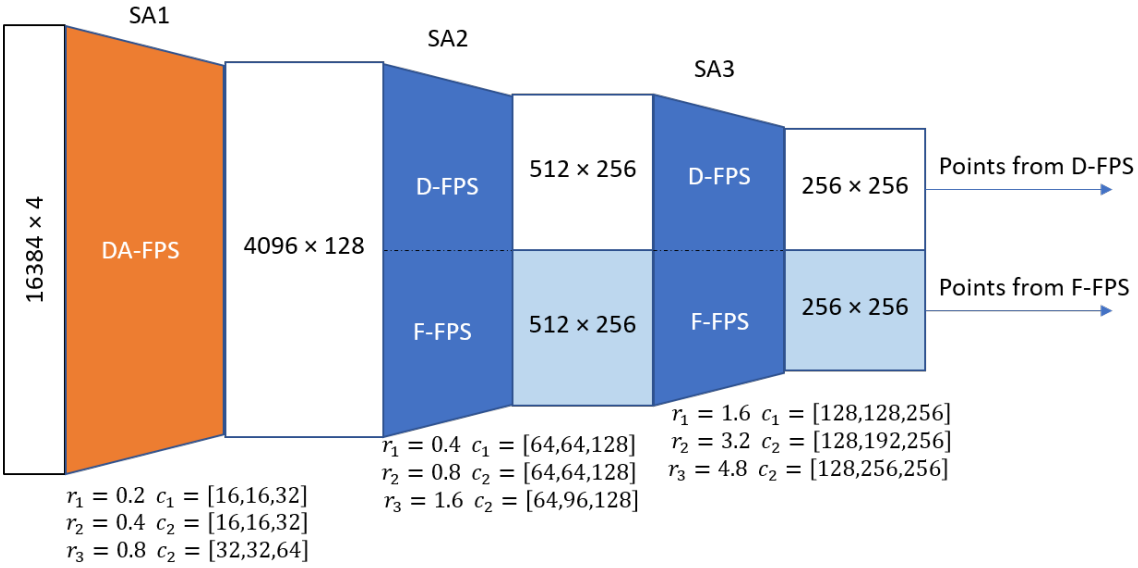


Figure 3.6: The backbone based on DA-FPS

Semantic-aware Sampling

Foreground points obviously contain rich information on predicting the object's location, dimension and orientation. Through F-FPS, a lot of foreground points are sampled and demonstrate the capability to boost detection accuracy. Since the SA layers can encode informative features of every downsampled point for classification and semantic segmentation. A question naturally comes out? why not design a sampling strategy based on the semantic prediction of whether a point is a foreground point or a background point? Wang, et al. [47] introduce a segmentation head to score every downsampled point to be a foreground point or not after the first SA layer. Zhang, el at. [54] propose extra branches on the backbone to estimate the semantic categories of each point, i.e., predict the class of each point. We adopt the idea of the simpler one from Wang and design a backbone network with a semantic-aware sampling strategy, as shown in Figure 3.7.

After an SA layer that implements multi-scale PointNet++, the point cloud is downsampled to a point set with 4096 points encoded with 128-channel features. The

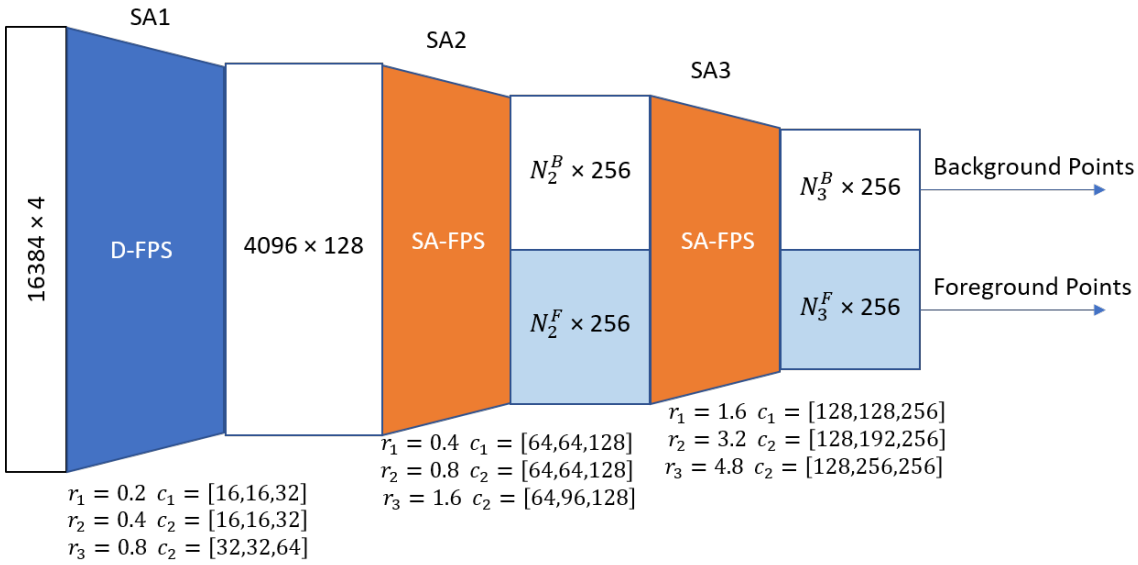


Figure 3.7: The backbone based on semantic-aware sampling

downsampled point set can be denoted as $D = \{d_i | i = 1, 2, ..., N_d\}$, $d_i = (x_i, y_i, z_i)$ with the features denoted as $f_i = (v_1^i, v_2^i, ...v_c^i)$. After the first SA layer, the point set D has $N_d = 4096$, $c = 128$. In SA2 where the first semantic-aware sampling is implemented. We first apply a segmentation head $\mathcal{F}^{seg}$ that consists of 3 MLP layers. $\mathcal{F}^{seg}$ can be represented by Equation 3.6. Based on the segmentation score $\{\varepsilon_i | i = 1, 2, ..., N_d\}$, top K points are selected as the foreground point set $D^{(f)} = \{d_j^{(f)} | j = 1, 2, ..., K\}$, the rest are the background point set $D^{(b)} = \{d_j^{(b)} | j = 1, 2, ..., N_d - K\}$. So far the points entering the SA2 layer have been segmented into foreground points and background points. We implement furthest point sampling on the foreground and background points separately, then combine them into one set for the input of the next SA layer. The process can be denoted as Equation 3.7.

$$\varepsilon_i = \mathcal{F}^{seg}(d_i, f_i) \in [0, 1] \tag{3.6}$$

where

ε_i is the segmentation score of point d_i in range [0,1].

$$D^{(\hat{f})} = FPS(D^{(f)}), D^{(\hat{b})} = FPS(D^{(b)}), S = D^{(\hat{f})} \oplus D^{(\hat{b})} \tag{3.7}$$

where

$D^{(\hat{f})} = \{d_j^{(\hat{f})} | j = 1, 2, ..., N_l^F\}$, N_l^F is the sample number of foreground points in

the lth layer,

$D^{(\hat{b})} = \{d_j^{(\hat{b})} | j = 1, 2, ..., N_l^B\}$, N_l^B is the sample number of background points in the lth layer,

S is the final sampled point set.

In our research, we try different ratios of foreground and background points and compare the results. The foreground points will be treated as seed points at the candidate generation layer to generate instance clusters for detection proposals. While the kept background points are utilized at later stages for offering complementary information for tasks like classification.

3.2.2 Candidate Generation Layer

The candidate generation layer is the key to making the 3D object detection model anchor-free through a novel clustering operation. The gathered points in every cluster tend to form an object instance in the point cloud and the associated features of the points functionally generate object proposals through aggregation. The novel clustering is based on a point shifting layer that implements Deep Hough Voting [32] to generate virtual points shifted to the instance centers from the seed points.

Figure 3.8 shows the architecture of candidate generation layer. In the architecture, Deep Hough Voting is in essence a shared MLP network that shifts the seed points into voted points. Given a set of seed points $\{s_i | i = 1, 2, ..., M\}$, where $s_i = [d_i; f_i]$ with $d_i \in \mathbb{R}^3$ and $f_i \in \mathbb{R}^C$ (As denoted in Subsection 3.2.1), the MLP takes in seed feature f_i and outputs the offset $\triangle d_i \in \mathbb{R}^3$ in Euclidean space and a feature offset $\triangle f_i \in \mathbb{R}^C$. So the voted points $v_i = [c_i; g_i]$ can be generated from the calculation that $c_i = d_i + \triangle d_i$ and $g_i = f_i + \triangle f_i$. The predicted $\triangle d_i$ is the offset of point d_i from the center of its belonged instance and is supervised by a regression loss shown in Equation 3.24 (Described in detail in Subsection 3.2.4). The addition of d_i and $\triangle d_i$ results in the voted c_i which is located closer to the instance center.

Unlike the seed points that are semantic-agnostic in VoteNet [32], seed points from our designed backbones are prone to be foreground points that are located on the surface of the objects. These seed points are treated as initial center points and fed into vote layers to be shifted closer to the object centers. Points closer to object centers tend to get more accurate bounding boxes. So the shifted point set $C = \{c_i | i = 1, 2, ..., M\}$ are candidate points that act as the centroids for the grouping

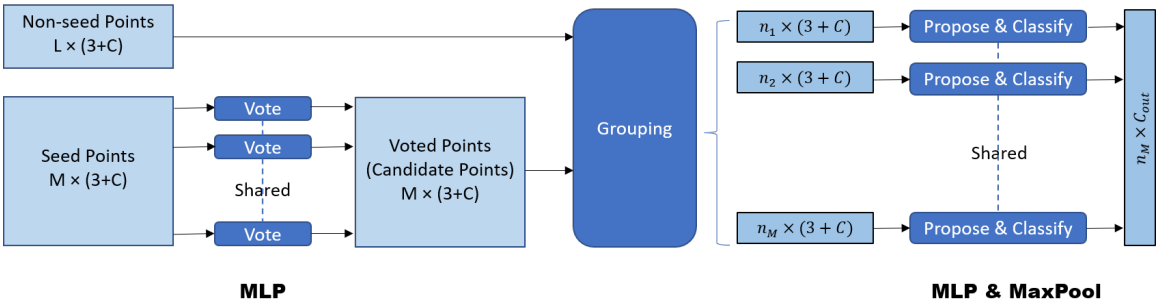


Figure 3.8: Candidate Generation Layer

layer. The union of seed points $P_{seed} = \{s_i | i = 1, 2, ..., M\}$ and non-seed points $P_{non-seed} = \{n_i | i = 1, 2, ..., L\}$ forms the input point set of candidate generation layer $P = P_{seed} \oplus P_{non-seed}$. The grouping layer group point set P around the candidate point set C within a radius r. M clusters are formed by finding neighboring points to every c_m's 3D location: $\mathcal{C}_m = \{p_i^{(m)} | \parallel p_i - c_m \parallel \leq r\}$ for $m = 1, 2, ..., M$. The next step is to aggregate the features of each cluster for the proposal and classification.

A shared PointNet is used to aggregate the points in each cluster. Given a cluster $\mathcal{C}_m = \{p_i^{(m)} | i = 1, 2, ..., n\}$ and its cluster center c_m. To exploit the local geometry of points in each cluster, points are transformed into a local normalized coordinate system by $p_i^{(m)'} = (p_i^{(m)} - c_m)/r$ with $i = 1, 2, ..., n$. For cluster $\mathcal{C}_m$, normalized coordinates $\{p_i^{(m)'} | i = 1, 2, ..., n\}$ and features $\{f_i^{(m)} | i = 1, 2, ..., n\}$ are concatenated and fed into a PointNet-like module to realized aggregation then prediction, as shown in Equation 3.8.

$$\mathcal{P}(\mathcal{C}_m) = MLP_2 \left\{ \max_{i=1,...,n} \left\{ MLP_1([p_i^{(m)'}; f_i^{(m)}]) \right\} \right\} \tag{3.8}$$

The PointNet-like module first extracts point-wise features through MLP_1, then the channel-wise max-pooling flattens all the features into a single feature vector. MLP_2 takes the single feature vector to generate a proposal for the corresponding cluster. The prediction head is included in MLP_2 and is discussed in the next subsection.

3.2.3 Prediction Head

For 3D object detection, the expected proposal form is the center, dimension and yaw of the bounding boxes of the detected objects, along with the class labels. In the sequence as mentioned, a bounding box is denoted as $(c_x, c_y, c_z, h, w, l, \theta)$. A predic-

tion head is a module designed to predict the bounding box and category information directly or indirectly. The design of the prediction head in our research is shown in Figure 3.9. Each candidate point plus the aggregated features goes through shared MLPs then two prediction branches, classification and regression, to generate a proposal. The classification branch is to predict the class label of the proposal. The regression branch predicts centroid offsets, dimension and yaw (or heading) of the bounding box.

The candidate generation layer selected and voted out the candidate points which are potential centroids of the objects. To utilize this geometric prior information, the prediction head predicts the distance between each candidate point and its corresponding object, denoted as $\triangle d = (\triangle x, \triangle y, \triangle z)$. So a predicted centroid can be acquired by Equation 3.9.

$$(c_x, c_y, c_z) = (c_x^{(can)} + \triangle x, c_y^{(can)} + \triangle y, c_z^{(can)} + \triangle z) \tag{3.9}$$

where
$(c_x^{(can)}, c_y^{(can)}, c_z^{(can)})$ is the coordinate of a candidate point,
$\triangle d = (\triangle x, \triangle y, \triangle z)$ is the offset directly predicted by the prediction head.

The dimensions of the bounding boxes (h, w, l) are predicted directly by the regression head.

For the yaw prediction, there is no prior orientation, we implement a hybrid of classification and regression formulations following [33]. The strategy is to split the direction range of yaw $[0, 2\pi]$ into NH equal bins, so the angle of each bin is $\angle Bin = \frac{2\pi}{NH}$. The final yaw is achieved by finding the bin in which the yaw is located and the residual angle. Suppose the angle of a yaw is $\angle H$, then shift the $\angle H$ to an angle of $\frac{\angle Bin}{2}$ to ensure the class boundaries are at the midpoint between two class centers, getting $\angle H_{shifted} = (\angle H + \frac{\angle Bin}{2})\%2\pi$. The class and residual of an angle can be decided through Equation 3.10and Equation 3.11.

$$angle\_cls = \lfloor \angle H_{shifted} / \angle Bin \rfloor \tag{3.10}$$

$$angle\_res = \angle H_{shifted} - (angle\_cls \times \angle Bin + \frac{\angle Bin}{2}) \tag{3.11}$$

For each candidate point, NH pairs of numbers are regressed, which are the

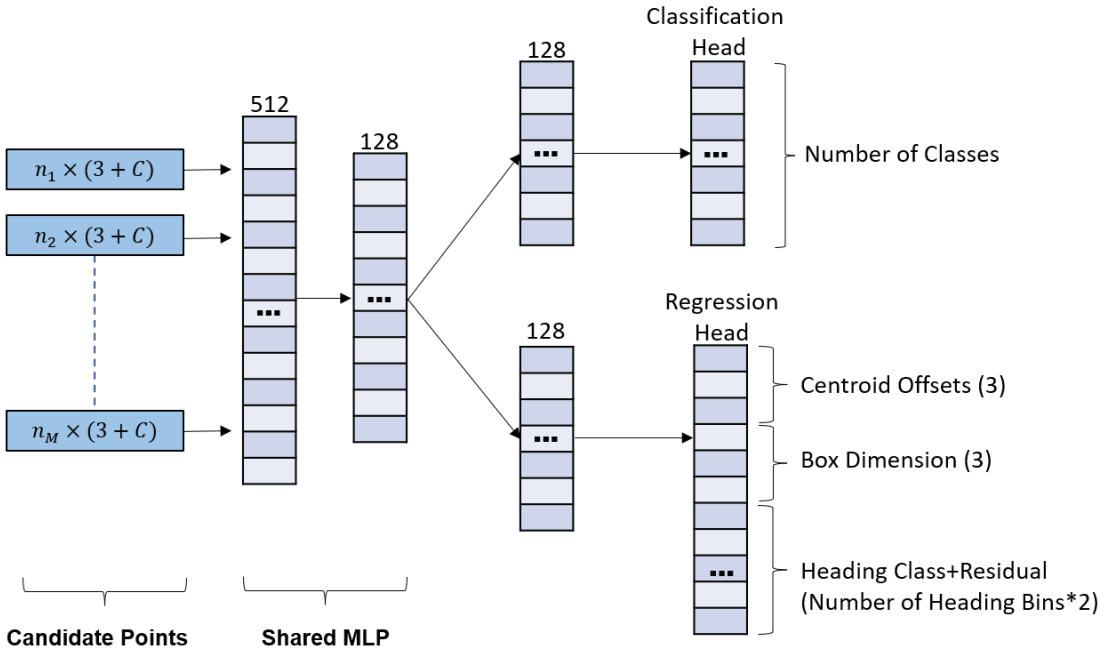


Figure 3.9: Prediction Head Design

probabilities that yaw falls into the bins and the associated residuals. The bin with the largest probability is picked as the *angle_cls* and the associated *angle_res* is used to reversely calculate $\angle H$ for the bounding box.

The classification head predicts the confidence score for each class. The head number is the number of categories to be classified. The class with the highest confidence score is picked during inference.

3.2.4 Loss Function

A novel design of loss functions in accordance with the prediction head can lead to efficient training and satisfactory model performance. We borrow the loss function design from [52] for the prediction head and from [47] for the segmentation head. In this section, we first introduce the loss functions we use for the classification and regression tasks. Then the methods to achieve the loss target of each task are elaborated.

Cross Entropy Loss

The cross entropy loss function is adopted in our models due to its effectiveness and wide usage in deep learning-based classification tasks. The ability of cross entropy loss to heavily penalize incorrect predictions, especially those made with high confidence, makes it well-suited to our task of accurately identifying and classifying objects in 3D space. The mathematical formulation of the cross entropy loss for binary classification is given by Equation 3.12.

$$L_c(y, \hat{y}) = -(y \log \hat{y} + (1 - y) \log (1 - \hat{y})) \tag{3.12}$$

where y represents the true label (0 or 1), and $\hat{y}$ is the predicted probability from the model, ranging between 0 and 1.

For multi-class classification, the cross entropy loss is extended to multiple classes using softmax function on the output layer and is defined as Equation 3.13.

$$L_c(y, \hat{y}_s) = -\sum_{i=1}^{C} y_i \log (\hat{y}_{i_s}) \tag{3.13}$$

where C is the number of classes, y_i are the elements of the one-hot encoded true label, and $\hat{y}_{i_s}$ are the elements of probability distribution after softmax operation on the predicted logit-form $\hat{y}_i$. The calculation of $\hat{y}_{i_s}$ from $\hat{y}_i$ can be acquired through Equantion 3.14.

$$\hat{y}_{i_s} = \frac{\exp(\hat{y}_i)}{\sum_{j=1}^{n} \exp(\hat{y}_j)} \tag{3.14}$$

Smooth-L1 Loss

In this research, smooth-L1 loss is adopted for all the regression tasks. It is a robust variant of the standard L1 loss, and has emerged as a popular choice due to its unique characteristics that help to mitigate certain issues present in other commonly used loss functions.

The L1 loss function is defined as the absolute difference between the predicted and true values. While it is more robust to outliers compared to the L2 loss (Mean Squared Error), it has a discontinuity at zero that can make the gradient undefined. The smooth-L1 loss, also known as the Huber loss, is designed to be less sensitive

to outliers in the data than the L2 loss while overcoming the issue of an undefined gradient at zero in the L1 loss. The mathematical definition of smooth-L1 loss is shown as Equation 3.15.

$$L_c(y, \hat{y}) = \begin{cases} 0.5(y - \hat{y})^2 & if|y - \hat{y}| < 1 \\ |y - \hat{y}| - 0.5 & otherwise \end{cases} \tag{3.15}$$

where y represents the true value, and $\hat{y}$ is the predicted value from the model.

In the context of 3D object detection, the prediction task often involves predicting continuous values like bounding box coordinates or object dimensions. These tasks can be sensitive to outliers or large errors. For example, a single mispredicted bounding box that is far from the ground truth can have a large impact on the L2 loss because it squares the error. In this case, the smooth L1 loss was employed for its ability to handle outliers effectively and its computational advantages. Its application proved to be pivotal in achieving reliable and robust object detection performance.

According to the prediction head, an overall loss function of our model can be represented as Equation 3.16.

$$L = L_{obj\_cls} + L_{center\_offset} + L_{size} + L_{dir\_class} + L_{dir\_reg} + L_{corner} + L_{vote} \tag{3.16}$$

where

L_{obj_cls} is the object classification loss of the points, a cross entropy loss,

L_{center_offset} is the regression loss of the offsets between the predicted centroids and the true centroids, a smooth-L1 loss,

L_{size} is the regression loss of the predicted dimensions, a smooth-L1 loss,

L_{dir_class} is the classification loss of the predicted class of the headings, a cross entropy loss,

L_{dir_reg} is the regression loss of the heading residuals, a smooth-L1 loss,

L_{corner} is the regression loss of the predicted 8 corners, a smooth-L1 loss,

L_{vote} is the regression loss of the offset distance between the seed points and its corresponding instance center.

Each of the loss functions on the right side of Equation 3.16 is introduced in detail in the following content.

Object Classification Loss

For each candidate point, the prediction head directly outputs the confidence scores for each class. However, there are multiple ways to assign a class label to a candidate point, which can influence the performance of the trained model. Inspired by FCOS [45], Yang, et al. [52] consider that candidate points closer to the object centers contribute more to accurate localization predictions. Based on this assumption, a 3D center-ness assignment strategy is implemented to indicate how close a point is located in the center of an object. The center-ness label is calculated as Equation 3.17.

$$l^{(ctrness)} = \sqrt[3]{\frac{min(f,b)}{max(f,b)} \times \frac{min(l,r)}{max(l,r)} \times \frac{min(t,d)}{max(t,d)}} \tag{3.17}$$

where (f, b, l, r, t, d) represent the distance to the front, back, left, right, top and bottom surfaces respectively.

In addition to the center-ness label, a semantic mask $l^{(mask)}$ indicating which instance the candidate belongs to is also necessary. The target of object classification loss $t^{(obj)}$ is the multiplication of $l^{(ctrness)}$ and $l^{(mask)}$, as Equation 3.18 shows.

$$L_{obj\_cls} = \frac{1}{N_c} \sum_{i=1}^{N_c} L_c(s_i, t_i^{(obj)}) \tag{3.18}$$

where
N_c is the number of candidate points,
s_i is the predicted object classification score of candidate point i,
$L_c(\cdot, \cdot)$ represents cross entropy loss function.

Center Offset Loss

Center offset is predicted directly by the regression head. To get the target of center offset, every candidate point is masked according to which instance it is located in. Then the distance between the point and the corresponding center of the bounding box is calculated and taken as the target. The smooth-L1 loss function of center offset is formulated as Equation 3.19

$$L_{center\_offset} = \frac{1}{N_p}\sum_{i=1}^{N_p} L_r(\triangle d_i, t_i^{(cen)})\mathbb{1}[t_i^{(obj)} > 0] \tag{3.19}$$

where
N_p is the number of positive candidate points,
$\triangle d_i$ is the predicted center offset of candidate point i,
$t_i^{(obj)}$ is the object classification target of point i,
$t_i^{(cen)}$ is the true offset of point i to the center its corresponding bounding box,
$L_r(\cdot,\cdot)$ represents smooth-L1 loss function.

Dimension Loss

Since the height, width, and length of each bounding box in a point cloud frame are given by the KITTI dataset. The dimension loss is directly calculated as Equation 3.20.

$$L_{size} = \frac{1}{N_p}\sum_{i=1}^{N_p} L_r(d_i, t_i^{(size)})\mathbb{1}[t_i^{(obj)} > 0] \tag{3.20}$$

where
N_p is the number of positive candidate points,
d_i is the predicted dimension from candidate point i, $d_i = (h_i, w_i, l_i)$,
$t_i^{(size)}$ is the target dimension of candidate point i, $t_i^{(size)} = (h_i^{(GT)}, w_i^{(TG)}, l_i^{(GT)})$,
$t_i^{(obj)}$ is the object classification target of point i.

Heading Classification Loss

Suppose the direction is evenly divided into NH bins, the probability of the heading angle falling into each bin is predicted by the regression head. To be more specific, the probabilities are achieved by the softmax operation on the NH logit number from the regression head. The target of heading classification is calculated through Equation 3.10 discussed in Subsection 3.2.3. Equation 3.21 shows the cross entorpy loss function of heading classification.

$$L_{dir\_class} = \frac{1}{N_p} \sum_{i=1}^{N_p} L_c(h_i^{(c)}, t_i^{(dir\_class)}) \mathbb{1}[t_i^{(obj)} > 0] \tag{3.21}$$

where
N_p is the number of positive candidate points,
$h_i^{(c)}$ is the predicted probability distribution of heading class of candidate point i,
$t_i^{(dir\_class)}$ is the one-hot form of target heading class,
$t_i^{(obj)}$ is the object classification target of point i.

Heading Residual Loss

The head residuals are directly predicted by the regression head. The target of heading residual regression is calculated through Equation 3.11 discussed in Subsection 3.2.3. Equation 3.22 shows the smooth-L1 loss function of heading residual regression.

$$L_{dir\_reg} = \frac{1}{N_p} \sum_{i=1}^{N_p} L_r(h_i^{(r)}, t_i^{(dir\_reg)}) \mathbb{1}[t_i^{(obj)} > 0] \tag{3.22}$$

where
N_p is the number of positive candidate points,
$h_i^{(r)}$ is the predicted heading residual of candidate point i,
$t_i^{(dir\_class)}$ is the target heading residual,
$t_i^{(obj)}$ is the object classification target of point i.

Corner Loss

Based on the loss functions mentioned above, the 3D box parameters (center, size, and heading) are optimized separately, not for the final 3D IoU metric. This may lead to poor results due to the error of one parameter. To deal with this problem, Qi, et al. [33] propose the corner loss, which regularizes the training for the bounding box parameters jointly. Yang, et al. [52] simplify the corner loss by removing the anchor boxes. We implement the simplified version in this research. The 8 corners of a bounding box can be calculated With the predicted center, size, and heading. Then the corner loss is the sum of the distances between the 8 corners of a predicted

box and a ground truth box, in a predefined order. The loss function is shown as Equation 3.23.

$$L_{corner} = \frac{1}{N_p} \sum_{i=1}^{N_p} \sum_{m=1}^{8} L_r(P_{im}, t_{im}^{(corner)}) \mathbb{1}[t_i^{(obj)} > 0] \tag{3.23}$$

where

N_p is the number of positive candidate points,

m is the index of the 8 corners,

P_{im} is the coordinate of corner m for candidate point i,

$t_{im}^{(corner)}$ is the ground truth coordinate of corner m for candidate point i,

$t_i^{(obj)}$ is the object classification target of point i.

Vote Loss

The shift of a seed point to a candidate point in the candidate generation layer is supervised by vote loss. As discussed in Subsection 3.2.2, the offset between a seed point $s_i = (x_i, y_i, z_i))$ and its instance center $c_i = (x_i^{(c)}, y_i^{(c)}, z_i^{(c)})$ is to be predicted by the Deep Hough Voting module. The target of the offset $\triangle d_i^*$ can be calculated by $\triangle d_i^* = \parallel c_i - s_i \parallel$. Equation 3.24 shows the calculation of the overall loss of Deep voting module.

$$L_{vote} = \frac{1}{N_{p^*}} \sum_{i}^{N_{p^*}} \parallel \triangle d_i - \triangle d_i^* \parallel \mathbb{1}[s_i \text{ on object}] \tag{3.24}$$

where

N_p^* is the number of seed points on an object, i.e., locating in the ground truth bounding box of an instance,

$\triangle d_i$ is the predicted offset between the seed point i and its corresponding instance,

$\triangle d_i^*$ is the ground truth offset.

In this chapter, we present the design of our model and the mathematical abstraction. A groundwork for point-based 3D object recognition tasks is first introduced. Then the backbone design according to different sampling strategies is discussed in detail. The influence of hyperparameters from different sampling strategies will be discussed in the next chapter. The candidate generation layer, prediction head and

loss functions are designed to be fit into the backbones. The next chapter will present the experiments and discuss the results.

Chapter 4

Experiments

In this chapter, we describe the experiment setup and benchmark used to assess the performance of models incorporating the sampling strategies discussed earlier. The evaluation is conducted using the KITTI benchmark, which measures the average precision for 3D object detection, providing a comprehensive assessment of the models' overall capabilities. Additionally, we analyze the inference time as another crucial metric. To gain deeper insights into the effects of the sampling strategies, we conduct a comparison of geometric visualizations of the points and perform statistical analyses to reveal finer details. These analyses complement the evaluation metric and allow us to better understand the performance of the models. After detailing the experimental setup and the evaluation metric, we present the results, showcasing the outcomes of our exploration and providing a clear picture of how the models fare with the different sampling strategies.

4.1 Experiment Setup

Chapter 3 outlines the design of models integrating various sampling strategies. To facilitate a fair comparison among these models, we establish a consistent experiment setup, illustrated in Figure 4.1. This ensures that all the models go through training and evaluation under identical scenarios. The specific configurations for each experimental step and running environment are elaborated in the subsequent subsections.

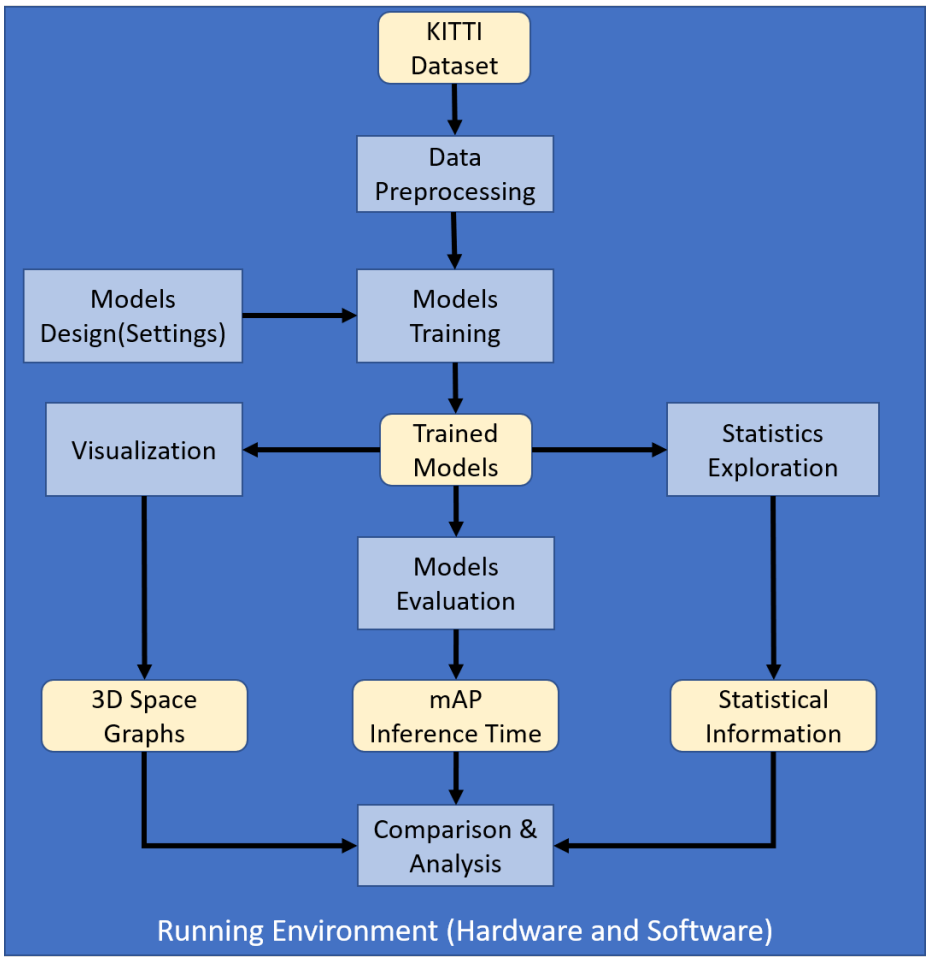


Figure 4.1: Experiment Setup

4.1.1 Data Preprocessing

As previously mentioned, we utilize the KITTI dataset for training and evaluating our models. Prior to inputting the raw data into the models for training, several essential data preprocessing steps are applied. These steps encompass data filtering, data augmentation, and data sampling, which are elaborated below:

Data Filtering

During data filtering, we apply filtering on both the raw data and ground truth labels. To optimize computation and prioritize the forward direction, we discard points located in the backward direction. The resulting filtered points lie within specific ranges: (0, 70) meters along the x axis, (-40, 40) meters along the y axis, and

(-5, 3) meters along the z axis.

Given that the "car" class represents the largest quantity in the dataset, our focus lies on car detection. Hence, we filter out other classes from the ground truth bounding boxes. Moreover, these filtered bounding boxes are also constrained to lie within the range of the filtered points. This ensures that we streamline the data to enhance the accuracy and efficiency of our models.

Data Augmentation

To enhance the models' generalization capability and mitigate overfitting, we employ several data augmentation strategies on the dataset. These strategies include random flipping, noise addition, global rotation, and scaling. The specifics of each strategy are illustrated below:

Random Flipping We randomly flip the points and bounding boxes along the horizontal axis (y axis). This augmentation technique not only increases the diversity of the dataset but also simulates different viewpoints of the objects, making the models more robust. The flipping ratio is set to 0.5, which means there is a 50% chance the whole point cloud will be flipped. This randomness ensures that the models encounter both original and flipped instances, effectively exposing them to various perspectives of the data.

Noise Addition To mimic real-world variations, we introduce random noise to the data. By doing so, the models learn to handle noisy and less ideal scenarios, thus improving their resilience. In our research, we specifically apply translation and rotation noise to the objects, with a focus on cars in this case. The translation noise is added along the x axis and y axis, which represents the plane on which cars typically move. The added noise follows a normal distribution, with a standard deviation of 1 for both x and y axes. Additionally, rotation noise is incorporated, ranging between $-\frac{\pi}{3}$ and $\frac{\pi}{3}$, following a uniform distribution. This rotation noise introduces variation in the object orientations, allowing the models to adapt to different angles of the objects during detection.

Global Scaling We incorporate random scaling to the point cloud, focusing on adjusting the aspect ratio. This technique plays a vital role in enhancing the models' capability to detect and estimate objects of various scales, making them more versatile

and adaptable in real-world scenarios with different object sizes. During the scaling augmentation, the scale ratio is determined by a uniform distribution between 0.9 and 1.1. This means that the point cloud may be uniformly scaled down to 90% or up to 110% of its original size.

Data Sampling

Dealing with large volumes of LiDAR data in datasets like KITTI can pose significant challenges due to memory limitations. A common practice is to subsample the raw LiDAR data, reducing it to a more manageable size. For the KITTI dataset, a widely used subsampling choice involves reducing the number of points per frame to 16,384. This specific value hits a balance between retaining critical information and easing the computational burden during training and inferencing. To achieve this, a random sampling approach is applied, ensuring that 16,384 points are randomly selected from the original raw data.

It's important to note that the data sampling process performed here, i.e., the subsampling to 16,384 points, is distinct from the sampling strategies being researched in the models. While the former is primarily for computational efficiency, the latter aims to investigate diverse sampling approaches that can enhance model performance and object detection capabilities.

4.1.2 Models Settings

In Chapter 3, we have detailed the model architectures that implement the researched sampling strategies. In this section, we propose distinct experiment settings to explore each sampling strategy comprehensively. The settings are introduced following the sequence in Chapter 3.

D-FPS and F-FPS

The backbone implementing D-FPS and F-FPS introduced in Chapter 3 shows a fusion sampling strategy that samples half points from the previous layer with D-FPS and F-FPS respectively. In the original paper of 3DSSD [52], Yang, et al. present the recall and AP of implementing D-FPS only, F-FPS only, and fusion sampling. In the experiment, we retrain the model implementing these 3 settings and see if the results meet those in the paper. In addition, we dive into each SA layer to see how are the points distributed in 3D space and the relations between points and ground

truth bounding boxes. The backbones that implement D-FPS only and F-FPS only sampling strategies are shown in Figure 4.2 and 4.3.

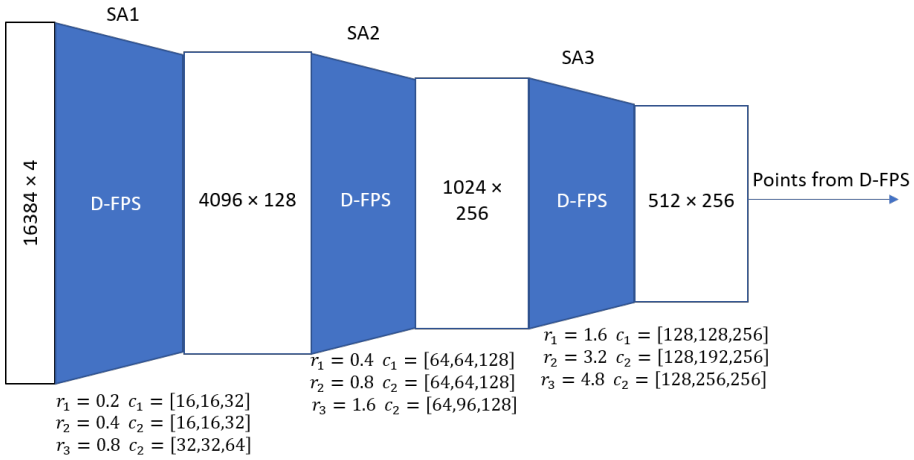


Figure 4.2: Backbone Implementing D-FPS Only

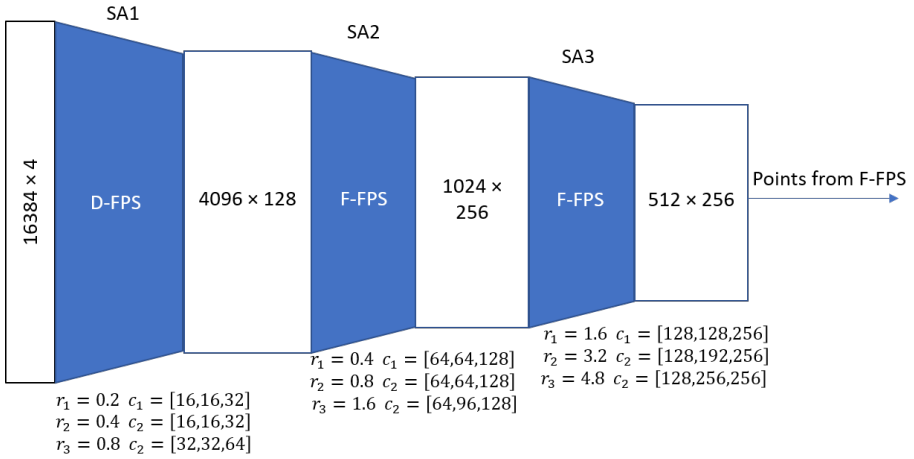


Figure 4.3: Backbone Implementing F-FPS Only

It is notable that for the F-FPS only backbone, the first SA layer implements D-FPS strategy, this is because the input of this layer is the point coordinates and reflectance of the point cloud, which do not carry semantic information that is necessary for the effectiveness of feature distance calculation. With more feature channels generated by the first SA layer, F-FPS is implemented only in the following SA layers.

To compare the effect of different sampling strategies, other parts of backbone structures and parameters such as grouping radii are kept consistent. The voting

module following the backbone takes 256 out of 512 points as the seed points to meet the number of F-FPS sampled points in the fusion sampling setup.

Density-aware Sampling

The original idea was to implement density-aware sampling across all the SA layers, replacing the D-FPS sampling. However, experiments demonstrated significant challenges with this approach. The loss functions exhibited poor convergence, resulting in mAP scores below 30%. Furthermore, the training and inference times are more than doubled compared to other models. As a remedy, the decision was made to apply DA-FPS only in the first SA layer with the expectation that the sampled points would tend to exhibit a more uniform distribution after this layer.

As described in Section 3.2.1, λ is a hyperparameter used to balance the influence of density on metric distance. A larger λ corresponds to a larger density-aware factor k, thereby increasing the contribution of density to the metric distance. To thoroughly explore this hyperparameter's impact, the experiment investigates four distinct λ values: $\lambda = 0.1, 0.2, 0.4, 1$. These values are evaluated to comprehend the relationship between the density-aware factor and overall model performance.

Semantic-aware Sampling

The backbone implementing semantic-aware sampling shown in Figure 3.7 is a generic structure that does not indicate the quantity of foreground and background points. To explore how many foreground points work better for the detection, we investigate different ratios between foreground and background points in layers SA2 and SA3 in the experiment. Three settings of ratios are experimented as shown in Table 4.1, the numbers in the table are the actual quantities of the points.

Setting Number	**F/B Ratio in SA2**	**F/B Ratio in SA3**
1	384:640	192:320
2	512:512	256:256
3	896:128	448:64

Table 4.1: Foreground and Background Points Ratio Setting

In the experiments on semantic-aware sampling, only the ratios between foreground and background points are changed, and the total number of sampled points

in each SA layer remains consistent. In the subsequent candidate generation layer, foreground points are taken as seed points and shifted into candidate points.

4.1.3 Models Training

We implement several training strategies in pursuit of training efficiency and performance of the trained model. They are detailed in the following paragraphs.

Mini-batch Training For a dataset containing a huge number of points, it is inefficient and memory-demanding to implement full-batch training which consumes the whole dataset. With a large training dataset and a 3D object detection model with high complexity, online training is noisy and takes a lot of time to converge. In this research, we adopt mini-batch training which strikes the balance between full-batch training and online training. The batch size is set to 4, which fits into the memory of the GPU we use.

AdamW Optimization We utilize AdamW as an optimization strategy to facilitate fast training and stable convergence. AdamW is an improved version of Adam, which makes use of the moving averages of the gradients (first moment) and squared gradients (second moment) to update the effective learning rate adaptively. Compared to Adam, AdamW decouples weight decay from the adaptive learning rate, offering improved generalization and training stability, especially in deep and large-scale models.

Learning Rate Adjustment A step decay strategy is implemented to adjust the learning rate during training. The learning rate for all models is set to 0.002 initially. At the 45th epoch, the learning rate decays by a factor of 10, to 0.0002. There is one more decay at the 65th epoch by the same factor, reducing the learning rate to 0.00002. The larger starting learning rate leads to fast convergence, and a decayed rate at the end of training avoids overshooting.

Consistent Epoches For the training of every model, we keep the epoch number at 80. This number has been proven to be working for the model to converge.

4.1.4 Models Evaluation

We adopt the metrics from KITTI to calculate the mean average precision (mAP) of the 3D object detection results. AP is obtained by calculating the area under the precision-recall curve (AUC-PR) for each point cloud frame. mAP is the average of the AP values across all frames.

In 3D object detection, the judgment for a true positive is based on the intersection-over-union (IoU) of a predicted bounding box. When the IoU between a predicted bounding box and ground truth bounding box is over a threshold (0.7 for car class in KITTI), it is seen as a true positive. The predicted confidence scores of a frame are sorted and ranked in descending order. For each step of the confidence scores from the highest to the lowest, precision and recall are calculated following Equation 4.1 and 4.2.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{4.1}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negtives} \tag{4.2}$$

Precisions and recalls obtained at all the confidence scores are used to draw the precision-recall curve. Then the area under the precision-recall curve is calculated to get the AP of a frame.

For the KITTI dataset evaluation, there are three difficulty levels according to the size, occlusion and truncation of the objects in the scene. The definition is shown in Table 4.2. In the research, we calculate the mAP of the car class for each difficulty level.

Difficulty Level	**Min. Bounding Box Height**	**Max. Occlusion Level**	**Max. Truncation**
Easy	40 Pixels	Fully visible	15%
Moderate	25 Pixels	Partly occluded	30%
Hard	25 Pixels	Difficult to see	50%

Table 4.2: The Difficulty Definition of KITTI Dataset

The mAP metric reflects a model's proficiency to accurately locate and classify objects in a point cloud. However, it is meaningless if the inference time is too long under scenarios such as autonomous driving. So inference time is another necessary metric to evaluate a model. During the testing phase, we compute the inference time by measuring the duration between inputting data into the model and obtaining the

prediction results for a given data frame. By calculating the average inference time across all frames, we establish a basis for comparing models in terms of their efficiency during the prediction process.

4.1.5 Visualization

The established metrics of mAP and inference time serve as comprehensive evaluative tools for gauging the ultimate model performance. However, these metrics alone don't provide insight into how sampling strategies function within different modules of the network. To gain a deeper understanding of the effects of diverse sampling strategies on object detection, we propose incorporating visualization techniques that offer intuitive insights.

During the testing phase, the following information is collected and recorded for visualization:

1. **Sampled Points in Each SA Layer**: Points sampled in SA layers (SA1, SA2, and SA3) are saved separately, differentiated by the employed strategies.

2. **Voted Points in Candidate Generation Layer**: Candidate points that have been shifted by the voting module in the candidate generation layer are collected.

3. **Predicted Bounding Boxes**: The predicted bounding boxes for each frame are saved.

4. **Ground Truth Bounding Boxes**: The ground truth bounding boxes for each frame are also included.

By visualizing the above information, we can see: 1) How do the points distribute after sampling; 2) Do the points distributed in the way we expect as the sampling progresses; 3) The ratio between foreground points and background points; 4) Do the candidate points approaching the instance center.

All the points and bounding boxes are stored as .obj files, compatible with various visualization software. In our research, we've chosen Meshlab for visualization purposes. This approach allows us to gain a deeper, visual understanding of how different sampling strategies impact various aspects of object detection within the network's modules.

4.1.6 Statistics Exploration

While visualization offers intuitive insights, statistical metrics are equally vital for comprehending the functioning of various sampling strategies within the network. We collect the following statistics for each frame within the test dataset and carry out an analysis based on the numbers.

1. **Bounding Box Recall Ratio**: A ground truth bounding box represents an object to be detected in a point cloud. If the survived points in a ground truth bounding box are below a specific quantity after sampling, we consider the bounding box being missed as the information carried by the survived internal points is not enough for a successful detection. The bounding box recall ratio is the percentage of ground truth bounding boxes that retain a specific number of internal points after a sampling layer. In our experiments, we try the number of internal points at 1, 5, and 10.

2. **Positive Points Capture Rate**: In SA layers where two sampling strategies are used, one of them is designed to capture more positive points (for example, F-FPS and FBS are intended to gather positive points). To understand how effective a sampling strategy is at capturing positive points, we compute the positive point capture rate. This rate represents the portion of positive points that are selected from all the positive points within a specific layer.

3. **Unique Point Ratio**: In SA layers that implement the first and second sampling strategies, some layers employ fusion sampling, combining points sampled through two distinct strategies. However, the extent of duplication within these samples has not been thoroughly explored. The reason for duplication is points to be sampled may be selected by both sampling strategies and then fused into one point set. It's important to understand how many duplicate points are being sampled, as given a fixed number of samples, duplicate points could potentially lead to information loss. We count the number of unique point ratios after each layer implementing fusion sampling.

4. **Voting Effect**: The voting module shifts the seed points, moving them closer to the center of an object. When more shifted points are situated within the ground truth bounding boxes, it results in more precise predictions. To quantify the impact of voting, we compute a ratio denoted as $N^{pos}_{vote}/N^{pos}_{SA1}$, which signifies

the proportion of positive points after voting compared to those present in the SA1 layer. This effect is influenced by different sampling strategies.

4.1.7 Comparison and Analysis

The combination of evaluation metrics, visualization, and statistical analysis offers a robust framework for conducting comprehensive comparisons within and between the researched sampling strategies.

First, within each sampling strategy, an exploration of different settings is conducted. This process helps in understanding the nuances of each setting and identifying the most effective configuration. This configuration, yielding the best performance, is then selected as the representative setting for subsequent comparison.

The chosen best-performing models from different sampling strategies are compared. This step reveals which strategy demonstrates the most promising results and holds potential for further exploration in the context of 3D object detection. Visualization and statistics are also compared between the various sampling strategies to provide further insights.

After the comprehensive comparison, the strengths and shortcomings of each sampling strategy are thoroughly analyzed. By employing this multifaceted approach, the research gains a comprehensive understanding of how different sampling strategies influence object detection models.

4.1.8 Running Environment

Hardware Training and evaluation of the models are performed on a machine equipped with 2 NVIDIA GPUs. The main specifications of this machine are provided in Table 4.3 below:

Processor	AMD EPYC 7601 (16 CPUs)
Memory	64 GB DDR4 RAM
Graphics Card	NVIDIA GeForce RTX 2080 Ti * 2
Hard Disk Storage	70 GB
OS	Ubuntu 20.04

Table 4.3: Hardware Information

Software The research is conducted on a Linux operating system, with the version specified in Table 4.3. To facilitate our investigations, we leverage an open-source 3D object detection toolbox called MMDetection3D [5]. Developed as part of the OpenMMLab project and built on PyTorch, MMDetection3D offers a rich model zoo comprising SOTA 3D object detection algorithms. It is capable of handling data from multiple sensor modalities such as LiDAR, radar, and camera images. Moreover, it provides seamless access to datasets from KITTI, Waymo, nuScenes, SUN RGB-D, and ScanNet. The flexible design of MMDetection3D makes configuration and modification of the models easy for various experiments and tasks. For reference, Table 4.4 lists the essential supporting software used in this research.

Software	**Version**
Python	3.8.15
CUDA	11.0
CuDNN	8.0.5
PyTorch	1.7.1
TorchVision	0.8.2
OpenCV	4.7.0
MMCV	1.6.2
MMDetection	2.26.0
MMDetection3D	1.0.0

Table 4.4: Software Environment

4.2 Experiment Results and Analysis

The evaluation results according to the setup within each sampling strategy are presented in Table 4.5. Statistic exploration results are presented in Table 4.6, 4.7, 4.8 and 4.9.

As shown in Table 4.5, the semantic-aware sampling strategy, particularly when biased toward foreground points (with a ratio of 7:1), yields the highest mean Average Precision (mAP) scores in the moderate difficulty category, with a 0.19% improvement than that of the 3DSSD baseline. The moderate evaluation level is the one officially used by KITTI to rank 3D object detectors. This result is supported by the statistics in Table 4.6, 4.9 that this setting achieved the highest bounding box recall rate no matter how many internal points are included and the highest ratio between positive points after voting and in the SA1 layer. However, this setting lags behind the

3DSSD fusion strategy and Semantic (F:B=1:1) setting in the hard category. All the models utilizing density-aware sampling strategies have yielded considerably lower mAP scores compared to their counterparts. Additionally, these density-aware models exhibit significantly longer inference times, exceeding 10 times the inference time of the 3DSSD baseline and the semantic-aware strategies, which are all between 70 and 80 milliseconds.

Detailed analysis of various strategies and settings is carried out subsequently with supportive visualizations.

Sampling Strategies & Settings	3D Car mAP (IoU=0.7)			Inference Time (ms)
	Easy	Moderate	Hard	
D-FPS Only	86.8906	77.7105	75.0167	71.55
F-FPS Only	91.0640	79.4326	78.1684	73.53
Fusion (D-FPS & FPS)	91.2986	80.1203	78.7814	72.97
DA-FPS $\lambda = 0.1$	86.8105	73.4519	69.6043	796.58
DA-FPS $\lambda = 0.2$	84.2788	72.0890	67.1594	941.02
DA-FPS $\lambda = 0.4$	83.3943	70.6628	67.2632	796.55
DA-FPS $\lambda = 1$	78.1530	69.0015	64.0699	839.12
S-FPS (F:B=3:5)	88.1598	76.8452	75.4076	71.41
S-FPS (F:B=1:1)	89.2553	79.2137	78.0572	74.31
S-FPS (F:B=7:1)	89.9251	80.3199	76.7134	78.11

Table 4.5: Evaluation results on KITTI val dataset

Sampling Strategies & Settings	Bounding Box Recall Ratio (Internal Point Number = 1)			Bounding Box Recall Ratio (Internal Point Number = 5)			Bounding Box Recall Ratio (Internal Point Number = 10)		
	SA1	SA2	SA3	SA1	SA2	SA3	SA1	SA2	SA3
D-FPS Only	0.9607	0.9198	0.8171	0.9147	0.5849	0.2289	0.8258	0.3142	0.0558
F-FPS Only	0.9607	0.9503	0.9064	0.9147	0.8425	0.6099	0.8258	0.6432	0.3707
Fusion (D-FPS & FPS)	0.9607	0.9531	0.8981	0.9147	0.7943	0.5110	0.8258	0.5498	0.2192
DA-FPS $\lambda = 0.1$	0.9626	0.9373	0.8770	0.9223	0.5968	0.3682	0.8475	0.2640	0.1092
DA-FPS $\lambda = 0.2$	0.9627	0.9347	0.8747	0.9205	0.5983	0.3516	0.8381	0.2654	0.0980
DA-FPS $\lambda = 0.4$	0.9606	0.9411	0.8758	0.9112	0.6173	0.3561	0.8253	0.2620	0.1030
DA-FPS $\lambda = 1$	0.9457	0.9369	0.8706	0.8772	0.6070	0.3477	0.7906	0.2510	0.0932
S-FPS (F:B=3:5)	0.9601	0.9083	0.6128	0.9151	0.5994	0.1220	0.8247	0.3373	0.0285
S-FPS (F:B=1:1)	0.9601	0.9359	0.8112	0.9151	0.7289	0.3335	0.8247	0.4823	0.1427
S-FPS (F:B=7:1)	0.9601	0.9496	0.9327	0.9151	0.8771	0.8309	0.8247	0.7486	0.6830

Table 4.6: Bounding box recall ratio. The bounding box recall ratio indicates the ratio of kept bounding boxes (internal points exist) after sampling.

4.2.1 D-FPS and F-FPS Analysis

The first three rows in Table 4.5 are the experiment results of D-FPS and F-FPS strategies.

	Fusion (D-FPS & F-FPS)	DA-FPS $\lambda = 0.1$	DA-FPS $\lambda = 0.2$	DA-FPS $\lambda = 0.4$	DA-FPS $\lambda = 1$	Semantic (F:B=3:5)	Semantic (F:B = 1:1)	Semantic (F:B = 7:1)
SA2	0.7157	0.4945	0.4913	0.4985	0.4987	0.8601	0.8587	0.9835
SA3	0.711	0.6328	0.6233	0.6267	0.6252	0.9654	0.9165	0.9944

Table 4.7: Positive point capture ratio. For fusion sampling of the 3DSSD baseline and DA-FPS that implements fusion sampling in SA2 and SA3, the positive point capture ratio indicates the ratio of the positive points being sampled by F-FPS in the specific layer. For semantic-aware sampling, it is the ratio of the positive points being sampled by foreground sampling.

	Fusion (D-FPS & F-FPS)	DA-FPS $\lambda = 0.1$	DA-FPS $\lambda = 0.2$	DA-FPS $\lambda = 0.4$	DA-FPS $\lambda = 1$
SA2	0.8924	0.6262	0.6174	0.5325	0.5039
SA3	0.8747	0.7537	0.7536	0.7125	0.7057

Table 4.8: Unique point ratio. The ratio applies to the layers that implement the fusion strategy. DA-FPS is implemented in the SA1 layer, the SA2 and SA3 are fusion layers employing D-FPS and F-FPS.

In terms of inference time, the D-FPS demonstrates the fastest performance while the F-FPS only model requires the longest time among the three settings. This variation in inference time is attributed to the distance calculation process during farthest point sampling. D-FPS is based on 3D Euclidean distance, which calculates the distance in 3D space. F-FPS performs distance calculations in a significantly higher-dimensional space specified by the feature vectors associated with each point. In the SA2 and SA3 layers, the dimensions utilized for feature distance calculation are 128 and 256 respectively. The increased computational demands imposed by F-FPS are reflected in the extended inference time required for this strategy.

The fusion strategy implementing both D-FPS and F-FPS achieves the highest mAP in the experiments, followed by strategies implementing F-FPS only and D-FPS only. The ranking is consistent with that in the original 3DSSD paper, but the mAPs achieved in our experiments are higher, we attribute the increase to an improved optimization method and more training epochs. As discussed in the original paper, a lack of enough negative points leads to poor performance in classification. So a fusion of D-FPS and F-FPS which samples both enough negative and positive points achieves the best performance.

The comparison of the Bounding Box Recall Ratio indicates that F-FPS tends to bring more positive points to the next SA layer and, consequently higher bounding box recall ratio. After going through 3 SA layers, 81% of the bounding boxes survive

	D-FPS Only	F-FPS Only	Fusion (D-FPS & F-FPS)	DA-FPS $\lambda = 0.1$	DA-FPS $\lambda = 0.2$	DA-FPS $\lambda = 0.4$	DA-FPS $\lambda = 1$	Semantic (F:B=3:5)	Semantic (F:B=1:1)	Semantic (F:B=7:1)
$N_{vote}^{pos}/N_{SA1}^{pos}$	0.1236	0.2022	0.2033	0.1377	0.13	0.135	0.1439	0.1458	0.2989	0.7604

Table 4.9: The ratio between the number of positive points existing after voting and in SA1 layer.

with more than 1 internal point using D-FPS only, and around 90% bounding boxes survive when using F-FPS only and fusion strategy. As the threshold of internal point points increases, the recall ratio decreases over all settings. However, the recall ratio is only 5% when the threshold is 10 with the D-FPS only strategy, compared to 37% when using F-FPS only, reflecting that F-FPS samples much more positive points than D-FPS.

In the model that incorporates fusion sampling, the Positive Point Capture Rates achieved by F-FPS in both SA2 and SA3 are around 71%, which implies D-FPS captures the remaining 29% of positive points. The Unique Point Ratios are 89% and 87% in SA2 and SA3 respectively, indicating that around 10% of the points from the preceding layer are sampled by both D-FPS and F-FPS. This overlap leads to information redundancy within these layers, an aspect that was not explored in the original paper.

$N_{vote}^{pos}/N_{SA1}^{pos}$ is the ratio between the number of positive points present after voting and the number of positive points in SA1. It reflects the impact that SA2 and SA3 impose on the voting effect. In the model employing fusion sampling, this ratio is the highest, closely followed by the model using F-FPS only. On the other hand, the D-FPS only model exhibits the lowest ratio. The ranking of $N_{vote}^{pos}/N_{SA1}^{pos}$ mirrors the same ranking as the mAP metric.

Figure 4.4 showcases the detection results and point distribution of three settings on the same data frame. When implementing D-FPS only, the sampled points evenly distribute over the whole space, with much more negative points (out of the boxes) compared to the fusion and F-FPS only strategy. For the fusion strategy, most yellow points (sampled by F-FPS) are in the bounding boxes, offering a certain quantity of positive candidate points. Meanwhile, the red points (sampled by D-FPS) distribute in an even pattern, supporting the detection as background points. F-FPS only strategy makes most of the sampled points in the SA3 layer concentrated in the bounding boxes, the other points scatter in a random way.

4.2.2 Density-aware Sampling Analysis

Density-aware sampling strategy yields lower mAPs than other strategies and much longer inference times. Longer inference time can be expected as the calculation of point-wise density is time-consuming. However, the magnitude of the increased time is beyond real-time application. Optimization of the density calculation may reduce the time but a more important question is why the strategy can not improve the mAP on the baseline, or how the strategy influences the result. From Table 4.6 we can see that when λ= 0.1 and 0.2, the bounding box recall ratios in the SA1 layer (where DA-FPS is applied) are lifted no matter how many internal points are included. But the improvement does not benefit the bounding box recall ratio in the following SA2 and SA3 layers, which are fusion layers the same as 3DSSD. The bounding box recall ratios in SA2 and SA3 are lower than those of the 3DSSD fusion strategy. The DA-FPS in the SA1 layer also decreases the positive point capture ratio in the SA2 and SA3 layers by 0.2 and 0.1 respectively. The unique point ratio in SA3 decreases from 0.75 to 0.70 when λ increases from 0.1 to 1. Nearly 30% of the sampled points in the SA3 layer are redundant in this scenario, serving as a prominent reason for the deteriorated performance when compared to the baseline.

Figure 4.5 clearly demonstrates the influence that density-aware sampling imposes on the point distribution in the SA1 layer. As the value of λ increases, we can see from the figure that the dense areas close to the sensor (bottom of the figure) become sparser. When $\lambda = 1$, the bottom part conserves such a small number of points that more bounding boxes are missed compared to the results with smaller λ. The density-aware strategy can effectively influence the density imbalance to an extent controlled by the density-aware factor. But figuring out the best-working factor according to the point cloud is a problem to be researched. Methods used in the following layers for avoiding duplicate sampling are also necessary for a density-aware strategy to achieve improvement on the baseline.

4.2.3 Semantic-aware Sampling Analysis

Semantic-aware sampling is based on confidence in point-wise semantic segmentation. The evaluation results from various settings reflect that it is the foreground point that mostly matters in point-based 3D object detection. The inference times are comparable with the 3DSSD baseline. A higher percentage of foreground points leads to a slight increase in inference time. The reason is that more foreground points

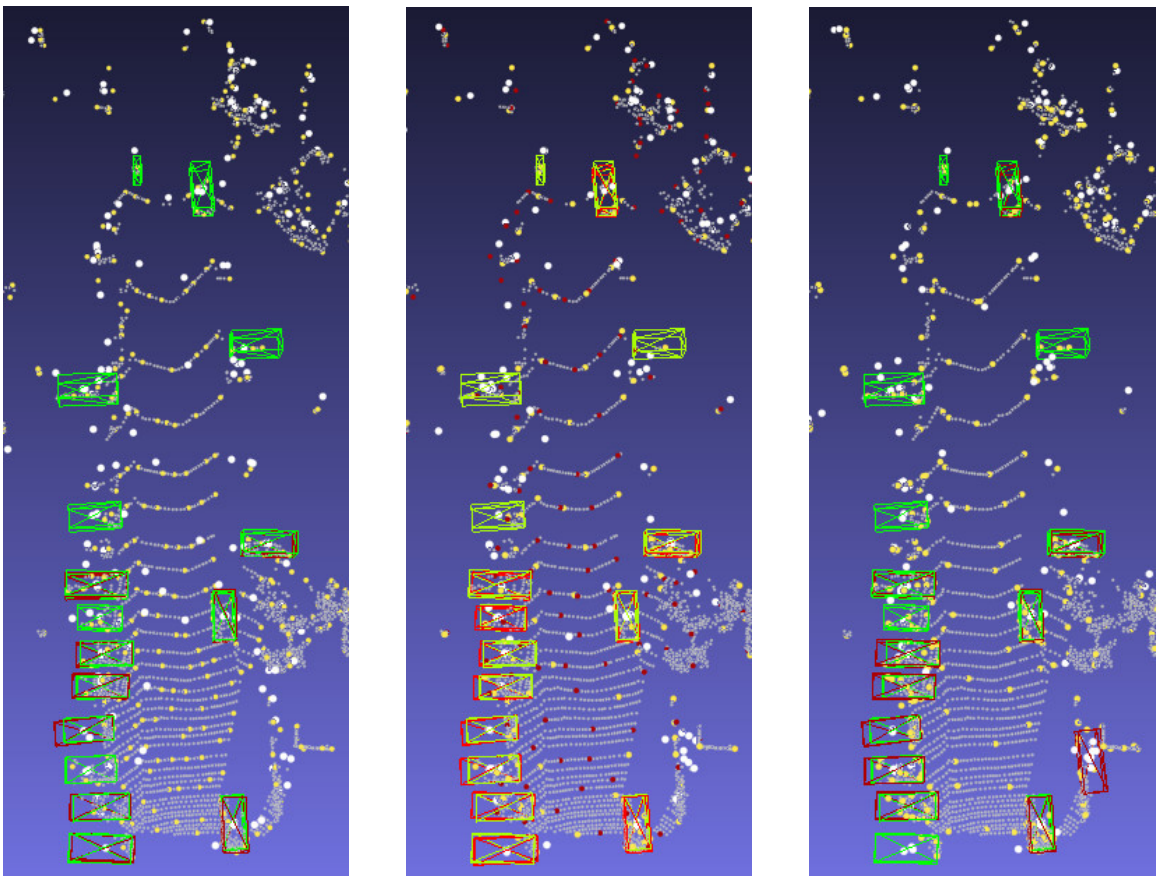
introduce more ball query operations at the voting stage.

The bounding box recall ratios are closely related to the ratio between foreground and background points. It can be seen from Table 4.6 that when the foreground points take the majority, the bounding box recall ratio is higher than those of any other strategies and settings with a big gap. When the threshold of internal points is 10, this setting can still recall 68.30% of the bounding boxes at the SA3 layer which consists of 512 points.

An advantage of semantic-aware sampling is that there is no duplicate point in the sampled points, as the sampling is carried out separately on segmented foreground points and background points. The positive point capture ratio is close to 1 as the segmentation is accurate that no positive point exists in segmented background points. $N^{pos}_{vote}/N^{pos}_{SA1}$ in Table 4.9 indicates that after voting, more than 76% foreground points in SA1 are kept by the foreground-biased semantic-aware sampling, the corresponding value of 3DSSD fusion sampling is only 20.33%. The big gap of $N^{pos}_{vote}/N^{pos}_{SA1}$ between these two settings does not reflect on the mAP. Because a certain quantity of foreground points is enough for a good 3D object detector to classify and localize the objects.

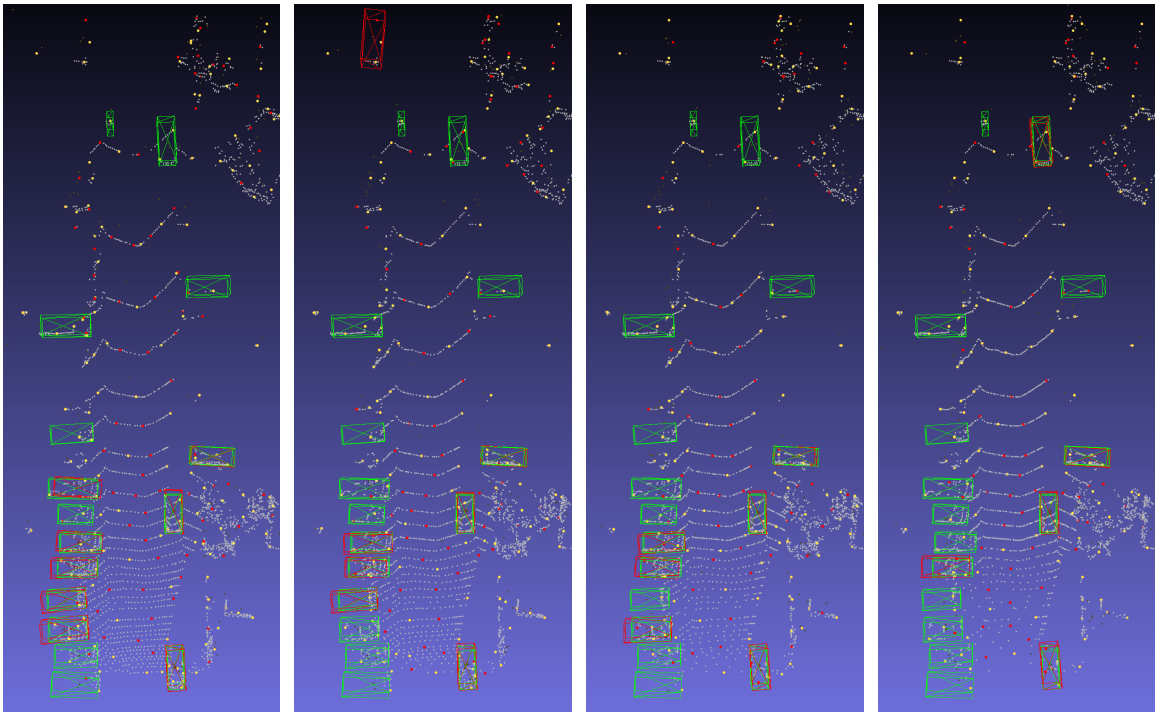
Figure 4.6 shows that semantic-aware sampling can clearly distinguish foreground and background points. When there are more background points, the sparse foreground points mostly scatter on the boundary of bounding boxes. When the ratio between foreground and background points is high, points close to the sensor distribute in obvious clusters and produce high-confidence bounding boxes. However, this can cause training imbalance towards the clustered points, while objects in the distance with fewer internal points are missed, as shown in Figure 4.6c

The above analysis shows both pros and cons of the three sampling strategies and offers us a deeper understanding of how they can affect the 3D object detection networks. Among three of them, we see the semantic-aware strategy as a more promising one as it delivers more information for the subsequent layers. Future research will be biased toward the subsequent layers that utilize the downsampled points.



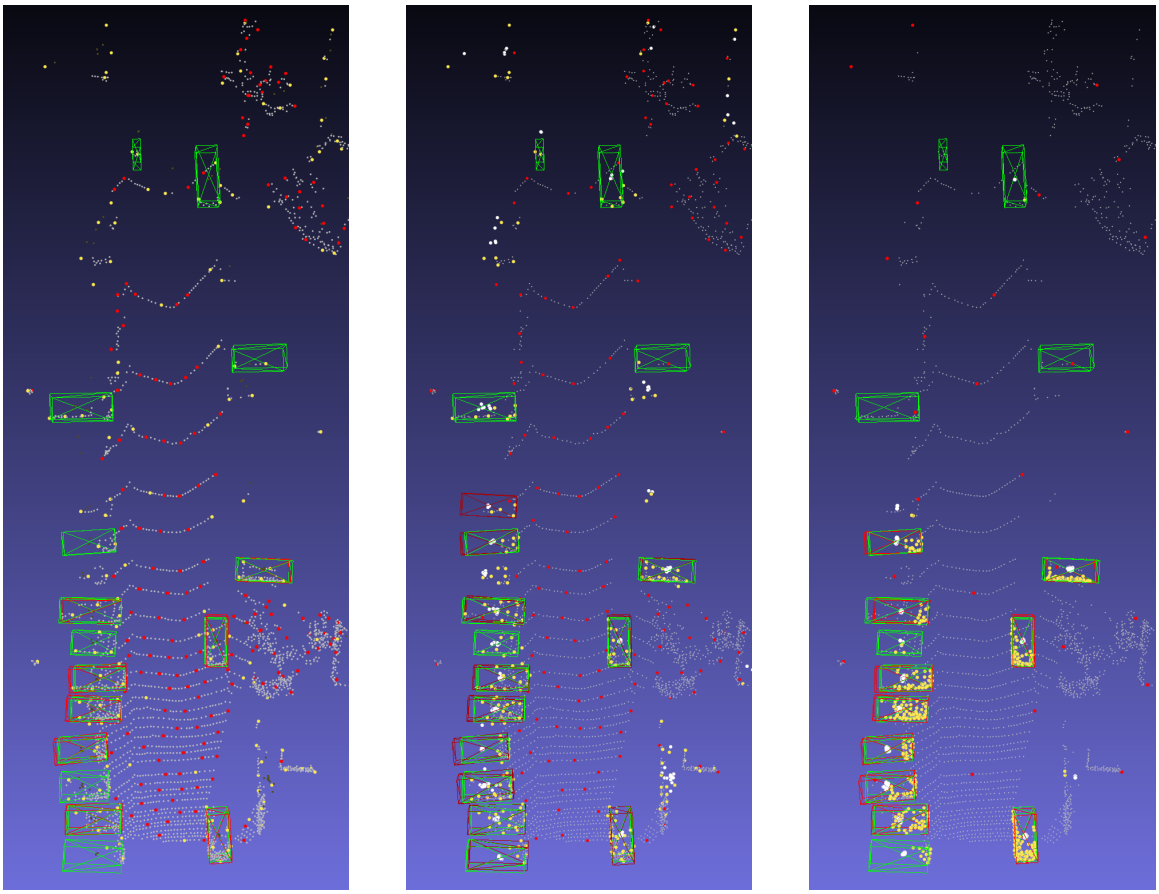
(a) D-FPS Only (b) Fusion (c) F-FPS Only

Figure 4.4: Results of different FPS settings on 3DSSD. The vast small grey points are sampled points in the SA1 layers. The big white dots are voted points. Green and red boxes are ground truth and predicted bounding boxes respectively. Yellow points in (a) and (c) are points sampled by D-FPS and F-FPS respectively in the SA3 layers. Yellow points in (b) are sampled by F-FPS while the red ones are sampled by D-FPS.



(a) $\lambda = 0.1$ (b) $\lambda = 0.2$ (c) $\lambda = 0.4$ (d) $\lambda = 1$

Figure 4.5: Results of different settings on density-aware sampling. The vast grey points are sampled points in the SA1 layers. Green and red boxes are ground truth and predicted bounding boxes respectively. Yellow points and red points are points sampled by F-FPS and D-FPS respectively in the SA3 layers.



(a) F:B=3:5 (b) F:B=1:1 (c) F:B=7:1

Figure 4.6: Results of different settings on semantic-aware sampling. The vast small grey points are sampled points in the SA1 layers. The big white dots are voted points. Green and red boxes are ground truth and predicted bounding boxes respectively. Yellow points and red points are points sampled from foreground points and background points respectively in the SA3 layers.

Chapter 5

Conclusions

In this research, we argue that point sampling plays an important role in point-based 3D object detection in terms of both performance and efficiency. Besides the farthest point sampling based on Euclidean distance and feature distance, we explored density-aware sampling and semantic-aware sampling on the 3DSSD framework. Density-aware sampling controls the density imbalance as designed but consumes too much time to calculate the point-wise density and introduces duplicate sampling. It is not satisfying in both precision and inference speed in our experiments. Semantic-aware sampling demonstrates a strong capability of capturing foreground points and achieves improvement on the 3DSSD baseline with a high ratio of foreground points. Future work includes optimizing the calculation of point density and subsequent grouping to reduce resampling. Another direction is to make better use of the semantic information for further improvement.

Bibliography

[1] Simegnew Yihunie Alaba and John E Ball. A survey on deep-learning-based lidar 3d object detection for autonomous driving. *Sensors*, 22(24):9577, 2022.

[2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[4] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[5] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. `https://github.com/open-mmlab/mmdetection3d`, 2020.

[6] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[7] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[8] Johannes Deichmann, Eike Ebel, Kersten Heineke, Ruth Heuss, Martin Kellner, and Fabian Steiner. Autonomous driving's future: Convenient and connected, Jan 2023.

[9] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1201–1209, 2021.

[10] Energy Technology Policy Division. Global EV Outlook 2023. Technical report, INTERNATIONAL ENERGY AGENCY, 2023.

[11] Marc-Antoine Drouin and Lama Seoud. Consumer-grade RGB-D cameras. In Yonghuai Liu, Nick Pears, Paul L. Rosin, and Patrik Huber, editors, *3D Imaging, Analysis and Applications*, pages 215–264. Springer International Publishing, Cham, 2020.

[12] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

[13] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. Ieee, 2008.

[14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[18] Jordan SK Hu, Tianshu Kuai, and Steven L Waslander. Point density-aware voxels for lidar 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8469–8478, 2022.

[19] Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2702–2719, 2019.

[20] Kirsten Korosec. Lidar companies face a "make it or break it" year, Jan 2023.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[22] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[23] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

[24] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518. IEEE, 2017.

[25] Hongyu Li, Youhui Guo, Yu Zhou, and Weiping Wang. Density-net: A density-aware network for 3d object detection. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1105–1112. IEEE, 2021.

[26] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 641–656, 2018.

[27] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[28] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[29] Paul F. McManamon. *LiDAR Technologies and Systems*. SPIE Press, Bellingham, Washington, USA, 2019.

[30] Jingmei Ning, Feipeng Da, and Shaoyan Gai. Density aware 3d object single stage detector. *IEEE Sensors Journal*, 21(20):23108–23117, 2021.

[31] Felix Nobis, Ehsan Shafiei, Phillip Karle, Johannes Betz, and Markus Lienkamp. Radar voxel fusion for 3d object detection. *Applied Sciences*, 11(12):5598, 2021.

[32] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019.

[33] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[34] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[35] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[36] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[38] John Rendleman. Engines of change, Aug 2007.

[39] Zhu Rui, Li Kin, Fun, Guo Chenchen, and Huang Jinmin. A comparative analysis of point sampling strategies in point-based 3d object detection. Paper presentation at IVPAI, Shenzhen, 2023.

[40] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[41] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[42] Zhaoyu Su, Pin Siang Tan, and Yu-Hsing Wang. Dv-det: Efficient 3d point cloud object detection with dynamic voxelization. *arXiv preprint arXiv:2107.12707*, 2021.

[43] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[44] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[45] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[46] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[47] Haiyang Wang, Shaoshuai Shi, Ze Yang, Rongyao Fang, Qi Qian, Hongsheng Li, Bernt Schiele, and Liwei Wang. Rbgnet: Ray-based grouping for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1110–1119, 2022.

[48] Hai Wu, Chenglu Wen, Shaoshuai Shi, Xin Li, and Cheng Wang. Virtual sparse convolution for multimodal 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21653–21662, 2023.

[49] Yutian Wu, Yueyu Wang, Shuwei Zhang, and Harutoshi Ogai. Deep 3d object detection networks using lidar data: A review. *IEEE Sensors Journal*, 21(2):1152–1171, 2020.

[50] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[51] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[52] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11040–11048, 2020.

[53] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016.

[54] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18953–18962, 2022.

[55] Ziyu Zhang, Feipeng Da, and Yi Yu. Data-free point cloud network for 3d face recognition. *arXiv preprint arXiv:1911.04731*, 2019.

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

[57] Michael Zollhöfer. Commodity RGB-D Sensors: Data Acquisition. In Paul L. Rosin, Yu-Kun Lai, Ling Shao, and Yonghuai Liu, editors, *RGB-D Image Analysis and Processing*, pages 3–13. Springer International Publishing, Cham, 2019.

[58] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object Detection in 20 Years: A Survey. *Proceedings of the IEEE*, 111(3):257–276, 2023.